

4. Instruction tables

Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD, and VIA CPUs

By Agner Fog. Technical University of Denmark.
Copyright © 1996 – 2019. Last updated 2019-08-15.

Introduction

This is the fourth in a series of five manuals:

1. Optimizing software in C++: An optimization guide for Windows, Linux and Mac platforms.
2. Optimizing subroutines in assembly language: An optimization guide for x86 platforms.
3. The microarchitecture of Intel, AMD and VIA CPUs: An optimization guide for assembly programmers and compiler makers.
4. Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs.
5. Calling conventions for different C++ compilers and operating systems.

The latest versions of these manuals are always available from www.agner.org/optimize.
Copyright conditions are listed below.

The present manual contains tables of instruction latencies, throughputs and micro-operation breakdown and other tables for x86 family microprocessors from Intel, AMD and VIA.

The figures in the instruction tables represent the results of my measurements rather than the official values published by microprocessor vendors. Some values in my tables are higher or lower than the values published elsewhere. The discrepancies can be explained by the following factors:

- My figures are experimental values while figures published by microprocessor vendors may be based on theory or simulations.
- My figures are obtained with a particular test method under particular conditions. It is possible that different values can be obtained under other conditions.
- Some latencies are difficult or impossible to measure accurately, especially for memory access and type conversions that cannot be chained.
- Latencies for moving data from one execution unit to another are listed explicitly in some of my tables while they are included in the general latencies in some tables published by Intel.

Most values are the same in all microprocessor modes (real, virtual, protected, 16-bit, 32-bit, 64-bit). Values for far calls and interrupts may be different in different modes. Call gates have not been tested.

Instructions with a LOCK prefix have a long latency that depends on cache organization and possibly RAM speed. If there are multiple processors or cores or direct memory access (DMA) devices then all locked instructions will lock a cache line for exclusive access, which may involve RAM access. A LOCK prefix typically costs more than a hundred clock cycles, even on single-processor systems. This also applies to the XCHG instruction with a memory operand.

If any text in the pdf version of this manual is unreadable, then please refer to the spreadsheet version.

Copyright notice

Introduction

This series of five manuals is copyrighted by Agner Fog. Public distribution and mirroring is not allowed. Non-public distribution to a limited audience for educational purposes is allowed. A creative commons license CC-BY-SA shall automatically come into force when I die. See <https://creativecommons.org/licenses/by-sa/4.0/legalcode>

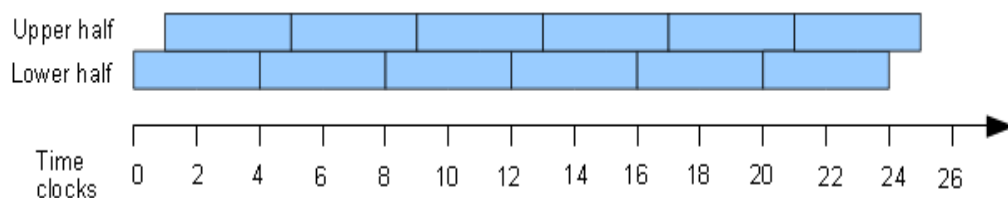
Definition of terms

Instruction The instruction name is the assembly code for the instruction. Multiple instructions or multiple variants of the same instruction may be joined into the same line. Instructions with and without a 'v' prefix to the name have the same values unless otherwise noted.

Operands Operands can be different types of registers, memory, or immediate constants. Abbreviations used in the tables are: i = immediate constant, r = any general purpose register, r32 = 32-bit register, etc., mm = 64 bit mmx register, x or xmm = 128 bit xmm register, y = 256 bit ymm register, z = 512 bit zmm register, v = any vector register, sr = segment register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc.

Latency The latency of an instruction is the delay that the instruction generates in a dependency chain. The measurement unit is clock cycles. Where the clock frequency is varied dynamically, the figures refer to the core clock frequency. The numbers listed are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity may increase the latencies by possibly more than 100 clock cycles on many processors, except in move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results may give a similar delay. A missing value in the table means that the value has not been measured or that it cannot be measured in a meaningful way.

Some processors have a pipelined execution unit that is smaller than the largest register size so that different parts of the operand are calculated at different times. Assume, for example, that we have a long dependency chain of 128-bit vector instructions running in a fully pipelined 64-bit execution unit with a latency of 4. The lower 64 bits of each operation will be calculated at times 0, 4, 8, 12, 16, etc. And the upper 64 bits of each operation will be calculated at times 1, 5, 9, 13, 17, etc. as shown in the figure below. If we look at one 128-bit instruction in isolation, the latency will be 5. But if we look at a long chain of 128-bit instructions, the total latency will be 4 clock cycles per instruction plus one extra clock cycle in the end. The latency in this case is listed as 4 in the tables because this is the value it adds to a dependency chain.



Reciprocal throughput The throughput is the maximum number of instructions of the same kind that can be executed per clock cycle when the operands of each instruction are independent of the preceding instructions. The values listed are the reciprocals of the throughputs, i.e. the average number of clock cycles per instruction when the instructions are not part of a limiting dependency chain. For example, a reciprocal throughput of 2 for FMUL means that a new FMUL instruction can start executing 2 clock cycles after a previous FMUL. A reciprocal throughput of 0.33 for ADD means that the execution units can handle 3 integer additions per clock cycle. The reason for listing the reciprocal values is that this makes comparisons between latency and throughput easier. The reciprocal throughput is also called issue latency. The values listed are for a single thread or a single core. A missing value in the table means that the value has not been measured.

Definition of terms

| | |
|------------------------|---|
| μops | Uop or μop is an abbreviation for micro-operation. Processors with out-of-order cores are capable of splitting complex instructions into μops. For example, a read-modify instruction may be split into a read-μop and a modify-μop. The number of μops that an instruction generates is important when certain bottlenecks in the pipeline limit the number of μops per clock cycle. |
| Execution unit | The execution core of a microprocessor has several execution units. Each execution unit can handle a particular category of μops, for example floating point additions. The information about which execution unit a particular μop goes to can be useful for two purposes. Firstly, two μops cannot execute simultaneously if they need the same execution unit. And secondly, some processors have a latency of an extra clock cycle when the result of a μop executing in one execution unit is needed as input for a μop in another execution unit. |
| Execution port | The execution units are clustered around a few execution ports on most Intel processors. Each μop passes through an execution port to get to the right execution unit. An execution port can be a bottleneck because it can handle only one μop at a time. Two μops cannot execute simultaneously if they need the same execution port, even if they are going to different execution units. |
| Instruction set | <p>This indicates which instruction set an instruction belongs to. The instruction is only available in processors that support this instruction set. The most important instruction sets are listed on the next page. Availability in processors prior to 80386 does not apply for 32-bit and 64-bit operands. Availability in the MMX instruction set does not apply to 128-bit packed integer instructions, which require SSE2. Availability in the SSE instruction set does not apply to double precision floating point instructions, which require SSE2.</p> <p>32-bit instructions are available in 80386 and later. 64-bit instructions in general purpose registers are available only under 64-bit operating systems. Instructions that use XMM registers (SSE and later), YMM registers (AVX and later), and ZMM registers (AVX512 and later) are only available under operating systems that support these register sets.</p> |

How the values were measured

The values in the tables are measured with the use of my own test programs, which are available from www.agner.org/optimize/testp.zip

The time unit for all measurements is CPU clock cycles. It is attempted to obtain the highest clock frequency if the clock frequency is varying with the workload. Many Intel processors have a performance counter named "core clock cycles". This counter gives measurements that are independent of the varying clock frequency. Where no "core clock cycles" counter is available, the "time stamp counter" is used (RDTSC instruction). In cases where this gives inconsistent results (e.g. in AMD Bobcat) it is necessary to make the processor boost the clock frequency by executing a large number of instructions (> 1 million) or turn off the power-saving features in the BIOS setup.

Instruction throughputs are measured with a long sequence of instructions of the same kind, where subsequent instructions use different registers in order to avoid dependence of each instruction on the previous one. The input registers are cleared in the cases where it is impossible to use different registers. The test code is carefully constructed in each case to make sure that no other bottleneck is limiting the throughput than the one that is being measured.

Instruction latencies are measured in a long dependency chain of identical instructions where the output of each instruction is needed as input for the next instruction.

The sequence of instructions should be long, but not so long that it doesn't fit into the level-1 code cache. A typical length is 100 instructions of the same type. This sequence is repeated in a loop if a larger number of instructions is desired.

Definition of terms

It is not possible to measure the latency of a memory read or write instruction with software methods. It is only possible to measure the combined latency of a memory write followed by a memory read from the same address. What is measured here is not actually the cache access time, because in most cases the microprocessor is smart enough to make a "store forwarding" directly from the write unit to the read unit rather than waiting for the data to go to the cache and back again. The latency of this store forwarding process is arbitrarily divided into a write latency and a read latency in the tables. But in fact, the only value that makes sense to performance optimization is the sum of the write time and the read time.

A similar problem occurs where the input and the output of an instruction use different types of registers. For example, the MOVD instruction can transfer data between general purpose registers and XMM vector registers. The value that can be measured is the combined latency of data transfer from one type of registers to another type and back again ($A \rightarrow B \rightarrow A$). The division of this latency between the $A \rightarrow B$ latency and the $B \rightarrow A$ latency is sometimes obvious, sometimes based on guesswork, μ op counts, indirect evidence, or triangular sequences such as $A \rightarrow B \rightarrow \text{Memory} \rightarrow A$. In many cases, however, the division of the total latency between $A \rightarrow B$ latency and $B \rightarrow A$ latency is arbitrary. However, what cannot be measured cannot matter for performance optimization. What counts is the sum of the $A \rightarrow B$ latency and the $B \rightarrow A$ latency, not the individual terms.

The μ op counts are usually measured with the use of the performance monitor counters (PMCs) that are built into modern microprocessors. The PMCs for VIA processors are undocumented, and the interpretation of these PMCs is based on experimentation.

The execution ports and execution units that are used by each instruction or μ op are detected in different ways depending on the particular microprocessor. Some microprocessors have PMCs that can give this information directly. In other cases it is necessary to obtain this information indirectly by testing whether a particular instruction or μ op can execute simultaneously with another instruction/ μ op that is known to go to a particular execution port or execution unit. On some processors, there is a delay for transmitting data from one execution unit (or cluster of execution units) to another. This delay can be used for detecting whether two different instructions/ μ ops are using the same or different execution units.

Instruction sets

Explanation of instruction sets for x86 processors

| | |
|---------|---|
| x86 | This is the name of the common instruction set, supported by all processors in this lineage. |
| 80186 | This is the first extension to the x86 instruction set. New integer instructions: PUSH i, PUSHA, POPA, IMUL r,r,i, BOUND, ENTER, LEAVE, shifts and rotates by immediate $\neq 1$. |
| 80286 | System instructions for 16-bit protected mode. |
| 80386 | The eight general purpose registers are extended from 16 to 32 bits. 32-bit addressing. 32-bit protected mode. Scaled index addressing. MOVZX, MOVSX, IMUL r,r, SHLD, SHRD, BT, BTR, BTS, BTC, BSF, BSR, SETcc. |
| 80486 | BSWAP. Later versions have CPUID. |
| x87 | This is the floating point instruction set. Supported when a 8087 or later coprocessor is present. Some 486 processors and all processors since Pentium/K5 have built-in support for floating point instructions without the need for a coprocessor. |
| 80287 | FSTSW AX |
| 80387 | FPREM1, FSIN, FCOS, FSINCOS. |
| Pentium | RDTSC, RDPMS. |
| PPro | Conditional move (CMOV, FCMOV) and fast floating point compare (FCOMI) instructions introduced in Pentium Pro. These instructions are not supported in Pentium MMX, but are supported in all processors with SSE and later. |
| MMX | Integer vector instructions with packed 8, 16 and 32-bit integers in the 64-bit MMX registers MM0 - MM7, which are aliased upon the floating point stack registers ST(0) - ST(7). |
| SSE | Single precision floating point scalar and vector instructions in the new 128-bit XMM registers XMM0 - XMM7. PREFETCH, SFENCE, FXSAVE, FXRSTOR, MOVNTQ, MOVNTPS. The use of XMM registers requires operating system support. |
| SSE2 | Double precision floating point scalar and vector instructions in the 128-bit XMM registers XMM0 - XMM7. 64-bit integer arithmetics in the MMX registers. Integer vector instructions with packed 8, 16, 32 and 64-bit integers in the XMM registers. MOVNTI, MOVNTPD, PAUSE, LFENCE, MFENCE. |
| SSE3 | FISTTP, LDDQU, MOVDDUP, MOVSHDUP, MOVSLLDUP, ADDSUBPS, ADDSPPD, HADDPS, HADDPD, HSUBPS, HSUBPD. |
| SSSE3 | (Supplementary SSE3): PSHUFB, PHADDW, PHADDSW, PHADDD, PMADDUBSW, PHSUBW, PHSUBSW, PHSUBD, PSIGNB, PSIGNW, PSIGND, PMULHRWS, PABSB, PABSW, PABSD, PALIGNR. |
| 64 bit | <p>This instruction set is called x86-64, x64, AMD64 or EM64T. It defines a new 64-bit mode with 64-bit addressing and the following extensions: The general purpose registers are extended to 64 bits, and the number of general purpose registers is extended from eight to sixteen. The number of XMM registers is also extended from eight to sixteen, but the number of MMX and ST registers is still eight. Data can be addressed relative to the instruction pointer. There is no way to get access to these extensions in 32-bit mode</p> <p>Most instructions that involve segmentation are not available in 64 bit mode. Direct far jumps and calls are not allowed, but indirect far jumps, indirect far calls and far returns are allowed. These are used in system code for switching mode. Segment registers DS, ES, and SS cannot be used. The FS and GS segments and segment prefixes are available in 64 bit mode and are used for addressing thread environment blocks and processor environment blocks</p> |

Instruction sets

| | |
|---|---|
| Instructions not available in 64 bit mode | <p>The following instructions are not available in 64-bit mode: PUSHA, POPA, BOUND, INTO, BCD instructions: AAA, AAS, DAA, DAS, AAD, AAM, undocumented instructions (SALC, ICEBP, 82H alias for 80H opcode), SYSENTER, SYSEXIT, ARPL. On some early Intel processors, LAHF and SAHF are not available in 64 bit mode. Increment and decrement register instructions cannot be coded in the short one-byte opcode form because these codes have been reassigned as REX prefixes.</p> <p>Most instructions that involve segmentation are not available in 64 bit mode. Direct far jumps and calls are not allowed, but indirect far jumps, indirect far calls and far returns are allowed. These are used in system code for switching mode. PUSH CS, PUSH DS, PUSH ES, PUSH SS, POP DS, POP ES, POP SS, LDS and LES instructions are not allowed. CS, DS, ES and SS prefixes are allowed but ignored. The FS and GS segments and segment prefixes are available in 64 bit mode and are used for addressing thread environment blocks and processor environment blocks.</p> |
| SSE4.1 | MPSADBW, PHMINPOSUW, PMULDQ, PMULLD, DPPS, DPPD, BLEND..., PMIN..., PMAX..., ROUND..., INSERT..., EXTRACT..., PMOVXS..., PMOVZX..., PTEST, PCMPEQQ, PACKUSDW, MOVNTDQA |
| SSE4.2 | CRC32, PCMPESTRI, PCMPESTRM, PCMPISTRI, PCMPISTRM, PCMPGTQ, POPCNT. |
| AES | AESDEC, AESDECLAST, AESENC, AESENCLAST, AESIMC, AESKEYGENASSIST. |
| CLMUL | PCLMULQDQ. |
| AVX | The sixteen 128-bit XMM registers are extended to 256-bit YMM registers with room for further extension in the future. The use of YMM registers requires operating system support. Floating point vector instructions are available in 256-bit versions. Almost all previous XMM instructions now have two versions: with and without zero-extension into the full YMM register. The zero-extension versions have three operands in most cases. Furthermore, the following instructions are added in AVX: VBROADCASTSS, VBROADCASTSD, VEXTRACTF128, VINSERTF128, VLDMXCSR, VMASKMOVPS, VMASKMOVPD, VPERMILPD, VPERMIL2PD, VPERMILPS, VPERMIL2PS, VPERM2F128, VSTMXCSR, VZEROALL, VZERoupper. |
| AVX2 | Integer vector instructions are available in 256-bit versions. Furthermore, the following instructions are added in AVX2: ANDN, BEXTR, BLSI, BLSMSK, BLSR, BZHI, INVPCID, LZCNT, MULX, PEXT, PDEP, RORX, SARX, SHLX, SHRX, TZCNT, VBROADCASTI128, VBROADCASTSS, VBROADCASTSD, VEXTRACTI128, VGATHERDPD, VGATHERQPD, VGATHERDPS, VGATHERQPS, VPGATHERDD, VPGATHERQD, VPGATHERDQ, VPGATHERQQ, VINSERTI128, VPERM2I128, VPERMD, VPERMPD, VPERMPS, VPERMQ, VPMASKMOVD, VPMASKMOVQ, VPSLLVD, VPSLLVQ, VPSRAVD, VPSRLVD, VPSRLVQ. |
| FMA3 | (FMA): Fused multiply and add instructions: VFMADDxxxPD, VFMADDxxxPS, VFMADDxxxSD, VFMADDxxxSS, VFMA DDSUBxxxPD, VFMA DDSUBxxxPS, VFMSUBADDxxxPD, VFMSUBADDxxxPS, VFMSUBxxxPD, VFMSUBxxxPS, VFMSUBxxxSD, VFMSUBxxxSS, VFNMADDxxxPD, VFNMADDxxxPS, VFNMADDxxxSD, VFNMADDxxxSS, VFNM SUBxxxPD, VFNM SUBxxxPS, VFNM SUBxxxSD, VFNM SUBxxxSS. |
| FMA4 | Same as Intel FMA, but with 4 different operands according to a preliminary Intel specification which is now supported only by AMD. Intel's FMA specification has later been changed to FMA3, which is now also supported by AMD. |
| MOVBE | MOVBE |
| POPCNT | POPCNT |
| PCLMUL | PCLMULQDQ |
| XSAVE | |
| XSAVEOPT | |

Instruction sets

| | |
|-------------|--|
| RDRAND | RDRAND |
| RDSEED | RDSEED |
| BMI1 | ANDN, BEXTR, BLSI, BLSMSK, BLSR, LZCNT, TXCNT |
| BMI2 | BZHI, MULX, PDEP, PEXT, RORX, SARX, SHRX, SHLX |
| ADX | ADCX, ADOX, CLAC |
| AVX512F | The 256-bit YMM registers are extended to 512-bit ZMM registers. The number of vector registers is extended to 32 in 64-bit mode, while there are still only 8 vector registers in 32-bit mode. 8 new vector mask registers k0 – k7. Masked vector instructions. Many new instructions. Single- and double precision floating point vectors are always supported. Other instructions are supported if the various optional AVX512 variants, listed below, are supported as well. |
| AVX512BW | Vectors of 8-bit and 16-bit integers in ZMM registers. |
| AVX512DQ | Some additional instructions with vectors of 32-bit and 64-bit integers in ZMM registers. |
| AVX512VL | The vector operations defined for 512-bit vectors in the various AVX512 subsets, including masked operations, can be applied to 128-bit and 256-bit vectors as well. |
| AVX512CD | Conflict detection instructions |
| AVX512ER | Approximate exponential function, reciprocal and reciprocal square root |
| AVX512PF | Gather and scatter prefetch |
| SHA | Secure hash algorithm |
| MPX | Memory protection extensions |
| SMAP | CLAC, STAC |
| CVT16 | VCVTPH2PS, VCVTPS2PH. |
| 3DNow | (AMD only. Obsolete). Single precision floating point vector instructions in the 64-bit MMX registers. Only available on AMD processors. The 3DNow instructions are: FEMMS, PAVGUSB, PF2ID, PFACC, PFADD, PFCMPEQ/GE/LE, PFMAX, PFMIN, PFRCP/IT1/IT2, PFRSQRT/IT1, PFSUB, PFSUBR, PI2FD, PMULHRW, PREFETCH/W. |
| 3DNowE | (AMD only. Obsolete). PF2IW, PFNACC, PFPNACC, PI2FW, PSWAPD. |
| PREFETCHW | This instruction has survived from 3DNow and now has its own feature name |
| PREFETCHWT1 | PREFETCHWT1 |
| SSE4A | (AMD only). EXTRQ, INSERTQ, LZCNT, MOVNTSD, MOVNTSS, POPCNT. (POPCNT shared with Intel SSE4.2). |
| XOP | (AMD only). VFRCZPD, VFRCZPS, VFRCZSD, VFRCZSS, VPCMOV, VPCOMB, VPCOMD, VPCOMQ, PCOMW, VPCOMUB, VPCOMUD, VPCOMUQ, VPCOMUW, VPHADDBD, VPHADDBQ, VPHADDBW, VPHADDQ, VPHADDUBD, VPHADDUBQ, VPHADDUBW, VPHADDUDQ, VPHADDUWD, VPHADDUWQ, VPHADDWD, VPHADDWQ, VPHSUBBW, VPHSUBDQ, VPHSUBWD, VPMACSD, VPMACSDQH, VPMACSDQL, VPMACSSD, VPMACSSDQH, VPMACSSDQL, VPMACSSWD, VPMACSSWW, VPMACSWD, VPMACSWW, VPMADCSSWD, VPMADCSSWW, VPPERM, VPROTB, VPROTD, VPROTQ, VPROTW, VPSHAB, VPSHAD, VPSHAQ, VPSHAW, VPSHLB, VPSHLD, VPSHLQ, VPSHLW. |

Microprocessor versions tested

The tables in this manual are based on testing of the following microprocessors

| Processor name | Microarchitecture Code name | Family number (hex) | Model number (hex) | Comment |
|-----------------------|--------------------------------|---------------------------|--------------------------|-----------------------------|
| AMD K7 Athlon | | 6 | 6 | Step. 2, rev. A5 |
| AMD K8 Opteron | | F | 5 | Stepping A |
| AMD K10 Opteron | | 10 | 2 | 2350, step. 1 |
| AMD Bulldozer | Bulldozer, Zambezi | 15 | 1 | FX-6100, step 2 |
| AMD Piledriver | Piledriver | 15 | 2 | FX-8350, step 0. And others |
| AMD Steamroller | Steamroller, Kaveri | 15 | 30 | A10-7850K, step 1 |
| AMD Excavator | Bristol Ridge | 15 | 65 | A10-9700E, step 1 |
| AMD Ryzen | Zen | 17 | 1 | Ryzen 7 1800X, step 1 |
| AMD Bobcat | Bobcat | 14 | 1 | E350, step. 0 |
| AMD Kabini | Jaguar | 16 | 0 | A4-5000, step 1 |
| Intel Pentium | P5 | 5 | 2 | |
| Intel Pentium MMX | P5 | 5 | 4 | Stepping 4 |
| Intel Pentium II | P6 | 6 | 6 | |
| Intel Pentium III | P6 | 6 | 7 | |
| Intel Pentium 4 | Netburst | F | 2 | Stepping 4, rev. B0 |
| Intel Pentium 4 EM64T | Netburst, Prescott | F | 4 | Xeon. Stepping 1 |
| Intel Pentium M | Dothan | 6 | D | Stepping 6, rev. B1 |
| Intel Core Duo | Yonah | 6 | E | Not fully tested |
| Intel Core 2 (65 nm) | Merom | 6 | F | T5500, Step. 6, rev. B2 |
| Intel Core 2 (45 nm) | Wolfdale | 6 | 17 | E8400, Step. 6 |
| Intel Core i7 | Nehalem | 6 | 1A | i7-920, Step. 5, rev. D0 |
| Intel 2nd gen. Core | Sandy Bridge | 6 | 2A | i5-2500, Step 7 |
| Intel 3rd gen. Core | Ivy Bridge | 6 | 3A | i7-3770K, Step 9 |
| Intel 4th gen. Core | Haswell | 6 | 3C | i7-4770K, step. 3 |
| Intel 5th gen. Core | Broadwell | 6 | 56 | D1540, step 2 |
| Intel 6th gen. Core | Skylake | 6 | 5E | Step. 3 |
| Intel 7th gen. Core | Skylake-X | 6 | 55 | Step. 4 |
| Intel Atom 330 | Diamondville | 6 | 1C | Step. 2 |
| Intel Bay Trail | Silvermont | 6 | 37 | Step. 3 |
| Intel Apollo Lake | Goldmont | 6 | 5C | Step. 9 |
| Intel Gemini Lake | Goldmont Plus | 6 | 7A | Step. 1 |
| Intel Xeon Phi | Knights Landing | 6 | 57 | Step. 1 |
| VIA Nano L2200 | | 6 | F | Step. 2 |
| VIA Nano L3050 | Isaiah | 6 | F | Step. 8 (prerelease sample) |

AMD K7**List of instruction timings and macro-operation breakdown***Explanation of column headings:*

| | |
|-------------------------------|--|
| Instruction: | Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc. |
| Operands: | i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, xmm = 128 bit xmm register, sr = segment register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc. |
| Ops: | Number of macro-operations issued from instruction decoder to schedulers. Instructions with more than 2 macro-operations use microcode. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latency listed does not include the memory operand and where the operand is listed as register or memory (r/m). |
| Reciprocal throughput: | This is also called issue latency. This value indicates the average number of clock cycles from the execution of an instruction begins to a subsequent independent instruction of the same kind can begin to execute. A value of 1/3 indicates that the execution units can handle 3 instructions per clock cycle in one thread. However, the throughput may be limited by other bottlenecks in the pipeline. |
| Execution unit: | Indicates which execution unit is used for the macro-operations. ALU means any of the three integer ALU's. ALU0_1 means that ALU0 and ALU1 are both used. AGU means any of the three integer address generation units. FADD means floating point adder unit. FMUL means floating point multiplier unit. FMISC means floating point store and miscellaneous unit. FA/M means FADD or FMUL is used. FANY means any of the three floating point units can be used. Two macro-operations can execute simultaneously if they go to different execution units. |

Integer instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution unit | Notes |
|--------------------------|----------|-----|---------|-----------------------|----------------|--|
| Move instructions | | | | | | |
| MOV | r,r | 1 | 1 | 1/3 | ALU | Any addr. mode. Add 1 clk if code segment base ≠ 0 |
| MOV | r,i | 1 | 1 | 1/3 | ALU | |
| MOV | r8,m8 | 1 | 4 | 1/2 | ALU, AGU | |
| MOV | r16,m16 | 1 | 4 | 1/2 | ALU, AGU | |
| MOV | r32,m32 | 1 | 3 | 1/2 | AGU | do. |
| MOV | m8,r8H | 1 | 8 | 1/2 | AGU | do. |
| MOV | m8,r8L | 1 | 2 | 1/2 | AGU | AH, BH, CH, DH |
| MOV | m16/32,r | 1 | 2 | 1/2 | AGU | Any other 8-bit register |
| MOV | m,i | 1 | 2 | 1/2 | AGU | Any addressing mode |
| MOV | r,sr | 1 | 2 | 1 | | |
| MOV | sr,r/m | 6 | 9-13 | 8 | | |
| MOVZX, MOVSX | r,r | 1 | 1 | 1/3 | ALU | |
| MOVZX, MOVSX | r,m | 1 | 4 | 1/2 | ALU, AGU | |

AMD K7

| | | | | | | |
|--------------------------------|-------------|----|----|-----|----------|----------------------------------|
| CMOVcc | r,r | 1 | 1 | 1/3 | ALU | Timing depends on hw |
| CMOVcc | r,m | 1 | | 1/2 | ALU, AGU | |
| XCHG | r,r | 3 | 2 | 1 | ALU | |
| XCHG | r,m | 3 | 16 | 16 | ALU, AGU | |
| XLAT | | 2 | 5 | | ALU, AGU | |
| PUSH | r | 1 | | 1 | ALU, AGU | |
| PUSH | i | 1 | | 1 | ALU, AGU | |
| PUSH | m | 2 | | 1 | ALU, AGU | |
| PUSH | sr | 2 | | 1 | ALU, AGU | |
| PUSHF(D) | | 1 | | 1 | ALU, AGU | |
| PUSHA(D) | | 9 | | 4 | ALU, AGU | Any addr. size Any addr. size |
| POP | r | 2 | | 1 | ALU, AGU | |
| POP | m | 3 | | 1 | ALU, AGU | |
| POP | DS/ES/FS/GS | 6 | | 10 | ALU, AGU | |
| POP | SS | 9 | | 18 | ALU, AGU | |
| POPF(D) | | 2 | | 1 | ALU, AGU | |
| POPA(D) | | 9 | | 4 | ALU, AGU | |
| LEA | r16,[m] | 2 | 3 | 1 | AGU | |
| LEA | r32,[m] | 1 | 2 | 1/3 | AGU | |
| LAHF | | 4 | 3 | 2 | ALU | latency ax=3, dx=4 |
| SAHF | | 2 | 2 | 2 | ALU | |
| SALC | | 1 | 1 | 1 | ALU | |
| LDS, LES, ... | r,m | 10 | | 9 | | |
| BSWAP | r | 1 | 1 | 1/3 | ALU | |
| Arithmetic instructions | | | | | | |
| ADD, SUB | r,r/i | 1 | 1 | 1/3 | ALU | |
| ADD, SUB | r,m | 1 | 1 | 1/2 | ALU, AGU | |
| ADD, SUB | m,r | 1 | 7 | 2.5 | ALU, AGU | |
| ADC, SBB | r,r/i | 1 | 1 | 1/3 | ALU | |
| ADC, SBB | r,m | 1 | 1 | 1/2 | ALU, AGU | |
| ADC, SBB | m,r/i | 1 | 7 | 2.5 | ALU, AGU | |
| CMP | r,r/i | 1 | 1 | 1/3 | ALU | |
| CMP | r,m | 1 | | 1/2 | ALU, AGU | |
| INC, DEC, NEG | r | 1 | 1 | 1/3 | ALU | |
| INC, DEC, NEG | m | 1 | 7 | 3 | ALU, AGU | |
| AAA, AAS | | 9 | 5 | 5 | ALU | latency ax=3, dx=4 |
| DAA | | 12 | 6 | 6 | ALU | |
| DAS | | 16 | 7 | 7 | ALU | |
| AAD | | 4 | 5 | | ALU0 | |
| AAM | | 31 | 13 | | ALU | |
| MUL, IMUL | r8/m8 | 3 | 3 | 2 | ALU0 | |
| MUL, IMUL | r16/m16 | 3 | 3 | 2 | ALU0_1 | |
| MUL, IMUL | r32/m32 | 3 | 4 | 3 | ALU0_1 | |
| IMUL | r16,r16/m16 | 2 | 3 | 2 | ALU0 | |
| IMUL | r32,r32/m32 | 2 | 4 | 2.5 | ALU0 | |
| IMUL | r16,(r16),i | 2 | 4 | 1 | ALU0 | |
| IMUL | r32,(r32),i | 2 | 5 | 2 | ALU0 | |
| IMUL | r16,m16,i | 3 | | 2 | ALU0 | |
| IMUL | r32,m32,i | 3 | | 2 | ALU0 | |
| DIV | r8/m8 | 32 | 24 | 23 | ALU | |

AMD K7

| | | | | | |
|---------------------------|----------|----|----|-----|----------|
| DIV | r16/m16 | 47 | 24 | 23 | ALU |
| DIV | r32/m32 | 79 | 40 | 40 | ALU |
| IDIV | r8 | 41 | 17 | 17 | ALU |
| IDIV | r16 | 56 | 25 | 25 | ALU |
| IDIV | r32 | 88 | 41 | 41 | ALU |
| IDIV | m8 | 42 | 17 | 17 | ALU |
| IDIV | m16 | 57 | 25 | 25 | ALU |
| IDIV | m32 | 89 | 41 | 41 | ALU |
| CBW, CWDE | | 1 | 1 | 1/3 | ALU |
| CWD, CDQ | | 1 | 1 | 1/3 | ALU |
| Logic instructions | | | | | |
| AND, OR, XOR | r,r | 1 | 1 | 1/3 | ALU |
| AND, OR, XOR | r,m | 1 | 1 | 1/2 | ALU, AGU |
| AND, OR, XOR | m,r | 1 | 7 | 2.5 | ALU, AGU |
| TEST | r,r | 1 | 1 | 1/3 | ALU |
| TEST | r,m | 1 | 1 | 1/2 | ALU, AGU |
| NOT | r | 1 | 1 | 1/3 | ALU |
| NOT | m | 1 | 7 | 2.5 | ALU, AGU |
| SHL, SHR, SAR | r,i/CL | 1 | 1 | 1/3 | ALU |
| ROL, ROR | r,i/CL | 1 | 1 | 1/3 | ALU |
| RCL, RCR | r,1 | 1 | 1 | 1/3 | ALU |
| RCL | r,i | 9 | 4 | 4 | ALU |
| RCR | r,i | 7 | 3 | 3 | ALU |
| RCL | r,CL | 9 | 3 | 3 | ALU |
| RCR | r,CL | 7 | 3 | 3 | ALU |
| SHL,SHR,SAR,ROL,ROR | m,i /CL | 1 | 7 | 3 | ALU, AGU |
| RCL, RCR | m,1 | 1 | 7 | 4 | ALU, AGU |
| RCL | m,i | 10 | 5 | 4 | ALU, AGU |
| RCR | m,i | 9 | 8 | 4 | ALU, AGU |
| RCL | m,CL | 9 | 6 | 4 | ALU, AGU |
| RCR | m,CL | 8 | 7 | 3 | ALU, AGU |
| SHLD, SHRD | r,r,i | 6 | 4 | 2 | ALU |
| SHLD, SHRD | r,r,cl | 7 | 4 | 3 | ALU |
| SHLD, SHRD | m,r,i/CL | 8 | 7 | 3 | ALU, AGU |
| BT | r,r/i | 1 | 1 | 1/3 | ALU |
| BT | m,i | 1 | | 1/2 | ALU, AGU |
| BT | m,r | 5 | | 2 | ALU, AGU |
| BTC, BTR, BTS | r,r/i | 2 | 2 | 1 | ALU |
| BTC | m,i | 5 | 7 | 2 | ALU, AGU |
| BTR, BTS | m,i | 4 | 7 | 2 | ALU, AGU |
| BTC, BTR, BTS | m,r | 8 | 6 | 3 | ALU, AGU |
| BSF | r,r | 19 | 7 | 7 | ALU |
| BSR | r,r | 23 | 9 | 9 | ALU |
| BSF | r,m | 20 | 8 | 8 | ALU, AGU |
| BSR | r,m | 23 | 10 | 10 | ALU, AGU |
| SETcc | r | 1 | 1 | 1/3 | ALU |
| SETcc | m | 1 | | 1/2 | ALU, AGU |
| CLC, STC | | 1 | | 1/3 | ALU |
| CMC | | 1 | 1 | 1/3 | ALU |
| CLD | | 2 | | 1 | ALU |
| STD | | 3 | | 2 | ALU |

AMD K7

| Control transfer instructions | | | | | | |
|-------------------------------|------------|-------|-------|---------|----------|------------------------|
| JMP | short/near | 1 | | 2 | ALU | low values = real mode |
| JMP | far | 16-20 | 23-32 | | | |
| JMP | r | 1 | | 2 | ALU | |
| JMP | m(near) | 1 | | 2 | ALU, AGU | low values = real mode |
| JMP | m(far) | 17-21 | 25-33 | | | |
| Jcc | short/near | 1 | | 1/3 - 2 | ALU | |
| J(E)CXZ | short | 2 | | 1/3 - 2 | ALU | rcp. t.= 2 if jump |
| LOOP | short | 7 | 3-4 | 3-4 | ALU | rcp. t.= 2 if jump |
| CALL | near | 3 | 2 | 2 | ALU | |
| CALL | far | 16-22 | 23-32 | | | low values = real mode |
| CALL | r | 4 | 3 | 3 | ALU | |
| CALL | m(near) | 5 | 3 | 3 | ALU, AGU | |
| CALL | m(far) | 16-22 | 24-33 | | | low values = real mode |
| RETN | | 2 | 3 | 3 | ALU | |
| RETN | i | 2 | 3 | 3 | ALU | |
| RETF | | 15-23 | 24-35 | | | low values = real mode |
| RETF | i | 15-24 | 24-35 | | | low values = real mode |
| IRET | | 32 | 81 | | | real mode |
| INT | i | 33 | 42 | | | real mode |
| BOUND | m | 6 | | 2 | | values are for no jump |
| INTO | | 2 | | 2 | | values are for no jump |
| String instructions | | | | | | |
| LODS | | 4 | 2 | 2 | | |
| REP LODS | | 5 | 2 | 2 | | values per count |
| STOS | | 4 | 2 | 2 | | |
| REP STOS | | 3 | 1 | 1 | | values per count |
| MOVS | | 7 | 3 | 3 | | |
| REP MOVS | | 4 | 1-4 | 1-4 | | values per count |
| SCAS | | 5 | 2 | 2 | | |
| REP SCAS | | 5 | 2 | 2 | | values per count |
| CMPS | | 7 | 6 | 6 | | |
| REP CMPS | | 6 | 3-4 | 3-4 | | values per count |
| Other | | | | | | |
| NOP (90) | | 1 | 0 | 1/3 | ALU | |
| Long NOP (0F 1F) | | 1 | 0 | 1/3 | ALU | |
| ENTER | | i,0 | 12 | 12 | 12 | |
| LEAVE | | 3 | | 3 | | 3 ops, 5 clk if 16 bit |
| CLI | | 8-9 | | 5 | | |
| STI | | 16-17 | | 27 | | |
| CPUID | | 19-28 | 44-74 | | | |
| RDTSC | | 5 | | 11 | | |
| RDPNC | | 9 | | 11 | | |

Floating point x87 instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution unit | Notes | |
|-------------------------|----------|-----|---------|-----------------------|----------------|---|-----|
| Move instructions | | | | | | | |
| FLD | r | 1 | 2 | 1/2 | FA/M | Low latency immediately after FCOMI | |
| FLD | m32/64 | 1 | 4 | 1/2 | FANY | | |
| FLD | m80 | 7 | 16 | 4 | | | |
| FBLD | m80 | 30 | 41 | 39 | | | |
| FST(P) | r | 1 | 2 | 1/2 | FA/M | | |
| FST(P) | m32/64 | 1 | 3 | 1 | FMISC | | |
| FSTP | m80 | 10 | 7 | 5 | | | |
| FBSTP | m80 | 260 | | 188 | | | |
| FXCH | r | 1 | 0 | 0.4 | | | |
| FILD | m | 1 | 9 | 1 | FMISC | | |
| FIST(P) | m | 1 | 7 | 1 | FMISC, FA/M | | |
| FLDZ, FLD1 | | 1 | | 1 | FMISC | | |
| FCMOVcc | st0,r | 9 | 6 | 5 | FMISC, FA/M | | |
| FFREE | r | 1 | | 1/3 | FANY | | |
| FINCSTP, FDECSTP | | 1 | 0 | 1/3 | FANY | | |
| FNSTSW | AX | 2 | 6-12 | 12 | FMISC, ALU | Low latency immediately after FCOM FTST | |
| FSTSW | AX | 3 | 6-12 | 12 | FMISC, ALU | | do. |
| FNSTSW | m16 | 2 | | 8 | FMISC, ALU | | do. |
| FNSTCW | m16 | 3 | | 1 | FMISC, ALU | | |
| FLDCW | m16 | 14 | | 42 | FMISC, ALU | faster if unchanged | |
| Arithmetic instructions | | | | | | | |
| FADD(P),FSUB(R)(P) | r/m | 1 | 4 | 1 | FADD | Low values are for round divisors | |
| FIADD,FISUB(R) | m | 2 | 4 | 1-2 | FADD,FMISC | | |
| FMUL(P) | r/m | 1 | 4 | 1 | FMUL | | |
| FIMUL | m | 2 | 4 | 2 | FMUL,FMISC | | |
| FDIV(R)(P) | r/m | 1 | 11-25 | 8-22 | FMUL | | |
| FIDIV(R) | m | 2 | 12-26 | 9-23 | FMUL,FMISC | | |
| FABS, FCHS | | 1 | 2 | 1 | FMUL | | |
| FCOM(P), FUCOM(P) | r/m | 1 | 2 | 1 | FADD | | |
| FCOMPP, FUCOMPP | | 1 | 2 | 1 | FADD | | |
| FCOMI(P) | r | 1 | 3 | 1 | FADD | | |
| FICOM(P) | m | 2 | | 1 | FADD, FMISC | | |
| FTST | | 1 | 2 | 1 | FADD | | |
| FXAM | | 2 | | 2 | FMISC, ALU | | |
| FRNDINT | | 5 | 10 | 3 | | | |
| FPREM | | 1 | 7-10 | 8 | FMUL | | |
| FPREM1 | | 1 | 8-11 | 8 | FMUL | | |
| Math | | | | | | | |
| FSQRT | | 1 | 35 | 12 | FMUL | | |
| FSIN | | 44 | 90-100 | | | | |

AMD K7

| | | | | | |
|--------------|----|---------|-----|-------|--|
| FCOS | 51 | 90-100 | | | |
| FSINCOS | 76 | 100-150 | | | |
| FPTAN | 46 | 100-200 | | | |
| FPATAN | 72 | 160-170 | | | |
| FSCALE | 5 | 8 | | | |
| FXTRACT | 7 | 11 | | | |
| F2XM1 | 8 | 27 | | | |
| FYL2X | 49 | 126 | | | |
| FYL2XP1 | 63 | 147 | | | |
| Other | | | | | |
| FNOP | 1 | 0 | 1/3 | FANY | |
| (F)WAIT | 1 | 0 | 1/3 | ALU | |
| FNCLEX | 7 | | 24 | FMISC | |
| FNINIT | 25 | | 92 | FMISC | |
| FNSAVE | 76 | | 147 | | |
| FRSTOR | 65 | | 120 | | |
| FXSAVE | 44 | | 59 | | |
| FXRSTOR | 85 | | 87 | | |

Integer MMX instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution unit | Notes |
|--------------------------------|----------|-----|---------|-----------------------|----------------|-------|
| Move instructions | | | | | | |
| MOVD | r32, mm | 2 | 7 | 2 | FMISC, ALU | |
| MOVD | mm, r32 | 2 | 9 | 2 | FANY, ALU | |
| MOVD | mm,m32 | 1 | | 1/2 | FANY | |
| MOVD | m32, r | 1 | | 1 | FMISC | |
| MOVQ | mm,mm | 1 | 2 | 1/2 | FA/M | |
| MOVQ | mm,m64 | 1 | | 1/2 | FANY | |
| MOVQ | m64,mm | 1 | | 1 | FMISC | |
| MOVNTQ | m,mm | 1 | | 2 | FMISC | |
| PACKSSWB/DW | | | | | | |
| PACKUSWB | mm,r/m | 1 | 2 | 2 | FA/M | |
| PUNPCKH/LBW/WD | mm,r/m | 1 | 2 | 2 | FA/M | |
| PSHUFW | mm,mm,i | 1 | 2 | 1/2 | FA/M | |
| MASKMOVQ | mm,mm | 32 | | 24 | | |
| PMOVBMSKB | r32,mm | 3 | | 3 | FADD | |
| PEXTRW | r32,mm,i | 2 | 5 | 2 | FMISC, ALU | |
| PINSRW | mm,r32,i | 2 | 12 | 2 | FA/M | |
| Arithmetic instructions | | | | | | |
| PADDB/W/D PADDSB/W | | | | | | |
| PADDUSB/W PSUBB/W/ | | | | | | |
| D PSUBSB/W PSUBUSB/ | | | | | | |
| W | mm,r/m | 1 | 2 | 1/2 | FA/M | |
| PCMPEQ/GT B/W/D | mm,r/m | 1 | 2 | 1/2 | FA/M | |
| PMULLW PMULHW | | | | | | |
| PMULHUW | mm,r/m | 1 | 3 | 1 | FMUL | |
| PMADDWD | mm,r/m | 1 | 3 | 1 | FMUL | |
| PAVGB/W | mm,r/m | 1 | 2 | 1/2 | FA/M | |
| PMIN/MAX SW/UB | mm,r/m | 1 | 2 | 1/2 | FA/M | |
| PSADBW | mm,r/m | 1 | 3 | 1 | FADD | |

AMD K7

| | | | | | | |
|--------------------------|-----------|---|---|-----|------|--|
| Logic | | | | | | |
| PAND PANDN POR PXOR | mm,r/m | 1 | 2 | 1/2 | FA/M | |
| PSLL/RLW/D/Q PSRAW/ D | mm,i/mm/m | 1 | 2 | 1/2 | FA/M | |
| Other | | | | | | |
| EMMS | | 1 | | 1/3 | FANY | |

Floating point XMM instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution unit | Notes |
|--------------------------|----------|-----|---------|--------------------------|-------------------|--|
| Move instructions | | | | | | |
| MOVAPS | r,r | 2 | 2 | 1 | FA/M | |
| MOVAPS | r,m | 2 | | 2 | FMISC | |
| MOVAPS | m,r | 2 | | 2 | FMISC | |
| MOVUPS | r,r | 2 | 2 | 1 | FA/M | |
| MOVUPS | r,m | 5 | | 2 | | |
| MOVUPS | m,r | 5 | | 2 | | |
| MOVSS | r,r | 1 | 2 | 1 | FA/M | |
| MOVSS | r,m | 2 | 4 | 1 | FANY FMISC | |
| MOVSS | m,r | 1 | 3 | 1 | FMISC | |
| MOVHPS, MOVLHPS | r,r | 1 | 2 | 1/2 | FA/M | |
| MOVHPS, MOVLPS | r,m | 1 | | 1/2 | FMISC | |
| MOVHPS, MOVLPS | m,r | 1 | | 1 | FMISC | |
| MOVNTPS | m,r | 2 | | 4 | FMISC | |
| MOVMSKPS | r32,r | 3 | | 2 | FADD | |
| SHUFPS | r,r/m,i | 3 | 3 | 3 | FMUL | |
| UNPCK H/L PS | r,r/m | 2 | 3 | 3 | FMUL | |
| Conversion | | | | | | |
| CVTPI2PS | xmm,mm | 1 | 4 | | FMISC | |
| CVT(T)PS2PI | mm,xmm | 1 | 6 | | FMISC | |
| CVTSS2SI | xmm,r32 | 4 | | 10 | FMISC | |
| CVT(T)SS2SI | r32,xmm | 2 | | 3 | FMISC | |
| Arithmetic | | | | | | |
| ADDSS SUBSS | r,r/m | 1 | 4 | 1 | FADD | |
| ADDPS SUBPS | r,r/m | 2 | 4 | 2 | FADD | |
| MULSS | r,r/m | 1 | 4 | 1 | FMUL | |
| MULPS | r,r/m | 2 | 4 | 2 | FMUL | |
| DIVSS | r,r/m | 1 | 11-16 | 8-13 | FMUL | Low values are for round divi- sors, e.g. powers of 2. do. |
| DIVPS | r,r/m | 2 | 18-30 | 18-30 | FMUL | |
| RCPSS | r,r/m | 1 | 3 | 1 | FMUL | |
| RCPPS | r,r/m | 2 | 3 | 2 | FMUL | |
| MAXSS MINSS | r,r/m | 1 | 2 | 1 | FADD | |
| MAXPS MINPS | r,r/m | 2 | 2 | 2 | FADD | |
| CMPccSS | r,r/m | 1 | 2 | 1 | FADD | |

AMD K7

| | | | | | | |
|------------------------------------|-------|---|----|----|------|--|
| CMPccPS | r,r/m | 2 | 2 | 2 | FADD | |
| COMISS UCOMISS | r,r/m | 1 | 2 | 1 | FADD | |
| Logic | | | | | | |
| ANDPS/D ANDNPS/D ORPS/D XORPS/D | r,r/m | 2 | 2 | 2 | FMUL | |
| Math | | | | | | |
| SQRTSS | r,r/m | 1 | 19 | 16 | FMUL | |
| SQRTPS | r,r/m | 2 | 36 | 36 | FMUL | |
| RSQRTSS | r,r/m | 1 | 3 | 1 | FMUL | |
| RSQRTPS | r,r/m | 2 | 3 | 2 | FMUL | |
| Other | | | | | | |
| LDMXCSR | m | 8 | | 9 | | |
| STMXCSR | m | 3 | | 10 | | |

3DNow instructions (obsolete)

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution unit | Notes |
|--------------------------------------|----------|-----|---------|-----------------------|----------------|---------|
| Move and convert instructions | | | | | | |
| PREFETCH(W) | m | 1 | | 1/2 | AGU | |
| PF2ID | mm,mm | 1 | 5 | 1 | FMISC | |
| PI2FD | mm,mm | 1 | 5 | 1 | FMISC | |
| PF2IW | mm,mm | 1 | 5 | 1 | FMISC | 3DNow E |
| PI2FW | mm,mm | 1 | 5 | 1 | FMISC | 3DNow E |
| PSWAPD | mm,mm | 1 | 2 | 1/2 | FA/M | 3DNow E |
| Integer instructions | | | | | | |
| PAVGUSB | mm,mm | 1 | 2 | 1/2 | FA/M | |
| PMULHRW | mm,mm | 1 | 3 | 1 | FMUL | |
| Floating point instructions | | | | | | |
| PFADD/SUB/SUBR | mm,mm | 1 | 4 | 1 | FADD | |
| PFCMPEQ/GE/GT | mm,mm | 1 | 2 | 1 | FADD | |
| PFMAX/MIN | mm,mm | 1 | 2 | 1 | FADD | |
| PFMUL | mm,mm | 1 | 4 | 1 | FMUL | |
| PFACC | mm,mm | 1 | 4 | 1 | FADD | |
| PFPNACC, PFPNACC | mm,mm | 1 | 4 | 1 | FADD | 3DNow E |
| PFRCP | mm,mm | 1 | 3 | 1 | FMUL | |
| PFRCPIT1/2 | mm,mm | 1 | 4 | 1 | FMUL | |
| PFRSQRT | mm,mm | 1 | 3 | 1 | FMUL | |
| PFRSQIT1 | mm,mm | 1 | 4 | 1 | FMUL | |
| Other | | | | | | |
| FEMMS | mm,mm | 1 | | 1/3 | FANY | |

AMD K8

List of instruction timings and macro-operation breakdown

Explanation of column headings:

| | |
|-------------------------------|--|
| Instruction: | Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc. |
| Operands: | i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, xmm = 128 bit xmm register, sr = segment register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc. |
| Ops: | Number of macro-operations issued from instruction decoder to schedulers. Instructions with more than 2 macro-operations use microcode. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latency listed does not include the memory operand where the operand is listed as register or memory (r/m). |
| Reciprocal throughput: | This is also called issue latency. This value indicates the average number of clock cycles from the execution of an instruction begins to a subsequent independent instruction of the same kind can begin to execute. A value of 1/3 indicates that the execution units can handle 3 instructions per clock cycle in one thread. However, the throughput may be limited by other bottlenecks in the pipeline. |
| Execution unit: | Indicates which execution unit is used for the macro-operations. ALU means any of the three integer ALU's. ALU0_1 means that ALU0 and ALU1 are both used. AGU means any of the three integer address generation units. FADD means floating point adder unit. FMUL means floating point multiplier unit. FMISC means floating point store and miscellaneous unit. FA/M means FADD or FMUL is used. FANY means any of the three floating point units can be used. Two macro-operations can execute simultaneously if they go to different execution units. |

Integer instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution unit | Notes |
|--------------------------|-------------|-----|---------|-----------------------|----------------|--------------------------------------|
| Move instructions | | | | | | |
| MOV | r,r | 1 | 1 | 1/3 | ALU | |
| MOV | r,i | 1 | 1 | 1/3 | ALU | |
| MOV | r8,m8 | 1 | 4 | 1/2 | ALU, AGU | Any addressing mode. |
| MOV | r16,m16 | 1 | 4 | 1/2 | ALU, AGU | Add 1 clock if code segment base ≠ 0 |
| MOV | r32,m32 | 1 | 3 | 1/2 | AGU | |
| MOV | r64,m64 | 1 | 3 | 1/2 | AGU | |
| MOV | m8,r8H | 1 | 8 | 1/2 | AGU | AH, BH, CH, DH |
| MOV | m8,r8L | 1 | 3 | 1/2 | AGU | Any other 8-bit register |
| MOV | m16/32/64,r | 1 | 3 | 1/2 | AGU | Any addressing mode |
| MOV | m,i | 1 | 3 | 1/2 | AGU | |
| MOV | m64,i32 | 1 | 3 | 1/2 | AGU | |
| MOV | r,sr | 1 | 2 | 1/2-1 | | |
| MOV | sr,r/m | 6 | 9-13 | 8 | | |
| MOVNTI | m,r | 1 | | 2-3 | AGU | |
| MOVZX, MOVZX | r,r | 1 | 1 | 1/3 | ALU | |
| MOVZX, MOVZX | r,m | 1 | 4 | 1/2 | ALU, AGU | |
| MOVSLD | r64,r32 | 1 | 1 | 1/3 | ALU | |

K8

| | | | | | | |
|--------------------------------|-------------|-----|-----|-----|----------|----------------------|
| MOVSXD | r64,m32 | 1 | | 1/2 | ALU, AGU | Timing depends on hw |
| CMOVcc | r,r | 1 | 1 | 1/3 | ALU | |
| CMOVcc | r,m | 1 | | 1/2 | ALU, AGU | |
| XCHG | r,r | 3 | 2 | 1 | ALU | |
| XCHG | r,m | 3 | 16 | 16 | ALU, AGU | |
| XLAT | | 2 | 5 | | ALU, AGU | |
| PUSH | r | 1 | 1 | 1 | ALU, AGU | |
| PUSH | i | 1 | 1 | 1 | ALU, AGU | |
| PUSH | m | 2 | 1 | 1 | ALU, AGU | |
| PUSH | sr | 2 | 1 | 1 | ALU, AGU | |
| PUSHF(D/Q) | | 5 | 2 | 2 | ALU, AGU | |
| PUSHA(D) | | 9 | 4 | 4 | ALU, AGU | |
| POP | r | 2 | 1 | 1 | ALU, AGU | |
| POP | m | 3 | 1 | 1 | ALU, AGU | |
| POP | DS/ES/FS/GS | 4-6 | 8 | 8 | ALU, AGU | |
| POP | SS | 7-9 | 28 | 28 | ALU, AGU | |
| POPF(D/Q) | | 25 | 10 | 10 | ALU, AGU | |
| POPA(D) | | 9 | 4 | 4 | ALU, AGU | |
| LEA | r16,[m] | 2 | 3 | 1 | AGU | Any address size |
| LEA | r32,[m] | 1 | 2 | 1/3 | AGU | Any address size |
| LEA | r64,[m] | 1 | 2 | 1/3 | AGU | Any address size |
| LAHF | | 4 | 3 | 2 | ALU | Any address size |
| SAHF | | 1 | 1 | 1/3 | ALU | |
| SALC | | 1 | 1 | 1/3 | ALU | |
| LDS, LES, ... | r,m | 10 | | 9 | | |
| BSWAP | r | 1 | 1 | 1/3 | ALU | |
| PREFETCHNTA | m | 1 | | 1/2 | AGU | |
| PREFETCHT0/1/2 | m | 1 | | 1/2 | AGU | |
| SFENCE | | 6 | | 8 | | |
| LFENCE | | 1 | | 5 | | |
| MFENCE | | 7 | | 16 | | |
| IN | r,i/DX | 270 | | | | |
| OUT | i/DX,r | 300 | | | | |
| Arithmetic instructions | | | | | | |
| ADD, SUB | r,r/i | 1 | 1 | 1/3 | ALU | latency ax=3, dx=4 |
| ADD, SUB | r,m | 1 | 1 | 1/2 | ALU, AGU | |
| ADD, SUB | m,r | 1 | 7 | 2.5 | ALU, AGU | |
| ADC, SBB | r,r/i | 1 | 1 | 1/3 | ALU | |
| ADC, SBB | r,m | 1 | 1 | 1/2 | ALU, AGU | |
| ADC, SBB | m,r/i | 1 | 7 | 2.5 | ALU, AGU | |
| CMP | r,r/i | 1 | 1 | 1/3 | ALU | |
| CMP | r,m | 1 | | 1/2 | ALU, AGU | |
| INC, DEC, NEG | r | 1 | 1 | 1/3 | ALU | |
| INC, DEC, NEG | m | 1 | 7 | 3 | ALU, AGU | |
| AAA, AAS | | 9 | 5 | 5 | ALU | |
| DAA | | 12 | 6 | 6 | ALU | |
| DAS | | 16 | 7 | 7 | ALU | |
| AAD | | 4 | 5 | | ALU0 | |
| AAM | | 31 | 13 | | ALU | |
| MUL, IMUL | r8/m8 | 1 | 3 | 1 | ALU0 | |
| MUL, IMUL | r16/m16 | 3 | 3-4 | 2 | ALU0_1 | |

K8

| | | | | | | |
|---------------------------|-------------|-----|-----|-----|----------|----------------------|
| MUL, IMUL | r32/m32 | 2 | 3 | 1 | ALU0_1 | latency rax=4, rdx=5 |
| MUL, IMUL | r64/m64 | 2 | 4-5 | 2 | ALU0_1 | |
| IMUL | r16,r16/m16 | 1 | 3 | 1 | ALU0 | |
| IMUL | r32,r32/m32 | 1 | 3 | 1 | ALU0 | |
| IMUL | r64,r64/m64 | 1 | 4 | 2 | ALU0_1 | |
| IMUL | r16,(r16),i | 2 | 4 | 1 | ALU0 | |
| IMUL | r32,(r32),i | 1 | 3 | 1 | ALU0 | |
| IMUL | r64,(r64),i | 1 | 4 | 2 | ALU0 | |
| IMUL | r16,m16,i | 3 | | 2 | ALU0 | |
| IMUL | r32,m32,i | 3 | | 2 | ALU0 | |
| IMUL | r64,m64,i | 3 | | 2 | ALU0_1 | |
| DIV | r8/m8 | 31 | 15 | 15 | ALU | |
| DIV | r16/m16 | 46 | 23 | 23 | ALU | |
| DIV | r32/m32 | 78 | 39 | 39 | ALU | |
| DIV | r64/m64 | 143 | 71 | 71 | ALU | |
| IDIV | r8 | 40 | 17 | 17 | ALU | |
| IDIV | r16 | 55 | 25 | 25 | ALU | |
| IDIV | r32 | 87 | 41 | 41 | ALU | |
| IDIV | r64 | 152 | 73 | 73 | ALU | |
| IDIV | m8 | 41 | 17 | 17 | ALU | |
| IDIV | m16 | 56 | 25 | 25 | ALU | |
| IDIV | m32 | 88 | 41 | 41 | ALU | |
| IDIV | m64 | 153 | 73 | 73 | ALU | |
| CBW, CWDE, CDQE | | 1 | 1 | 1/3 | ALU | |
| CWD, CDQ, CQO | | 1 | 1 | 1/3 | ALU | |
| Logic instructions | | | | | | |
| AND, OR, XOR | r,r | 1 | 1 | 1/3 | ALU | |
| AND, OR, XOR | r,m | 1 | 1 | 1/2 | ALU, AGU | |
| AND, OR, XOR | m,r | 1 | 7 | 2.5 | ALU, AGU | |
| TEST | r,r | 1 | 1 | 1/3 | ALU | |
| TEST | r,m | 1 | 1 | 1/2 | ALU, AGU | |
| NOT | r | 1 | 1 | 1/3 | ALU | |
| NOT | m | 1 | 7 | 2.5 | ALU, AGU | |
| SHL, SHR, SAR | r,i/CL | 1 | 1 | 1/3 | ALU | |
| ROL, ROR | r,i/CL | 1 | 1 | 1/3 | ALU | |
| RCL, RCR | r,1 | 1 | 1 | 1/3 | ALU | |
| RCL | r,i | 9 | 3 | 3 | ALU | |
| RCR | r,i | 7 | 3 | 3 | ALU | |
| RCL | r,CL | 9 | 4 | 4 | ALU | |
| RCR | r,CL | 7 | 3 | 3 | ALU | |
| SHL,SHR,SAR,ROL,ROR | m,i /CL | 1 | 7 | 3 | ALU, AGU | |
| RCL, RCR | m,1 | 1 | 7 | 4 | ALU, AGU | |
| RCL | m,i | 10 | 9 | 4 | ALU, AGU | |
| RCR | m,i | 9 | 8 | 4 | ALU, AGU | |
| RCL | m,CL | 9 | 7 | 4 | ALU, AGU | |
| RCR | m,CL | 8 | 8 | 3 | ALU, AGU | |
| SHLD, SHRD | r,r,i | 6 | 3 | 3 | ALU | |
| SHLD, SHRD | r,r,cl | 7 | 3 | 3 | ALU | |
| SHLD, SHRD | m,r,i/CL | 8 | 6 | 3 | ALU, AGU | |
| BT | r,r/i | 1 | 1 | 1/3 | ALU | |
| BT | m,i | 1 | | 1/2 | ALU, AGU | |

K8

| | | | | | | |
|--------------------------------------|------------|-------|-------|---------|----------|-------------------------|
| BT | m,r | 5 | | 2 | ALU, AGU | |
| BTC, BTR, BTS | r,r/i | 2 | 2 | 1 | ALU | |
| BTC | m,i | 5 | 7 | 2 | ALU, AGU | |
| BTR, BTS | m,i | 4 | 7 | 2 | ALU, AGU | |
| BTC | m,r | 8 | 5 | 5 | ALU, AGU | |
| BTR, BTS | m,r | 8 | 8 | 3 | ALU, AGU | |
| BSF | r16/32,r | 21 | 8 | 8 | ALU | |
| BSF | r64,r | 22 | 9 | 9 | ALU | |
| BSR | r,r | 28 | 10 | 10 | ALU | |
| BSF | r16,m | 20 | 8 | 8 | ALU, AGU | |
| BSF | r32,m | 22 | 9 | 9 | ALU, AGU | |
| BSF | r64,m | 25 | 10 | 10 | ALU, AGU | |
| BSR | r,m | 28 | 10 | 10 | ALU, AGU | |
| SETcc | r | 1 | 1 | 1/3 | ALU | |
| SETcc | m | 1 | | 1/2 | ALU, AGU | |
| CLC, STC | | 1 | | 1/3 | ALU | |
| CMC | | 1 | 1 | 1/3 | ALU | |
| CLD | | 1 | | 1/3 | ALU | |
| STD | | 2 | | 1/3 | ALU | |
| Control transfer instructions | | | | | | |
| JMP | short/near | 1 | | 2 | ALU | |
| JMP | far | 16-20 | 23-32 | | | low values = real mode |
| JMP | r | 1 | | 2 | ALU | |
| JMP | m(near) | 1 | | 2 | ALU, AGU | |
| JMP | m(far) | 17-21 | 25-33 | | | low values = real mode |
| Jcc | short/near | 1 | | 1/3 - 2 | ALU | recip. thrp.= 2 if jump |
| J(E/R)CXZ | short | 2 | | 1/3 - 2 | ALU | recip. thrp.= 2 if jump |
| LOOP | short | 7 | 3-4 | 3-4 | ALU | |
| CALL | near | 3 | 2 | 2 | ALU | |
| CALL | far | 16-22 | 23-32 | | | low values = real mode |
| CALL | r | 4 | 3 | 3 | ALU | |
| CALL | m(near) | 5 | 3 | 3 | ALU, AGU | |
| CALL | m(far) | 16-22 | 24-33 | | | low values = real mode |
| RETN | | 2 | 3 | 3 | ALU | |
| RETN | i | 2 | 3 | 3 | ALU | |
| RETF | | 15-23 | 24-35 | | | low values = real mode |
| RETF | i | 15-24 | 24-35 | | | low values = real mode |
| IRET | | 32 | 81 | | | real mode |
| INT | i | 33 | 42 | | | real mode |
| BOUND | m | 6 | | 2 | | values are for no jump |
| INTO | | 2 | | 2 | | values are for no jump |
| String instructions | | | | | | |
| LODS | | 4 | 2 | 2 | | |
| REP LODS | | 5 | 2 | 2 | | values are per count |

K8

| | | | | | |
|------------------|---------|---------|---------|-----|------------------------|
| STOS | 4 | 2 | 2 | | |
| REP STOS | 1.5 - 2 | 0.5 - 1 | 0.5 - 1 | | values are per count |
| MOVS | 7 | 3 | 3 | | |
| REP MOVS | 3 | 1-2 | 1-2 | | values are per count |
| SCAS | 5 | 2 | 2 | | |
| REP SCAS | 5 | 2 | 2 | | values are per count |
| CMPS | 2 | 3 | 3 | | |
| REP CMPS | 6 | 2 | 2 | | values are per count |
| Other | | | | | |
| NOP (90) | 1 | 0 | 1/3 | ALU | |
| Long NOP (0F 1F) | 1 | 0 | 1/3 | ALU | |
| ENTER | i,0 | 12 | 12 | 12 | |
| LEAVE | 2 | | 3 | | 3 ops, 5 clk if 16 bit |
| CLI | 8-9 | | 5 | | |
| STI | 16-17 | | 27 | | |
| CPUID | 22-50 | 47-164 | | | |
| RDTSC | 6 | 10 | 7 | | |
| RDPNC | 9 | 12 | 7 | | |

Floating point x87 instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution unit | Notes |
|--------------------------------|----------|-----|---------|-----------------------|----------------|-------------------------------------|
| Move instructions | | | | | | |
| FLD | r | 1 | 2 | 1/2 | FA/M | |
| FLD | m32/64 | 1 | 4 | 1/2 | FANY | |
| FLD | m80 | 7 | 16 | 4 | | |
| FBLD | m80 | 30 | 41 | 39 | | |
| FST(P) | r | 1 | 2 | 1/2 | FA/M | |
| FST(P) | m32/64 | 1 | 3 | 1 | FMISC | |
| FSTP | m80 | 10 | 7 | 5 | | |
| FBSTP | m80 | 260 | 173 | 160 | | |
| FXCH | r | 1 | 0 | 0.4 | | |
| FILD | m | 1 | 9 | 1 | FMISC | |
| FIST(P) | m | 1 | 7 | 1 | FMISC, FA/M | |
| FLDZ, FLD1 | | 1 | | 1 | FMISC | |
| FCMOVcc | st0,r | 9 | 4-15 | 4 | FMISC, FA/M | Low latency immediately after FCOMI |
| FFREE | r | 1 | | 2 | FANY | |
| FINCSTP, FDECSTP | | 1 | 0 | 1/3 | FANY | |
| FNSTSW | AX | 2 | 6-12 | 12 | FMISC, ALU | Low latency immediately after FCOM |
| FSTSW | AX | 3 | 6-12 | 12 | FMISC, ALU | do. |
| FNSTSW | m16 | 2 | | 8 | FMISC, ALU | do. |
| FNSTCW | m16 | 3 | | 1 | FMISC, ALU | |
| FLDCW | m16 | 18 | | 50 | FMISC, ALU | faster if unchanged |
| Arithmetic instructions | | | | | | |
| FADD(P),FSUB(R)(P) | r/m | 1 | 4 | 1 | FADD | |
| FIADD,FISUB(R) | m | 2 | 4 | 1-2 | FADD,FMISC | |
| FMUL(P) | r/m | 1 | 4 | 1 | FMUL | |

K8

| | | | | | | |
|-------------------|-----|-----|---------|------|-------------|---------------------------------------|
| FIMUL | m | 2 | 4 | 2 | FMUL, FMISC | Low values are for round divisors do. |
| FDIV(R)(P) | r/m | 1 | 11-25 | 8-22 | FMUL | |
| FIDIV(R) | m | 2 | 12-26 | 9-23 | FMUL, FMISC | |
| FABS, FCHS | | 1 | 2 | 1 | FMUL | |
| FCOM(P), FUCOM(P) | r/m | 1 | 2 | 1 | FADD | |
| FCOMPP, FUCOMPP | | 1 | 2 | 1 | FADD | |
| FCOMI(P) | r | 1 | 3 | 1 | FADD | |
| FICOM(P) | m | 2 | | 1 | FADD, FMISC | |
| FTST | | 1 | 2 | 1 | FADD | |
| FXAM | | 2 | | 1 | FMISC, ALU | |
| FRNDINT | | 5 | 10 | 3 | | |
| FPREM | | 1 | 7-10 | 8 | FMUL | |
| FPREM1 | | 1 | 8-11 | 8 | FMUL | |
| Math | | | | | | |
| FSQRT | | 1 | 27 | 12 | FMUL | |
| FLDPI, etc. | | 1 | | 1 | FMISC | |
| FSIN | | 66 | 140-190 | | | |
| FCOS | | 73 | 150-190 | | | |
| FSINCOS | | 98 | 170-200 | | | |
| FPTAN | | 67 | 150-180 | | | |
| FPATAN | | 97 | 217 | | | |
| FSCALE | | 5 | 8 | | | |
| FXTRACT | | 7 | 12 | 7 | | |
| F2XM1 | | 53 | 126 | | | |
| FYL2X | | 72 | 179 | | | |
| FYL2XP1 | | 75 | 175 | | | |
| Other | | | | | | |
| FNOP | | 1 | 0 | 1/3 | FANY | |
| (F)WAIT | | 1 | 0 | 1/3 | ALU | |
| FNCLEX | | 8 | | 27 | FMISC | |
| FNINIT | | 26 | | 100 | FMISC | |
| FNSAVE | | 77 | | 171 | | |
| FRSTOR | | 70 | | 136 | | |
| FXSAVE | | 61 | | 56 | | |
| FXRSTOR | | 101 | | 95 | | |

Integer MMX and XMM instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution unit | Notes |
|--------------------------|-------------|-----|---------|-----------------------|----------------|--|
| Move instructions | | | | | | |
| MOVD | r32, mm | 2 | 4 | 2 | FMISC, ALU | |
| MOVD | mm, r32 | 2 | 9 | 2 | FANY, ALU | |
| MOVD | mm, m32 | 1 | | 1/2 | FANY | |
| MOVD | r32, xmm | 3 | 2 | 2 | FMISC, ALU | |
| MOVD | xmm, r32 | 3 | 3 | 2 | | |
| MOVD | xmm, m32 | 2 | | 1 | FANY | |
| MOVD | m32, r | 1 | | 1 | FMISC | |
| MOVD (MOVQ) | r64, mm/xmm | 2 | 4 | 2 | FMISC, ALU | Moves 64 bits. Name of instruction differs |

K8

| | | | | | | |
|--------------------------------|------------|----|----|-----|-------------|-----|
| MOVD (MOVQ) | mm,r64 | 2 | 9 | 2 | FANY, ALU | do. |
| MOVD (MOVQ) | xmm,r64 | 3 | 9 | 2 | FANY, ALU | do. |
| MOVQ | mm,mm | 1 | 2 | 1/2 | FA/M | |
| MOVQ | xmm,xmm | 2 | 2 | 1 | FA/M, FMISC | |
| MOVQ | mm,m64 | 1 | | 1/2 | FANY | |
| MOVQ | xmm,m64 | 2 | | 1 | FANY, FMISC | |
| MOVQ | m64,mm/x | 1 | | 1 | FMISC | |
| MOVDQA | xmm,xmm | 2 | 2 | 1 | FA/M | |
| MOVDQA | xmm,m | 2 | | 2 | FMISC | |
| MOVDQA | m,xmm | 2 | | 2 | FMISC | |
| MOVDQU | xmm,m | 4 | | 2 | | |
| MOVDQU | m,xmm | 5 | | 2 | | |
| MOVDQ2Q | mm,xmm | 1 | 2 | 1/2 | FA/M | |
| MOVQ2DQ | xmm,mm | 2 | 2 | 1 | FA/M, FMISC | |
| MOVNTQ | m,mm | 1 | | 2 | FMISC | |
| MOVNTDQ | m,xmm | 2 | | 3 | FMISC | |
| PACKSSWB/DW | | | | | | |
| PACKUSWB | mm,r/m | 1 | 2 | 2 | FA/M | |
| PACKSSWB/DW | | | | | | |
| PACKUSWB | xmm,r/m | 3 | 3 | 2 | FA/M | |
| PUNPCKH/LBW/WD/DQ | mm,r/m | 1 | 2 | 2 | FA/M | |
| PUNPCKH/LBW/WD/DQ | xmm,r/m | 2 | 2 | 2 | FA/M | |
| PUNPCKHQDQ | xmm,r/m | 2 | 2 | 1 | FA/M | |
| PUNPCKLQDQ | xmm,r/m | 1 | 2 | 1/2 | FA/M | |
| PSHUFD | xmm,xmm,i | 3 | 3 | 1.5 | FA/M | |
| PSHUFW | mm,mm,i | 1 | 2 | 1/2 | FA/M | |
| PSHUFL/HW | xmm,xmm,i | 2 | 2 | 1 | FA/M | |
| MASKMOVQ | mm,mm | 32 | | 13 | | |
| MASKMOVDQU | xmm,xmm | 64 | | 26 | | |
| PMOVMASKB | r32,mm/xmm | 1 | 2 | 1 | FADD | |
| PEXTRW | r32,mm/x,i | 2 | 5 | 2 | FMISC, ALU | |
| PINSRW | mm,r32,i | 2 | 12 | 2 | FA/M | |
| PINSRW | xmm,r32,i | 3 | 12 | 3 | FA/M | |
| Arithmetic instructions | | | | | | |
| PADDB/W/D/Q | | | | | | |
| PADDSB/W | | | | | | |
| PADDUSB/W PSUBB/W/D/Q PSUBSB/W | | | | | | |
| PSUBUSB/W | mm,r/m | 1 | 2 | 1/2 | FA/M | |
| PADDB/W/D/Q | | | | | | |
| PADDSB/W ADDUSB/W PSUBB/W/D/Q | | | | | | |
| PSUBSB/W | | | | | | |
| PSUBUSB/W | xmm,r/m | 2 | 2 | 1 | FA/M | |
| PCMPEQ/GT B/W/D | mm,r/m | 1 | 2 | 1/2 | FA/M | |
| PCMPEQ/GT B/W/D | xmm,r/m | 2 | 2 | 1 | FA/M | |
| PMULLW PMULHW | | | | | | |
| PMULHUW | | | | | | |
| PMULUDQ | mm,r/m | 1 | 3 | 1 | FMUL | |
| PMULLW PMULHW | | | | | | |
| PMULHUW | | | | | | |
| PMULUDQ | xmm,r/m | 2 | 3 | 2 | FMUL | |

K8

| | | | | | |
|--------------------------|-----------|---|---|-----|------|
| PMADDWD | mm,r/m | 1 | 3 | 1 | FMUL |
| PMADDWD | xmm,r/m | 2 | 3 | 2 | FMUL |
| PAVGB/W | mm,r/m | 1 | 2 | 1/2 | FA/M |
| PAVGB/W | xmm,r/m | 2 | 2 | 1 | FA/M |
| PMIN/MAX SW/UB | mm,r/m | 1 | 2 | 1/2 | FA/M |
| PMIN/MAX SW/UB | xmm,r/m | 2 | 2 | 1 | FA/M |
| PSADBW | mm,r/m | 1 | 3 | 1 | FADD |
| PSADBW | xmm,r/m | 2 | 3 | 2 | FADD |
| Logic | | | | | |
| PAND PANDN POR PXOR | mm,r/m | 1 | 2 | 1/2 | FA/M |
| PAND PANDN POR PXOR | xmm,r/m | 2 | 2 | 1 | FA/M |
| PSLL/RL W/D/Q PSRAW/D | mm,i/mm/m | 1 | 2 | 1/2 | FA/M |
| PSLL/RL W/D/Q PSRAW/D | x,i/x/m | 2 | 2 | 1 | FA/M |
| PSLLDQ, PSRLDQ | xmm,i | 2 | 2 | 1 | FA/M |
| Other | | | | | |
| EMMS | | 1 | | 1/3 | FANY |

Floating point XMM instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution unit | Notes |
|--------------------------|----------|-----|---------|--------------------------|-------------------|-------|
| Move instructions | | | | | | |
| MOVAPS/D | r,r | 2 | 2 | 1 | FA/M | |
| MOVAPS/D | r,m | 2 | | 2 | FMISC | |
| MOVAPS/D | m,r | 2 | | 2 | FMISC | |
| MOVUPS/D | r,r | 2 | 2 | 1 | FA/M | |
| MOVUPS/D | r,m | 4 | | 2 | | |
| MOVUPS/D | m,r | 5 | | 2 | | |
| MOVSS/D | r,r | 1 | 2 | 1 | FA/M | |
| MOVSS/D | r,m | 2 | 4 | 1 | FANY FMISC | |
| MOVSS/D | m,r | 1 | 3 | 1 | FMISC | |
| MOVHPS, MOVLHPS | r,r | 1 | 2 | 1/2 | FA/M | |
| MOVHPS/D, MOVLPS/D | r,m | 1 | | 1 | FMISC | |
| MOVHPS/D, MOVLPS/D | m,r | 1 | | 1 | FMISC | |
| MOVDDUP | r,r | 2 | 2 | 1 | | SSE3 |
| MOVSH/LDUP | r,r | 2 | 2 | 2 | | SSE3 |
| MOVNTPS/D | m,r | 2 | | 3 | FMISC | |
| MOVMSKPS/D | r32,r | 1 | 8 | 1 | FADD | |
| SHUFPS/D | r,r/m,i | 3 | 3 | 2 | FMUL | |
| UNPCK H/L PS/D | r,r/m | 2 | 3 | 3 | FMUL | |
| Conversion | | | | | | |
| CVTTPS2PD | r,r/m | 2 | 4 | 2 | FMISC | |
| CVTPD2PS | r,r/m | 4 | 8 | 3 | FMISC | |

K8

| | | | | | | |
|--------------------|---------|---|-------|-------|-------|--|
| CVTSD2SS | r,r/m | 3 | 8 | 8 | FMISC | |
| CVTSS2SD | r,r/m | 1 | 2 | 1 | FMISC | |
| CVTDQ2PS | r,r/m | 2 | 5 | 2 | FMISC | |
| CVTDQ2PD | r,r/m | 2 | 5 | 2 | FMISC | |
| CVT(T)PS2DQ | r,r/m | 2 | 5 | 2 | FMISC | |
| CVT(T)PD2DQ | r,r/m | 4 | 8 | 3 | FMISC | |
| CVTPI2PS | xmm,mm | 1 | 4 | 1 | FMISC | |
| CVTPI2PD | xmm,mm | 2 | 5 | 2 | FMISC | |
| CVT(T)PS2PI | mm,xmm | 1 | 6 | 1 | FMISC | |
| CVT(T)PD2PI | mm,xmm | 3 | 8 | 2 | FMISC | |
| CVTSI2SS | xmm,r32 | 3 | 14 | 2 | FMISC | |
| CVTSI2SD | xmm,r32 | 2 | 12 | 2 | FMISC | |
| CVT(T)SD2SI | r32,xmm | 2 | 10 | 2 | FMISC | |
| CVT(T)SS2SI | r32,xmm | 2 | 9 | 2 | FMISC | |
| Arithmetic | | | | | | |
| ADDSS/D SUBSS/D | r,r/m | 1 | 4 | 1 | FADD | |
| ADDPS/D SUBPS/D | r,r/m | 2 | 4 | 2 | FADD | |
| HADDPS/D HSUBPS/D | r,r/m | 2 | 4 | 2 | FADD | SSE3 |
| MULSS/D | r,r/m | 1 | 4 | 1 | FMUL | |
| MULPS/D | r,r/m | 2 | 4 | 2 | FMUL | |
| | | | | | | Low values are for round divisors, e.g. powers of 2. |
| DIVSS | r,r/m | 1 | 11-16 | 8-13 | FMUL | do. |
| DIVPS | r,r/m | 2 | 18-30 | 18-30 | FMUL | do. |
| DIVSD | r,r/m | 1 | 11-20 | 8-17 | FMUL | do. |
| DIVPD | r,r/m | 2 | 16-34 | 16-34 | FMUL | do. |
| RCPSS | r,r/m | 1 | 3 | 1 | FMUL | |
| RCPPS | r,r/m | 2 | 3 | 2 | FMUL | |
| MAXSS/D MINSS/D | r,r/m | 1 | 2 | 1 | FADD | |
| MAXPS/D MINPS/D | r,r/m | 2 | 2 | 2 | FADD | |
| CMPccSS/D | r,r/m | 1 | 2 | 1 | FADD | |
| CMPccPS/D | r,r/m | 2 | 2 | 2 | FADD | |
| COMISS/D UCOMISS/D | r,r/m | 1 | 2 | 1 | FADD | |
| Logic | | | | | | |
| ANDPS/D ANDNPS/D | | | | | | |
| ORPS/D XORPS/D | r,r/m | 2 | 2 | 2 | FMUL | |
| Math | | | | | | |
| SQRTSS | r,r/m | 1 | 19 | 16 | FMUL | |
| SQRTPS | r,r/m | 2 | 36 | 36 | FMUL | |
| SQRTSD | r,r/m | 1 | 27 | 24 | FMUL | |
| SQRTPD | r,r/m | 2 | 48 | 48 | FMUL | |
| RSQRTSS | r,r/m | 1 | 3 | 1 | FMUL | |
| RSQRTPS | r,r/m | 2 | 3 | 2 | FMUL | |
| Other | | | | | | |
| LDMXCSR | m | 8 | | 9 | | |
| STMXCSR | m | 3 | | 10 | | |

AMD K10

List of instruction timings and macro-operation breakdown

Explanation of column headings:

| | |
|-------------------------------|--|
| Instruction: | Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc. |
| Operands: | i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, xmm = 128 bit xmm register, sr = segment register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc. |
| Ops: | Number of macro-operations issued from instruction decoder to schedulers. Instructions with more than 2 macro-operations use microcode. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latency listed does not include the memory operand where the operand is listed as register or memory (r/m). |
| Reciprocal throughput: | This is also called issue latency. This value indicates the average number of clock cycles from the execution of an instruction begins to a subsequent independent instruction of the same kind can begin to execute. A value of 1/3 indicates that the execution units can handle 3 instructions per clock cycle in one thread. However, the throughput may be limited by other bottlenecks in the pipeline. |
| Execution unit: | Indicates which execution unit is used for the macro-operations. ALU means any of the three integer ALU's. ALU0_1 means that ALU0 and ALU1 are both used. AGU means any of the three integer address generation units. FADD means floating point adder unit. FMUL means floating point multiplier unit. FMISC means floating point store and miscellaneous unit. FA/M means FADD or FMUL is used. FANY means any of the three floating point units can be used. Two macro-operations can execute simultaneously if they go to different execution units. |

Integer instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution unit | Notes |
|--------------------------|-------------|-----|---------|-----------------------|----------------|--|
| Move instructions | | | | | | |
| MOV | r,r | 1 | 1 | 1/3 | ALU | |
| MOV | r,i | 1 | 1 | 1/3 | ALU | |
| MOV | r8,m8 | 1 | 4 | 1/2 | ALU, AGU | Any addr. mode. Add 1 clock if code segment base ≠ 0 |
| MOV | r16,m16 | 1 | 4 | 1/2 | ALU, AGU | |
| MOV | r32,m32 | 1 | 3 | 1/2 | AGU | |
| MOV | r64,m64 | 1 | 3 | 1/2 | AGU | |
| MOV | m8,r8H | 1 | 8 | 1/2 | AGU | AH, BH, CH, DH |
| MOV | m8,r8L | 1 | 3 | 1/2 | AGU | Any other 8-bit reg. |
| MOV | m16/32/64,r | 1 | 3 | 1/2 | AGU | Any addressing mode |
| MOV | m,i | 1 | 3 | 1/2 | AGU | |
| MOV | m64,i32 | 1 | 3 | 1/2 | AGU | |
| MOV | r,sr | 1 | 3-4 | 1/2 | | |
| MOV | sr,r/m | 6 | 8-26 | 8 | | from AMD manual |
| MOVNTI | m,r | 1 | | 1 | AGU | |
| MOVZX, MOVZX | r,r | 1 | 1 | 1/3 | ALU | |
| MOVZX, MOVZX | r,m | 1 | 4 | 1/2 | ALU, AGU | |
| MOVSLD | r64,r32 | 1 | 1 | 1/3 | ALU | |

K10

| | | | | | | |
|--------------------------------|-------------|------|----|-----|----------|---|
| MOVSXD | r64,m32 | 1 | 4 | 1/2 | ALU, AGU | Timing depends on hw |
| CMOVcc | r,r | 1 | 1 | 1/3 | ALU | |
| CMOVcc | r,m | 1 | 4 | 1/2 | ALU, AGU | |
| XCHG | r,r | 2 | 1 | 1 | ALU | |
| XCHG | r,m | 2 | 21 | 19 | ALU, AGU | |
| XLAT | | 2 | 5 | 5 | ALU, AGU | |
| PUSH | r | 1 | | 1/2 | ALU, AGU | |
| PUSH | i | 1 | | 1/2 | ALU, AGU | |
| PUSH | m | 2 | | 1 | ALU, AGU | |
| PUSH | sr | 2 | | 1 | ALU, AGU | |
| PUSHF(D/Q) | | 9 | | 3 | ALU, AGU | |
| PUSHA(D) | | 9 | 6 | 6 | ALU, AGU | |
| POP | r | 1 | | 1/2 | ALU, AGU | |
| POP | m | 3 | 3 | 1 | ALU, AGU | |
| POP | DS/ES/FS/GS | 6 | 10 | 8 | ALU, AGU | |
| POP | SS | 10 | 26 | 16 | ALU, AGU | |
| POPF(D/Q) | | 28 | 16 | 11 | ALU, AGU | |
| POPA(D) | | 9 | 6 | 6 | ALU, AGU | |
| LEA | r16,[m] | 2 | 3 | 1 | ALU, AGU | Any address size ≤ 2 source operands W. scale or 3 opr. |
| LEA | r32/64,[m] | 1 | 1 | 1/3 | ALU | |
| LEA | r32/64,[m] | 1 | 2 | 1/3 | AGU | |
| LAHF | | 4 | 3 | 2 | ALU | |
| SAHF | | 1 | 1 | 1/3 | ALU | |
| SALC | | 1 | 1 | 1 | ALU | 3DNow |
| LDS, LES, ... | r,m | 10 | | 10 | | |
| BSWAP | r | 1 | 1 | 1/3 | ALU | |
| PREFETCHNTA | m | 1 | | 1/2 | AGU | |
| PREFETCHT0/1/2 | m | 1 | | 1/2 | AGU | |
| PREFETCH(W) | m | 1 | | 1/2 | AGU | |
| SFENCE | | 6 | | 8 | | |
| LFENCE | | 1 | | 1 | | |
| MFENCE | | 4 | | 33 | | |
| IN | r,i/DX | ~270 | | | | |
| OUT | i/DX,r | ~300 | | | | |
| Arithmetic instructions | | | | | | |
| ADD, SUB | r,r/i | 1 | 1 | 1/3 | ALU | latency ax=3, dx=4 |
| ADD, SUB | r,m | 1 | | 1/2 | ALU, AGU | |
| ADD, SUB | m,r | 1 | 4 | 1 | ALU, AGU | |
| ADC, SBB | r,r/i | 1 | 1 | 1/3 | ALU | |
| ADC, SBB | r,m | 1 | | 1/2 | ALU, AGU | |
| ADC, SBB | m,r/i | 1 | 4 | 1 | ALU, AGU | |
| CMP | r,r/i | 1 | 1 | 1/3 | ALU | |
| CMP | r,m | 1 | | 1/2 | ALU, AGU | |
| INC, DEC, NEG | r | 1 | 1 | 1/3 | ALU | |
| INC, DEC, NEG | m | 1 | 7 | 2 | ALU, AGU | |
| AAA, AAS | | 9 | 5 | 5 | ALU | |
| DAA | | 12 | 6 | 6 | ALU | |
| DAS | | 16 | 7 | 7 | ALU | |
| AAD | | 4 | 5 | 5 | ALU0 | |
| AAM | | 30 | 13 | 13 | ALU | |
| MUL, IMUL | r8/m8 | 1 | 3 | 1 | ALU0 | |
| MUL, IMUL | r16/m16 | 3 | 3 | 2 | ALU0_1 | |

K10

| | | | | | | |
|---------------------|-------------|----|-------|-------|----------|---|
| MUL, IMUL | r32/m32 | 2 | 3 | 1 | ALU0_1 | latency rax=4, rdx=5 |
| MUL, IMUL | r64/m64 | 2 | 4 | 2 | ALU0_1 | |
| IMUL | r16,r16/m16 | 1 | 3 | 1 | ALU0 | |
| IMUL | r32,r32/m32 | 1 | 3 | 1 | ALU0 | |
| IMUL | r64,r64/m64 | 1 | 4 | 2 | ALU0_1 | |
| IMUL | r16,(r16),i | 2 | 4 | 1 | ALU0 | |
| IMUL | r32,(r32),i | 1 | 3 | 1 | ALU0 | |
| IMUL | r64,(r64),i | 1 | 4 | 2 | ALU0 | |
| IMUL | r16,m16,i | 3 | | 2 | ALU0 | |
| IMUL | r32,m32,i | 3 | | 2 | ALU0 | |
| IMUL | r64,m64,i | 3 | | 2 | ALU0_1 | |
| DIV | r8/m8 | | 17 | 17 | ALU | |
| IDIV | r8 | | 19 | 19 | ALU | |
| IDIV | m8 | | 22 | 22 | ALU | |
| DIV | r16/m16 | | 15-30 | 15-30 | ALU | Depends on number of significant bits in absolute value of dividend. See AMD software optimization guide. |
| DIV | r32/m32 | | 15-46 | 15-46 | ALU | |
| DIV | r64/m64 | | 15-78 | 15-78 | ALU | |
| IDIV | r16/m16 | | 24-39 | 24-39 | ALU | |
| IDIV | r32/m32 | | 24-55 | 24-55 | ALU | |
| IDIV | r64/m64 | | 24-87 | 24-87 | ALU | |
| CBW, CWDE, CDQE | | 1 | 1 | 1/3 | ALU | |
| CWD, CDQ, CQO | | 1 | 1 | 1/3 | ALU | |
| Logic instructions | | | | | | |
| AND, OR, XOR | r,r | 1 | 1 | 1/3 | ALU | |
| AND, OR, XOR | r,m | 1 | | 1/2 | ALU, AGU | |
| AND, OR, XOR | m,r | 1 | 4 | 1 | ALU, AGU | |
| TEST | r,r | 1 | 1 | 1/3 | ALU | |
| TEST | r,m | 1 | | 1/2 | ALU, AGU | |
| NOT | r | 1 | 1 | 1/3 | ALU | |
| NOT | m | 1 | 7 | 1 | ALU, AGU | |
| SHL, SHR, SAR | r,i/CL | 1 | 1 | 1/3 | ALU | |
| ROL, ROR | r,i/CL | 1 | 1 | 1/3 | ALU | |
| RCL, RCR | r,1 | 1 | 1 | 1 | ALU | |
| RCL | r,i | 9 | 3 | 3 | ALU | |
| RCR | r,i | 7 | 3 | 3 | ALU | |
| RCL | r,CL | 9 | 4 | 4 | ALU | |
| RCR | r,CL | 7 | 3 | 3 | ALU | |
| SHL,SHR,SAR,ROL,RCR | m,i /CL | 1 | 7 | 1 | ALU, AGU | |
| RCL, RCR | m,1 | 1 | 7 | 1 | ALU, AGU | |
| RCL | m,i | 10 | 7 | 5 | ALU, AGU | |
| RCR | m,i | 9 | 7 | 6 | ALU, AGU | |
| RCL | m,CL | 9 | 8 | 6 | ALU, AGU | |
| RCR | m,CL | 8 | 7 | 5 | ALU, AGU | |
| SHLD, SHRD | r,r,i | 6 | 3 | 2 | ALU | |
| SHLD, SHRD | r,r,cl | 7 | 3 | 3 | ALU | |
| SHLD, SHRD | m,r,i/CL | 8 | 7.5 | 6 | ALU, AGU | |
| BT | r,r/i | 1 | 1 | 1/3 | ALU | |
| BT | m,i | 1 | | 1/2 | ALU, AGU | |
| BT | m,r | 5 | 7 | 2 | ALU, AGU | |
| BTC, BTR, BTS | r,r/i | 2 | 2 | 1/3 | ALU | |
| BTC | m,i | 5 | 9 | 1.5 | ALU, AGU | |
| BTR, BTS | m,i | 4 | 9 | 1.5 | ALU, AGU | |

K10

| | | | | | | |
|--------------------------------------|------------|-------|-------|---------|----------|-------------------------|
| BTC | m,r | 8 | 8 | 10 | ALU, AGU | |
| BTR, BTS | m,r | 8 | 8 | 7 | ALU, AGU | |
| BSF | r,r | 6 | 4 | 3 | ALU | |
| BSR | r,r | 7 | 4 | 3 | ALU | |
| BSF | r,m | 7 | 7 | 3 | ALU, AGU | |
| BSR | r,m | 8 | 7 | 3 | ALU, AGU | |
| POPCNT | r,r/m | 1 | 2 | 1 | ALU | SSE4.A / SSE4.2 |
| LZCNT | r,r/m | 1 | 2 | 1 | ALU | SSE4.A, AMD only |
| SETcc | r | 1 | 1 | 1/3 | ALU | |
| SETcc | m | 1 | | 1/2 | ALU, AGU | |
| CLC, STC | | 1 | | 1/3 | ALU | |
| CMC | | 1 | 1 | 1/3 | ALU | |
| CLD | | 1 | | 1/3 | ALU | |
| STD | | 2 | | 2/3 | ALU | |
| Control transfer instructions | | | | | | |
| JMP | short/near | 1 | | 2 | ALU | |
| JMP | far | 16-20 | 23-32 | | | low values = real mode |
| JMP | r | 1 | | 2 | ALU | |
| JMP | m(near) | 1 | | 2 | ALU, AGU | |
| JMP | m(far) | 17-21 | 25-33 | | | low values = real mode |
| Jcc | short/near | 1 | | 1/3 - 2 | ALU | recip. thrp.= 2 if jump |
| J(E/R)CXZ | short | 2 | | 2/3 - 2 | ALU | recip. thrp.= 2 if jump |
| LOOP | short | 7 | | 3 | ALU | |
| CALL | near | 3 | 2 | 2 | ALU | |
| CALL | far | 16-22 | 23-32 | | | low values = real mode |
| CALL | r | 4 | 3 | 3 | ALU | |
| CALL | m(near) | 5 | 3 | 3 | ALU, AGU | |
| CALL | m(far) | 16-22 | 24-33 | | | low values = real mode |
| RETN | | 2 | 3 | 3 | ALU | |
| RETN | i | 2 | 3 | 3 | ALU | |
| RETF | | 15-23 | 24-35 | | | low values = real mode |
| RETF | i | 15-24 | 24-35 | | | low values = real mode |
| IRET | | 32 | 81 | | | real mode |
| INT | i | 33 | 42 | | | real mode |
| BOUND | m | 6 | | 2 | | values are for no jump |
| INTO | | 2 | | 2 | | values are for no jump |
| String instructions | | | | | | |
| LODS | | 4 | 2 | 2 | | |
| REP LODS | | 5 | 2 | 2 | | values are per count |
| STOS | | 4 | 2 | 2 | | |
| REP STOS | | 2 | 1 | 1 | | values are per count |
| MOVS | | 7 | 3 | 3 | | |
| REP MOVS | | 3 | 1 | 1 | | values are per count |
| SCAS | | 5 | 2 | 2 | | |
| REP SCAS | | 5 | 2 | 2 | | values are per count |
| CMPS | | 7 | 3 | 3 | | |
| REP CMPS | | 3 | 1 | 1 | | values are per count |
| Other | | | | | | |
| NOP (90) | | 1 | 0 | 1/3 | ALU | |
| Long NOP (0F 1F) | | 1 | 0 | 1/3 | ALU | |

K10

| | | | | | | |
|-------|--|-------|--------|----|----|------------------------|
| ENTER | | i,0 | 12 | | 12 | 3 ops, 5 clk if 16 bit |
| LEAVE | | 2 | | 3 | | |
| CLI | | 8-9 | | 5 | | |
| STI | | 16-17 | | 27 | | |
| CPUID | | 22-50 | 47-164 | | | |
| RDTSC | | 30 | | 67 | | |
| RDPMC | | 13 | | 5 | | |

Floating point x87 instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution unit | Notes |
|--------------------------------|----------|-----|---------|-----------------------|----------------|--|
| Move instructions | | | | | | |
| FLD | r | 1 | 2 | 1/2 | FA/M | |
| FLD | m32/64 | 1 | 4 | 1/2 | FANY | |
| FLD | m80 | 7 | 13 | 4 | | |
| FBLD | m80 | 20 | 94 | 30 | | |
| FST(P) | r | 1 | 2 | 1/2 | FA/M | |
| FST(P) | m32/64 | 1 | 2 | 1 | FMISC | |
| FSTP | m80 | 10 | 8 | 7 | | |
| FBSTP | m80 | 218 | 167 | 163 | | |
| FXCH | r | 1 | 0 | 1/3 | | |
| FILD | m | 1 | 6 | 1 | FMISC | |
| FIST(P) | m | 1 | 4 | 1 | FMISC | |
| FLDZ, FLD1 | | 1 | | 1 | FMISC | |
| FCMOVcc | st0,r | 9 | | | FMISC, FA/M | Low latency immediately after FCOMI |
| FFREE | r | 1 | | 1/3 | FANY | |
| FINCSTP, FDECSTP | | 1 | 0 | 1/3 | FANY | |
| FNSTSW | AX | 2 | | 16 | FMISC, ALU | Low latency immediately after FCOM FTST do. do. faster if unchanged |
| FSTSW | AX | 3 | | 14 | FMISC, ALU | |
| FNSTSW | m16 | 2 | | 9 | FMISC, ALU | |
| FNSTCW | m16 | 3 | | 2 | FMISC, ALU | |
| FLDCW | m16 | 12 | | 14 | FMISC, ALU | |
| Arithmetic instructions | | | | | | |
| FADD(P),FSUB(R)(P) | r/m | 1 | 4 | 1 | FADD | |
| FIADD,FISUB(R) | m | 2 | | 4 | FADD,FMISC | |
| FMUL(P) | r/m | 1 | 4 | 1 | FMUL | |
| FIMUL | m | 2 | | 4 | FMUL,FMISC | |
| FDIV(R)(P) | r/m | 1 | ? | 24 | FMUL | |
| FIDIV(R) | m | 2 | 31 | 24 | FMUL,FMISC | |
| FABS, FCHS | | 1 | 2 | 2 | FMUL | |
| FCOM(P), FUCOM(P) | r/m | 1 | | 1 | FADD | |
| FCOMPP, FUCOMPP | | 1 | | 1 | FADD | |
| FCOMI(P) | r | 1 | | 1 | FADD | |
| FICOM(P) | m | 2 | | 1 | FADD, FMISC | |
| FTST | | 1 | | 1 | FADD | |
| FXAM | | 2 | | 1 | FMISC, ALU | |
| FRNDINT | | 6 | | 37 | | |
| FPREM | | 1 | | 7 | FMUL | |
| FPREM1 | | 1 | | 7 | FMUL | |

K10

| | | | | | | |
|--------------|---|----|-------|-----|-------|--|
| Math | | | | | | |
| FSQRT | | 1 | 35 | 35 | FMUL | |
| FLDPI, etc. | | 1 | | 1 | FMISC | |
| FSIN | | 45 | ~51? | | | |
| FCOS | | 51 | ~90? | | | |
| FSINCOS | | 76 | ~125? | | | |
| FPTAN | | 45 | ~119 | | | |
| FPATAN | | 9 | 151? | 45? | | |
| FSCALE | | 5 | 9 | 29 | | |
| FXTRACT | | 11 | 9 | 41 | | |
| F2XM1 | | 8 | 65 | 30? | | |
| FYL2X | | 8 | 13 | 30? | | |
| FYL2XP1 | | 12 | 114 | 44? | | |
| Other | | | | | | |
| FNOP | | 1 | 0 | 1/3 | FANY | |
| (F)WAIT | | 1 | 0 | 1/3 | ALU | |
| FNCLEX | | 8 | | 28 | FMISC | |
| FNINIT | | 26 | | 103 | FMISC | |
| FNSAVE | m | 77 | 162 | 149 | | |
| FRSTOR | m | 70 | 133 | 149 | | |
| FXSAVE | m | 61 | 63 | 58 | | |
| FXRSTOR | m | 85 | 89 | 79 | | |

Integer MMX and XMM instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution unit | Notes |
|--------------------------|-----------|-----|---------|-----------------------|----------------|--|
| Move instructions | | | | | | |
| MOVD | r32, mm | 1 | 3 | 1 | FADD | |
| MOVD | mm, r32 | 2 | 6 | 3 | | |
| MOVD | mm, m32 | 1 | 4 | 1/2 | FANY | |
| MOVD | r32, xmm | 1 | 3 | 1 | FADD | |
| MOVD | xmm, r32 | 2 | 6 | 3 | | |
| MOVD | xmm, m32 | 1 | 2 | 1/2 | | |
| MOVD | m32, mm/x | 1 | 2 | 1 | FMISC | |
| MOVD (MOVQ) | r64, mm/x | 1 | 3 | 1 | FADD | Moves 64 bits. Name of instruction differs |
| MOVD (MOVQ) | mm, r64 | 2 | 6 | 3 | | do. |
| MOVD (MOVQ) | xmm, r64 | 2 | 6 | 3 | FMUL, ALU | do. |
| MOVQ | mm, mm | 1 | 2 | 1/2 | FA/M | |
| MOVQ | xmm, xmm | 1 | 2.5 | 1/3 | FANY | |
| MOVQ | mm, m64 | 1 | 4 | 1/2 | FANY | |
| MOVQ | xmm, m64 | 1 | 2 | 1/2 | ? | |
| MOVQ | m64, mm/x | 1 | 2 | 1 | FMISC | |
| MOVDQA | xmm, xmm | 1 | 2.5 | 1/3 | FANY | |
| MOVDQA | xmm, m | 1 | 2 | 1/2 | ? | |
| MOVDQA | m, xmm | 2 | 2 | 1 | FMUL, FMISC | |
| MOVDQU | xmm, m | 1 | 2 | 1/2 | | |
| MOVDQU | m, xmm | 3 | 3 | 2 | | |
| MOVDQ2Q | mm, xmm | 1 | 2 | 1/3 | FANY | |
| MOVQ2DQ | xmm, mm | 1 | 2 | 1/3 | FANY | |

K10

| | | | | | | |
|--------------------------------|-------------|----|---|-----|------------|------------------|
| MOVNTQ | m,mm | 1 | | 1 | FMISC | |
| MOVNTDQ | m,xmm | 2 | | 1 | FMUL,FMISC | |
| PACKSSWB/DW | | | | | | |
| PACKUSWB | mm,r/m | 1 | 2 | 1/2 | FA/M | |
| PACKSSWB/DW | | | | | | |
| PACKUSWB | xmm,r/m | 1 | 3 | 1/2 | FA/M | |
| PUNPCKH/LBW/WD/DQ | mm,r/m | 1 | 2 | 1/2 | FA/M | |
| PUNPCKH/LBW/WD/DQ | xmm,r/m | 1 | 3 | 1/2 | FA/M | |
| PUNPCKHQDQ | xmm,r/m | 1 | 3 | 1/2 | FA/M | |
| PUNPCKLQDQ | xmm,r/m | 1 | 3 | 1/2 | FA/M | |
| PSHUFD | xmm,xmm,i | 1 | 3 | 1/2 | FA/M | |
| PSHUFW | mm,mm,i | 1 | 2 | 1/2 | FA/M | |
| PSHUFL/HW | xmm,xmm,i | 1 | 2 | 1/2 | FA/M | |
| MASKMOVQ | mm,mm | 32 | | 13 | | |
| MASKMOVDQU | xmm,xmm | 64 | | 24 | | |
| PMOVMSKB | r32,mm/xmm | 1 | 3 | 1 | FADD | |
| PEXTRW | r32,mm/x,i | 2 | 6 | 1 | | |
| PINSRW | mm/x,r32,i | 2 | 9 | 3 | FA/M | |
| INSERTQ | xmm,xmm | 3 | 6 | 2 | FA/M | SSE4.A, AMD only |
| INSERTQ | xmm,xmm,i,i | 3 | 6 | 2 | FA/M | SSE4.A, AMD only |
| EXTRQ | xmm,xmm | 1 | 2 | 1/2 | FA/M | SSE4.A, AMD only |
| EXTRQ | xmm,xmm,i,i | 1 | 2 | 1/2 | FA/M | SSE4.A, AMD only |
| Arithmetic instructions | | | | | | |
| PADDB/W/D/Q | | | | | | |
| PADDSB/W | | | | | | |
| PADDUSB/W PSUBB/W/D/Q PSUBSB/W | mm/xmm,r/m | 1 | 2 | 1/2 | FA/M | |
| PSUBUSB/W | mm/xmm,r/m | 1 | 2 | 1/2 | FA/M | |
| PCMPEQ/GT B/W/D | | | | | | |
| PMULLW PMULHW | | | | | | |
| PMULHUW | | | | | | |
| PMULUDQ | mm/xmm,r/m | 1 | 3 | 1 | FMUL | |
| PMADDWD | mm/xmm,r/m | 1 | 3 | 1 | FMUL | |
| PAVGB/W | mm/xmm,r/m | 1 | 2 | 1/2 | FA/M | |
| PMIN/MAX SW/UB | mm/xmm,r/m | 1 | 2 | 1/2 | FA/M | |
| PSADBW | mm/xmm,r/m | 1 | 3 | 1 | FADD | |
| Logic | | | | | | |
| PAND PANDN POR | | | | | | |
| PXOR | mm/xmm,r/m | 1 | 2 | 1/2 | FA/M | |
| PSLL/RL W/D/Q | | | | | | |
| PSRAW/D | mm,i/mm/m | 1 | 2 | 1/2 | FA/M | |
| PSLL/RL W/D/Q | | | | | | |
| PSRAW/D | x,i/mm/x | 1 | 3 | 1/2 | FA/M | |
| PSLLDQ, PSRLDQ | xmm,i | 1 | 3 | 1/2 | FA/M | |
| Other | | | | | | |
| EMMS | | 1 | | 1/3 | FANY | |

Floating point XMM instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution unit | Notes | |
|-----------------------|----------|-----|---------|-----------------------|----------------|------------------|--|
| Move instructions | | | | | | | |
| MOVAPS/D | r,r | 1 | 2.5 | 1/2 | FANY | SSE4.A, AMD only | |
| MOVAPS/D | r,m | 1 | 2 | 1/2 | ? | | |
| MOVAPS/D | m,r | 2 | 2 | 1 | FMUL,FMISC | | |
| MOVUPS/D | r,r | 1 | 2.5 | 1/2 | FANY | | |
| MOVUPS/D | r,m | 1 | 2 | 1/2 | ? | | |
| MOVUPS/D | m,r | 3 | 3 | 2 | FMISC | | |
| MOVSS/D | r,r | 1 | 2 | 1/2 | FA/M | | |
| MOVSS/D | r,m | 1 | 2 | 1/2 | ? | | |
| MOVSS/D | m,r | 1 | 2 | 1 | FMISC | | |
| MOVHLPS, MOVLHPS | r,r | 1 | 3 | 1/2 | FA/M | | |
| MOVHPS/D, MOVLPS/D | r,m | 1 | 4 | 1/2 | FA/M | | |
| MOVHPS/D, MOVLPS/D | m,r | 1 | | 1 | FMISC | | |
| MOVNTPS/D | m,r | 2 | | 3 | FMUL,FMISC | | |
| MOVNTSS/D | m,r | 1 | | 1 | FMISC | | |
| MOVMSKPS/D | r32,r | 1 | 3 | 1 | FADD | | |
| SHUFPS/D | r,r/m,i | 1 | 3 | 1/2 | FA/M | | |
| UNPCK H/L PS/D | r,r/m | 1 | 3 | 1/2 | FA/M | | |
| Conversion | | | | | | | |
| CVTPS2PD | r,r/m | 1 | 2 | 1 | FMISC | | |
| CVTPD2PS | r,r/m | 2 | 7 | 1 | | | |
| CVTSD2SS | r,r/m | 3 | 8 | 2 | | | |
| CVTSS2SD | r,r/m | 3 | 7 | 2 | | | |
| CVTDQ2PS | r,r/m | 1 | 4 | 1 | FMISC | | |
| CVTDQ2PD | r,r/m | 1 | 4 | 1 | FMISC | | |
| CVT(T)PS2DQ | r,r/m | 1 | 4 | 1 | FMISC | | |
| CVT(T)PD2DQ | r,r/m | 2 | 7 | 1 | | | |
| CVTPI2PS | xmm,mm | 2 | 7 | 1 | | | |
| CVTPI2PD | xmm,mm | 1 | 4 | 1 | FMISC | | |
| CVT(T)PS2PI | mm,xmm | 1 | 4 | 1 | FMISC | | |
| CVT(T)PD2PI | mm,xmm | 2 | 7 | 1 | | | |
| CVTSI2SS | xmm,r32 | 3 | 14 | 3 | | | |
| CVTSI2SD | xmm,r32 | 3 | 14 | 3 | | | |
| CVT(T)SD2SI | r32,xmm | 2 | 8 | 1 | FADD,FMISC | | |
| CVT(T)SS2SI | r32,xmm | 2 | 8 | 1 | FADD,FMISC | | |
| Arithmetic | | | | | | | |
| ADDSS/D SUBSS/D | r,r/m | 1 | 4 | 1 | FADD | | |
| ADDPS/D SUBPS/D | r,r/m | 1 | 4 | 1 | FADD | | |
| MULSS/D | r,r/m | 1 | 4 | 1 | FMUL | | |
| MULPS/D | r,r/m | 1 | 4 | 1 | FMUL | | |
| DIVSS | r,r/m | 1 | 16 | 13 | FMUL | | |
| DIVPS | r,r/m | 1 | 18 | 15 | FMUL | | |
| DIVSD | r,r/m | 1 | 20 | 17 | FMUL | | |
| DIVPD | r,r/m | 1 | 20 | 17 | FMUL | | |
| RCPSS RCPPS | r,r/m | 1 | 3 | 1 | FMUL | | |

K10

| | | | | | | |
|------------------------------------|-------|----|----|-----|------|--|
| MAXSS/D MINSS/D | r,r/m | 1 | 2 | 1 | FADD | |
| MAXPS/D MINPS/D | r,r/m | 1 | 2 | 1 | FADD | |
| CMPccSS/D | r,r/m | 1 | 2 | 1 | FADD | |
| CMPccPS/D | r,r/m | 1 | 2 | 1 | FADD | |
| COMISS/D UCOMISS/D | r,r/m | 1 | | 1 | FADD | |
| Logic | | | | | | |
| ANDPS/D ANDNPS/D ORPS/D XORPS/D | r,r/m | 1 | 2 | 1/2 | FA/M | |
| Math | | | | | | |
| SQRTSS | r,r/m | 1 | 19 | 16 | FMUL | |
| SQRTPS | r,r/m | 1 | 21 | 18 | FMUL | |
| SQRTSD | r,r/m | 1 | 27 | 24 | FMUL | |
| SQRTPD | r,r/m | 1 | 27 | 24 | FMUL | |
| RSQRTSS | r,r/m | 1 | 3 | 1 | FMUL | |
| RSQRTPS | r,r/m | 1 | 3 | 1 | FMUL | |
| Other | | | | | | |
| LDMXCSR | m | 12 | 12 | 10 | | |
| STMXCSR | m | 3 | 12 | 11 | | |

Obsolete 3DNow instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution unit | Notes | |
|-------------------------------|----------|-----|---------|-----------------------|----------------|---|--|
| Move and convert instructions | | | | | | 3DNow extension 3DNow extension 3DNow extension | |
| PF2ID | mm,mm | | | | | | |
| PI2FD | mm,mm | | | | | | |
| PF2IW | mm,mm | | | | | | |
| PI2FW | mm,mm | | | | | | |
| PSWAPD | mm,mm | | | | | | |
| Integer instructions | | | | | | | |
| PAVGUSB | mm,mm | | | | | | |
| PMULHRW | mm,mm | | | | | | |
| Floating point instructions | | | | | | | |
| PFADD/SUB/SUBR | mm,mm | 1 | 4 | 1 | FADD | 3DNow extension | |
| PFCMPEQ/GE/GT | mm,mm | 1 | 2 | 1 | FADD | | |
| PFMAX/MIN | mm,mm | 1 | 2 | 1 | FADD | | |
| PFMUL | mm,mm | 1 | 4 | 1 | FMUL | | |
| PFACC | mm,mm | 1 | 4 | 1 | FADD | | |
| PFNACC, PFPNACC | mm,mm | 1 | 4 | 1 | FADD | | |
| PFRCP | mm,mm | 1 | 3 | 1 | FMUL | | |
| PFRCPIT1/2 | mm,mm | 1 | 4 | 1 | FMUL | | |
| PFRSQRT | mm,mm | 1 | 3 | 1 | FMUL | | |
| PFRSQIT1 | mm,mm | 1 | 4 | 1 | FMUL | | |
| Other | | | | | | | |
| FEMMS | mm,mm | | | | | | |

K10

Thank you to Xucheng Tang for doing the measurements on the K10.

AMD Bulldozer

List of instruction timings and macro-operation breakdown

Explanation of column headings:

| | |
|-------------------------------|--|
| Instruction: | Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc. |
| Operands: | i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, x = 128 bit xmm register, y = 256 bit ymm register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc. |
| Ops: | Number of macro-operations issued from instruction decoder to schedulers. Instructions with more than 2 macro-operations use microcode. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latency listed does not include the memory operand where the listing for register and memory operand are joined (r/m). |
| Reciprocal throughput: | This is also called issue latency. This value indicates the average number of clock cycles from the execution of an instruction begins to a subsequent independent instruction of the same kind can begin to execute. A value of 1/3 indicates that the execution units can handle 3 instructions per clock cycle in one thread. However, the throughput may be limited by other bottlenecks in the pipeline. |
| Execution pipe: | Indicates which execution pipe or unit is used for the macro-operations: Integer pipes: EX0: integer ALU, division EX1: integer ALU, multiplication, jump EX01: can use either EX0 or EX1 AG01: address generation unit 0 or 1 Floating point and vector pipes: P0: floating point add, mul, div, convert, shuffle, shift P1: floating point add, mul, div, shuffle, shift P2: move, integer add, boolean P3: move, integer add, boolean, store P01: can use either P0 or P1 P23: can use either P2 or P3 Two macro-operations can execute simultaneously if they go to different execution pipes |
| Domain: | Tells which execution unit domain is used: ivec: integer vector execution unit. fp: floating point execution unit. fma: floating point multiply/add subunit. inherit: the output operand inherits the domain of the input operand. ivec/fma means the input goes to the ivec domain and the output comes from the fma domain. There is an additional latency of 1 clock cycle if the output of an ivec instruction goes to the input of a fp or fma instruction, and when the output of a fp or fma instruction goes to the input of an ivec or store instruction. There is no latency between the fp and fma units. All other latencies after memory load and before memory store instructions are included in the latency counts. An fma instruction has a latency of 5 if the output goes to another fma instruction, 6 if the output goes to an fp instruction, and 6+1 if the output goes to an ivec or store instruction. |

Integer instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Notes |
|-------------|----------|-----|---------|-----------------------|-----------------|-------|
|-------------|----------|-----|---------|-----------------------|-----------------|-------|

Bulldozer

| Move instructions | | | | | | |
|-------------------------|------------|----|-----|------|-----------|---|
| MOV | r,r | 1 | 1 | 0.5 | EX01 | all addr. modes all addr. modes |
| MOV | r,i | 1 | 1 | 0.5 | EX01 | |
| MOV | r,m | 1 | 4 | 0.5 | AG01 | |
| MOV | m,r | 1 | 4 | 1 | EX01 AG01 | |
| MOV | m,i | 1 | | 1 | | |
| MOVNTI | m,r | 1 | 5 | 2 | | |
| MOVZX, MOVZX | r,r | 1 | 1 | 0.5 | EX01 | |
| MOVZX | r,m | 1 | 5 | 0.5 | EX01 | |
| MOVZX | r,m | 1 | 4 | 0.5 | EX01 | |
| MOVSLD | r64,r32 | 1 | 1 | 0.5 | EX01 | |
| MOVSLD | r64,m32 | 1 | 5 | 0.5 | EX01 | |
| CMOVcc | r,r | 1 | 1 | 0.5 | EX01 | |
| CMOVcc | r,m | 1 | | 0.5 | EX01 | |
| XCHG | r,r | 2 | 1 | 1 | EX01 | |
| XCHG | r,m | 2 | ~50 | ~50 | EX01 | Timing depends on hw |
| XLAT | | 2 | 6 | 2 | | |
| PUSH | r | 1 | | 1 | | |
| PUSH | i | 1 | | 1 | | |
| PUSH | m | 2 | | 1.5 | | |
| PUSHF(D/Q) | | 8 | | 4 | | |
| PUSHA(D) | | 9 | | 9 | | |
| POP | r | 1 | | 1 | | |
| POP | m | 2 | | 1 | | |
| POPF(D/Q) | | 34 | | 19 | | |
| POPA(D) | | 14 | | 8 | | |
| LEA | r16,[m] | 2 | 2-3 | | EX01 | |
| LEA | r32,[m] | 2 | 2-3 | | EX01 | |
| LEA | r32/64,[m] | 1 | 2 | 0.5 | EX01 | any addr. size 16 bit addr. size scale factor > 1 or 3 operands all other cases |
| LEA | r32/64,[m] | 1 | 1 | 0.5 | EX01 | |
| LAHF | | 4 | 3 | 2 | | |
| SAHF | | 2 | 2 | 1 | | |
| SALC | | 1 | 1 | 1 | | |
| BSWAP | r | 1 | 1 | 0.5 | EX01 | |
| PREFETCHNTA | m | 1 | | 0.5 | | |
| PREFETCHT0/1/2 | m | 1 | | 0.5 | | |
| PREFETCH/W | m | 1 | | 0.5 | | |
| SFENCE | | 6 | | 89 | | |
| LFENCE | | 1 | | 0.25 | | |
| MFENCE | | 6 | | 89 | | |
| Arithmetic instructions | | | | | | |
| ADD, SUB | r,r | 1 | 1 | 0.5 | EX01 | AMD 3DNow |
| ADD, SUB | r,i | 1 | 1 | 0.5 | EX01 | |
| ADD, SUB | r,m | 1 | | 0.5 | EX01 | |
| ADD, SUB | m,r | 1 | 7-8 | 1 | EX01 | |
| ADD, SUB | m,i | 1 | 7-8 | 1 | EX01 | |
| ADC, SBB | r,r | 1 | 1 | | EX01 | |
| ADC, SBB | r,i | 1 | 1 | | EX01 | |
| ADC, SBB | r,m | 1 | 1 | 1 | EX01 | |
| ADC, SBB | m,r | 1 | 9 | 1 | EX01 | |

Bulldozer

| | | | | | |
|---------------------------|-------------|----|-------|-------|------|
| ADC, SBB | m,i | 1 | 9 | 1 | EX01 |
| CMP | r,r | 1 | 1 | 0.5 | EX01 |
| CMP | r,i | 1 | 1 | 0.5 | EX01 |
| CMP | r,m | 1 | | 0.5 | EX01 |
| INC, DEC, NEG | r | 1 | 1 | 0.5 | EX01 |
| INC, DEC, NEG | m | 1 | 7-8 | 1 | EX01 |
| AAA, AAS | | 10 | 6 | | |
| DAA | | 16 | 9 | | |
| DAS | | 20 | 10 | | |
| AAD | | 4 | 6 | | |
| AAM | | 9 | 20 | 20 | |
| MUL, IMUL | r8/m8 | 1 | 4 | 2 | EX1 |
| MUL, IMUL | r16/m16 | 2 | 4 | 2 | EX1 |
| MUL, IMUL | r32/m32 | 1 | 4 | 2 | EX1 |
| MUL, IMUL | r64/m64 | 1 | 6 | 4 | EX1 |
| IMUL | r16,r16/m16 | 1 | 4 | 2 | EX1 |
| IMUL | r32,r32/m32 | 1 | 4 | 2 | EX1 |
| IMUL | r64,r64/m64 | 1 | 6 | 4 | EX1 |
| IMUL | r16,(r16),i | 2 | 5 | 2 | EX1 |
| IMUL | r32,(r32),i | 1 | 4 | 2 | EX1 |
| IMUL | r64,(r64),i | 1 | 6 | 4 | EX1 |
| IMUL | r16,m16,i | 2 | | 2 | EX1 |
| IMUL | r32,m32,i | 2 | | 2 | EX1 |
| IMUL | r64,m64,i | 2 | | 4 | EX1 |
| DIV | r8/m8 | 14 | 20 | 20 | EX0 |
| DIV | r16/m16 | 18 | 15-27 | 15-28 | EX0 |
| DIV | r32/m32 | 16 | 16-43 | 16-43 | EX0 |
| DIV | r64/m64 | 16 | 16-75 | 16-75 | EX0 |
| IDIV | r8/m8 | 33 | 23 | 20 | EX0 |
| IDIV | r16/m16 | 36 | 23-33 | 20-27 | EX0 |
| IDIV | r32/m32 | 36 | 22-48 | 20-43 | EX0 |
| IDIV | r64/m64 | 36 | 22-79 | 20-75 | EX0 |
| CBW, CWDE, CDQE | | 1 | 1 | | EX01 |
| CDQ, CQO | | 1 | 1 | 0.5 | EX01 |
| CWD | | 2 | 1 | 1 | EX01 |
| Logic instructions | | | | | |
| AND, OR, XOR | r,r | 1 | 1 | 0.5 | EX01 |
| AND, OR, XOR | r,i | 1 | 1 | 0.5 | EX01 |
| AND, OR, XOR | r,m | 1 | | 0.5 | EX01 |
| AND, OR, XOR | m,r | 1 | 7-8 | 1 | EX01 |
| AND, OR, XOR | m,i | 1 | 7-8 | 1 | EX01 |
| TEST | r,r | 1 | 1 | 0.5 | EX01 |
| TEST | r,i | 1 | 1 | 0.5 | EX01 |
| TEST | m,r | 1 | | 0.5 | EX01 |
| TEST | m,i | 1 | | 0.5 | EX01 |
| NOT | r | 1 | 1 | 0.5 | EX01 |
| NOT | m | 1 | 7 | 1 | EX01 |
| SHL, SHR, SAR | r,i/CL | 1 | 1 | 0.5 | EX01 |
| ROL, ROR | r,i/CL | 1 | 1 | 0.5 | EX01 |
| RCL | r,1 | 1 | 1 | | EX01 |
| RCL | r,i | 16 | 8 | | EX01 |
| RCL | r,cl | 17 | 9 | | EX01 |

Bulldozer

| | | | | | | |
|--------------------------------------|---------------|----|---|-----|------|--------------|
| RCR | r,1 | 1 | 1 | | EX01 | |
| RCR | r,i | 15 | 8 | | EX01 | |
| RCR | r,cl | 16 | 8 | | EX01 | |
| SHLD, SHRD | r,r,i | 6 | 3 | 3 | EX01 | |
| SHLD, SHRD | r,r,cl | 7 | 4 | 3.5 | EX01 | |
| SHLD, SHRD | m,r,i/CL | 8 | | 3.5 | EX01 | |
| BT | r,r/i | 1 | 1 | 0.5 | EX01 | |
| BT | m,i | 1 | | 0.5 | EX01 | |
| BT | m,r | 7 | | 3.5 | EX01 | |
| BTC, BTR, BTS | r,r/i | 2 | 2 | 1 | EX01 | |
| BTC, BTR, BTS | m,i | 4 | | 2 | EX01 | |
| BTC, BTR, BTS | m,r | 10 | | 5 | EX01 | |
| BSF | r,r | 6 | 3 | 3 | EX01 | |
| BSF | r,m | 8 | 4 | 4 | EX01 | |
| BSR | r,r | 7 | 4 | 4 | EX01 | |
| BSR | r,m | 9 | | 5 | EX01 | |
| LZCNT | r,r | 1 | 2 | 2 | EX0 | SSE4.A |
| POPCNT | r,r/m | 1 | 4 | 2 | EX1 | SSE4.2 |
| SETcc | r | 1 | 1 | 0.5 | EX01 | |
| SETcc | m | 1 | | 1 | EX01 | |
| CLC, STC | | 1 | | 0.5 | EX01 | |
| CMC | | 1 | 1 | | EX01 | |
| CLD | | 2 | | 3 | | |
| STD | | 2 | | 4 | | |
| POPCNT | r16/32,r16/32 | 1 | 4 | 2 | | SSE4A |
| POPCNT | r64,r64 | 1 | 4 | 4 | | SSE4A |
| LZCNT | r,r | 2 | 2 | 2 | | SSE4A |
| EXTRQ | x,i,i | 1 | 3 | 1 | P1 | SSE4A |
| EXTRQ | x,x | 1 | 3 | 1 | P1 | SSE4A |
| INSERTQ | x,x,i,i | 1 | 3 | 1 | P1 | SSE4A |
| INSERTQ | x,x | 1 | 3 | 1 | P1 | SSE4A |
| Control transfer instructions | | | | | | |
| JMP | short/near | 1 | | 2 | EX1 | |
| JMP | r | 1 | | 2 | EX1 | |
| JMP | m | 1 | | 2 | EX1 | |
| Jcc | short/near | 1 | | 1-2 | EX1 | 2 if jumping |
| fused CMP+Jcc | short/near | 1 | | 1-2 | EX1 | 2 if jumping |
| J(E/R)CXZ | short | 1 | | 1-2 | EX1 | 2 if jumping |
| LOOP | short | 1 | | 1-2 | EX1 | 2 if jumping |
| LOOPE LOOPNE | short | 1 | | 1-2 | EX1 | 2 if jumping |
| CALL | near | 2 | | 2 | EX1 | |
| CALL | r | 2 | | 2 | EX1 | |
| CALL | m | 3 | | 2 | EX1 | |
| RET | | 1 | | 2 | EX1 | |
| RET | i | 4 | | 2-3 | EX1 | |
| BOUND | m | 11 | | 5 | | for no jump |
| INTO | | 4 | | 24 | | for no jump |
| String instructions | | | | | | |
| LODS | | 3 | | 3 | | |
| REP LODS | | 6n | | 3n | | |
| STOS | | 3 | | 3 | | |

Bulldozer

| | | | | | | |
|------------------------|-------------|-----------|-----|-----------|------|-----------|
| REP STOS | | 2n | | 2n | | small n |
| REP STOS | | 3 per 16B | | 3 per 16B | | best case |
| MOVS | | 5 | | 3 | | |
| REP MOVS | | 2n | | 2n | | small n |
| REP MOVS | | 4 per 16B | | 3 per 16B | | best case |
| SCAS | | 3 | | 3 | | |
| REP SCAS | | 7n | | 4n | | |
| CMPS | | 6 | | 3 | | |
| REP CMPS | | 9n | | 4n | | |
| Synchronization | | | | | | |
| LOCK ADD | m,r | 1 | ~55 | | | |
| XADD | m,r | 4 | 10 | | | |
| LOCK XADD | m,r | 4 | ~51 | | | |
| CMPXCHG | m8,r8 | 5 | 15 | | | |
| LOCK CMPXCHG | m8,r8 | 5 | ~51 | | | |
| CMPXCHG | m,r16/32/64 | 6 | 14 | | | |
| LOCK CMPXCHG | m,r16/32/64 | 6 | ~52 | | | |
| CMPXCHG8B | m64 | 18 | 15 | | | |
| LOCK CMPXCHG8B | m64 | 18 | ~53 | | | |
| CMPXCHG16B | m128 | 22 | 52 | | | |
| LOCK CMPXCHG16B | m128 | 22 | ~94 | | | |
| Other | | | | | | |
| NOP (90) | | 1 | | 0.25 | none | |
| Long NOP (0F 1F) | | 1 | | 0.25 | none | |
| PAUSE | | 40 | | 43 | | |
| ENTER | a,0 | 13 | | 22 | | |
| ENTER | a,b | 11+5b | | 16+4b | | |
| LEAVE | | 2 | | 4 | | |
| CPUID | | 37-63 | | 112-280 | | |
| RDTSC | | 36 | | 42 | | |
| RDPMSR | | 22 | | 300 | | |
| CRC32 | r32,r8 | 3 | 3 | 2 | | |
| CRC32 | r32,r16 | 5 | 5 | 5 | | |
| CRC32 | r32,r32 | 5 | 6 | 6 | | |
| XGETBV | | 4 | | 31 | | |

Floating point x87 instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Domain, notes |
|--------------------------|----------|-----|---------|-----------------------|-----------------|---------------|
| Move instructions | | | | | | |
| FLD | r | 1 | 2 | 0.5 | P01 | fp |
| FLD | m32/64 | 1 | 8 | 1 | | fp |
| FLD | m80 | 8 | 14 | 4 | | fp |
| FBLD | m80 | 60 | 61 | 40 | P0 P1 P2 P3 | fp |
| FST(P) | r | 1 | 2 | 0.5 | P01 | fp |
| FST(P) | m32/64 | 2 | 8 | 1 | | fp |
| FSTP | m80 | 13 | 9 | 20 | | fp |
| FBSTP | m80 | 239 | 240 | 244 | P0 P1 F3 | fp |
| FXCH | r | 1 | 0 | 0.5 | P01 | inherit |
| FILD | m | 1 | 12 | 1 | F3 | fp |

Bulldozer

| | | | | | | |
|--------------------------------|-------|---------|--------|--------|-------------|---------|
| FIST(P) | m | 2 | 8 | 1 | P0 F3 | fp |
| FLDZ, FLD1 | | 1 | | 0.5 | P01 | fp |
| FCMOVcc | st0,r | 8 | 3 | 3 | P0 P1 F3 | fp |
| FFREE | r | 1 | | 0.25 | none | |
| FINCSTP, FDECSTP | | 1 | 0 | 0.25 | none | inherit |
| FNSTSW | AX | 4 | ~13 | 22 | P0 P2 P3 | |
| FNSTSW | m16 | 3 | ~13 | 19 | P0 P2 P3 | |
| FLDCW | m16 | 1 | | 3 | | |
| FNSTCW | m16 | 3 | | 2 | | |
| Arithmetic instructions | | | | | | |
| FADD(P),FSUB(R)(P) | r/m | 1 | 5-6 | 1 | P01 | fma |
| FIADD,FISUB(R) | m | 2 | | 2 | P01 | fma |
| FMUL(P) | r/m | 1 | 5-6 | 1 | P01 | fma |
| FIMUL | m | 2 | | 2 | P01 | fma |
| FDIV(R)(P) | r | 1 | 10-42 | 5-18 | P01 | fp |
| FDIV(R) | m | 2 | | | P01 | fp |
| FIDIV(R) | m | 2 | | | P01 | fp |
| FABS, FCHS | | 1 | 2 | 0.5 | P01 | fp |
| FCOM(P), FUCOM(P) | r/m | 1 | | 0.5 | P01 | fp |
| FCOMPP, FUCOMPP | | 1 | | 0.5 | P01 | fp |
| FCOMI(P) | r | 2 | 2 | 1 | P0 P1 F3 | fp |
| FICOM(P) | m | 2 | | 1 | P01 | fp |
| FTST | | 1 | | 0.5 | P01 | fp |
| FXAM | | 1 | ~20 | 0.5 | P01 | fp |
| FRNDINT | | 1 | 4 | 1 | P0 | fp |
| FPREM | | 1 | 19-62 | | P0 | fp |
| FPREM1 | | 1 | 19-65 | | P0 | fp |
| Math | | | | | | |
| FSQRT | | 1 | 10-53 | | P01 | |
| FLDPI, etc. | | 1 | | 0.5 | P01 | |
| FSIN | | 10-162 | 65-210 | 65-210 | P0 P1 P3 | |
| FCOS | | 160-170 | ~160 | ~160 | P0 P1 P3 | |
| FSINCOS | | 12-166 | 95-160 | 95-160 | P0 P1 P3 | |
| FPTAN | | 11-190 | 95-245 | 95-245 | P0 P1 P3 | |
| FPATAN | | 10-355 | 60-440 | 60-440 | P0 P1 P3 | |
| FSCALE | | 8 | 52 | | P0 P1 P3 | |
| FXTRACT | | 12 | 10 | 5 | P0 P1 P3 | |
| F2XM1 | | 10 | 64-71 | | P0 P1 P3 | |
| FYL2X | | 10-175 | | | P0 P1 P3 | |
| FYL2XP1 | | 10-175 | | | P0 P1 P3 | |
| Other | | | | | | |
| FNOP | | 1 | | 0.25 | none | |
| (F)WAIT | | 1 | | 0.25 | none | |
| FNCLEX | | 18 | | 57 | P0 | |
| FNINIT | | 31 | | 170 | P0 | |
| FNSAVE | m864 | 103 | 300 | 300 | P0 P1 P2 P3 | |
| FRSTOR | m864 | 76 | 312 | 312 | P0 P3 | |

Integer vector instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Notes |
|--------------------------|--------------|-----|---------|-----------------------|-----------------|----------------|
| Move instructions | | | | | | |
| MOVD | r32/64, mm/x | 1 | 8 | 1 | | |
| MOVD | mm/x, r32/64 | 2 | 10 | 1 | | |
| MOVD | mm/x, m32 | 1 | 6 | 0.5 | | |
| MOVD | m32, mm/x | 1 | 5 | 1 | | |
| MOVQ | mm/x, mm/x | 1 | 2 | 0.5 | P23 | |
| MOVQ | mm/x, m64 | 1 | 6 | 0.5 | | |
| MOVQ | m64, mm/x | 1 | 5 | 1 | P3 | |
| MOVDQA | xmm, xmm | 1 | 0 | 0.25 | none | inherit domain |
| MOVDQA | xmm, m | 1 | 6 | 0.5 | | |
| MOVDQA | m, xmm | 1 | 5 | 1 | P3 | |
| VMOVDQA | ymm, ymm | 2 | 2 | 0.5 | P23 | |
| VMOVDQA | ymm, m256 | 2 | 6 | 1 | | |
| VMOVDQA | m256, ymm | 4 | 5 | 3 | P3 | |
| MOVDQU | xmm, xmm | 1 | 0 | 0.25 | none | inherit domain |
| MOVDQU | xmm, m | 1 | 6 | 0.5 | | |
| MOVDQU | m, xmm | 1 | 5 | 1 | P3 | |
| LDDQU | xmm, m | 1 | 6 | 0.5 | | |
| VMOVDQU | ymm, m256 | 2 | 6 | 1-2 | | |
| VMOVDQU | m256, ymm | 8 | 6 | 10 | P2 P3 | |
| MOVDQ2Q | mm, xmm | 1 | 2 | 0.5 | P23 | |
| MOVQ2DQ | xmm, mm | 1 | 2 | 0.5 | P23 | |
| MOVNTQ | m, mm | 1 | 6 | 2 | P3 | |
| MOVNTDQ | m, xmm | 1 | 6 | 2 | P3 | |
| MOVNTDQA | xmm, m | 1 | 6 | 0.5 | | |
| PACKSSWB/DW | mm/x, r/m | 1 | 2 | 1 | P1 | |
| PACKUSWB | mm/x, r/m | 1 | 2 | 1 | P1 | |
| PUNPCKH/LBW/WD/DQ | mm/x, r/m | 1 | 2 | 1 | P1 | |
| PUNPCKHQDQ | xmm, r/m | 1 | 2 | 1 | P1 | |
| PUNPCKLQDQ | xmm, r/m | 1 | 2 | 1 | P1 | |
| PSHUFB | mm/x, r/m | 1 | 3 | 1 | P1 | |
| PSHUFD | xmm, xmm, i | 1 | 2 | 1 | P1 | |
| PSHUFW | mm, mm, i | 1 | 2 | 1 | P1 | |
| PSHUFL/HW | xmm, xmm, i | 1 | 2 | 1 | P1 | |
| PALIGNR | mm/x, r/m, i | 1 | 2 | 1 | P1 | |
| PBLENDW | xmm, r/m | 1 | 2 | 0.5 | P23 | SSE4.1 |
| MASKMOVQ | mm, mm | 31 | 38 | 37 | P3 | |
| MASKMOVDQU | xmm, xmm | 64 | 48 | 61 | P1 P3 | |
| PMOVMASKB | r32, mm/x | 2 | 10 | 1 | P1 P3 | |
| PEXTRB/W/D/Q | r, x/mm, i | 2 | 10 | 1 | P1 P3 | AVX |
| PINSRB/W/D/Q | x/mm, r, i | 2 | 12 | 2 | P1 | |
| PMOVSXBW/BD/BQ/WD/WQ/DQ | xmm, xmm | 1 | 2 | 1 | P1 | SSE4.1 |
| PMOVZXBW/BD/BQ/WD/WQ/DQ | xmm, xmm | 1 | 2 | 1 | P1 | SSE4.1 |
| VPCMOV | x, x, x, x/m | 1 | 2 | 1 | P1 | AMD XOP |
| VPCMOV | y, y, y, y/m | 2 | 2 | 2 | P1 | AMD XOP |
| VPPERM | x, x, x, x/m | 1 | 2 | 1 | P1 | AMD XOP |

Bulldozer

| Arithmetic instructions | | | | | | | |
|---------------------------|-----------|---|---|-----|--------|-------------------------------|--|
| PADDB/W/D/Q/SB/SW/USB/USW | mm/x,r/m | 1 | 2 | 0.5 | P23 | | |
| PSUBB/W/D/Q/SB/SW/USB/USW | mm/x,r/m | 1 | 2 | 0.5 | P23 | | |
| PHADD/SUB(S)W/D | x,x | 3 | 5 | 2 | P1 P23 | SSSE3 | |
| PHADD/SUB(S)W/D | x,m | 4 | 5 | 2 | P1 P23 | SSSE3 | |
| PCMPEQ/GT B/W/D | mm/x,r/m | 1 | 2 | 0.5 | P23 | | |
| PCMPEQQ | mm/x,r/m | 1 | 2 | 0.5 | P23 | SSE4.1 | |
| PCMPGTQ | mm/x,r/m | 1 | 2 | 0.5 | P23 | SSE4.2 | |
| PMULLW PMULHW | | | | | | | |
| PMULHUW PMULUDQ | mm/x,r/m | 1 | 4 | 1 | P0 | | |
| PMULLD | xmm,r/m | 1 | 5 | 2 | P0 | SSE4.1 | |
| PMULDQ | xmm,r/m | 1 | 4 | 1 | P0 | SSE4.1 | |
| PMULHRSW | mm/x,r/m | 1 | 4 | 1 | P0 | SSSE3 | |
| PMADDWD | mm/x,r/m | 1 | 4 | 1 | P0 | | |
| PMADDUBSW | mm/x,r/m | 1 | 4 | 1 | P0 | | |
| PAVGB/W | mm/x,r/m | 1 | 2 | 0.5 | P23 | | |
| PMIN/MAX SB/SW/ SD | | | | | | | |
| UB/UW/UD | mm/x,r/m | 1 | 2 | 0.5 | P23 | | |
| PHMINPOSUW | xmm,r/m | 2 | 4 | 1 | P1 P23 | SSE4.1 | |
| PASB/W/D | mm/x,r/m | 1 | 2 | 0.5 | P23 | SSSE3 | |
| PSIGNB/W/D | mm/x,r/m | 1 | 2 | 0.5 | P23 | SSSE3 | |
| PSADBW | mm/x,r/m | 2 | 4 | 1 | P23 | | |
| MPSADBW | x,x,i | 8 | 8 | 4 | P1 P23 | SSE4.1 | |
| VPCOMB/W/D/Q | x,x,x/m,i | 1 | 2 | 0.5 | P23 | AMD XOP latency 0 if i=6,7 | |
| VPCOMUB/W/D/Q | x,x,x/m,i | 1 | 2 | 0.5 | P23 | AMD XOP latency 0 if i=6,7 | |
| VPHADDBW/BD/BQ/WD/WQ/DQ | x,x/m | 1 | 2 | 0.5 | P23 | AMD XOP | |
| VPHADDUBW/BD/BQ/WD/WQ/DQ | x,x/m | 1 | 2 | 0.5 | P23 | AMD XOP | |
| VPHSUBBW/WD/DQ | x,x/m | 1 | 2 | 0.5 | P23 | AMD XOP | |
| VPMACSWW/WD | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP | |
| VPMACSD | x,x,x/m,x | 1 | 5 | 2 | P0 | AMD XOP | |
| VPMACSDQH/L | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP | |
| VPMACSSWW/WD | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP | |
| VPMACSSD | x,x,x/m,x | 1 | 5 | 2 | P0 | AMD XOP | |
| VPMACSSDQH/L | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP | |
| VPMADCSWD | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP | |
| VPMADCSSWD | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP | |
| Logic | | | | | | | |
| PAND PANDN POR | | | | | | | |
| PXOR | mm/x,r/m | 1 | 2 | 0.5 | P23 | | |
| PSLL/RL W/D/Q | | | | | | | |
| PSRAW/D | mm/x,r/m | 1 | 3 | 1 | P1 | | |
| PSLL/RL W/D/Q | | | | | | | |
| PSRAW/D | mm/x,i | 1 | 2 | 1 | P1 | | |
| PSLLDQ, PSRLDQ | xmm,i | 1 | 2 | 1 | P1 | | |
| PTEST | xmm,r/m | 2 | | 1 | P1 P3 | SSE4.1 | |
| VPROTB/W/D/Q | x,x,x/m | 1 | 3 | 1 | P1 | AMD XOP | |
| VPROTB/W/D/Q | x,x,i | 1 | 2 | 1 | P1 | AMD XOP | |
| VPSHAB/W/D/Q | x,x,x/m | 1 | 3 | 1 | P1 | AMD XOP | |

Bulldozer

| | | | | | | |
|----------------------------|---------|----|----|------|----------|---------|
| VPSHLB/W/D/Q | x,x,x/m | 1 | 3 | 1 | P1 | AMD XOP |
| String instructions | | | | | | |
| PCMPESTRI | x,x,i | 27 | 17 | 10 | P1 P2 P3 | SSE4.2 |
| PCMPESTRM | x,x,i | 27 | 10 | 10 | P1 P2 P3 | SSE4.2 |
| PCMPISTRI | x,x,i | 7 | 14 | 3 | P1 P2 P3 | SSE4.2 |
| PCMPISTRM | x,x,i | 7 | 7 | 4 | P1 P2 P3 | SSE4.2 |
| Encryption | | | | | | |
| PCLMULQDQ | x,x/m,i | 5 | 12 | 7 | P1 | pclmul |
| AESDEC | x,x | 2 | 5 | 2 | P01 | aes |
| AESDECLAST | x,x | 2 | 5 | 2 | P01 | aes |
| AESENC | x,x | 2 | 5 | 2 | P01 | aes |
| AESENCLAST | x,x | 2 | 5 | 2 | P01 | aes |
| AESIMC | x,x | 1 | 5 | 1 | P0 | aes |
| AESKEYGENASSIST | x,x,i | 1 | 5 | 1 | P0 | aes |
| Other | | | | | | |
| EMMS | | 1 | | 0.25 | | |

Floating point XMM and YMM instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Domain, notes |
|--------------------------|-----------|-----|---------|-----------------------|-----------------|----------------|
| Move instructions | | | | | | |
| MOVAPS/D MOVUPS/D | x,x | 1 | 0 | 0.25 | none | inherit domain |
| VMOVAPS/D | y,y | 2 | 2 | 0.5 | P23 | ivec |
| MOVAPS/D MOVUPS/D | x,m128 | 1 | 6 | 0.5 | | |
| VMOVAPS/D | | | | | | |
| VMOVUPS/D | y,m256 | 2 | 6 | 1-2 | | |
| MOVAPS/D MOVUPS/D | m128,x | 1 | 5 | 1 | P3 | |
| VMOVAPS/D | m256,y | 4 | 5 | 3 | P3 | |
| VMOVUPS/D | m256,y | 8 | 6 | 10 | P2 P3 | |
| MOVSS/D | x,x | 1 | 2 | 0.5 | P01 | fp |
| MOVSS/D | x,m32/64 | 1 | 6 | 0.5 | | |
| MOVSS/D | m32/64,x | 1 | 5 | 1 | | |
| MOVHPS/D MOVLPS/D | x,m64 | 1 | 7 | 1 | | |
| MOVHPS/D | m64,x | 2 | 8 | 1 | P1 P3 | |
| MOVLPS/D | m64,x | 1 | 7 | 1 | P3 | |
| MOVLHPS MOVHLPS | x,x | 1 | 2 | 1 | P1 | ivec |
| MOVMSKPS/D | r32,x | 2 | 10 | 1 | P1 P3 | |
| VMOVMSKPS/D | r32,y | | | | | |
| MOVNTPS/D | m128,x | 1 | 6 | 2 | P3 | |
| VMOVNTPS/D | m256,y | | | | | |
| MOVNTSS/SD | m,x | 1 | | 4 | P3 | SSE4A |
| SHUFPS/D | x,x/m,i | 1 | 2 | 1 | P1 | ivec |
| VSHUFPS/D | y,y,y/m,i | 2 | 2 | 2 | P1 | ivec |
| VPERMILPS/PD | x,x,x/m | 1 | 3 | 1 | P1 | ivec |
| VPERMILPS/PD | y,y,y/m | 2 | 3 | 2 | P1 | ivec |

Bulldozer

| | | | | | | |
|-------------------|------------|----|----|-----|-------------|------|
| VPERMILPS/PD | x,x/m,i | 1 | 2 | 1 | P1 | ivec |
| VPERMILPS/PD | y,y/m,i | 2 | 2 | 2 | P1 | ivec |
| VPERM2F128 | y,y,y,i | 8 | 4 | 3 | P23 | ivec |
| VPERM2F128 | y,y,m,i | 10 | | 4 | P23 | ivec |
| BLENDPS/PD | x,x/m,i | 1 | 2 | 0.5 | P23 | ivec |
| VBLENDPS/PD | y,y,y/m,i | 2 | 2 | 1 | P23 | ivec |
| BLENDVPS/PD | x,x/m,xmm0 | 1 | 2 | 1 | P1 | ivec |
| VBLENDVPS/PD | y,y,y/m,y | 2 | 2 | 2 | P1 | ivec |
| MOVDDUP | x,x | 1 | 2 | 1 | P1 | ivec |
| MOVDDUP | x,m64 | 1 | | 0.5 | | |
| VMOVDDUP | y,y | 2 | 2 | 2 | P1 | ivec |
| VMOVDDUP | y,m256 | 2 | | 1 | | |
| VBROADCASTSS | x,m32 | 1 | 6 | 0.5 | | |
| VBROADCASTSS | y,m32 | 2 | 6 | 0.5 | P23 | |
| VBROADCASTSD | y,m64 | 2 | 6 | 0.5 | P23 | |
| VBROADCASTF128 | y,m128 | 2 | 6 | 0.5 | P23 | |
| MOVSH/LDUP | x,x | 1 | 2 | 1 | P1 | ivec |
| MOVSH/LDUP | x,m128 | 1 | | 0.5 | | |
| VMOVSH/LDUP | y,y | 2 | 2 | 2 | P1 | ivec |
| VMOVSH/LDUP | y,m256 | 2 | | 1 | | |
| UNPCKH/LPS/D | x,x/m | 1 | 2 | 1 | P1 | ivec |
| VUNPCKH/LPS/D | y,y,y/m | 2 | 2 | 2 | P1 | ivec |
| EXTRACTPS | r32,x,i | 2 | 10 | 1 | P1 P3 | |
| EXTRACTPS | m32,x,i | 2 | 14 | 1 | P1 P3 | |
| VEXTRACTF128 | x,y,i | 1 | 2 | 1 | P23 | ivec |
| VEXTRACTF128 | m128,y,i | 2 | 7 | 1 | P23 | |
| INSERTPS | x,x,i | 1 | 2 | 1 | P1 | |
| INSERTPS | x,m32,i | 1 | | 1 | P1 | |
| VINSERTF128 | y,y,x,i | 2 | 2 | 1 | P23 | ivec |
| VINSERTF128 | y,y,m128,i | 2 | 9 | 1 | P23 | |
| VMASKMOVPS/D | x,x,m128 | 1 | 9 | 0.5 | P01 | |
| VMASKMOVPS/D | y,y,m256 | 2 | 9 | 1 | P01 | |
| VMASKMOVPS/D | m128,x,x | 18 | 22 | 7 | P0 P1 P2 P3 | |
| VMASKMOVPS/D | m256,y,y | 34 | 25 | 13 | P0 P1 P2 P3 | |
| Conversion | | | | | | |
| CVTPD2PS | x,x | 2 | 7 | 1 | P01 | fp |
| VCVTPD2PS | x,y | 4 | 7 | 2 | P01 | fp |
| CVTPS2PD | x,x | 2 | 7 | 1 | P01 | fp |
| VCVTPS2PD | y,x | 4 | 7 | 2 | P01 | fp |
| CVTSD2SS | x,x | 1 | 4 | 1 | P0 | fp |
| CVTSS2SD | x,x | 1 | 4 | 1 | P0 | fp |
| CVTDQ2PS | x,x | 1 | 4 | 1 | P0 | fp |
| VCVTDQ2PS | y,y | 2 | 4 | 2 | P0 | fp |
| CVT(T) PS2DQ | x,x | 1 | 4 | 1 | P0 | fp |
| VCVT(T) PS2DQ | y,y | 2 | 4 | 2 | P0 | fp |
| CVTDQ2PD | x,x | 2 | 7 | 1 | P01 | fp |
| VCVTDQ2PD | y,x | 4 | 8 | 2 | P01 | fp |
| CVT(T)PD2DQ | x,x | 2 | 7 | 1 | P01 | fp |
| VCVT(T)PD2DQ | x,y | 4 | 7 | 2 | P01 | fp |
| CVTPI2PS | x,mm | 1 | 4 | 1 | P0 | fp |
| CVT(T)PS2PI | mm,x | 1 | 4 | 1 | P0 | fp |
| CVTPI2PD | x,mm | 2 | 7 | 1 | P0 P1 | fp |

Bulldozer

| | | | | | | |
|--|-----------|----|------|---------|---------|----------|
| CVT(T) PD2PI | mm,x | 2 | 7 | 1 | P0 P1 | fp |
| CVTSI2SS | x,r32 | 2 | 14 | 1 | P0 | fp |
| CVT(T)SS2SI | r32,x | 2 | 13 | 1 | P0 | fp |
| CVTSI2SD | x,r32/64 | 2 | 14 | 1 | P0 | fp |
| CVT(T)SD2SI | r32/64,x | 2 | 13 | 1 | P0 | fp |
| Arithmetic | | | | | | |
| ADDSS/D SUBSS/D | x,x/m | 1 | 5-6 | 0.5 | P01 | fma |
| ADDPSS/D SUBPSS/D | x,x/m | 1 | 5-6 | 0.5 | P01 | fma |
| VADDPSS/D VSUBPSS/D | y,y,y/m | 2 | 5-6 | 1 | P01 | fma |
| ADDSUBPSS/D | x,x/m | 1 | 5-6 | 0.5 | P01 | fma |
| VADDSUBPSS/D | y,y,y/m | 2 | 5-6 | 1 | P01 | fma |
| HADDPSS/D HSUBPSS/D | x,x | 3 | 10 | 2 | P01 P1 | ivec/fma |
| HADDPSS/D HSUBPSS/D | x,m128 | 4 | | 2 | P01 P1 | ivec/fma |
| VHADDPSS/D | | | | | | |
| VHSUBPSS/D | y,y,y | 8 | 10 | 4 | P01 P1 | ivec/fma |
| VHADDPSS/D | | | | | | |
| VHSUBPSS/D | y,y,m | 10 | | 4 | P01 P1 | ivec/fma |
| MULSS MULSD | x,x/m | 1 | 5-6 | 0.5 | P01 | fma |
| MULPSS MULPD | x,x/m | 1 | 5-6 | 0.5 | P01 | fma |
| VMULPSS VMULPD | y,y,y/m | 2 | 5-6 | 1 | P01 | fma |
| DIVSS DIVPSS | x,x/m | 1 | 9-24 | 4.5-9.5 | P01 | fp |
| VDIVPSS | y,y,y/m | 2 | 9-24 | 9-19 | P01 | fp |
| DIVSD DIVPD | x,x/m | 1 | 9-27 | 4.5-11 | P01 | fp |
| VDIVPD | y,y,y/m | 2 | 9-27 | 9-22 | P01 | fp |
| RCPSS/PS | x,x/m | 1 | 5 | 1 | P01 | fp |
| VRCPSS | y,y/m | 2 | 5 | 2 | P01 | fp |
| CMPSS/D | | | | | | |
| CMPSS/D | x,x/m | 1 | 2 | 0.5 | P01 | fp |
| VCMPSS/D | y,y,y/m | 2 | 2 | 1 | P01 | fp |
| COMISS/D UCOMISS/D | x,x/m | 2 | | 1 | P01 P3 | fp |
| MAXSS/SD/PS/PD | | | | | | |
| MINSS/SD/PS/PD | x,x/m | 1 | 2 | 0.5 | P01 | fp |
| VMAXPSS/D VMINPSS/D | y,y,y/m | 2 | 2 | 1 | P01 | fp |
| ROUNDSS/SD/PS/PD | x,x/m,i | 1 | 4 | 1 | P0 | fp |
| VROUNDSS/SD/PS/PD | | | | | | |
| PD | y,y/m,i | 2 | 4 | 2 | P0 | fp |
| DPPS | x,x,i | 16 | 25 | 6 | P01 P23 | fma |
| DPPS | x,m128,i | 18 | | 7 | P01 P23 | fma |
| VDPPS | y,y,y,i | 25 | 27 | 13 | P01 P3 | fma |
| VDPPS | y,m256,i | 29 | | 13 | P01 P3 | fma |
| DPPD | x,x,i | 15 | 15 | 5 | P01 P23 | fma |
| DPPD | x,m128,i | 17 | | 6 | P01 P23 | fma |
| VFMADDSS/SD | x,x,x,x/m | 1 | 5-6 | 0.5 | P01 | AMD FMA4 |
| VFMADDPSS/PD | x,x,x,x/m | 1 | 5-6 | 0.5 | P01 | AMD FMA4 |
| VFMADDPSS/PD | y,y,y,y/m | 2 | 5-6 | 1 | P01 | AMD FMA4 |
| All other FMA4 instructions: same as above | | | | | | AMD FMA4 |
| Math | | | | | | |

Bulldozer

| | | | | | | |
|---------------------------|---------|-----|-------|----------|-------------|-------------|
| SQRTSS/PS | x,x/m | 1 | 14-15 | 4.5-12 | P01 | fp |
| VSQRTPS | y,y/m | 2 | 14-15 | 9-24 | P01 | fp |
| SQRTSD/PD | x,x/m | 1 | 24-26 | 4.5-16.5 | P01 | fp |
| VSQRTPD | y,y/m | 2 | 24-26 | 9-33 | P01 | fp |
| RSQRTSS/PS | x,x/m | 1 | 5 | 1 | P01 | fp |
| VRSQRTPS | y,y/m | 2 | 5 | 2 | P01 | fp |
| VFRCZSS/SD/PS/PD | x,x | 2 | 10 | 2 | P01 | AMD XOP |
| VFRCZSS/SD/PS/PD | x,m | 3 | 10 | 2 | P01 | AMD XOP |
| Logic | | | | | | |
| AND/ANDN/OR/XORPS/ PD | x,x/m | 1 | 2 | 0.5 | P23 | ivec |
| VAND/ANDN/OR/ XORPS/PD | y,y,y/m | 2 | 2 | 1 | P23 | ivec |
| Other | | | | | | |
| VZEROUPPER | | 9 | | 4 | | 32 bit mode |
| VZEROUPPER | | 16 | | 5 | | 64 bit mode |
| VZEROALL | | 17 | | 6 | P2 P3 | 32 bit mode |
| VZEROALL | | 32 | | 10 | P2 P3 | 64 bit mode |
| LDMXCSR | m32 | 1 | 10 | 4 | P0 P3 | |
| STMXCSR | m32 | 2 | 19 | 19 | P0 P3 | |
| FXSAVE | m4096 | 67 | 136 | 136 | P0 P1 P2 P3 | |
| FXRSTOR | m4096 | 116 | 176 | 176 | P0 P1 P2 P3 | |
| XSAVE | m | 122 | 196 | 196 | P0 P1 P2 P3 | |
| XRSTOR | m | 177 | 250 | 250 | P0 P1 P2 P3 | |

AMD Piledriver

List of instruction timings and macro-operation breakdown

Explanation of column headings:

| | |
|-------------------------------|--|
| Instruction: | Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc. |
| Operands: | i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, x = 128 bit xmm register, y = 256 bit ymm register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc. |
| Ops: | Number of macro-operations issued from instruction decoder to schedulers. Instructions with more than 2 macro-operations use microcode. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latency listed does not include the memory operand where the listing for register and memory operand are joined (r/m). |
| Reciprocal throughput: | This is also called issue latency. This value indicates the average number of clock cycles from the execution of an instruction begins to a subsequent independent instruction of the same kind can begin to execute. A value of 1/3 indicates that the execution units can handle 3 instructions per clock cycle in one thread. However, the throughput may be limited by other bottlenecks in the pipeline. |
| Execution pipe: | Indicates which execution pipe or unit is used for the macro-operations: Integer pipes: EX0: integer ALU, division EX1: integer ALU, multiplication, jump EX01: can use either EX0 or EX1 AG01: address generation unit 0 or 1 Floating point and vector pipes: P0: floating point add, mul, div, convert, shuffle, shift P1: floating point add, mul, div, shuffle, shift P2: move, integer add, boolean P3: move, integer add, boolean, store P01: can use either P0 or P1 P23: can use either P2 or P3 Two macro-operations can execute simultaneously if they go to different execution pipes |
| Domain: | Tells which execution unit domain is used: ivec: integer vector execution unit. fp: floating point execution unit. fma: floating point multiply/add subunit. inherit: the output operand inherits the domain of the input operand. ivec/fma means the input goes to the ivec domain and the output comes from the fma domain. There is an additional latency of 1 clock cycle if the output of an ivec instruction goes to the input of a fp or fma instruction, and when the output of a fp or fma instruction goes to the input of an ivec or store instruction. There is no latency between the fp and fma units. All other latencies after memory load and before memory store instructions are included in the latency counts. An fma instruction has a latency of 5 if the output goes to another fma instruction, 6 if the output goes to an fp instruction, and 6+1 if the output goes to an ivec or store instruction. |

Integer instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Notes |
|-------------|----------|-----|---------|-----------------------|-----------------|-------|
|-------------|----------|-----|---------|-----------------------|-----------------|-------|

Piledriver

| Move instructions | | | | | | |
|-------------------------|------------|----|-----|------|--------------|---|
| MOV | r8,r8 | 1 | 1 | 0.5 | EX01 | all addr. modes all addr. modes |
| MOV | r16,r16 | 1 | 1 | 0.5 | EX01 | |
| MOV | r32,r32 | 1 | 1 | 0.3 | EX01 or AG01 | |
| MOV | r64,r64 | 1 | 1 | 0.3 | EX01 or AG01 | |
| MOV | r,i | 1 | 1 | 0.5 | EX01 | |
| MOV | r,m | 1 | 4 | 0.5 | AG01 | |
| MOV | m,r | 1 | 4 | 1 | EX01 AG01 | |
| MOV | m,i | 1 | | 1 | | |
| MOVNTI | m,r | 1 | 4 | 2 | | |
| MOVZX, MOVZX | r16,r8 | 1 | 1 | 1 | EX01 | |
| MOVZX, MOVZX | r32,r | 1 | 1 | 0.5 | EX01 | |
| MOVZX, MOVZX | r64,r | 1 | 1 | 0.5 | EX01 | |
| MOVZX | r,m | 1 | 5 | 0.5 | EX01 | |
| MOVZX | r,m | 1 | 4 | 0.5 | EX01 | |
| MOVSLD | r64,r32 | 1 | 1 | 0.5 | EX01 | Timing depends on hw |
| MOVSLD | r64,m32 | 1 | 5 | 0.5 | EX01 | |
| CMOVcc | r,r | 1 | 1 | 0.5 | EX01 | |
| CMOVcc | r,m | 1 | | 0.5 | EX01 | |
| XCHG | r8,r8 | 2 | 1 | 1 | EX01 | |
| XCHG | r16,r16 | 2 | 1 | 1 | EX01 | |
| XCHG | r32,r32 | 2 | 1 | 0.5 | EX01 | |
| XCHG | r64,r64 | 2 | 1 | 0.5 | EX01 | |
| XCHG | r,m | 2 | ~40 | ~40 | EX01 | |
| XLAT | | 2 | 6 | 2 | | |
| PUSH | r | 1 | | 1 | | |
| PUSH | i | 1 | | 1 | | |
| PUSH | m | 2 | | 1 | | |
| PUSHF(D/Q) | | 8 | | 4 | | any addr. size 16 bit addr. size scale factor > 1 or 3 operands all other cases |
| PUSHA(D) | | 9 | | 9 | | |
| POP | r | 1 | | 1 | | |
| POP | m | 2 | | 1 | | |
| POPF(D/Q) | | 34 | | 18 | | |
| POPA(D) | | 14 | | 8 | | |
| LEA | r16,[m] | 2 | 2-3 | | EX01 | |
| LEA | r32,[m] | 2 | 2-3 | | EX01 | |
| LEA | r32/64,[m] | 1 | 2 | 0.5 | EX01 | |
| LEA | r32/64,[m] | 1 | 1 | 0.5 | EX01 | |
| LAHF | | 4 | 3 | 2 | | |
| SAHF | | 2 | 2 | 1 | | |
| SALC | | 1 | 1 | 1 | | |
| BSWAP | r | 1 | 1 | 0.5 | EX01 | PREFETCHW |
| PREFETCHNTA | m | 1 | | 0.5 | | |
| PREFETCHT0/1/2 | m | 1 | | 0.5 | | |
| PREFETCH/W | m | 1 | | 0.5 | | |
| SFENCE | | 7 | | 81 | | |
| LFENCE | | 1 | | 0.25 | | |
| MFENCE | | 7 | | 81 | | |
| Arithmetic instructions | | | | | | |
| ADD, SUB | r,r | 1 | 1 | 0.5 | EX01 | |

Piledriver

| | | | | | |
|---------------------------|-------------|----|-------|-------|------|
| ADD, SUB | r,i | 1 | 1 | 0.5 | EX01 |
| ADD, SUB | r,m | 1 | | 0.5 | EX01 |
| ADD, SUB | m,r | 1 | 7-8 | 1 | EX01 |
| ADD, SUB | m,i | 1 | 7-8 | 1 | EX01 |
| ADC, SBB | r,r | 1 | 1 | | EX01 |
| ADC, SBB | r,i | 1 | 1 | | EX01 |
| ADC, SBB | r,m | 1 | 1 | 1 | EX01 |
| ADC, SBB | m,r | 1 | 9 | 1 | EX01 |
| ADC, SBB | m,i | 1 | 9 | 1 | EX01 |
| CMP | r,r | 1 | 1 | 0.5 | EX01 |
| CMP | r,i | 1 | 1 | 0.5 | EX01 |
| CMP | r,m | 1 | | 0.5 | EX01 |
| CMP | m,i | 1 | | 0.5 | EX01 |
| INC, DEC, NEG | r | 1 | 1 | 0.5 | EX01 |
| INC, DEC, NEG | m | 1 | 7-8 | 1 | EX01 |
| AAA, AAS | | 10 | 6 | | |
| DAA | | 16 | 9 | | |
| DAS | | 20 | 10 | | |
| AAD | | 4 | 6 | | |
| AAM | | 10 | 15 | 15 | |
| MUL, IMUL | r8/m8 | 1 | 4 | 2 | EX1 |
| MUL, IMUL | r16/m16 | 2 | 4 | 2 | EX1 |
| MUL, IMUL | r32/m32 | 1 | 4 | 2 | EX1 |
| MUL, IMUL | r64/m64 | 1 | 6 | 4 | EX1 |
| IMUL | r16,r16/m16 | 1 | 4 | 2 | EX1 |
| IMUL | r32,r32/m32 | 1 | 4 | 2 | EX1 |
| IMUL | r64,r64/m64 | 1 | 6 | 4 | EX1 |
| IMUL | r16,(r16),i | 2 | 5 | 2 | EX1 |
| IMUL | r32,(r32),i | 1 | 4 | 2 | EX1 |
| IMUL | r64,(r64),i | 1 | 6 | 4 | EX1 |
| IMUL | r16,m16,i | 2 | | 2 | EX1 |
| IMUL | r32,m32,i | 2 | | 2 | EX1 |
| IMUL | r64,m64,i | 2 | | 4 | EX1 |
| DIV | r8/m8 | 9 | 17-22 | 13-22 | EX0 |
| DIV | r16/m16 | 7 | 13-26 | 13-25 | EX0 |
| DIV | r32/m32 | 2 | 12-40 | 12-40 | EX0 |
| DIV | r64/m64 | 2 | 13-71 | 13-71 | EX0 |
| IDIV | r8/m8 | 9 | 17-21 | 13-18 | EX0 |
| IDIV | r16/m16 | 7 | 13-26 | 13-25 | EX0 |
| IDIV | r32/m32 | 2 | 13-40 | 13-40 | EX0 |
| IDIV | r64/m64 | 2 | 13-71 | 13-71 | EX0 |
| CBW, CWDE, CDQE | | 1 | 1 | | EX01 |
| CDQ, CQO | | 1 | 1 | 0.5 | EX01 |
| CWD | | 2 | 1 | 1 | EX01 |
| Logic instructions | | | | | |
| AND, OR, XOR | r,r | 1 | 1 | 0.5 | EX01 |
| AND, OR, XOR | r,i | 1 | 1 | 0.5 | EX01 |
| AND, OR, XOR | r,m | 1 | | 0.5 | EX01 |
| AND, OR, XOR | m,r | 1 | 7-8 | 1 | EX01 |
| AND, OR, XOR | m,i | 1 | 7-8 | 1 | EX01 |
| TEST | r,r | 1 | 1 | 0.5 | EX01 |
| TEST | r,i | 1 | 1 | 0.5 | EX01 |

Piledriver

| | | | | | | |
|--------------------------------------|---------------|----|-----|------|------|---------|
| TEST | m,r | 1 | | 0.5 | EX01 | |
| TEST | m,i | 1 | | 0.5 | EX01 | |
| NOT | r | 1 | 1 | 0.5 | EX01 | |
| NOT | m | 1 | 7-8 | 1 | EX01 | |
| ANDN | r,r,r | 1 | 1 | 0.5 | EX01 | BMI1 |
| SHL, SHR, SAR | r,i/CL | 1 | 1 | 0.5 | EX01 | |
| ROL, ROR | r,i/CL | 1 | 1 | 0.5 | EX01 | |
| RCL | r,1 | 1 | 1 | | EX01 | |
| RCL | r,i | 16 | 7 | | EX01 | |
| RCL | r,cl | 17 | 7 | | EX01 | |
| RCR | r,1 | 1 | 1 | | EX01 | |
| RCR | r,i | 15 | 7 | | EX01 | |
| RCR | r,cl | 16 | 6 | | EX01 | |
| SHLD, SHRD | r,r,i | 6 | 3 | 3 | EX01 | |
| SHLD, SHRD | r,r,cl | 7 | 3 | 3 | EX01 | |
| SHLD, SHRD | m,r,i/CL | 8 | | 3.5 | EX01 | |
| BT | r,r/i | 1 | 1 | 0.5 | EX01 | |
| BT | m,i | 1 | | 0.5 | EX01 | |
| BT | m,r | 7 | | 4 | EX01 | |
| BTC, BTR, BTS | r,r/i | 2 | 2 | 1 | EX01 | |
| BTC, BTR, BTS | m,i | 4 | 20 | | EX01 | |
| BTC, BTR, BTS | m,r | 10 | 21 | | EX01 | |
| BSF | r,r | 6 | 3 | 3 | EX01 | |
| BSF | r,m | 8 | 4 | 4 | EX01 | |
| BSR | r,r | 7 | 4 | 4 | EX01 | |
| BSR | r,m | 9 | | 5 | EX01 | |
| SETcc | r | 1 | 1 | 0.5 | EX01 | |
| SETcc | m | 1 | | 1 | EX01 | |
| CLC, STC | | 1 | | 0.5 | EX01 | |
| CMC | | 1 | 1 | | EX01 | |
| CLD | | 2 | | 3 | | |
| STD | | 2 | | 4 | | |
| POPCNT | r16/32,r16/32 | 1 | 4 | 2 | | SSE4.2 |
| POPCNT | r64,r64 | 1 | 4 | 4 | | SSE4.2 |
| LZCNT | r,r | 1 | 2 | 2 | EX0 | LZCNT |
| TZCNT | r,r | 2 | 2 | 2 | | BMI1 |
| BEXTR | r,r,r | 2 | 2 | 0.67 | | BMI1 |
| BEXTR | r,r,i | 2 | 2 | 0.67 | | AMD TBM |
| BLSI | r,r | 2 | 2 | 1 | | BMI1 |
| BLSMSK | r,r | 2 | 2 | 1 | | BMI1 |
| BLSR | r,r | 2 | 2 | 1 | | BMI1 |
| BLCFILL | r,r | 2 | 2 | 1 | | AMD TBM |
| BLCI | r,r | 2 | 2 | 1 | | AMD TBM |
| BLCIC | r,r | 2 | 2 | 1 | | AMD TBM |
| BLCMSK | r,r | 2 | 2 | 1 | | AMD TBM |
| BLCS | r,r | 2 | 2 | 1 | | AMD TBM |
| BLSFILL | r,r | 2 | 2 | 1 | | AMD TBM |
| BLSI | r,r | 2 | 2 | 1 | | AMD TBM |
| BLSIC | r,r | 2 | 2 | 1 | | AMD TBM |
| T1MSKC | r,r | 2 | 2 | 1 | | AMD TBM |
| TZMSK | r,r | 2 | 2 | 1 | | AMD TBM |
| Control transfer instructions | | | | | | |

Piledriver

| | | | | | | |
|----------------------------|------------|------------|-----|-----------|------|--------------|
| JMP | short/near | 1 | | 2 | EX1 | |
| JMP | r | 1 | | 2 | EX1 | |
| JMP | m | 1 | | 2 | EX1 | |
| Jcc | short/near | 1 | | 1-2 | EX1 | 2 if jumping |
| fused CMP+Jcc | short/near | 1 | | 1-2 | EX1 | 2 if jumping |
| J(E/R)CXZ | short | 1 | | 1-2 | EX1 | 2 if jumping |
| LOOP | short | 1 | | 1-2 | EX1 | 2 if jumping |
| LOOPE LOOPNE | short | 1 | | 1-2 | EX1 | 2 if jumping |
| CALL | near | 2 | | 2 | EX1 | |
| CALL | r | 2 | | 2 | EX1 | |
| CALL | m | 3 | | 2 | EX1 | |
| RET | | 1 | | 2 | EX1 | |
| RET | i | 4 | | 2 | EX1 | |
| BOUND | m | 11 | | 5 | | for no jump |
| INTO | | 4 | | 2 | | for no jump |
| String instructions | | | | | | |
| LODS | | 3 | | 3 | | |
| REP LODS | m8/m16 | 6n | | 3n | | |
| REP LODS | m32/m64 | 6n | | 2.5n | | |
| STOS | | 3 | | 3 | | |
| REP STOS | | 1n | | 1n | | small n |
| REP STOS | | 3 per 16B | | 3 per 16B | | best case |
| MOVS | | 5 | | 3 | | |
| REP MOVS | | 1-3n | | 1n | | small n |
| REP MOVS | | 4.5 pr 16B | | 3 per 16B | | best case |
| SCAS | | 3 | | 3 | | |
| REP SCAS | | 7n | | 3-4n | | |
| CMPS | | 6 | | 3 | | |
| REP CMPS | | 9n | | 4n | | |
| Synchronization | | | | | | |
| LOCK ADD | m,r | 1 | ~40 | | | |
| XADD | m,r | 4 | 20 | | | |
| LOCK XADD | m,r | 4 | ~39 | | | |
| CMPXCHG | m,r8/16 | 5 | 23 | | | |
| LOCK CMPXCHG | m,r8/16 | 5 | ~40 | | | |
| CMPXCHG | m,r32/64 | 6 | 20 | | | |
| LOCK CMPXCHG | m,r32/64 | 6 | ~40 | | | |
| CMPXCHG8B | m64 | 18 | 25 | | | |
| LOCK CMPXCHG8B | m64 | 18 | ~42 | | | |
| CMPXCHG16B | m128 | 22 | 66 | | | |
| LOCK CMPXCHG16B | m128 | 22 | ~80 | | | |
| Other | | | | | | |
| NOP (90) | | 1 | | 0.25 | none | |
| Long NOP (0F 1F) | | 1 | | 0.25 | none | |
| PAUSE | | 40 | | 40 | | |
| ENTER | a,0 | 13 | | 21 | | |
| ENTER | a,b | 20+3b | | 16+4b | | |
| LEAVE | | 2 | | 4 | | |
| CPUID | | 38-64 | | 105-271 | | |
| XGETBV | | 4 | | 30 | | |

Piledriver

| | | | | | |
|-------|---------|----|---|-----|--|
| RDTSC | | 36 | | 42 | |
| RDPMC | | 21 | | 310 | |
| CRC32 | r32,r8 | 3 | 3 | 2 | |
| CRC32 | r32,r16 | 5 | 5 | 5 | |
| CRC32 | r32,r32 | 5 | 6 | 6 | |

Floating point x87 instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Domain, notes |
|--------------------------------|----------|-----|---------|-----------------------|-----------------|---------------|
| Move instructions | | | | | | |
| FLD | r | 1 | 2 | 0.5 | P01 | fp |
| FLD | m32/64 | 1 | 7 | 1 | | fp |
| FLD | m80 | 8 | 20 | 4 | | fp |
| FBLD | m80 | 60 | 64 | 35 | P0 P1 P2 P3 | fp |
| FST(P) | r | 1 | 2 | 0.5 | P01 | fp |
| FST(P) | m32/64 | 2 | 7 | 1 | | fp |
| FSTP | m80 | 13 | 22 | 20 | | fp |
| FBSTP | m80 | 239 | 220 | | P0 P1 F3 | fp |
| FXCH | r | 1 | 0 | 0.5 | P01 | inherit |
| FILD | m | 1 | 11 | 1 | F3 | fp |
| FIST(T)(P) | m | 2 | 7 | 1 | P0 F3 | fp |
| FLDZ, FLD1 | | 1 | | 0.5 | P01 | fp |
| FCMOVcc | st0,r | 8 | 3 | 3 | P0 P1 F3 | fp |
| FFREE | r | 1 | | 0.25 | none | |
| FINCSTP, FDECSTP | | 1 | 0 | 0.25 | none | inherit |
| FNSTSW | AX | 3 | | 19 | P0 P2 P3 | |
| FNSTSW | m16 | 2 | | 17 | P0 P2 P3 | |
| FLDCW | m16 | 1 | | 3 | | |
| FNSTCW | m16 | 2 | | 2 | | |
| Arithmetic instructions | | | | | | |
| FADD(P),FSUB(R)(P) | r/m | 1 | 5-6 | 1 | P01 | fma |
| FIADD,FISUB(R) | m | 2 | | 2 | P01 | fma |
| FMUL(P) | r/m | 1 | 5-6 | 1 | P01 | fma |
| FIMUL | m | 2 | | 2 | P01 | fma |
| FDIV(R)(P) | r | 1 | 9-40 | 4-16 | P01 | fp |
| FDIV(R) | m | 1 | | | P01 | fp |
| FIDIV(R) | m | 2 | | | P01 | fp |
| FABS, FCHS | | 1 | 2 | 0.5 | P01 | fp |
| FCOM(P), FUCOM(P) | r/m | 1 | | 0.5 | P01 | fp |
| FCOMPP, FUCOMPP | | 1 | | 0.5 | P01 | fp |
| FCOMI(P) | r | 2 | 2 | 1 | P0 P1 F3 | fp |
| FICOM(P) | m | 2 | | 1 | P01 | fp |
| FTST | | 1 | | 0.5 | P01 | fp |
| FXAM | | 1 | ~20 | 0.5 | P01 | fp |
| FRNDINT | | 1 | 4 | 1 | P0 | fp |
| FPREM | | 1 | 17-60 | | P0 | fp |
| FPREM1 | | 1 | 17-60 | | P0 | fp |
| Math | | | | | | |
| FSQRT | | 1 | 14-50 | 5-20 | P01 | |
| FLDPI, etc. | | 1 | | 0.5 | P01 | |

Piledriver

| | | | | | |
|--------------|------|---------|---------|--------|-------------|
| FSIN | | 10-162 | 60-210 | 60-146 | P0 P1 P3 |
| FCOS | | 160-170 | ~154 | ~154 | P0 P1 P3 |
| FSINCOS | | 12-166 | 86-141 | 86-141 | P0 P1 P3 |
| FPTAN | | 11-190 | 166-231 | 86-204 | P0 P1 P3 |
| FPATAN | | 10-355 | 60-352 | 60-352 | P0 P1 P3 |
| FSCALE | | 8 | 44 | 5 | P0 P1 P3 |
| FXTRACT | | 12 | 7 | 5 | P0 P1 P3 |
| F2XM1 | | 10 | 60-73 | | P0 P1 P3 |
| FYL2X | | 10-176 | | | P0 P1 P3 |
| FYL2XP1 | | 10-176 | | | P0 P1 P3 |
| Other | | | | | |
| FNOP | | 1 | | 0.25 | none |
| (F)WAIT | | 1 | | 0.25 | none |
| FNCLEX | | 18 | | 54 | P0 |
| FNINIT | | 31 | | 134 | P0 |
| FNSAVE | m864 | 103 | 300 | 300 | P0 P1 P2 P3 |
| FRSTOR | m864 | 76 | 236 | 236 | P0 P3 |

Integer vector instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Notes |
|--------------------------|--------------|-----|---------|-----------------------|-----------------|----------------|
| Move instructions | | | | | | |
| MOVD | r32/64, mm/x | 1 | 8 | 1 | | P3 |
| MOVD | mm/x, r32/64 | 2 | 10 | 1 | | |
| MOVD | mm/x, m32 | 1 | 6 | 0.5 | | |
| MOVD | m32, mm/x | 1 | 5 | 1 | | P3 |
| MOVQ | mm/x, mm/x | 1 | 2 | 0.5 | P23 | |
| MOVQ | mm/x, m64 | 1 | 6 | 0.5 | | |
| MOVQ | m64, mm/x | 1 | 5 | 1 | P3 | |
| MOVDQA | xmm, xmm | 1 | 0 | 0.25 | none | inherit domain |
| MOVDQA | xmm, m | 1 | 6 | 0.5 | | |
| MOVDQA | m, xmm | 1 | 5 | 1 | P3 | |
| VMOVDQA | ymm, ymm | 2 | 2 | 0.5 | P23 | |
| VMOVDQA | ymm, m256 | 2 | 6 | 1 | | |
| VMOVDQA | m256, ymm | 4 | 11 | 17 | P3 | |
| MOVDQU | xmm, xmm | 1 | 0 | 0.25 | none | inherit domain |
| MOVDQU | xmm, m | 1 | 6 | 0.5 | | |
| MOVDQU | m, xmm | 1 | 5 | 1 | P3 | |
| LDDQU | xmm, m | 1 | 6 | 0.5 | | |
| VMOVDQU | ymm, m256 | 2 | 6 | 1 | | |
| VMOVDQU | m256, ymm | 8 | 14 | 20 | P2 P3 | |
| MOVDQ2Q | mm, xmm | 1 | 2 | 0.5 | P23 | |
| MOVQ2DQ | xmm, mm | 1 | 2 | 0.5 | P23 | |
| MOVNTQ | m, mm | 1 | 5 | 2 | P3 | |
| MOVNTDQ | m, xmm | 1 | 5 | 2 | P3 | |
| MOVNTDQA | xmm, m | 1 | 6 | 0.5 | | |
| PACKSSWB/DW | mm/x, r/m | 1 | 2 | 1 | P1 | |
| PACKUSWB | mm/x, r/m | 1 | 2 | 1 | P1 | |
| PUNPCKH/LBW/WD/DQ | mm/x, r/m | 1 | 2 | 1 | P1 | |

Piledriver

| | | | | | | |
|--------------------------------|------------|----|----|-----|--------|-------------------------------|
| PUNPCKHQDQ | xmm,r/m | 1 | 2 | 1 | P1 | |
| PUNPCKLQDQ | xmm,r/m | 1 | 2 | 1 | P1 | |
| PSHUFB | mm/x,r/m | 1 | 3 | 1 | P1 | |
| PSHUFD | xmm,xmm,i | 1 | 2 | 1 | P1 | |
| PSHUFW | mm,mm,i | 1 | 2 | 1 | P1 | |
| PSHUFL/HW | xmm,xmm,i | 1 | 2 | 1 | P1 | |
| PALIGNR | mm/x,r/m,i | 1 | 2 | 1 | P1 | |
| PBLENDW | xmm,r/m | 1 | 2 | 0.5 | P23 | SSE4.1 |
| MASKMOVQ | mm,mm | 31 | 36 | 59 | P3 | |
| MASKMOVDQU | xmm,xmm | 64 | 59 | 92 | P1 P3 | |
| PMOVMSKB | r32,mm/x | 2 | 10 | 1 | P1 P3 | |
| PEXTRB/W/D/Q | r,x/mm,i | 2 | 10 | 1 | P1 P3 | SSE4.1 |
| PINSRB/W/D/Q | x/mm,r,i | 2 | 12 | 2 | P1 | |
| EXTRQ | x,i,i | 1 | 3 | 1 | P1 | AMD SSE4A |
| EXTRQ | x,x | 1 | 1 | 1 | P1 | AMD SSE4A |
| INSERTQ | x,x,i,i | 1 | 1 | 1 | P1 | AMD SSE4A |
| INSERTQ | x,x | 1 | 1 | 1 | P1 | AMD SSE4A |
| PMOVSXBW/BD/BQ/ WD/WQ/DQ | x,x | 1 | 2 | 1 | P1 | SSE4.1 |
| PMOVZXBW/BD/BQ/ WD/WQ/DQ | x,x | 1 | 2 | 1 | P1 | SSE4.1 |
| VPCMOV | x,x,x,x/m | 1 | 2 | 1 | P1 | AMD XOP |
| VPCMOV | y,y,y,y/m | 2 | 2 | 2 | P1 | AMD XOP |
| VPPERM | x,x,x,x/m | 1 | 2 | 1 | P1 | AMD XOP |
| Arithmetic instructions | | | | | | |
| PADDB/W/D/Q/SB/SW/ USB/USW | mm/x,r/m | 1 | 2 | 0.5 | P23 | |
| PSUBB/W/D/Q/SB/SW/ USB/USW | mm/x,r/m | 1 | 2 | 0.5 | P23 | |
| PHADD/SUB(S)W/D | x,x | 3 | 5 | 2 | P1 P23 | SSSE3 |
| PHADD/SUB(S)W/D | x,m | 4 | 5 | 2 | P1 P23 | SSSE3 |
| PCMPEQ/GT B/W/D | mm/x,r/m | 1 | 2 | 0.5 | P23 | |
| PCMPEQQ | mm/x,r/m | 1 | 2 | 0.5 | P23 | SSE4.1 |
| PCMPGTQ | mm/x,r/m | 1 | 2 | 0.5 | P23 | SSE4.2 |
| PMULLW PMULHW | | | | | | |
| PMULHUW PMULUDQ | mm/x,r/m | 1 | 4 | 1 | P0 | |
| PMULLD | x,r/m | 1 | 5 | 2 | P0 | SSE4.1 |
| PMULDQ | x,r/m | 1 | 4 | 1 | P0 | SSE4.1 |
| PMULHRSW | mm/x,r/m | 1 | 4 | 1 | P0 | SSSE3 |
| PMADDWD | mm/x,r/m | 1 | 4 | 1 | P0 | |
| PMADDUBSW | mm/x,r/m | 1 | 4 | 1 | P0 | |
| PAVGB/W | mm/x,r/m | 1 | 2 | 0.5 | P23 | |
| PMIN/MAX SB/SW/ SD UB/UW/UD | mm/x,r/m | 1 | 2 | 0.5 | P23 | |
| PHMINPOSUW | x,r/m | 2 | 4 | 1 | P1 P23 | SSE4.1 |
| PABSB/W/D | mm/x,r/m | 1 | 2 | 0.5 | P23 | SSSE3 |
| PSIGNB/W/D | mm/x,r/m | 1 | 2 | 0.5 | P23 | SSSE3 |
| PSADBW | mm/x,r/m | 2 | 4 | 1 | P23 | |
| MPSADBW | x,x,i | 8 | 8 | 4 | P1 P23 | SSE4.1 |
| VPCOMB/W/D/Q | x,x,x/m,i | 1 | 2 | 0.5 | P23 | AMD XOP latency 0 if i=6,7 |
| VPCOMUB/W/D/Q | x,x,x/m,i | 1 | 2 | 0.5 | P23 | AMD XOP latency 0 if i=6,7 |

Piledriver

| | | | | | | |
|----------------------------|-----------|----|----|------|----------|---------|
| VPHADDBW/BD/BQ/WD/WQ/DQ | x,x/m | 1 | 2 | 0.5 | P23 | AMD XOP |
| VPHADDUBW/BD/BQ/WD/WQ/DQ | x,x/m | 1 | 2 | 0.5 | P23 | AMD XOP |
| VPHSUBBW/WD/DQ | x,x/m | 1 | 2 | 0.5 | P23 | AMD XOP |
| VPMACSWW/WD | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP |
| VPMACSD | x,x,x/m,x | 1 | 5 | 2 | P0 | AMD XOP |
| VPMACSDQH/L | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP |
| VPMACSSWW/WD | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP |
| VPMACSSD | x,x,x/m,x | 1 | 5 | 2 | P0 | AMD XOP |
| VPMACSSDQH/L | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP |
| VPMADCSWD | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP |
| VPMADCSSWD | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP |
| Logic | | | | | | |
| PAND PANDN POR PXOR | mm/x,r/m | 1 | 2 | 0.5 | P23 | |
| PSLL/RL W/D/Q | | | | | | |
| PSRAW/D | mm/x,r/m | 1 | 3 | 1 | P1 | |
| PSLL/RL W/D/Q | | | | | | |
| PSRAW/D | mm/x,i | 1 | 2 | 1 | P1 | |
| PSLLDQ, PSRLDQ | x,i | 1 | 2 | 1 | P1 | |
| PTEST | x,r/m | 2 | | 1 | P1 P3 | SSE4.1 |
| VPROTB/W/D/Q | x,x,x/m | 1 | 3 | 1 | P1 | AMD XOP |
| VPROTB/W/D/Q | x,x,i | 1 | 2 | 1 | P1 | AMD XOP |
| VPSHAB/W/D/Q | x,x,x/m | 1 | 3 | 1 | P1 | AMD XOP |
| VPSHLB/W/D/Q | x,x,x/m | 1 | 3 | 1 | P1 | AMD XOP |
| String instructions | | | | | | |
| PCMPESTRI | x,x,i | 27 | 16 | 10 | P1 P2 P3 | SSE4.2 |
| PCMPESTRM | x,x,i | 27 | 10 | 10 | P1 P2 P3 | SSE4.2 |
| PCMPISTRI | x,x,i | 7 | 13 | 3 | P1 P2 P3 | SSE4.2 |
| PCMPISTRM | x,x,i | 7 | 7 | 4 | P1 P2 P3 | SSE4.2 |
| Encryption | | | | | | |
| PCLMULQDQ | x,x/m,i | 5 | 12 | 7 | P1 | pclmul |
| VPCLMULQDQ | x,x,x,i | 6 | 12 | 7 | P1 | pclmul |
| PCLMULQDQ | x,x,m,i | 7 | 12 | 7 | P1 | pclmul |
| AESDEC | x,x | 2 | 5 | 2 | P01 | aes |
| AESDECLAST | x,x | 2 | 5 | 2 | P01 | aes |
| AESENC | x,x | 2 | 5 | 2 | P01 | aes |
| AESENC LAST | x,x | 2 | 5 | 2 | P01 | aes |
| AESIMC | x,x | 1 | 5 | 1 | P0 | aes |
| AESKEYGENASSIST | x,x,i | 1 | 5 | 1 | P0 | aes |
| Other | | | | | | |
| EMMS | | 1 | | 0.25 | | |

Floating point XMM and YMM instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Domain, notes |
|--------------------------|----------|-----|---------|-----------------------|-----------------|---------------|
| Move instructions | | | | | | |

Piledriver

| | | | | | | |
|-------------------|------------|----|----|------|-------|----------------|
| MOVAPS/D MOVUPS/D | x,x | 1 | 0 | 0.25 | none | inherit domain |
| VMOVAPS/D | y,y | 2 | 2 | 0.5 | P23 | ivec |
| MOVAPS/D MOVUPS/D | x,m128 | 1 | 6 | 0.5 | | |
| VMOVAPS/D | | | | | | |
| VMOVUPS/D | y,m256 | 2 | 6 | 1 | | |
| MOVAPS/D MOVUPS/D | m128,x | 1 | 5 | 1 | P3 | |
| VMOVAPS/D | m256,y | 4 | 11 | 17 | P3 | |
| VMOVUPS/D | m256,y | 8 | 15 | 20 | P2 P3 | |
| MOVSS/D | x,x | 1 | 2 | 0.5 | P01 | fp |
| MOVSS/D | x,m32/64 | 1 | 6 | 0.5 | | |
| MOVSS/D | m32/64,x | 1 | 5 | 1 | | |
| MOVHPS/D | x,m64 | 1 | 8 | 1 | P1 | |
| MOVLPS/D | x,m64 | 1 | 7 | 0.5 | P01 | |
| MOVHPS/D | m64,x | 2 | 7 | 1 | P1 P3 | |
| MOVLPS/D | m64,x | 1 | 6 | 1 | P3 | |
| MOVLHPS MOVHLPS | x,x | 1 | 2 | 1 | P1 | ivec |
| MOVMSKPS/D | r32,x | 2 | 10 | 1 | P1 P3 | |
| VMOVMSKPS/D | r32,y | 2 | | 1 | | |
| MOVNTPS/D | m128,x | 1 | 5 | 2 | P3 | |
| VMOVNTPS/D | m256,y | 4 | | 18 | | |
| MOVNTSS/SD | m,x | 1 | | 4 | P3 | AMD SSE4A |
| SHUFPS/D | x,x/m,i | 1 | 2 | 1 | P1 | ivec |
| VSHUFPS/D | y,y,y/m,i | 2 | 2 | 2 | P1 | ivec |
| VPERMILPS/PD | x,x,x/m | 1 | 3 | 1 | P1 | ivec |
| VPERMILPS/PD | y,y,y/m | 2 | 3 | 2 | P1 | ivec |
| VPERMILPS/PD | x,x/m,i | 1 | 2 | 1 | P1 | ivec |
| VPERMILPS/PD | y,y/m,i | 2 | 2 | 2 | P1 | ivec |
| VPERM2F128 | y,y,y,i | 8 | 4 | 3 | P23 | ivec |
| VPERM2F128 | y,y,m,i | 10 | | 4 | P23 | ivec |
| BLENDPS/PD | x,x/m,i | 1 | 2 | 0.5 | P23 | ivec |
| VBLENDPS/PD | y,y,y/m,i | 2 | 2 | 1 | P23 | ivec |
| BLENDVPS/PD | x,x/m,xmm0 | 1 | 2 | 1 | P1 | ivec |
| VBLENDVPS/PD | y,y,y/m,y | 2 | 2 | 2 | P1 | ivec |
| MOVDDUP | x,x | 1 | 2 | 1 | P1 | ivec |
| MOVDDUP | x,m64 | 1 | | 0.5 | | |
| VMOVDDUP | y,y | 2 | 2 | 2 | P1 | ivec |
| VMOVDDUP | y,m256 | 2 | | 1 | | |
| VBROADCASTSS | x,m32 | 1 | 6 | 0.5 | | |
| VBROADCASTSS | y,m32 | 2 | 6 | 0.5 | P23 | |
| VBROADCASTSD | y,m64 | 2 | 6 | 0.5 | P23 | |
| VBROADCASTF128 | y,m128 | 2 | 6 | 0.5 | P23 | |
| MOVSH/LDUP | x,x | 1 | 2 | 1 | P1 | ivec |
| MOVSH/LDUP | x,m128 | 1 | | 0.5 | | |
| VMOVSH/LDUP | y,y | 2 | 2 | 2 | P1 | ivec |
| VMOVSH/LDUP | y,m256 | 2 | | 1 | | |
| UNPCKH/LPS/D | x,x/m | 1 | 2 | 1 | P1 | ivec |
| VUNPCKH/LPS/D | y,y,y/m | 2 | 2 | 2 | P1 | ivec |
| EXTRACTPS | r32,x,i | 2 | | 1 | P1 P3 | |
| EXTRACTPS | m32,x,i | 2 | 6 | 1 | P1 P3 | |
| VEEXTRACTF128 | x,y,i | 1 | 2 | 0.5 | P23 | ivec |
| VEEXTRACTF128 | m128,y,i | 2 | 6 | 1 | P23 | |

Piledriver

| | | | | | | |
|-------------------|------------|----|------|------|-------------|----------|
| INSERTPS | x,x,i | 1 | 2 | 1 | P1 | |
| INSERTPS | x,m32,i | 1 | 7 | 2 | P1 | |
| VINSERTF128 | y,y,x,i | 2 | 2 | 1 | P23 | ivec |
| VINSERTF128 | y,y,m128,i | 2 | 13 | 1 | P23 | |
| VMASKMOVPS/D | x,x,m128 | 1 | 7 | 0.5 | P01 | |
| VMASKMOVPS/D | y,y,m256 | 2 | 13 | 1 | P01 | |
| VMASKMOVPS/D | m128,x,x | 18 | ~100 | ~90 | P0 P1 P2 P3 | |
| VMASKMOVPS/D | m256,y,y | 34 | ~190 | ~180 | P0 P1 P2 P3 | |
| Conversion | | | | | | |
| CVTPD2PS | x,x | 2 | 8 | 1 | P01 | ivec/fp |
| VCVTPD2PS | x,y | 4 | 7 | 2 | P01 | ivec/fp |
| CVTPS2PD | x,x | 2 | 8 | 1 | P01 | ivec/fp |
| VCVTPS2PD | y,x | 4 | 8 | 2 | P01 | ivec/fp |
| CVTSD2SS | x,x | 1 | 4 | 1 | P0 | fp |
| CVTSS2SD | x,x | 1 | 4 | 1 | P0 | fp |
| CVTDQ2PS | x,x | 1 | 4 | 1 | P0 | fp |
| VCVTDQ2PS | y,y | 2 | 4 | 2 | P0 | fp |
| CVT(T) PS2DQ | x,x | 1 | 4 | 1 | P0 | fp |
| VCVT(T) PS2DQ | y,y | 2 | 4 | 2 | P0 | fp |
| CVTDQ2PD | x,x | 2 | 8 | 1 | P01 | ivec/fp |
| VCVTDQ2PD | y,x | 4 | 8 | 2 | P01 | ivec/fp |
| CVT(T)PD2DQ | x,x | 2 | 8 | 1 | P01 | fp/ivec |
| VCVT(T)PD2DQ | x,y | 4 | 7 | 2 | P01 | fp/ivec |
| CVTPI2PS | x,mm | 2 | 8 | 1 | P0 P23 | ivec/fp |
| CVT(T)PS2PI | mm,x | 1 | 4 | 1 | P0 | fp |
| CVTPI2PD | x,mm | 2 | 7 | 1 | P0 P1 | ivec/fp |
| CVT(T) PD2PI | mm,x | 2 | 7 | 1 | P0 P1 | fp/ivec |
| CVTSI2SS | x,r32 | 2 | 13 | 1 | P0 | fp |
| CVT(T)SS2SI | r32,x | 2 | 12 | 1 | P0 P3 | fp |
| CVTSI2SD | x,r32/64 | 2 | 13 | 1 | P0 | fp |
| CVT(T)SD2SI | r32/64,x | 2 | 12 | 1 | P0 P3 | fp |
| VCVTPS2PH | x/m,x,i | 2 | 8 | 2 | P0 P1 | F16C |
| VCVTPS2PH | x/m,y,i | 4 | 8 | 2 | P0 P1 | F16C |
| VCVTPH2PS | x,x/m | 2 | 8 | 2 | P0 P1 | F16C |
| VCVTPH2PS | y,x/m | 4 | 8 | 2 | P0 P1 | F16C |
| Arithmetic | | | | | | |
| ADDSS/D SUBSS/D | x,x/m | 1 | 5-6 | 0.5 | P01 | fma |
| ADDPS/D SUBPS/D | x,x/m | 1 | 5-6 | 0.5 | P01 | fma |
| VADDPS/D VSUBPS/D | y,y,y/m | 2 | 5-6 | 1 | P01 | fma |
| ADDSUBPS/D | x,x/m | 1 | 5-6 | 0.5 | P01 | fma |
| VADDSUBPS/D | y,y,y/m | 2 | 5-6 | 1 | P01 | fma |
| HADDPS/D HSUBPS/D | x,x | 3 | 10 | 2 | P01 P1 | ivec/fma |
| HADDPS/D HSUBPS/D | x,m | 4 | | 2 | P01 P1 | ivec/fma |
| VHADDPS/D | | | | | | |
| VHSUBPS/D | y,y,y/m | 8 | 10 | 4 | P01 P1 | ivec/fma |
| MULSS MULSD | x,x/m | 1 | 5-6 | 0.5 | P01 | fma |
| MULPS MULPD | x,x/m | 1 | 5-6 | 0.5 | P01 | fma |
| VMULPS VMULPD | y,y,y/m | 2 | 5-6 | 1 | P01 | fma |
| DIVSS DIVPS | x,x/m | 1 | 9-24 | 5-10 | P01 | fp |

Piledriver

| | | | | | | |
|--|-----------|----|-------|------|---------|-------------|
| VDIVPS | y,y,y/m | 2 | 9-24 | 9-20 | P01 | fp |
| DIVSD DIVPD | x,x/m | 1 | 9-27 | 5-10 | P01 | fp |
| VDIVPD | y,y,y/m | 2 | 9-27 | 9-18 | P01 | fp |
| RCPSS/PS | x,x/m | 1 | 5 | 1 | P01 | fp |
| VRCPPS | y,y/m | 2 | 5 | 2 | P01 | fp |
| CMPSS/D | | | | | | |
| CMPPS/D | x,x/m | 1 | 2 | 0.5 | P01 | fp |
| VCMPSS/D | y,y,y/m | 2 | 2 | 1 | P01 | fp |
| COMISS/D UCOMISS/ D | x,x/m | 2 | | 1 | P01 P3 | fp |
| MAXSS/SD/PS/PD | | | | | | |
| MINSS/SD/PS/PD | x,x/m | 1 | 2 | 0.5 | P01 | fp |
| VMAXPS/D VMINPS/D | y,y,y/m | 2 | 2 | 1 | P01 | fp |
| ROUNDSS/SD/PS/PD | x,x/m,i | 1 | 4 | 1 | P0 | fp |
| VROUNDSS/SD/PS/ PD | y,y/m,i | 2 | 4 | 2 | P0 | fp |
| DPPS | x,x,i | 16 | 25 | 6 | P01 P23 | SSE4.1 |
| DPPS | x,m,i | 18 | | 7 | P01 P23 | SSE4.1 |
| VDPPS | y,y,y,i | 25 | 27 | 13 | P01 P3 | SSE4.1 |
| VDPPS | y,m,i | 29 | | 13 | P01 P3 | SSE4.1 |
| DPPD | x,x,i | 15 | 15 | 5 | P01 P23 | SSE4.1 |
| DPPD | x,m,i | 17 | | 6 | P01 P23 | SSE4.1 |
| VFMADD132SS/SD | x,x,x/m | 1 | 5-6 | 1 | P01 | FMA3 |
| VFMADD132PS/PD | x,x,x/m | 1 | 5-6 | 1 | P01 | FMA3 |
| VFMADD132PS/PD | y,y,y/m | 2 | 5-6 | 1 | P01 | FMA3 |
| All other FMA3 instructions: same as above | | | | | | FMA3 |
| VFMADDSS/SD | x,x,x,x/m | 1 | 5-6 | 0.5 | P01 | AMD FMA4 |
| VFMADDPS/PD | x,x,x,x/m | 1 | 5-6 | 0.5 | P01 | AMD FMA4 |
| VFMADDPS/PD | y,y,y,y/m | 2 | 5-6 | 1 | P01 | AMD FMA4 |
| All other FMA4 instructions: same as above | | | | | | AMD FMA4 |
| Math | | | | | | |
| SQRTSS/PS | x,x/m | 1 | 13-15 | 5-12 | P01 | fp |
| VSQRTPS | y,y/m | 2 | 14-15 | 9-24 | P01 | fp |
| SQRTSD/PD | x,x/m | 1 | 24-26 | 5-15 | P01 | fp |
| VSQRTPD | y,y/m | 2 | 24-26 | 9-29 | P01 | fp |
| RSQRTSS/PS | x,x/m | 1 | 5 | 1 | P01 | fp |
| VRSQRTPS | y,y/m | 2 | 5 | 2 | P01 | fp |
| VFRCZSS/SD/PS/PD | x,x | 2 | 10 | 2 | P01 | AMD XOP |
| VFRCZSS/SD/PS/PD | x,m | 3 | 10 | 2 | P01 | AMD XOP |
| Logic | | | | | | |
| AND/ANDN/OR/XORPS/ PD | x,x/m | 1 | 2 | 0.5 | P23 | ivec |
| VAND/ANDN/OR/ XORPS/PD | y,y,y/m | 2 | 2 | 1 | P23 | ivec |
| Other | | | | | | |
| VZEROUPPER | | 9 | | 4 | P2 P3 | 32 bit mode |
| VZEROUPPER | | 16 | | 5 | P2 P3 | 64 bit mode |
| VZEROALL | | 17 | | 6 | P2 P3 | 32 bit mode |
| VZEROALL | | 32 | | 10 | P2 P3 | 64 bit mode |
| LDMXCSR | m32 | 7 | | 34 | P0 P3 | |

Piledriver

| | | | | | | |
|---------|-------|-----|-----|-----|-------------|--|
| STMXCSR | m32 | 2 | | 17 | P0 P3 | |
| FXSAVE | m4096 | 67 | 136 | 136 | P0 P1 P2 P3 | |
| FXRSTOR | m4096 | 116 | 176 | 176 | P0 P1 P2 P3 | |
| XSAVE | m | 122 | 196 | 196 | P0 P1 P2 P3 | |
| XRSTOR | m | 177 | 250 | 250 | P0 P1 P2 P3 | |

AMD Steamroller

List of instruction timings and macro-operation breakdown

Explanation of column headings:

| | |
|-------------------------------|--|
| Instruction: | Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc. |
| Operands: | i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, x = 128 bit xmm register, y = 256 bit ymm register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc. |
| Ops: | Number of macro-operations issued from instruction decoder to schedulers. Instructions with more than 2 macro-operations use microcode. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. The latency listed does not include the memory operand where the listing for register and memory operand are joined (r/m). |
| Reciprocal throughput: | This is also called issue latency. This value indicates the average number of clock cycles from the execution of an instruction begins to a subsequent independent instruction of the same kind can begin to execute. A value of 1/3 indicates that the execution units can handle 3 instructions per clock cycle in one thread. However, the throughput may be limited by other bottlenecks in the pipeline. |
| Execution pipe: | Indicates which execution pipe or unit is used for the macro-operations: Integer pipes: EX0: integer ALU, division EX1: integer ALU, multiplication, jump EX01: can use either EX0 or EX1 AG01: address generation unit 0 or 1 Floating point and vector pipes: P0: floating point add, mul, div. Integer add, mul, bool P1: floating point add, mul, div. Shuffle, shift, pack P2: Integer add. Bool, store P01: can use either P0 or P1 P02: can use either P0 or P2 Two macro-operations can execute simultaneously if they go to different execution pipes |
| Domain: | Tells which execution unit domain is used: ivec: integer vector execution unit. fp: floating point execution unit. fma: floating point multiply/add subunit. inherit: the output operand inherits the domain of the input operand. ivec/fma means the input goes to the ivec domain and the output comes from the fma domain. There is an additional latency of 1 clock cycle if the output of an ivec instruction goes to the input of a fp or fma instruction, and when the output of a fp or fma instruction goes to the input of an ivec or store instruction. There is no latency between the fp and fma units. All other latencies after memory load and before memory store instructions are included in the latency counts. An fma instruction has a latency of 5 if the output goes to another fma instruction, 6 if the output goes to an fp instruction, and 6+1 if the output goes to an ivec or store instruction. |

Integer instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Notes |
|--------------------------|----------|-----|---------|-----------------------|-----------------|-------|
| Move instructions | | 1 | 1 | 0.5 | EX01 | |
| MOV | r8,r8 | | | | | |

Steamroller

| | | | | | | |
|--------------------------------|------------|----|-----|------|--------------|----------------------|
| MOV | r16,r16 | 1 | 1 | 0.5 | EX01 | |
| MOV | r32,r32 | 1 | 1 | 0.25 | EX01 or AG01 | |
| MOV | r64,r64 | 1 | 1 | 0.25 | EX01 or AG01 | |
| MOV | r,i | 1 | 1 | 0.5 | EX01 | |
| MOV | r,m | 1 | 3 | 0.5 | AG01 | all addr. modes |
| MOV | m,r | 1 | 4 | 1 | EX01 AG01 | all addr. modes |
| MOV | m,i | 1 | | 1 | | |
| MOVNTI | m,r | 1 | 4 | 1 | | |
| MOVZX, MOVSX | r,r | 1 | 1 | 0.5 | EX01 | |
| MOVSX | r,m | 1 | 5 | 0.5 | EX01 | |
| MOVZX | r,m | 1 | 4 | 0.5 | EX01 | |
| MOVSXD | r64,r32 | 1 | 1 | 0.5 | EX01 | |
| MOVSXD | r64,m32 | 1 | 5 | 0.5 | EX01 | |
| CMOVcc | r,r | 1 | 1 | 0.5 | EX01 | |
| CMOVcc | r,m | 1 | | 0.5 | EX01 | |
| XCHG | r8,r8 | 2 | 1 | 1 | EX01 | |
| XCHG | r16,r16 | 2 | 1 | 1 | EX01 | |
| XCHG | r32,r32 | 2 | 1 | 0.5 | EX01 | |
| XCHG | r64,r64 | 2 | 1 | 0.5 | EX01 | |
| XCHG | r,m | 2 | ~38 | ~38 | EX01 | Timing depends on hw |
| XLAT | | 2 | 6 | 2 | | |
| PUSH | r | 1 | | 1 | | |
| PUSH | i | 1 | | 1 | | |
| PUSH | m | 2 | | 1 | | |
| PUSHF(D/Q) | | 8 | | 4 | | |
| PUSHA(D) | | 9 | | 9 | | |
| POP | r | 1 | | 1 | | |
| POP | m | 2 | | 1 | | |
| POPF(D/Q) | | 34 | | 19 | | |
| POPA(D) | | 14 | | 8 | | |
| POP | sp | 1 | 2 | | | |
| LEA | r16,[m] | 2 | 2-3 | | EX01 | any addr. size |
| LEA | r32,[m] | 1 | 2 | | EX01 | 16 bit addr. size |
| LEA | r32/64,[m] | 1 | 2 | 0.5 | EX01 | scale factor > 1 |
| LEA | r32/64,[m] | 1 | 1 | 0.5 | EX01 | or 3 operands |
| LAHF | | 4 | 3 | 2 | | all other cases |
| SAHF | | 2 | 2 | 1 | | |
| SALC | | 1 | 1 | 1 | | |
| BSWAP | r | 1 | 1 | 0.5 | EX01 | |
| PREFETCHNTA | m | 1 | | 0.5 | | |
| PREFETCHT0/1/2 | m | 1 | | 0.5 | | |
| PREFETCH/W | m | 1 | | 0.5 | | PREFETCHW |
| SFENCE | | 7 | | ~80 | | |
| LFENCE | | 1 | | 0.25 | | |
| MFENCE | | 7 | | ~80 | | |
| Arithmetic instructions | | | | | | |
| ADD, SUB | r,r | 1 | 1 | 0.5 | EX01 | |
| ADD, SUB | r,i | 1 | 1 | 0.5 | EX01 | |
| ADD, SUB | r,m | 1 | | 0.5 | EX01 | |
| ADD, SUB | m,r | 1 | 7 | 1 | EX01 | |

Steamroller

| | | | | | |
|---------------------------|-------------|----|-------|-------|------|
| ADD, SUB | m,i | 1 | 7 | 1 | EX01 |
| ADC, SBB | r,r | 1 | 1 | | EX01 |
| ADC, SBB | r,i | 1 | 1 | | EX01 |
| ADC, SBB | r,m | 1 | 1 | 1 | EX01 |
| ADC, SBB | m,r | 1 | 9 | 1 | EX01 |
| ADC, SBB | m,i | 1 | 9 | 1 | EX01 |
| CMP | r,r | 1 | 1 | 0.5 | EX01 |
| CMP | r,i | 1 | 1 | 0.5 | EX01 |
| CMP | r,m | 1 | | 0.5 | EX01 |
| CMP | m,i | 1 | | 0.5 | EX01 |
| INC, DEC, NEG | r | 1 | 1 | 0.5 | EX01 |
| INC, DEC, NEG | m | 1 | 7 | 1 | EX01 |
| AAA, AAS | | 10 | 6 | | |
| DAA | | 16 | 8 | | |
| DAS | | 20 | 10 | | |
| AAD | | 4 | 6 | | |
| AAM | | 10 | 15 | 15 | |
| MUL, IMUL | r8/m8 | 1 | 4 | 2 | EX1 |
| MUL, IMUL | r16/m16 | 2 | 4 | 2 | EX1 |
| MUL, IMUL | r32/m32 | 1 | 4 | 2 | EX1 |
| MUL, IMUL | r64/m64 | 1 | 6 | 4 | EX1 |
| IMUL | r16,r16/m16 | 1 | 4 | 2 | EX1 |
| IMUL | r32,r32/m32 | 1 | 4 | 2 | EX1 |
| IMUL | r64,r64/m64 | 1 | 6 | 4 | EX1 |
| IMUL | r16,(r16),i | 2 | 5 | 2 | EX1 |
| IMUL | r32,(r32),i | 1 | 4 | 2 | EX1 |
| IMUL | r64,(r64),i | 1 | 6 | 4 | EX1 |
| IMUL | r16,m16,i | 2 | | 2 | EX1 |
| IMUL | r32,m32,i | 2 | | 2 | EX1 |
| IMUL | r64,m64,i | 2 | | 4 | EX1 |
| DIV | r8/m8 | 9 | 17-22 | 13-17 | EX0 |
| DIV | r16/m16 | 7 | 15-25 | 15-25 | EX0 |
| DIV | r32/m32 | 2 | 13-39 | 13-39 | EX0 |
| DIV | r64/m64 | 2 | 13-70 | 13-70 | EX0 |
| IDIV | r8/m8 | 9 | 17-22 | 13-17 | EX0 |
| IDIV | r16/m16 | 7 | 14-25 | 14-24 | EX0 |
| IDIV | r32/m32 | 2 | 13-39 | 13-39 | EX0 |
| IDIV | r64/m64 | 2 | 13-70 | 13-70 | EX0 |
| CBW, CWDE, CDQE | | 1 | 1 | | EX01 |
| CDQ, CQO | | 1 | 1 | 0.5 | EX01 |
| CWD | | 2 | 1 | 1 | EX01 |
| Logic instructions | | | | | |
| AND, OR, XOR | r,r | 1 | 1 | 0.5 | EX01 |
| AND, OR, XOR | r,i | 1 | 1 | 0.5 | EX01 |
| AND, OR, XOR | r,m | 1 | | 0.5 | EX01 |
| AND, OR, XOR | m,r | 1 | 7 | 1 | EX01 |
| AND, OR, XOR | m,i | 1 | 7 | 1 | EX01 |
| TEST | r,r | 1 | 1 | 0.5 | EX01 |
| TEST | r,i | 1 | 1 | 0.5 | EX01 |
| TEST | m,r | 1 | | 0.5 | EX01 |
| TEST | m,i | 1 | | 0.5 | EX01 |
| NOT | r | 1 | 1 | 0.5 | EX01 |

Steamroller

| | | | | | | |
|--------------------------------------|---------------|-----|---|-----|------|---------|
| NOT | m | 1 | 7 | 1 | EX01 | BMI1 |
| ANDN | r,r,r | 1 | 1 | 0.5 | EX01 | |
| SHL, SHR, SAR | r,i/CL | 1 | 1 | 0.5 | EX01 | |
| ROL, ROR | r,i/CL | 1 | 1 | 0.5 | EX01 | |
| RCL | r,1 | 1 | 1 | | EX01 | |
| RCL | r,i | 16 | 7 | | EX01 | |
| RCL | r,cl | 17 | 7 | | EX01 | |
| RCR | r,1 | 1 | 1 | | EX01 | |
| RCR | r,i | 15 | 7 | | EX01 | |
| RCR | r,cl | 16 | 7 | | EX01 | |
| SHLD, SHRD | r,r,i | 6 | 3 | 3 | EX01 | |
| SHLD, SHRD | r,r,cl | 7-8 | 4 | 4 | EX01 | |
| SHLD, SHRD | m,r,i/CL | 8 | | 4 | EX01 | |
| BT | r,r/i | 1 | 1 | 0.5 | EX01 | |
| BT | m,i | 1 | | 0.5 | EX01 | |
| BT | m,r | 7 | | 3.5 | EX01 | |
| BTC, BTR, BTS | r,r/i | 2 | 2 | 1 | EX01 | |
| BTC, BTR, BTS | m,i | 4 | | 2 | EX01 | |
| BTC, BTR, BTS | m,r | 10 | | 5 | EX01 | |
| BSF | r,r | 6 | 3 | 3 | EX01 | |
| BSF | r,m | 8 | 4 | 4 | EX01 | |
| BSR | r,r | 7 | 4 | 4 | EX01 | |
| BSR | r,m | 9 | | 5 | EX01 | |
| SETcc | r | 1 | 1 | 0.5 | EX01 | |
| SETcc | m | 1 | | 1 | EX01 | |
| CLC, STC | | 1 | | 0.5 | EX01 | |
| CMC | | 1 | 1 | | EX01 | |
| CLD | | 2 | | 3 | | |
| STD | | 2 | | 4 | | |
| POPCNT | r16/32,r16/32 | 1 | 4 | 2 | | SSE4.2 |
| POPCNT | r64,r64 | 1 | 4 | 4 | | SSE4.2 |
| LZCNT | r,r | 1 | 2 | 2 | EX0 | LZCNT |
| TZCNT | r,r | 2 | 2 | 2 | | BMI1 |
| BEXTR | r,r,r | 2 | 2 | 1 | | BMI1 |
| BEXTR | r,r,i | 2 | 2 | 1 | | AMD TBM |
| BLSI | r,r | 2 | 2 | 1 | | BMI1 |
| BLSMSK | r,r | 2 | 2 | 1 | | BMI1 |
| BLSR | r,r | 2 | 2 | 1 | | BMI1 |
| BLCFILL | r,r | 2 | 2 | 1 | | AMD TBM |
| BLCI | r,r | 2 | 2 | 1 | | AMD TBM |
| BLCIC | r,r | 2 | 2 | 1 | | AMD TBM |
| BLCMSK | r,r | 2 | 2 | 1 | | AMD TBM |
| BLCS | r,r | 2 | 2 | 1 | | AMD TBM |
| BLSFILL | r,r | 2 | 2 | 1 | | AMD TBM |
| BLSI | r,r | 2 | 2 | 1 | | AMD TBM |
| BLSIC | r,r | 2 | 2 | 1 | | AMD TBM |
| T1MSKC | r,r | 2 | 2 | 1 | | AMD TBM |
| TZMSK | r,r | 2 | 2 | 1 | | AMD TBM |
| Control transfer instructions | | | | | | |
| JMP | short/near | 1 | | 2 | EX1 | |
| JMP | r | 1 | | 2 | EX1 | |
| JMP | m | 1 | | 2 | EX1 | |

Steamroller

| | | | | | | |
|----------------------------|------------|------------|------|------------|------|--------------|
| Jcc | short/near | 1 | | 1-2 | EX1 | 2 if jumping |
| fused CMP+Jcc | short/near | 1 | | 1-2 | EX1 | 2 if jumping |
| J(E/R)CXZ | short | 1 | | 1-2 | EX1 | 2 if jumping |
| LOOP | short | 1 | | 1-2 | EX1 | 2 if jumping |
| LOOPE LOOPNE | short | 1 | | 1-2 | EX1 | 2 if jumping |
| CALL | near | 2 | | 2 | EX1 | |
| CALL | r | 2 | | 2 | EX1 | |
| CALL | m | 3 | | 2 | EX1 | |
| RET | | 1 | | 2 | EX1 | |
| RET | i | 4 | | 2 | EX1 | |
| BOUND | m | 11 | | 5 | | for no jump |
| INTO | | 4 | | 2 | | for no jump |
| String instructions | | | | | | |
| LODS | | 3 | | 3 | | |
| REP LODS | m8/m16 | 6n | | 3n | | |
| REP LODS | m32/m64 | 6n | | 2.5n | | |
| STOS | | 3 | | 3 | | |
| REP STOS | | 1n | | ~1n | | small n |
| REP STOS | | 3 per 16B | | 2 per 16B | | best case |
| MOVS | | 5 | | 3 | | |
| REP MOVS | | ~1n | | ~1n | | small n |
| REP MOVS | | 4-5 pr 16B | | ~2 per 16B | | best case |
| SCAS | | 3 | | 3 | | |
| REP SCAS | | 7n | | 3-4n | | |
| CMPS | | 6 | | 3 | | |
| REP CMPS | | 9n | | 4n | | |
| Synchronization | | | | | | |
| LOCK ADD | m,r | 1 | ~39 | | | |
| XADD | m,r | 4 | 9-12 | | | |
| LOCK XADD | m,r | 4 | ~39 | | | |
| CMPXCHG | m,r8 | 5 | 15 | | | |
| CMPXCHG | m,r16 | 6 | 15 | | | |
| CMPXCHG | m,r32/64 | 6 | 13 | | | |
| LOCK CMPXCHG | m8,r8 | 5 | ~40 | | | |
| LOCK CMPXCHG | m16,r16 | 6 | ~40 | | | |
| LOCK CMPXCHG | m,r32/64 | 6 | ~40 | | | |
| CMPXCHG8B | m64 | 18 | ~14 | | | |
| LOCK CMPXCHG8B | m64 | 18 | ~42 | | | |
| CMPXCHG16B | m128 | 24 | ~47 | | | |
| LOCK CMPXCHG16B | m128 | 24 | ~80 | | | |
| Other | | | | | | |
| NOP (90) | | 1 | | 0.25 | none | |
| Long NOP (0F 1F) | | 1 | | 0.25 | none | |
| PAUSE | | 8 | | 4 | | |
| ENTER | a,0 | 13 | | 21 | | |
| ENTER | a,b | 11+5b | | 20-30 | | |
| LEAVE | | 2 | | 3 | | |
| CPUID | | 38-64 | | 100-300 | | |
| XGETBV | | 4 | | 30 | | |
| RDTSC | | 44 | | 78 | | |

Steamroller

| | | | | | | |
|--------|---------|----|---|-----|--|--------|
| RDTSCP | | 44 | | 105 | | rdtscp |
| RDPMC | | 22 | | 360 | | |
| CRC32 | r32,r8 | 3 | 3 | 2 | | |
| CRC32 | r32,r16 | 5 | 5 | 5 | | |
| CRC32 | r32,r32 | 7 | 6 | 6 | | |

Floating point x87 instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Domain, notes |
|--------------------------------|----------|--------|---------|-----------------------|-----------------|---------------|
| Move instructions | | | | | | |
| FLD | r | 1 | 2 | 0.5 | P01 | fp |
| FLD | m32/64 | 1 | 7 | 1 | | fp |
| FLD | m80 | 8 | 11 | 4 | | fp |
| FBLD | m80 | 60 | 52 | 34 | P0 P1 P2 | fp |
| FST(P) | r | 1 | 2 | 0.5 | P01 | fp |
| FST(P) | m32/64 | 2 | 7 | 1 | | fp |
| FSTP | m80 | 13 | 14 | 19 | | fp |
| FBSTP | m80 | 239 | 222 | 222 | P0 P1 P2 | fp |
| FXCH | r | 1 | 0 | 0.5 | P01 | inherit |
| FILD | m | 1 | 11 | 1 | P01 | fp |
| FIST(T)(P) | m | 2 | 7 | 1 | P0 P2 | fp |
| FLDZ, FLD1 | | 1 | | 0.5 | P01 | fp |
| FCMOVcc | st0,r | 8 | 3 | 3 | P0 P1 P2 | fp |
| FFREE | r | 1 | | 0.25 | none | |
| FINCSTP, FDECSTP | | 1 | 0 | 0.25 | none | inherit |
| FNSTSW | AX | 3 | 11 | 19 | P0 P2 | |
| FNSTSW | m16 | 2 | | 17 | P0 P2 | |
| FLDCW | m16 | 1 | | 3 | | |
| FNSTCW | m16 | 2 | | 2 | | |
| Arithmetic instructions | | | | | | |
| FADD(P),FSUB(R)(P) | r/m | 1 | 5 | 1 | P01 | fma |
| FIADD,FISUB(R) | m | 2 | | 2 | P01 | fma |
| FMUL(P) | r/m | 1 | 5 | 1 | P01 | fma |
| FIMUL | m | 2 | | 2 | P01 | fma |
| FDIV(R)(P) | r | 1 | 9-37 | 4-16 | P01 | fp |
| FDIV(R) | m | 1 | | | P01 | fp |
| FIDIV(R) | m | 2 | | 4 | P01 | fp |
| FABS, FCHS | | 1 | 2 | 0.5 | P01 | fp |
| FCOM(P), FUCOM(P) | r/m | 1 | | 0.5 | P01 | fp |
| FCOMPP, FUCOMPP | | 1 | | 0.5 | P01 | fp |
| FCOMI(P) | r | 2 | 2 | 1 | P01 P2 | fp |
| FICOM(P) | m | 2 | | 1 | P01 | fp |
| FTST | | 1 | | 0.5 | P01 | fp |
| FXAM | | 1 | 26 | 0.5 | P01 | fp |
| FRNDINT | | 1 | 4 | 1 | P0 | fp |
| FPREM FPREM1 | | 1 | 17-60 | 12-53 | P0 | fp |
| Math | | | | | | |
| FSQRT | | 1 | 10-50 | 5-20 | P01 | |
| FLDPI, etc. | | 1 | | 0.5 | P01 | |
| FSIN | | 10-164 | 60-210 | 60-165 | P0 P1 P2 | |

Steamroller

| | | | | | |
|--------------|------|--------|--------|--------|----------|
| FCOS | | 18-166 | 76-158 | | P0 P1 P2 |
| FSINCOS | | 12-168 | | 90-165 | P0 P1 P2 |
| FPTAN | | 11-192 | 90-245 | 90-210 | P0 P1 P2 |
| FPATAN | | 10-365 | 60-440 | 60-365 | P0 P1 P2 |
| FSCALE | | 10 | 49 | 5 | P0 P1 P2 |
| FXTRACT | | 12 | 8 | 5 | P0 P1 P2 |
| F2XM1 | | 10-18 | 60-74 | | P0 P1 P2 |
| FYL2X | | 9-183 | 60-280 | | P0 P1 P2 |
| FYL2XP1 | | 206 | ~390 | | P0 P1 P2 |
| Other | | | | | |
| FNOP | | 1 | | 0.25 | none |
| (F)WAIT | | 1 | | 0.25 | none |
| FNCLEX | | 18 | | 63 | P0 |
| FNINIT | | 31 | | 131 | P0 |
| FNSAVE | m864 | 98 | 256 | 256 | P0 P1 P2 |
| FRSTOR | m864 | 73 | 166 | 166 | P0 P2 |

Integer vector instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Notes |
|--------------------------|--------------|-----|---------|-----------------------|-----------------|----------------|
| Move instructions | | | | | | |
| MOVD | r32/64, mm/x | 1 | 8 | 1 | P2 | |
| MOVD | mm/x, r32/64 | 2 | 9 | 1 | | |
| MOVD | mm/x, m32 | 1 | 2 | 0.5 | | |
| MOVD | m32, mm/x | 1 | 3 | 1 | | |
| MOVQ | mm/x, mm/x | 1 | 2 | 0.5 | P02 | |
| MOVQ | mm/x, m64 | 1 | 2 | 0.5 | | |
| MOVQ | m64, mm/x | 1 | 3 | 1 | | |
| MOVDQA | x, x | 1 | 0 | 0.25 | none | inherit domain |
| MOVDQA | x, m | 1 | 6 | 0.5 | | |
| MOVDQA | m, x | 1 | 5 | 1 | P2 | |
| VMOVDQA | y, y | 2 | 6 | 0.5 | P02 | |
| VMOVDQA | y, m256 | 2 | 5 | 1 | | |
| VMOVDQA | m256, y | 2 | 4 | 1 | P2 | |
| MOVDQU | x, x | 1 | 0 | 0.25 | none | inherit domain |
| MOVDQU | x, m | 1 | 6 | 0.5 | | |
| MOVDQU | m, x | 1 | 5 | 1 | P2 | |
| LDDQU | x, m | 1 | 6 | 0.5 | | |
| VMOVDQU | y, m256 | 2 | 7 | 1 | | |
| VMOVDQU | m256, y | 2 | 6 | 1 | | |
| MOVDQ2Q | mm, x | 1 | 1 | 0.5 | P02 | |
| MOVQ2DQ | x, mm | 1 | 1 | 0.5 | P02 | |
| MOVNTQ | m, mm | 1 | 5 | 1 | P2 | |
| MOVNTDQ | m, x | 1 | 5 | 1 | P2 | |
| MOVNTDQA | x, m | 1 | 2 | 0.5 | | |
| PACKSSWB/DW | mm/x, r/m | 1 | 2 | 1 | P1 | |
| PACKUSWB | mm/x, r/m | 1 | 2 | 1 | P1 | |
| PUNPCKH/LBW/WD/DQ | mm/x, r/m | 1 | 2 | 1 | P1 | |
| PUNPCKHQDQ | x, r/m | 1 | 2 | 1 | P1 | |

Steamroller

| | | | | | | |
|----------------------------------|------------|----|----|-----|----------|-------------------------------|
| PUNPCKLQDQ | x,r/m | 1 | 2 | 1 | P1 | |
| PSHUFB | mm/x,r/m | 1 | 3 | 1 | P1 | |
| PSHUFD | x,x,i | 1 | 2 | 1 | P1 | |
| PSHUFW | mm,mm,i | 1 | 2 | 1 | P1 | |
| PSHUFL/HW | x,x,i | 1 | 2 | 1 | P1 | |
| PALIGNR | mm/x,r/m,i | 1 | 2 | 1 | P1 | |
| PBLENDW | x,r/m | 1 | 2 | 0.5 | P02 | SSE4.1 |
| MASKMOVQ | mm,mm | 31 | 32 | 16 | P2 | |
| MASKMOVDQU | x,x | 65 | 45 | 31 | P0 P1 P2 | |
| PMOVMASKB | r32,mm/x | 2 | 5 | 1 | P1 P2 | |
| PEXTRB/W/D/Q | r,x/mm,i | 2 | 5 | 1 | P1 P2 | SSE4.1 |
| PINSRB/W/D/Q | x/mm,r,i | 2 | 6 | 1 | P1 | |
| EXTRQ | x,i,i | 1 | 3 | 1 | P1 | AMD SSE4A |
| EXTRQ | x,x | 1 | 1 | 1 | P1 | AMD SSE4A |
| INSERTQ | x,x,i,i | 1 | 1 | 1 | P1 | AMD SSE4A |
| INSERTQ | x,x | 1 | 1 | 1 | P1 | AMD SSE4A |
| PMOVSBXW/BD/BQ/ WD/WQ/DQ | x,x | 1 | 2 | 1 | P1 | SSE4.1 |
| PMOVZXBW/BD/BQ/ WD/WQ/DQ | x,x | 1 | 2 | 1 | P1 | SSE4.1 |
| VPCMOV | x,x,x,x/m | 1 | 2 | 1 | P1 | AMD XOP |
| VPCMOV | y,y,y,y/m | 2 | 2 | 2 | P1 | AMD XOP |
| VPPERM | x,x,x,x/m | 1 | 2 | 1 | P1 | AMD XOP |
| Arithmetic instructions | | | | | | |
| PADDB/W/D/Q/SB/SW/ USB/USW | mm/x,r/m | 1 | 2 | 0.5 | P02 | |
| PSUBB/W/D/Q/SB/SW/ USB/USW | mm/x,r/m | 1 | 2 | 0.5 | P02 | |
| PHADD/SUB(S)W/D | x,x | 3 | 5 | 2 | P02 2P1 | SSSE3 |
| PCMPEQ/GT B/W/D | mm/x,r/m | 1 | 2 | 0.5 | P02 | |
| PCMPEQQ | mm/x,r/m | 1 | 2 | 0.5 | P02 | SSE4.1 |
| PCMPGTQ | mm/x,r/m | 1 | 2 | 0.5 | P02 | SSE4.2 |
| PMULLW PMULHW PMULHUW PMULUDQ | mm/x,r/m | 1 | 4 | 1 | P0 | |
| PMULLD | x,r/m | 1 | 5 | 2 | P0 | SSE4.1 |
| PMULDQ | x,r/m | 1 | 4 | 1 | P0 | SSE4.1 |
| PMULHRSW | mm/x,r/m | 1 | 4 | 1 | P0 | SSSE3 |
| PMADDWD | mm/x,r/m | 1 | 4 | 1 | P0 | |
| PMADDUBSW | mm/x,r/m | 1 | 4 | 1 | P0 | |
| PAVGB/W | mm/x,r/m | 1 | 2 | 0.5 | P02 | |
| PMIN/MAX SB/SW/ SD UB/UW/UD | mm/x,r/m | 1 | 2 | 0.5 | P02 | |
| PHMINPOSUW | x,r/m | 2 | 4 | 1 | P1 P02 | SSE4.1 |
| PABSB/W/D | mm/x,r/m | 1 | 2 | 0.5 | P02 | SSSE3 |
| PSIGNB/W/D | mm/x,r/m | 1 | 2 | 0.5 | P02 | SSSE3 |
| PSADBW | mm/x,r/m | 2 | 4 | 1 | P02 | |
| MPSADBW | x,x,i | 8 | 8 | 4 | P1 P02 | SSE4.1 |
| VPCOMB/W/D/Q | x,x,x/m,i | 1 | 2 | 0.5 | P02 | AMD XOP latency 0 if i=6,7 |
| VPCOMUB/W/D/Q | x,x,x/m,i | 1 | 2 | 0.5 | P02 | AMD XOP latency 0 if i=6,7 |
| VPHADDBW/BD/BQ/ WD/WQ/DQ | x,x/m | 1 | 2 | 0.5 | P02 | AMD XOP |

Steamroller

| | | | | | | |
|----------------------------|-----------|----|----|------|----------|---------|
| VPHADDUBW/BD/BQ/WD/WQ/DQ | x,x/m | 1 | 2 | 0.5 | P02 | AMD XOP |
| VPHSUBBW/WD/DQ | x,x/m | 1 | 2 | 0.5 | P02 | AMD XOP |
| VPMACSWW/WD | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP |
| VPMACSSD | x,x,x/m,x | 1 | 5 | 2 | P0 | AMD XOP |
| VPMACSDQH/L | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP |
| VPMACSSWW/WD | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP |
| VPMACSSD | x,x,x/m,x | 1 | 5 | 2 | P0 | AMD XOP |
| VPMACSSDQH/L | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP |
| VPMADCSWD | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP |
| VPMADCSSWD | x,x,x/m,x | 1 | 4 | 1 | P0 | AMD XOP |
| Logic | | | | | | |
| PAND PANDN POR PXOR | mm/x,r/m | 1 | 2 | 0.5 | P02 | |
| PSLL/RL W/D/Q | | | | | | |
| PSRAW/D | mm/x,r/m | 1 | 3 | 1 | P1 | |
| PSLL/RL W/D/Q | | | | | | |
| PSRAW/D | mm/x,i | 1 | 2 | 1 | P1 | |
| PSLLDQ, PSRLDQ | x,i | 1 | 2 | 1 | P1 | |
| PTEST | x,r/m | 2 | 14 | 1 | P1 P2 | SSE4.1 |
| VPROTB/W/D/Q | x,x,x/m | 1 | 3 | 1 | P1 | AMD XOP |
| VPROTB/W/D/Q | x,x,i | 1 | 2 | 1 | P1 | AMD XOP |
| VPSHAB/W/D/Q | x,x,x/m | 1 | 3 | 1 | P1 | AMD XOP |
| VPSHLB/W/D/Q | x,x,x/m | 1 | 3 | 1 | P1 | AMD XOP |
| String instructions | | | | | | |
| PCMPESTRI | x,x,i | 30 | 11 | 11 | P0 P1 P2 | SSE4.2 |
| PCMPESTRM | x,x,i | 30 | 10 | 10 | P0 P1 P2 | SSE4.2 |
| PCMPISTRI | x,x,i | 9 | 5 | 5 | P0 P1 P2 | SSE4.2 |
| PCMPISTRM | x,x,i | 8 | 6 | 6 | P0 P1 P2 | SSE4.2 |
| Encryption | | | | | | |
| PCLMULQDQ | x,x/m,i | 7 | 11 | 7 | P1 | pclmul |
| VPCLMULQDQ | x,x,x,i | 7 | 11 | 7 | P1 | pclmul |
| PCLMULQDQ | x,x,m,i | 8 | | 7 | P1 | pclmul |
| AESDEC | x,x | 2 | 5 | 1 | P01 | aes |
| AESDECLAST | x,x | 2 | 5 | 1 | P01 | aes |
| AESENC | x,x | 2 | 5 | 1 | P01 | aes |
| AESENCLAST | x,x | 2 | 5 | 1 | P01 | aes |
| AESIMC | x,x | 1 | 5 | 1 | P0 | aes |
| AESKEYGENASSIST | x,x,i | 1 | 5 | 1 | P0 | aes |
| Other | | | | | | |
| EMMS | | 1 | | 0.25 | | |

Floating point vector instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Domain, notes |
|--------------------------|----------|-----|---------|-----------------------|-----------------|----------------|
| Move instructions | | | | | | |
| MOVAPS/D MOVUPS/D | x,x | 1 | 0 | 0.25 | none | inherit domain |

Steamroller

| | | | | | | |
|-------------------|------------|----|----|-----|-------|-----------|
| VMOVAPS/D | y,y | 2 | 2 | 0.5 | P02 | ivec |
| MOVAPS/D MOVUPS/D | x,m128 | 1 | 6 | 0.5 | | |
| VMOVAPS/D | | | | | | |
| VMOVUPS/D | y,m256 | 2 | 6 | 1 | | |
| MOVAPS/D MOVUPS/D | m128,x | 1 | 5 | 1 | P2 | |
| VMOVAPS/D | m256,y | 2 | 5 | 2 | P2 | |
| VMOVUPS/D | m256,y | 2 | 5 | 2 | P2 | |
| MOVSS/D | x,x | 1 | 2 | 0.5 | P01 | fp |
| MOVSS/D | x,m32/64 | 1 | 6 | 0.5 | | |
| MOVSS/D | m32/64,x | 1 | 5 | 1 | P2 | |
| MOVHPS/D | x,m64 | 1 | 6 | 1 | P1 | |
| MOVLPS/D | x,m64 | 1 | 6 | 0.5 | P01 | |
| MOVHPS/D | m64,x | 2 | 6 | 1 | P1 P2 | |
| MOVLPS/D | m64,x | 1 | 5 | 1 | P2 | |
| MOVLHPS MOVHLPS | x,x | 1 | 2 | 1 | P1 | ivec |
| MOVMSKPS/D | r32,x | 2 | 5 | 1 | P1 P2 | |
| VMOVMSKPS/D | r32,y | 2 | 15 | 1 | P1 P2 | |
| MOVNTPS/D | m128,x | 1 | 3 | 1 | P2 | |
| VMOVNTPS/D | m256,y | 2 | 3 | 2-3 | P2 | |
| MOVNTSS/SD | m,x | 1 | | 3 | P2 | AMD SSE4A |
| SHUFPS/D | x,x/m,i | 1 | 2 | 1 | P2 | ivec |
| VSHUFPS/D | y,y,y/m,i | 2 | 2 | 2 | P2 | ivec |
| VPERMILPS/PD | x,x,x/m | 1 | 3 | 1 | P1 | ivec |
| VPERMILPS/PD | y,y,y/m | 2 | 3 | 2 | P1 | ivec |
| VPERMILPS/PD | x,x/m,i | 1 | 2 | 1 | P1 | ivec |
| VPERMILPS/PD | y,y/m,i | 2 | 2 | 2 | P1 | ivec |
| VPERM2F128 | y,y,y,i | 8 | 4 | 3.5 | P0 P2 | ivec |
| VPERM2F128 | y,y,m,i | 12 | | 4 | P0 P2 | ivec |
| BLENDPS/PD | x,x/m,i | 1 | 2 | 0.5 | P01 | fp |
| VBLENDPS/PD | y,y,y/m,i | 2 | 2 | 1 | P01 | fp |
| BLENDVPS/PD | x,x/m,xmm0 | 1 | 2 | 0.5 | P01 | |
| VBLENDVPS/PD | y,y,y/m,y | 2 | 2 | 1 | P01 | |
| MOVDDUP | x,x | 1 | 2 | 1 | P1 | ivec |
| MOVDDUP | x,m64 | 1 | | 0.5 | | |
| VMOVDDUP | y,y | 2 | 2 | 2 | P1 | ivec |
| VMOVDDUP | y,m256 | 2 | | 1 | | |
| VBROADCASTSS | x,m32 | 1 | 8 | 0.5 | | |
| VBROADCASTSS | y,m32 | 2 | 8 | 0.5 | P02 | |
| VBROADCASTSD | y,m64 | 2 | 8 | 0.5 | P02 | |
| VBROADCASTF128 | y,m128 | 2 | 8 | 0.5 | P02 | |
| MOVSH/LDUP | x,x | 1 | 2 | 1 | P1 | ivec |
| MOVSH/LDUP | x,m128 | 1 | | 0.5 | | |
| VMOVSH/LDUP | y,y | 2 | 2 | 2 | P1 | ivec |
| VMOVSH/LDUP | y,m256 | 2 | | 1 | | |
| UNPCKH/LPS/D | x,x/m | 1 | 2 | 1 | P1 | ivec |
| VUNPCKH/LPS/D | y,y,y/m | 2 | 2 | 2 | P1 | ivec |
| EXTRACTPS | r32,x,i | 2 | | 1 | P1 P2 | |
| EXTRACTPS | m32,x,i | 2 | 10 | 1 | P1 P2 | |
| VEEXTRACTF128 | x,y,i | 1 | 2 | 0.5 | P02 | ivec |
| VEEXTRACTF128 | m128,y,i | 2 | 10 | 1 | P0 P2 | |
| INSERTPS | x,x,i | 1 | 2 | 1 | P1 | |

Steamroller

| | | | | | | |
|-------------------|------------|----|------|------|----------|----------|
| INSERTPS | x,m32,i | 1 | 9 | 2 | P1 | |
| VINSERTF128 | y,y,x,i | 2 | 2 | 1 | P02 | ivec |
| VINSERTF128 | y,y,m128,i | 2 | 10 | 1 | P02 | |
| VMASKMOVPS/D | x,x,m128 | 1 | 9 | 0.5 | P01 | |
| VMASKMOVPS/D | y,y,m256 | 2 | 9 | 1 | P01 | |
| VMASKMOVPS/D | m128,x,x | 20 | ~35 | 8 | P0 P1 P2 | |
| VMASKMOVPS/D | m256,y,y | 41 | ~35 | 16 | P0 P1 P2 | |
| Conversion | | | | | | |
| CVTPD2PS | x,x | 2 | 6 | 1 | P01 | ivec/fp |
| VCVTPD2PS | x,y | 4 | 6 | 2 | P01 | ivec/fp |
| CVTPS2PD | x,x | 2 | 6 | 1 | P01 | ivec/fp |
| VCVTPS2PD | y,x | 4 | 6 | 2 | P01 | ivec/fp |
| CVTSD2SS | x,x | 1 | 4 | 1 | P0 | fp |
| CVTSS2SD | x,x | 1 | 4 | 1 | P0 | fp |
| CVTDQ2PS | x,x | 1 | 4 | 1 | P0 | fp |
| VCVTDQ2PS | y,y | 2 | 4 | 2 | P0 | fp |
| CVT(T) PS2DQ | x,x | 1 | 4 | 1 | P0 | fp |
| VCVT(T) PS2DQ | y,y | 2 | 4 | 2 | P0 | fp |
| CVTDQ2PD | x,x | 2 | 7 | 1 | P01 | ivec/fp |
| VCVTDQ2PD | y,x | 4 | 7 | 2 | P01 | ivec/fp |
| CVT(T)PD2DQ | x,x | 2 | 7 | 1 | P01 | fp/ivec |
| VCVT(T)PD2DQ | x,y | 4 | 7 | 2 | P01 | fp/ivec |
| CVTPI2PS | x,mm | 2 | 6 | 1 | P0 P2 | ivec/fp |
| CVT(T)PS2PI | mm,x | 1 | 5 | 1 | P0 | fp |
| CVTPI2PD | x,mm | 2 | 7 | 1 | P0 P1 | ivec/fp |
| CVT(T) PD2PI | mm,x | 2 | 7 | 1 | P0 P1 | fp/ivec |
| CVTSI2SS | x,r32 | 2 | 13 | 1 | P0 | fp |
| CVT(T)SS2SI | r32,x | 2 | 12 | 1 | P0 P2 | fp |
| CVTSI2SD | x,r32/64 | 2 | 12 | 1 | P0 | fp |
| CVT(T)SD2SI | r32/64,x | 2 | 12 | 1 | P0 P2 | fp |
| VCVTPS2PH | x/m,x,i | 2 | 7 | 2 | P0 P1 | F16C |
| VCVTPS2PH | x/m,y,i | 4 | 7 | 2 | P0 P1 | F16C |
| VCVTPH2PS | x,x/m | 2 | 7 | 2 | P0 P1 | F16C |
| VCVTPH2PS | y,x/m | 4 | 7 | 2 | P0 P1 | F16C |
| Arithmetic | | | | | | |
| ADDSS/D SUBSS/D | x,x/m | 1 | 5-6 | 1 | P01 | fma |
| ADDPS/D SUBPS/D | x,x/m | 1 | 5-6 | 1 | P01 | fma |
| VADDPS/D VSUBPS/D | y,y,y/m | 2 | 5-6 | 2 | P01 | fma |
| ADDSUBPS/D | x,x/m | 1 | 5-6 | 1 | P01 | fma |
| VADDSUBPS/D | y,y,y/m | 2 | 5-6 | 1 | P01 | fma |
| HADDPS/D HSUBPS/D | x,x | 4 | 10 | 2 | P0 P1 | ivec/fma |
| VHADDPS/D | | | | | | |
| VHSUBPS/D | y,y,y/m | 8 | 10 | 4 | P01 P1 | ivec/fma |
| MULSS MULSD | x,x/m | 1 | 5-6 | 0.5 | P01 | fma |
| MULPS MULPD | x,x/m | 1 | 5-6 | 0.5 | P01 | fma |
| VMULPS VMULPD | y,y,y/m | 2 | 5-6 | 1 | P01 | fma |
| DIVSS DIVPS | x,x/m | 1 | 9-17 | 4-6 | P01 | fp |
| VDIVPS | y,y,y/m | 2 | 9-17 | 9-12 | P01 | fp |
| DIVSD DIVPD | x,x/m | 1 | 9-32 | 4-13 | P01 | fp |

Steamroller

| | | | | | | |
|--|-----------|-------|-------|------|----------|-------------|
| VDIVPD | y,y,y/m | 2 | 9-32 | 9-27 | P01 | fp |
| RCPSS/PS | x,x/m | 1 | 5 | 1 | P01 | fp |
| VRCPPS | y,y/m | 2 | 5 | 2 | P01 | fp |
| CMPSS/D | | | | | | |
| CMPSS/D | x,x/m | 1 | 2 | 0.5 | P01 | fp |
| VCMPSS/D | y,y,y/m | 2 | 2 | 1 | P01 | fp |
| COMISS/D UCOMISS/ D | x,x/m | 2 | | 1 | P01 P2 | fp |
| MAXSS/SD/PS/PD | | | | | | |
| MINSS/SD/PS/PD | x,x/m | 1 | 2 | 0.5 | P01 | fp |
| VMAXPS/D VMINPS/D | y,y,y/m | 2 | 2 | 1 | P01 | fp |
| ROUNDSS/SD/PS/PD | x,x/m,i | 1 | 4 | 1 | P0 | fp |
| VROUNDSS/SD/PS/ PD | y,y/m,i | 2 | 4 | 2 | P0 | fp |
| DPPS | x,x,i | 9 | 25 | 4 | P0 P1 | SSE4.1 |
| DPPS | x,m,i | 10 | | 5 | P0 P1 | SSE4.1 |
| VDPPS | y,y,y,i | 13 | 25 | 8 | P0 P1 | SSE4.1 |
| VDPPS | y,m,i | 15 | | 8 | P0 P1 | SSE4.1 |
| DPPD | x,x,i | 7 | 14 | 3 | P0 P1 | SSE4.1 |
| DPPD | x,m,i | 8 | | 4 | P0 P1 | SSE4.1 |
| VFMADD132SS/SD | x,x,x/m | 1 | 5-6 | 0.5 | P01 | FMA3 |
| VFMADD132PS/PD | x,x,x/m | 1 | 5-6 | 0.5 | P01 | FMA3 |
| VFMADD132PS/PD | y,y,y/m | 2 | 5-6 | 1 | P01 | FMA3 |
| All other FMA3 instructions: same as above | | | | | | FMA3 |
| VFMADDSS/SD | x,x,x,x/m | 1 | 5-6 | 0.5 | P01 | AMD FMA4 |
| VFMADDPS/PD | x,x,x,x/m | 1 | 5-6 | 0.5 | P01 | AMD FMA4 |
| VFMADDPS/PD | y,y,y,y/m | 2 | 5-6 | 1 | P01 | AMD FMA4 |
| All other FMA4 instructions: same as above | | | | | | AMD FMA4 |
| Math | | | | | | |
| SQRTSS/PS | x,x/m | 1 | 12-13 | 4-9 | P01 | fp |
| VSQRTPS | y,y/m | 2 | 12-13 | 9-18 | P01 | fp |
| SQRTSD/PD | x,x/m | 1 | 26-29 | 4-18 | P01 | fp |
| VSQRTPD | y,y/m | 2 | 27-28 | 9-37 | P01 | fp |
| RSQRTSS/PS | x,x/m | 1 | 5 | 1 | P01 | fp |
| VRSQRTPS | y,y/m | 2 | 5 | 2 | P01 | fp |
| VFRCZSS/SD/PS/PD | x,x | 2 | 10 | 2 | P01 | AMD XOP |
| VFRCZSS/SD/PS/PD | x,m | 4 | | 2 | P01 | AMD XOP |
| Logic | | | | | | |
| AND/ANDN/OR/XORPS/ PD | x,x/m | 1 | 2 | 0.5 | P02 | ivec |
| VAND/ANDN/OR/ XORPS/PD | y,y,y/m | 2 | 2 | 1 | P02 | ivec |
| Other | | | | | | |
| VZEROUPPER | | 9 | | 4 | | 32 bit mode |
| VZEROUPPER | | 16 | | 5 | | 64 bit mode |
| VZEROALL | | 17 | | 6 | P02 | 32 bit mode |
| VZEROALL | | 32 | | 10 | P02 | 64 bit mode |
| LDMXCSR | m32 | 9 | | 36 | P0 P2 | |
| STMXCSR | m32 | 2 | | 17 | P0 P2 | |
| FXSAVE | m4096 | 59-67 | | 78 | P0 P1 P2 | |

Steamroller

| | | | | | | |
|---------|-------|---------|--|---------|----------|--|
| FXRSTOR | m4096 | 104-112 | | 160 | P0 P1 P2 | |
| XSAVE | m | 121-137 | | 147-166 | P0 P1 P2 | |
| XRSTOR | m | 191-209 | | 291-297 | P0 P1 P2 | |

AMD Excavator

List of instruction timings and macro-operation breakdown

Explanation of column headings:

| | |
|-------------------------------|--|
| Instruction: | Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc. |
| Operands: | i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, x = 128 bit xmm register, y = 256 bit ymm register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc. |
| Ops: | Number of macro-operations issued from instruction decoder to schedulers. Instructions with more than 2 macro-operations use microcode. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. The latency listed does not include the memory operand where the listing for register and memory operand are joined (r/m). The numbers are approximate. |
| Reciprocal throughput: | This is also called issue latency. This value indicates the average number of clock cycles from the execution of an instruction begins to a subsequent independent instruction of the same kind can begin to execute. A value of 1/3 indicates that the execution units can handle 3 instructions per clock cycle in one thread. However, the throughput may be limited by other bottlenecks in the pipeline. |
| Execution pipe: | Indicates which execution pipe or unit is used for the macro-operations: Integer pipes: EX0: integer ALU, division EX1: integer ALU, multiplication, jump EX01: can use either EX0 or EX1 AG01: address generation unit 0 or 1 Floating point and vector pipes: P0: floating point add, mul, div. Integer add, mul, bool P1: floating point add, mul, div. Shuffle, shift, pack P2: Integer add. Bool, store P01: can use either P0 or P1 P02: can use either P0 or P2 Two macro-operations can execute simultaneously if they go to different execution pipes |
| Domain: | Tells which execution unit domain is used: ivec: integer vector execution unit. fp: floating point execution unit. fma: floating point multiply/add subunit. inherit: the output operand inherits the domain of the input operand. ivec/fma means the input goes to the ivec domain and the output comes from the fma domain. There is an additional latency of 1 clock cycle if the output of an ivec instruction goes to the input of a fp or fma instruction, and when the output of a fp or fma instruction goes to the input of an ivec or store instruction. There is no latency between the fp and fma units. All other latencies after memory load and before memory store instructions are included in the latency counts. An fma instruction has a latency of 5 if the output goes to another fma instruction, 6 if the output goes to an fp instruction, and 6+1 if the output goes to an ivec or store instruction. |

Integer instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Notes |
|--------------------------|----------|-----|---------|-----------------------|-----------------|-------|
| Move instructions | | 1 | 1 | 0.5 | EX01 | |
| MOV | r8,r8 | | | | | |

Excavator

| | | | | | | |
|--------------------------------|------------|----|-----|------|--------------|----------------------------|
| MOV | r16,r16 | 1 | 1 | 0.5 | EX01 | |
| MOV | r32,r32 | 1 | 1 | 0.25 | EX01 or AG01 | |
| MOV | r64,r64 | 1 | 1 | 0.25 | EX01 or AG01 | |
| MOV | r,i | 1 | 1 | 0.25 | EX01 or AG01 | |
| MOV | r,m | 1 | 3 | 0.5 | AG01 | all addr. modes |
| MOV | m,r | 1 | 4 | 1 | EX01 AG01 | all addr. modes |
| MOV | m,i | 1 | | 1 | | |
| MOVNTI | m,r | 1 | 4 | 1 | | |
| MOVZX, MOVSX | r,r | 1 | 1 | 0.25 | | |
| MOVSX | r,m | 1 | 5 | 0.5 | EX01 | |
| MOVZX | r,m | 1 | 4 | 0.5 | EX01 | |
| MOVSXD | r64,r32 | 1 | 1 | 0.25 | EX01 | |
| MOVSXD | r64,m32 | 1 | 5 | 0.5 | EX01 | |
| CMOVcc | r,r | 1 | 2 | 0.5 | EX01 | |
| CMOVcc | r,m | 1 | | 0.5 | EX01 | |
| XCHG | r8,r8 | 2 | 1 | 1 | EX01 | |
| XCHG | r16,r16 | 2 | 1 | 1 | EX01 | |
| XCHG | r32,r32 | 2 | 1 | 0.5 | EX01 | |
| XCHG | r64,r64 | 2 | 1 | 0.5 | EX01 | |
| XCHG | r,m | 2 | ~50 | ~50 | EX01 | Timing depends on hardware |
| XLAT | | 2 | 6 | 2 | | |
| PUSH | r | 1 | | 1 | | |
| PUSH | i | 1 | | 1 | | |
| PUSH | m | 2 | | 1 | | |
| PUSHF(D/Q) | | 8 | | 4 | | |
| PUSHA(D) | | 9 | | 9 | | |
| POP | r | 1 | | 0.5 | | |
| POP | m | 2 | | 1 | | |
| POPF(D/Q) | | 34 | | 12 | | |
| POPA(D) | | 14 | | 8 | | |
| POP | sp | 1 | 1 | 2 | | |
| LEA | r16,[m] | 2 | 2-3 | | EX01 | |
| LEA | r32/64,[m] | 1 | 1 | 0.5 | EX01 | all other cases |
| LAHF | | 4 | 2 | 2 | | |
| SAHF | | 2 | 2 | 1 | | |
| SALC | | 1 | 1 | 1 | | |
| BSWAP | r | 1 | 1 | 0.5 | EX01 | |
| PREFETCHNTA | m | 1 | | 0.5 | | |
| PREFETCHT0/1/2 | m | 1 | | 0.5 | | |
| PREFETCH/W | m | 1 | | 0.5 | | PREFETCHW |
| SFENCE | | 7 | | ~100 | | |
| LFENCE | | 1 | | 0.25 | | |
| MFENCE | | 7 | | ~100 | | |
| Arithmetic instructions | | | | | | |
| ADD, SUB | r,r | 1 | 1 | 0.25 | | |
| ADD, SUB | r,i | 1 | 1 | 0.25 | | |
| ADD, SUB | r,m | 1 | | 0.5 | EX01 | |
| ADD, SUB | m,r | 1 | 7 | 1 | EX01 | |
| ADD, SUB | m,i | 1 | 7 | 1 | EX01 | |
| ADC, SBB | r,r | 1 | 1 | | EX01 | |
| ADC, SBB | r,i | 1 | 1 | | EX01 | |

Excavator

| | | | | | | |
|---------------------------|-------------|----|-------|------|------|------------------------|
| ADC, SBB | r,m | 1 | 1 | 1 | EX01 | approximate timings |
| ADC, SBB | m,r | 1 | 9 | 1 | EX01 | |
| ADC, SBB | m,i | 1 | 9 | 1 | EX01 | |
| CMP | r,r | 1 | 1 | 0.25 | EX01 | |
| CMP | r,i | 1 | 1 | 0.25 | EX01 | |
| CMP | r,m | 1 | | 0.5 | EX01 | |
| CMP | m,i | 1 | | 0.5 | EX01 | |
| INC, DEC, NEG | r | 1 | 1 | 0.25 | EX01 | |
| INC, DEC, NEG | m | 1 | 7 | 1 | EX01 | |
| AAA, AAS | | 10 | 5 | | | |
| DAA | | 16 | 8 | | | |
| DAS | | 20 | 9 | | | |
| AAD | | 4 | 5 | | | |
| AAM | | 10 | 15 | 15 | | |
| MUL, IMUL | r8/m8 | 1 | 3 | 2 | EX1 | |
| MUL, IMUL | r16/m16 | 2 | 3 | 2 | EX1 | |
| MUL, IMUL | r32/m32 | 1 | 3 | 2 | EX1 | |
| MUL, IMUL | r64/m64 | 1 | 5 | 4 | EX1 | |
| IMUL | r16,r16/m16 | 1 | 3 | 2 | EX1 | |
| IMUL | r32,r32/m32 | 1 | 3 | 2 | EX1 | |
| IMUL | r64,r64/m64 | 1 | 5 | 4 | EX1 | |
| IMUL | r16,(r16),i | 2 | 5 | 2 | EX1 | |
| IMUL | r32,(r32),i | 1 | 4 | 2 | EX1 | |
| IMUL | r64,(r64),i | 1 | 6 | 4 | EX1 | |
| IMUL | r16,m16,i | 2 | | 2 | EX1 | |
| IMUL | r32,m32,i | 2 | 4 | 2 | EX1 | |
| IMUL | r64,m64,i | 2 | 7 | 4 | EX1 | |
| DIV | r8/m8 | 9 | 17-19 | | EX0 | approximate timings |
| DIV | r16/m16 | 7 | 16-26 | | EX0 | |
| DIV | r32/m32 | 2 | 14-42 | | EX0 | |
| DIV | r64/m64 | 2 | 14-74 | | EX0 | |
| IDIV | r8/m8 | 9 | 15-19 | | EX0 | |
| IDIV | r16/m16 | 7 | 16-26 | | EX0 | |
| IDIV | r32/m32 | 2 | 14-42 | | EX0 | |
| IDIV | r64/m64 | 2 | 15-74 | | EX0 | |
| CBW, CWDE, CDQE | | 1 | 1 | | EX01 | |
| CDQ, CQO | | 1 | 1 | 0.5 | EX01 | |
| CWD | | 2 | 1 | 1 | EX01 | |
| Logic instructions | | | | | | BMI1 |
| AND, OR, XOR | r,r | 1 | 1 | 0.33 | | |
| AND, OR, XOR | r,i | 1 | 1 | 0.33 | | |
| AND, OR, XOR | r,m | 1 | | 0.5 | EX01 | |
| AND, OR, XOR | m,r | 1 | 7 | 1 | EX01 | |
| AND, OR, XOR | m,i | 1 | 7 | 1 | EX01 | |
| TEST | r,r | 1 | 1 | 0.25 | | |
| TEST | r,i | 1 | 1 | 0.25 | | |
| TEST | m,r | 1 | | 0.5 | EX01 | |
| TEST | m,i | 1 | | 0.5 | EX01 | |
| NOT | r | 1 | 1 | 0.33 | | BMI1 |
| NOT | m | 1 | 7 | 1 | EX01 | |
| ANDN | r,r,r | 1 | 1 | 0.33 | | |
| SHL, SHR, SAR | r,i/CL | 1 | 1 | 0.5 | EX01 | |

Excavator

| | | | | | | |
|--------------------------------------|------------|----|---|------|------|--------------|
| ROL, ROR | r,i/CL | 1 | 1 | 0.5 | EX01 | |
| RCL | r,1 | 1 | 1 | | EX01 | |
| RCL | r,i | 16 | 4 | | EX01 | |
| RCL | r,cl | 17 | 5 | | EX01 | |
| RCR | r,1 | 1 | 1 | | EX01 | |
| RCR | r,i | 15 | 5 | | EX01 | |
| RCR | r,cl | 16 | 5 | | EX01 | |
| SHLD, SHRD | r,r,i | 6 | 3 | 3 | EX01 | |
| SHLD, SHRD | r,r,cl | 7 | 3 | 3 | EX01 | |
| SHLD, SHRD | m,r,i/CL | 8 | | 4 | EX01 | |
| BT | r,r/i | 1 | 1 | 0.5 | EX01 | |
| BT | m,i | 1 | | 0.5 | EX01 | |
| BT | m,r | 7 | | 4 | EX01 | |
| BTC, BTR, BTS | r,r/i | 2 | 2 | 1 | EX01 | |
| BTC, BTR, BTS | m,i | 4 | | 2 | EX01 | |
| BTC, BTR, BTS | m,r | 10 | | 4 | EX01 | |
| BSF | r,r | 6 | 3 | 3 | EX01 | |
| BSF | r,m | 8 | 4 | 4 | EX01 | |
| BSR | r,r | 7 | 4 | 4 | EX01 | |
| BSR | r,m | 9 | | 5 | EX01 | |
| SETcc | r | 1 | 1 | 0.5 | EX01 | |
| SETcc | m | 1 | | 1 | EX01 | |
| CLC, STC | | 1 | | 0.25 | | |
| CMC | | 1 | 1 | | EX01 | |
| CLD | | 2 | | 3 | | |
| STD | | 2 | | 4 | | |
| POPCNT | r,r | 1 | 4 | 1 | | SSE4.2 |
| LZCNT | r,r | 2 | 2 | 2 | EX0 | LZCNT |
| TZCNT | r,r | 2 | 2 | 2 | | BMI1 |
| BEXTR | r,r,r | 1 | 1 | 0.5 | | BMI1 |
| BEXTR | r,r,i | 1 | 1 | 0.5 | | AMD TBM |
| BLSI | r,r | 2 | 2 | 0.5 | | BMI1 |
| BLSMSK | r,r | 2 | 2 | 0.5 | | BMI1 |
| BLSR | r,r | 2 | 2 | 0.5 | | BMI1 |
| BLCFILL | r,r | 2 | 2 | 0.5 | | AMD TBM |
| BLCI | r,r | 2 | 2 | 0.5 | | AMD TBM |
| BLCIC | r,r | 2 | 2 | 0.5 | | AMD TBM |
| BLCMSK | r,r | 2 | 2 | 0.5 | | AMD TBM |
| BLCS | r,r | 2 | 2 | 0.5 | | AMD TBM |
| BLSFILL | r,r | 2 | 2 | 0.5 | | AMD TBM |
| BLSI | r,r | 2 | 2 | 0.5 | | AMD TBM |
| BLSIC | r,r | 2 | 2 | 0.5 | | AMD TBM |
| T1MSKC | r,r | 2 | 2 | 0.5 | | AMD TBM |
| TZMSK | r,r | 2 | 2 | 0.5 | | AMD TBM |
| Control transfer instructions | | | | | | |
| JMP | short/near | 1 | | 2 | EX1 | |
| JMP | r | 1 | | 2 | EX1 | |
| JMP | m | 1 | | 2 | EX1 | |
| Jcc | short/near | 1 | | 1-2 | EX1 | 2 if jumping |
| fused CMP+Jcc | short/near | 1 | | 1-2 | EX1 | 2 if jumping |
| J(E/R)CXZ | short | 1 | | 1-2 | EX1 | 2 if jumping |
| LOOP | short | 1 | | 1-2 | EX1 | 2 if jumping |

Excavator

| | | | | | | |
|----------------------------|----------|------------|-----|------------|------|--------------|
| LOOPE LOOPNE | short | 1 | | 1-2 | EX1 | 2 if jumping |
| CALL | near | 2 | | 2 | EX1 | |
| CALL | r | 2 | | 2 | EX1 | |
| CALL | m | 3 | | 2 | EX1 | |
| RET | | 1 | | 2 | EX1 | |
| RET | i | 4 | | 2 | EX1 | |
| BOUND | m | 11 | | 4 | | for no jump |
| INTO | | 4 | | 2 | | for no jump |
| String instructions | | | | | | |
| LODS | | 3 | | 3 | | |
| REP LODS | m8/m16 | 6n | | 2n – 3n | | |
| STOS | | 3 | | 3 | | |
| REP STOS | | 1n | | ~1n | | small n |
| REP STOS | | 3 per 16B | | 2 per 16B | | best case |
| MOVS | | 5 | | 3 | | |
| REP MOVS | | ~1n | | ~1n | | small n |
| REP MOVS | | 4-5 pr 16B | | ~2 per 16B | | best case |
| SCAS | | 3 | | 3 | | |
| REP SCAS | | 7n | | 3-4n | | |
| CMPS | | 6 | | 3 | | |
| REP CMPS | | 9n | | 4n | | |
| Synchronization | | | | | | |
| LOCK ADD | m,r | 1 | ~35 | | | |
| XADD | m,r | 4 | ~10 | | | |
| LOCK XADD | m,r | 4 | ~35 | | | |
| CMPXCHG | m,r8 | 5 | 11 | | | |
| CMPXCHG | m,r16 | 6 | 11 | | | |
| CMPXCHG | m,r32/64 | 6 | 12 | | | |
| LOCK CMPXCHG | m8,r8 | 5 | ~40 | | | |
| LOCK CMPXCHG | m16,r16 | 6 | ~40 | | | |
| LOCK CMPXCHG | m,r32/64 | 6 | ~40 | | | |
| CMPXCHG8B | m64 | 18 | ~14 | | | |
| LOCK CMPXCHG8B | m64 | 18 | ~42 | | | |
| CMPXCHG16B | m128 | 24 | ~40 | | | |
| LOCK CMPXCHG16B | m128 | 24 | ~80 | | | |
| Other | | | | | | |
| NOP (90) | | 1 | | 0.25 | none | |
| Long NOP (0F 1F) | | 1 | | 0.25 | none | |
| PAUSE | | 8 | | 4 | | |
| ENTER | a,0 | 13 | | 21 | | |
| ENTER | a,b | 11+5b | | 20-30 | | |
| LEAVE | | 2 | | 3 | | |
| CPUID | | 38-64 | | 100-300 | | |
| XGETBV | | 4 | | 30 | | |
| RDTSC | | 44 | | 78 | | |
| RDTSCP | | 44 | | 105 | | rdtscp |
| RDPNC | | 22 | | 380 | | |
| CRC32 | r32,r8 | 3 | 3 | 2 | | |
| CRC32 | r32,r16 | 5 | 5 | 5 | | |
| CRC32 | r32,r32 | 7 | 6 | 6 | | |

Floating point x87 instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Domain, notes |
|--------------------------------|----------|--------|---------|-----------------------|-----------------|---------------|
| Move instructions | | | | | | |
| FLD | r | 1 | 2 | 0.5 | P01 | fp |
| FLD | m32/64 | 1 | 6 | 1 | | fp |
| FLD | m80 | 9 | 15 | 3 | | fp |
| FBLD | m80 | 61 | 52 | 34 | P0 P1 P2 | fp |
| FST(P) | r | 1 | 2 | 0.5 | P01 | fp |
| FST(P) | m32/64 | 2 | 6 | 1 | | fp |
| FSTP | m80 | 13 | 14 | 19 | | fp |
| FBSTP | m80 | 240 | 222 | 222 | P0 P1 P2 | fp |
| FXCH | r | 1 | 0 | 0.5 | P01 | inherit |
| FILD | m | 1 | 11 | 1 | P01 | fp |
| FIST(T)(P) | m | 2 | 7 | 1 | P0 P2 | fp |
| FLDZ, FLD1 | | 1 | | 0.5 | P01 | fp |
| FCMOVcc | st0,r | 8 | 3 | 3 | P0 P1 P2 | fp |
| FFREE | r | 1 | | 0.25 | none | |
| FINCSTP, FDECSTP | | 1 | 0 | 0.25 | none | inherit |
| FNSTSW | AX | 3 | 11 | 19 | P0 P2 | |
| FNSTSW | m16 | 2 | | 15 | P0 P2 | |
| FLDCW | m16 | 1 | | 3 | | |
| FNSTCW | m16 | 2 | | 2 | | |
| Arithmetic instructions | | | | | | |
| FADD(P),FSUB(R)(P) | r/m | 1 | 5 | 1 | P01 | fma |
| FIADD,FISUB(R) | m | 2 | | 2 | P01 | fma |
| FMUL(P) | r/m | 1 | 5 | 1 | P01 | fma |
| FIMUL | m | 2 | | 2 | P01 | fma |
| FDIV(R)(P) | r | 1 | 9-37 | 4-16 | P01 | fp |
| FDIV(R) | m | 1 | | | P01 | fp |
| FIDIV(R) | m | 2 | | 4 | P01 | fp |
| FABS, FCHS | | 1 | 2 | 0.5 | P01 | fp |
| FCOM(P), FUCOM(P) | r/m | 1 | | 0.5 | P01 | fp |
| FCOMPP, FUCOMPP | | 1 | | 0.5 | P01 | fp |
| FCOMI(P) | r | 2 | 2 | 1 | P01 P2 | fp |
| FICOM(P) | m | 2 | | 1 | P01 | fp |
| FTST | | 1 | | 0.5 | P01 | fp |
| FXAM | | 1 | 26 | 0.5 | P01 | fp |
| FRNDINT | | 1 | 4 | 1 | P0 | fp |
| FPREM FPREM1 | | 1 | 17-60 | 12-53 | P0 | fp |
| Math | | | | | | |
| FSQRT | | 1 | 10-20 | 5-20 | P01 | |
| FLDPI, etc. | | 1 | | 0.5 | P01 | |
| FSIN | | 10-164 | 60-210 | 60-165 | P0 P1 P2 | |
| FCOS | | 18-166 | 76-158 | | P0 P1 P2 | |
| FSINCOS | | 12-168 | | 90-165 | P0 P1 P2 | |
| FPTAN | | 11-192 | 90-245 | 90-210 | P0 P1 P2 | |
| FPATAN | | 10-365 | 60-440 | 60-365 | P0 P1 P2 | |
| FSCALE | | 8 | 15 | 4 | P0 P1 P2 | |

Excavator

| | | | | | | |
|--------------|------|-------|--------|------|----------|--|
| EXTRACT | | 13 | 8 | 5 | P0 P1 P2 | |
| F2XM1 | | 10-18 | 60-74 | | P0 P1 P2 | |
| FYL2X | | 9-183 | 60-280 | | P0 P1 P2 | |
| FYL2XP1 | | 206 | ~390 | | P0 P1 P2 | |
| Other | | | | | | |
| FNOP | | 1 | | 0.25 | none | |
| (F)WAIT | | 1 | | 0.25 | none | |
| FNCLEX | | 20 | | 63 | P0 | |
| FNINIT | | 34 | | 150 | P0 | |
| FNSAVE | m864 | 100 | ~250 | ~250 | P0 P1 P2 | |
| FRSTOR | m864 | 78 | ~150 | ~150 | P0 P2 | |

Integer vector instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Notes |
|--------------------------|------------|-----|---------|-----------------------|-----------------|--------------------|
| Move instructions | | | | | | |
| MOVD | r32, mm | 1 | 7 | 1 | | |
| MOVD | mm, r32 | 2 | 8 | 1 | | |
| MOVD/Q | r32/64, x | 1 | 7 | 1 | | |
| MOVD/Q | x, r32/64 | 2 | 9 | 1 | | |
| MOVD | mm/x, m32 | 1 | 5 | 0.5 | | |
| MOVD | m32, mm/x | 1 | 4 | 1 | P2 | |
| MOVQ | mm/x, mm/x | 1 | 2 | 0.5 | | |
| MOVQ | mm/x, m64 | 1 | 5 | 0.5 | | |
| MOVQ | m64, mm/x | 1 | 4 | 1 | P2 | |
| MOVDQA | x, x | 1 | 0 | 0.25 | | renaming |
| MOVDQA | x, m | 1 | 5 | 0.5 | | |
| MOVDQA | m, x | 1 | 4 | 1 | P2 | |
| VMOVDQA | y, y | 2 | 2 | 0.5 | | lower half renamed |
| VMOVDQA | y, m | 2 | 5 | 1 | | |
| VMOVDQA | m, y | 2 | 4 | 2 | P2 | |
| MOVDQU | x, x | 1 | 0 | 0.25 | | renaming |
| MOVDQU | x, m | 1 | 5 | 0.5 | | |
| MOVDQU | m, x | 1 | 4 | 1 | P2 | |
| LDDQU | x, m | 1 | 5 | 0.5 | | |
| VMOVDQU | y, m | 2 | 5 | 1 | | |
| VMOVDQU | m, y | 2 | 4 | 2 | P2 | |
| MOVDQ2Q | mm, x | 1 | 2 | 0.5 | | |
| MOVQ2DQ | x, mm | 1 | 2 | 0.5 | | |
| MOVNTQ | m, mm | 1 | 4 | 1 | P2 | |
| MOVNTDQ | m, x | 1 | 4 | 1 | P2 | |
| VMOVNTDQA | m, y | 2 | 5 | 2 | | |
| MOVNTDQA | x, m | 1 | 4 | 0.5 | | |
| VMOVNTDQA | y, m | 2 | 5 | 1 | | |
| PACKSSWB/DW | mm/x, r/m | 1 | 2 | 1 | P1 | |
| PACKUSWB | mm/x, r/m | 1 | 2 | 1 | P1 | |
| VPACKSSWB/DW | y, r/m | 2 | 2 | 2 | P1 | |
| VPACKUSWB | y, r/m | 2 | 2 | 2 | P1 | |
| PUNPCKH/LBW/WD/DQ | mm/x, r/m | 1 | 2 | 1 | P1 | |

Excavator

| | | | | | | |
|-----------------------------|-------------|----|-------|-----|-------|-----------|
| PUNPCKL/HQDQ | x,r/m | 1 | 2 | 1 | P1 | |
| VPUNPCKL/HQDQ | y,r/m | 2 | 2 | 2 | P1 | |
| PSHUFB | mm/x,r/m | 1 | 2 | 1 | P1 | |
| VPSHUFB | y,r/m | 2 | 2 | 2 | P1 | |
| PSHUFD | x,x,i | 1 | 2 | 1 | P1 | |
| VPSHUFD | y,y,i | 2 | 2 | 2 | P1 | |
| PSHUFW | mm,mm,i | 1 | 2 | 1 | P1 | |
| PSHUFL/HW | x,x,i | 1 | 2 | 1 | P1 | |
| VPSHUFL/HW | y,y,i | 2 | 2 | 2 | P1 | |
| PALIGNR | mm/x,r/m,i | 1 | 2 | 1 | P1 | |
| VPALIGNR | y,r/m,i | 2 | 2 | 2 | P1 | |
| PBLENDW | x,r/m,i | 1 | 2 | 0.5 | P02 | |
| PBLENDW | y,r/m,i | 2 | 2 | 1 | P02 | |
| MASKMOVQ | mm,mm | 31 | ~50 | ~50 | | |
| MASKMOVDQU | x,x | 65 | ~50 | ~50 | | |
| VPMASKMOVD/Q | x,x,m | 1 | 6 | 0.5 | | |
| VPMASKMOVD/Q | y,y,m | 2 | 6 | 1 | | |
| VPMASKMOVD/Q | m,x,x | 2 | 10-40 | 30 | | |
| VPMASKMOVD/Q | m,y,y | 4 | 10-40 | 60 | | |
| PMOVMASKB | r32,mm/x | 2 | 8 | 1 | P12 | |
| VPMOVMASKB | r32,y | 2 | 9 | 2 | P12 | |
| PEXTRB/W/D/Q | r,x/mm,i | 2 | 9 | 1 | P1 P2 | |
| PINSRB/W/D/Q | x/mm,r,i | 2 | 10 | 1 | P1 | |
| EXTRQ | x,i,i | 1 | 2 | 1 | P1 | AMD SSE4A |
| EXTRQ | x,x | 1 | 2 | 1 | P1 | AMD SSE4A |
| INSERTQ | x,x,i,i | 1 | 1 | 1 | P1 | AMD SSE4A |
| INSERTQ | x,x | 1 | 2 | 1 | P1 | AMD SSE4A |
| PMOVSXBW/BD/BQ/ WD/WQ/DQ | x,x | 1 | 2 | 1 | P1 | |
| PMOVSXBW/BD/BQ/ WD/WQ/DQ | y,x | 3 | 3 | 3 | P1 | |
| PMOVZXBW/BD/BQ/ WD/WQ/DQ | x,x | 1 | 2 | 1 | P1 | |
| PMOVZXBW/BD/BQ/ WD/WQ/DQ | y,x | 3 | 3 | 3 | P1 | |
| VINSERTI128 | y,y,x,i | 2 | 2 | 1 | P0 P2 | AVX2 |
| VINSERTI128 | y,y,m,i | 2 | 7 | 1 | P0 P2 | AVX2 |
| VPBROADCAST B/W/D/Q | x,x | 1 | 2 | 1 | P1 | AVX2 |
| VPBROADCAST B/W/D/Q | x,m | | | | | |
| VPBROADCAST B/W/D/Q | y,x | 2 | 3 | 2 | P1 | AVX2 |
| VPBROADCAST B/W/D/Q | y,m | | | | | |
| VBROADCASTI128 | y,m128 | 2 | 5 | 0.5 | P02 | AVX2 |
| VPGATHERDD | x,[r+s*x],x | 34 | | 15 | | AVX2 |
| VPGATHERDD | y,[r+s*y],y | 51 | | 21 | | AVX2 |
| VPGATHERQD | x,[r+s*x],x | 25 | | 13 | | AVX2 |
| VPGATHERQD | x,[r+s*y],x | 32 | | 14 | | AVX2 |
| VPGATHERDQ | x,[r+s*x],x | 24 | | 13 | | AVX2 |
| VPGATHERDQ | y,[r+s*x],y | 32 | | 15 | | AVX2 |
| VPGATHERQQ | x,[r+s*x],x | 24 | | 49 | | AVX2 |
| VPGATHERQQ | y,[r+s*y],y | 31 | | 15 | | AVX2 |

Excavator

| Arithmetic instructions | | | | | | |
|---|----------|----|---|-----|----------|--|
| PADDB/W/D/Q/SB/SW/USB/USW | mm/x,r/m | 1 | 2 | 0.5 | P02 | |
| VPADD.. | y,y,r/m | 2 | 2 | 1 | P02 | |
| PSUBB/W/D/Q/SB/SW/USB/USW | mm/x,r/m | 1 | 2 | 0.5 | P02 | |
| VPSUB.. | y,y,r/m | 2 | 2 | 1 | P02 | |
| PHADD/SUB(S)W/D | mm,mm | 3 | 4 | 2 | P0 P1 P2 | |
| PHADD/SUB(S)W/D | x,x | 4 | 4 | 2 | P0 P1 P2 | |
| VPHADD/SUB(S)W/D | y,y,y | 8 | 4 | 4 | P0 P1 P2 | |
| PCMPEQ B/W/D/Q | mm/x,r/m | 1 | 2 | 0.5 | P02 | |
| VPCMPEQ B/W/D/Q | y,y,r/m | 2 | 2 | 1 | P02 | |
| PCMPGT B/W/D/Q | mm/x,r/m | 1 | 2 | 0.5 | P02 | |
| VPCMPGT B/W/D/Q | y,y,r/m | 2 | 2 | 1 | P02 | |
| PMULLW PMULHW PMULHUW PMULLD PMULDQ PMULUDQ | mm/x,r/m | 1 | 4 | 1 | P0 | |
| VPMULLW VPMULHW PMULLD VPMULHUW VPMULDQ | y,y,r/m | 2 | 4 | 2 | P0 | |
| VPMULUDQ | mm/x,r/m | 1 | 4 | 1 | P0 | |
| PMULHSW | y,y,r/m | 2 | 4 | 2 | P0 | |
| VPMULHSW | mm/x,r/m | 1 | 4 | 1 | P0 | |
| PMADDWD | y,y,r/m | 2 | 4 | 2 | P0 | |
| VPMADDWD | mm/x,r/m | 1 | 4 | 1 | P0 | |
| PMADDUBSW | y,y,r/m | 2 | 4 | 2 | P0 | |
| VPMADDUBSW | mm/x,r/m | 1 | 4 | 1 | P0 | |
| PAVGB/W | mm/x,r/m | 1 | 2 | 0.5 | P02 | |
| VPAVGB/W | mm/x,r/m | 2 | 2 | 1 | P02 | |
| PMIN/MAX SB/SW/ SD UB/UW/UD | mm/x,r/m | 1 | 2 | 0.5 | P02 | |
| VPMIN/MAX SB/SW/ SD UB/UW/UD | y,y,r/m | 2 | 2 | 1 | P02 | |
| PHMINPOSUW | x,r/m | 2 | 3 | 1 | P02 P1 | |
| PABSB/W/D | mm/x,r/m | 1 | 2 | 0.5 | P02 | |
| VPABSB/W/D | y,r/m | 2 | 2 | 1 | P02 | |
| PSIGNB/W/D | mm/x,r/m | 1 | 2 | 0.5 | P02 | |
| VPSIGNB/W/D | y,r/m | 2 | 2 | 1 | P02 | |
| PSADBW | mm/x,r/m | 2 | 4 | 1 | P02 | |
| VPSADBW | y,y,r/m | 4 | 4 | 2 | P02 | |
| MPSADBW | x,x,i | 12 | 7 | 4 | P0 P1 P2 | |
| VMPSADBW | y,y,y,i | 8 | 7 | 8 | P0 P1 P2 | |
| Logic | | | | | | |
| PAND PANDN POR PXOR | mm/x,r/m | 1 | 2 | 0.5 | P02 | |
| VPAND VPANDN VPOR VPXOR | y,y,r/m | 2 | 2 | 1 | P02 | |
| PSLL/RL W/D/Q PSRAW/D | mm/x,r/m | 1 | 2 | 1 | P1 | |
| VPSLL/RL W/D/Q VP- SRAW/D | y,y,x/m | 2 | 2 | 2 | P1 | |
| PSLL/RL W/D/Q PSRAW/D | mm/x,i | 1 | 2 | 1 | P1 | |

Excavator

| | | | | | | |
|----------------------------|---------|----|----|------|-------|--------|
| VPSLL/RL W/D/Q VP-SRAW/D | y,y,i | 2 | 2 | 2 | P1 | |
| PSLLDQ, PSRLDQ | x,i | 1 | 2 | 1 | P1 | |
| VPSLLDQ VPSRLDQ | y,y,i | 2 | 2 | 2 | P1 | |
| VPSLLVD/Q | | | | | | |
| VPSRAVD | | | | | | |
| VPSRLVD/Q | x,x,x | 1 | 2 | 1 | P1 | AVX2 |
| VPSLLVD/Q | | | | | | |
| VPSRAVD | | | | | | |
| VPSRLVD/Q | y,y,y | 2 | 2 | 2 | P1 | AVX2 |
| PTEST | x,r/m | 2 | | 1 | P1 P2 | |
| VPTEST | y,y/m | 4 | | 2 | P1 P2 | |
| String instructions | | | | | | |
| PCMPESTRI | x,x,i | 31 | 42 | 9 | | SSE4.2 |
| PCMPESTRI | x,m,i | 31 | | | | SSE4.2 |
| PCMPESTRM | x,x,i | 30 | 37 | 8 | | SSE4.2 |
| PCMPESTRM | x,m,i | 31 | | | | SSE4.2 |
| PCMPISTRI | x,x,i | 10 | 21 | 6 | | SSE4.2 |
| PCMPISTRI | x,m,i | 9 | | | | SSE4.2 |
| PCMPISTRM | x,x,i | 8 | 26 | 5 | | SSE4.2 |
| PCMPISTRM | x,m,i | 10 | | | | SSE4.2 |
| Encryption | | | | | | |
| PCLMULQDQ | x,x/m,i | 4 | 5 | 5 | | pclmul |
| AESDEC | x,x | 2 | 4 | 1 | P01 | aes |
| AESDECLAST | x,x | 2 | 4 | 1 | P01 | aes |
| AESENC | x,x | 2 | 5 | 1 | P01 | aes |
| AESENCLAST | x,x | 2 | 4 | 1 | P01 | aes |
| AESIMC | x,x | 1 | 4 | 1 | P0 | aes |
| AESKEYGENASSIST | x,x,i | 1 | 4 | 1 | P0 | aes |
| Other | | | | | | |
| EMMS | | 1 | | 0.25 | | |

Floating point vector instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Domain, notes |
|--------------------------|----------|-----|---------|-----------------------|-----------------|----------------|
| Move instructions | | | | | | |
| MOVAPS/D MOVUPS/D | x,x | 1 | 0 | 0.25 | none | inherit domain |
| VMOVAPS/D | y,y | 2 | 2 | 0.5 | P02 | ivec |
| MOVAPS/D MOVUPS/D | x,m128 | 1 | 5 | 0.5 | | |
| VMOVAPS/D | | | | | | |
| VMOVUPS/D | y,m256 | 2 | 5 | 1 | | |
| MOVAPS/D MOVUPS/D | m128,x | 1 | 4 | 1 | P2 | |
| VMOVAPS/D | m256,y | 2 | 4 | 2 | P2 | |
| VMOVUPS/D | m256,y | 2 | 4 | 2 | P2 | |
| MOVSS/D | x,x | 1 | 2 | 0.5 | P01 | |
| MOVSS/D | x,m32/64 | 1 | 5 | 0.5 | | |
| MOVSS/D | m32/64,x | 1 | 4 | 1 | P2 | |

Excavator

| | | | | | | |
|-------------------|------------|----|-----|-----|----------|-----------|
| MOVHPS/D | x,m64 | 1 | 6 | 1 | P2 | |
| MOVLPS/D | x,m64 | 1 | 6 | 0.5 | P01 | |
| MOVHPS/D | m64,x | 2 | 6 | 2 | P2 | |
| MOVLPS/D | m64,x | 1 | 6 | 1 | P2 | |
| MOVLHPS MOVHLPS | x,x | 1 | 2 | 1 | P2 | |
| MOVMSKPS/D | r32,x | 2 | 8 | 1 | P1 P2 | |
| VMOVMSKPS/D | r32,y | 3 | 8 | 2 | P1 P2 | |
| MOVNTPS/D | m128,x | 1 | 4 | 1 | P2 | |
| VMOVNTPS/D | m256,y | 2 | 4 | 3 | P2 | |
| MOVNTSS/SD | m,x | 2 | | | | AMD SSE4A |
| SHUFPS/D | x,x/m,i | 1 | 2 | 1 | P2 | ivec |
| VSHUFPS/D | y,y,y/m,i | 2 | 2 | 2 | P2 | ivec |
| VPERMILPS/PD | x,x,x/m | 1 | 3 | 1 | P1 | ivec |
| VPERMILPS/PD | y,y,y/m | 2 | 3 | 2 | P1 | ivec |
| VPERMILPS/PD | x,x/m,i | 1 | 2 | 1 | P1 | ivec |
| VPERMILPS/PD | y,y/m,i | 2 | 2 | 2 | P1 | ivec |
| VPERM2F128 | y,y,y,i | 8 | 4 | 3 | P0 P2 | ivec |
| VPERM2F128 | y,y,m,i | 12 | | 4 | P0 P2 | ivec |
| BLENDPS/PD | x,x/m,i | 1 | 2 | 0.5 | P01 | fp |
| VBLENDPS/PD | y,y,y/m,i | 2 | 2 | 1 | P01 | fp |
| BLENDVPS/PD | x,x/m,xmm0 | 1 | 2 | 0.5 | P01 | |
| VBLENDVPS/PD | y,y,y/m,y | 2 | 2 | 0.5 | P01 | |
| MOVDDUP | x,x | 1 | 2 | 1 | P1 | ivec |
| MOVDDUP | x,m64 | 1 | | 0.5 | | |
| VMOVDDUP | y,y | 2 | 2 | 2 | P1 | ivec |
| VMOVDDUP | y,m256 | 2 | | 1 | | |
| VBROADCASTSS | x,x | 1 | 2 | 1 | P2 | |
| VBROADCASTSS | x,m32 | 1 | 5 | 0.5 | | |
| VBROADCASTSS/SD | y,x | 2 | 3 | 2 | P2 | |
| VBROADCASTSS/SD | y,m32 | 2 | 5 | 0.5 | P02 | |
| VBROADCASTF128 | y,m128 | 2 | 5 | 0.5 | P02 | |
| MOVSH/LDUP | x,x | 1 | 2 | 1 | P1 | ivec |
| MOVSH/LDUP | x,m128 | 1 | | 0.5 | | |
| VMOVSH/LDUP | y,y | 2 | 3 | 2 | P1 | ivec |
| VMOVSH/LDUP | y,m256 | 2 | | 1 | | |
| UNPCKH/LPS/D | x,x/m | 1 | 2 | 1 | P2 | |
| VUNPCKH/LPS/D | y,y,y/m | 2 | 2 | 2 | P2 | |
| EXTRACTPS | r32,x,i | 2 | 9 | 2 | P2 | |
| EXTRACTPS | m32,x,i | 2 | 6 | 2 | P2 | |
| VEXTRACTF128 | x,y,i | 1 | 2 | 0.5 | P02 | ivec |
| VEXTRACTF128 | m128,y,i | 2 | 7 | 1 | P0 P2 | |
| INSERTPS | x,x,i | 1 | 2 | 1 | P2 | |
| INSERTPS | x,m32,i | 1 | 7 | 1 | P2 | |
| VINSERTF128 | y,y,x,i | 2 | 2 | 1 | P02 | ivec |
| VINSERTF128 | y,y,m128,i | 2 | 8 | 1 | P02 | |
| VMASKMOVPS/D | x,x,m128 | 1 | 8 | 0.5 | P01 | |
| VMASKMOVPS/D | y,y,m256 | 2 | 8 | 1 | P01 | |
| VMASKMOVPS/D | m128,x,x | 20 | ~40 | | P0 P1 P2 | |
| VMASKMOVPS/D | m256,y,y | 41 | ~40 | | P0 P1 P2 | |
| Conversion | | | | | | |
| CVTPD2PS | x,x | 2 | 6 | 1 | P0 P2 | |
| VCVTPD2PS | x,y | 4 | 6 | 2 | P0 P2 | |

Excavator

| | | | | | | |
|------------------------|----------|---|------|------|--------|----------|
| CVTPS2PD | x,x | 2 | 6 | 1 | P0 P2 | |
| VCVTPS2PD | y,x | 4 | 6 | 2 | P0 P2 | |
| CVTSD2SS | x,x | 1 | 4 | 1 | P0 | |
| CVTSS2SD | x,x | 1 | 4 | 1 | P0 | |
| CVTDQ2PS | x,x | 1 | 4 | 1 | P0 | |
| VCVTDQ2PS | y,y | 2 | 4 | 2 | P0 | |
| CVT(T) PS2DQ | x,x | 1 | 4 | 1 | P0 | |
| VCVT(T) PS2DQ | y,y | 2 | 4 | 2 | P0 | |
| CVTDQ2PD | x,x | 2 | 5 | 1 | P0 P2 | |
| VCVTDQ2PD | y,x | 4 | 5 | 2 | P0 P2 | |
| CVT(T)PD2DQ | x,x | 2 | 5 | 1 | P0 P2 | |
| VCVT(T)PD2DQ | x,y | 4 | 5 | 2 | P0 P2 | |
| CVTPI2PS | x,mm | 2 | 5 | 1 | P0 P2 | |
| CVT(T)PS2PI | mm,x | 1 | 5 | 1 | P0 | |
| CVTPI2PD | x,mm | 2 | 6 | 1 | P0 P1 | |
| CVT(T) PD2PI | mm,x | 2 | 6 | 1 | P0 P1 | |
| CVTSI2SS | x,r32 | 2 | 12 | 1 | P0 int | |
| CVT(T)SS2SI | r32,x | 2 | 12 | 1 | P0 P2 | |
| CVTSI2SD | x,r32/64 | 2 | 12 | 1 | P0 int | |
| CVT(T)SD2SI | r32/64,x | 2 | 12 | 1 | P0 P2 | |
| VCVTPS2PH | x/m,x,i | 2 | 6 | 2 | P0 P1 | F16C |
| VCVTPS2PH | x/m,y,i | 4 | 6 | 2 | P0 P1 | F16C |
| VCVTPH2PS | x,x/m | 2 | 6 | 2 | P0 P1 | F16C |
| VCVTPH2PS | y,x/m | 4 | 6 | 2 | P0 P1 | F16C |
| Arithmetic | | | | | | |
| ADDSS/D SUBSS/D | x,x/m | 1 | 5 | 1 | P01 | fma |
| ADDPS/D SUBPS/D | x,x/m | 1 | 5 | 1 | P01 | fma |
| VADDPS/D VSUBPS/D | y,y,y/m | 2 | 5 | 2 | P01 | fma |
| ADDSUBPS/D | x,x/m | 1 | 5 | 1 | P01 | fma |
| VADDSUBPS/D | y,y,y/m | 2 | 5 | 1 | P01 | fma |
| HADDPS/D HSUBPS/D | x,x | 4 | 9 | 2 | P0 P1 | ivec/fma |
| VHADDPS/D | | | | | | |
| VHSUBPS/D | y,y,y/m | 8 | 9 | 3 | P01 P1 | ivec/fma |
| MULSS MULSD | x,x/m | 1 | 5 | 0.5 | P01 | fma |
| MULPS MULPD | x,x/m | 1 | 5 | 0.5 | P01 | fma |
| VMULPS VMULPD | y,y,y/m | 2 | 5 | 1 | P01 | fma |
| DIVSS DIVPS | x,x/m | 1 | 12 | 4 | P01 | fp |
| VDIVPS | y,y,y/m | 2 | 12 | 8 | P01 | fp |
| DIVSD DIVPD | x,x/m | 1 | 8-22 | 4-8 | P01 | fp |
| VDIVPD | y,y,y/m | 2 | 8-22 | 8-16 | P01 | fp |
| RCPSS/PS | x,x/m | 1 | 5 | 1 | P01 | fp |
| VRCPPS | y,y/m | 2 | 5 | 2 | P01 | fp |
| CMPSS/D | | | | | | |
| CMPPS/D | x,x/m | 1 | 2 | 0.5 | P01 | fp |
| VCMPPS/D | y,y,y/m | 2 | 2 | 1 | P01 | fp |
| COMISS/D UCOMISS/ D | x,x/m | 2 | 8 | 1 | P01 P2 | fp |
| MAXSS/SD/PS/PD | | | | | | |
| MINSS/SD/PS/PD | x,x/m | 1 | 2 | 0.5 | P01 | fp |
| VMAXPS/D VMINPS/D | y,y,y/m | 2 | 2 | 1 | P01 | fp |

Excavator

| | | | | | | |
|--|-----------|-----|----|------|-------|-------------|
| ROUNDSS/SD/PS/PD | x,x/m,i | 1 | 4 | 1 | P0 | fp |
| VROUNDSS/SD/PS/PD | y,y/m,i | 2 | 4 | 2 | P0 | fp |
| DPPS | x,x/m,i | 9 | 20 | 4 | P0 P1 | |
| VDPPS | y,y,y,i | 13 | 22 | 5 | P0 P1 | |
| VDPPS | y,m,i | 15 | | 6 | P0 P1 | |
| DPPD | x,x,i | 7 | 14 | 4 | P0 P1 | |
| DPPD | x,m,i | 8 | | 4 | P0 P1 | |
| VFMADD132SS/SD | x,x,x/m | 1 | 5 | 0.5 | P01 | FMA3 |
| VFMADD132PS/PD | x,x,x/m | 1 | 5 | 0.5 | P01 | FMA3 |
| VFMADD132PS/PD | y,y,y/m | 2 | 5 | 1 | P01 | FMA3 |
| All other FMA3 instructions: same as above | | | | | | FMA3 |
| VFMADDSS/SD | x,x,x,x/m | 1 | 5 | 0.5 | P01 | AMD FMA4 |
| VFMADDPS/PD | x,x,x,x/m | 1 | 5 | 0.5 | P01 | AMD FMA4 |
| VFMADDPS/PD | y,y,y,y/m | 2 | 5 | 1 | P01 | AMD FMA4 |
| All other FMA4 instructions: same as above | | | | | | AMD FMA4 |
| Math | | | | | | |
| SQRTSS/PS | x,x/m | 1 | 13 | 4 | P01 | fp |
| VSQRTPS | y,y/m | 2 | 13 | 9 | P01 | fp |
| SQRTSD/PD | x,x/m | 1 | 16 | 4-9 | P01 | fp |
| VSQRTPD | y,y/m | 2 | 16 | 8-19 | P01 | fp |
| RSQRTSS/PS | x,x/m | 1 | 4 | 1 | P01 | fp |
| VRSQRTPS | y,y/m | 2 | 4 | 2 | P01 | fp |
| VFRCZSS/SD/PS/PD | x,x | 2 | 38 | 4 | P01 | AMD XOP |
| VFRCZSS/SD/PS/PD | x,m | 3 | | | P01 | AMD XOP |
| Logic | | | | | | |
| AND/ANDN/OR/XORPS/PD | x,x/m | 1 | 2 | 0.5 | P02 | ivec |
| VAND/ANDN/OR/XORPS/PD | y,y,y/m | 2 | 2 | 1 | P02 | ivec |
| Other | | | | | | |
| VZEROUPPER | | 10 | | 4 | | 32 bit mode |
| VZEROUPPER | | 17 | | 5 | | 64 bit mode |
| VZEROALL | | 18 | | 6 | P02 | 32 bit mode |
| VZEROALL | | 33 | | 9 | P02 | 64 bit mode |
| LDMXCSR | m32 | 9 | | ~50 | | |
| STMXCSR | m32 | 2 | | ~50 | | |
| FXSAVE | m4096 | 76 | | 100 | | 32 bit mode |
| FXSAVE | m4096 | 84 | | 180 | | 64 bit mode |
| FXRSTOR | m4096 | 124 | | 250 | | 32 bit mode |
| FXRSTOR | m4096 | 129 | | 150 | | 64 bit mode |
| XSAVE | m | 139 | | 215 | | 32 bit mode |
| XSAVE | m | 155 | | 170 | | 64 bit mode |
| XRSTOR | m | 204 | | 350 | | 32 bit mode |
| XRSTOR | m | | | | | 64 bit mode |

AMD Ryzen

List of instruction timings and macro-operation breakdown

Explanation of column headings:

| | |
|-------------------------------|---|
| Instruction: | Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc. |
| Operands: | i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, x = 128 bit xmm register, y = 256 bit ymm register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc. |
| Ops: | Number of macro-operations issued from instruction decoder to schedulers. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. The latency listed does not include the memory operand where the listing for register and memory operand are joined (r/m). |
| Reciprocal throughput: | This is also called issue latency. This value indicates the average number of clock cycles from the execution of an instruction begins to a subsequent independent instruction of the same kind can begin to execute. A value of 1/3 indicates that the execution units can handle 3 instructions per clock cycle in one thread. However, the throughput may be limited by other bottlenecks in the pipeline. |
| Execution pipe: | Indicates which execution pipe or unit is used for the macro-operations: P0: Floating point and vector pipe 0 P1: Floating point and vector pipe 1 P2: Floating point and vector pipe 2 P3: Floating point and vector pipe 3 P0 P1: Uses both P0 and P1 P01: Uses either P0 and P1 Where no unit is specified, it uses one or more integer pipe or address generation units. Two micro-operations can execute simultaneously if they go to different execution pipes |
| Domain: | Tells which execution unit domain is used: ivec: integer vector execution unit. fp: floating point execution unit. inherit: the output operand inherits the domain of the input operand. There is an additional latency of 1 clock cycle if the output of an ivec instruction goes to the input of a fp instruction, and when the output of a fp instruction goes to the input of an ivec instruction. All other latencies after memory load and before memory store instructions are included in the latency counts. |

Integer instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Notes |
|--------------------------|----------|-----|---------|-----------------------|-----------------|----------|
| Move instructions | | | | | | |
| MOV | r8,r8 | 1 | 1 | 0.25 | | |
| MOV | r16,r16 | 1 | 1 | 0.25 | | |
| MOV | r32,r32 | 1 | 0 | 0.3 | | renaming |
| MOV | r64,r64 | 1 | 0 | 0.2 | | renaming |
| MOV | r8,i | 1 | | 0.25 | | |
| MOV | r16,i | 1 | | 0.25 | | |
| MOV | r32,i | 1 | | 0.25 | | |
| MOV | r64,i | 1 | | 0.25 | | |
| MOV | r,m | 1 | 3 | 0.5 | | |
| MOV | m,r | 1 | 4 | 1 | | |

Ryzen

| | | | | | |
|--------------------------------|------------|----|------|------|-----------------|
| MOV | m,i | 1 | | 1 | |
| MOVNTI | m,r | 1 | high | 1 | |
| MOVZX, MOVZX | r,r | 1 | 1 | 0.25 | |
| MOVZX, MOVZX | r,m | 1 | 3 | 0.5 | |
| MOVSXD | r64,r32 | 1 | 1 | 0.25 | |
| MOVSXD | r64,m32 | 1 | 3 | 0.5 | |
| CMOVcc | r,r | 1 | 1 | 0.25 | |
| CMOVcc | r,m | 1 | | 0.5 | |
| XCHG | r8,r8 | 2 | 1 | 1 | |
| XCHG | r16,r16 | 2 | 1 | 1 | |
| XCHG | r32,r32 | 2 | 0 | 0.33 | renaming |
| XCHG | r64,r64 | 2 | 0 | 0.33 | renaming |
| XCHG | r,m | 2 | ~30 | ~30 | depends on hw |
| XLAT | | 2 | 5 | 2 | |
| PUSH | r | 1 | | 1 | |
| PUSH | i | 1 | | 1 | |
| PUSH | m | 2 | | 1 | |
| PUSHF(D/Q) | | 9 | | 4 | |
| PUSH | sp | 2 | | 1 | |
| PUSHA(D) | | 9 | | 8 | |
| POP | r | 1 | | 0.5 | |
| POP | m | 2 | | 1 | |
| POPF(D/Q) | | 35 | | 13 | |
| POPA(D) | | 9 | | 4 | |
| POP | sp | 1 | 2 | 2 | |
| LEA | r16,[m] | 2 | 2 | 1 | |
| LEA | r32,[m] | 1 | 1 | 0.5 | 64 bit mode |
| LEA | r32/64,[m] | | 1 | 0.5 | rip relative |
| LEA | r32/64,[m] | 1 | 1 | 0.25 | all other cases |
| LAHF | | 4 | 3 | 2 | |
| SAHF | | 2 | 2 | 0.5 | |
| SALC | | 1 | 1 | 1 | |
| BSWAP | r | 1 | 1 | 0.25 | |
| MOVBE | r,[m] | 1 | | 0.5 | MOVBE |
| MOVBE | [m],r | 1 | | 1 | MOVBE |
| PREFETCHNTA | m | 1 | | 0.5 | |
| PREFETCHT0/1/2 | m | 1 | | 0.5 | |
| PREFETCH/W | m | 1 | | 0.5 | PREFETCHW |
| SFENCE | | 4 | | ~20 | |
| LFENCE | | 1 | | 0.25 | |
| MFENCE | | 7 | | ~70 | |
| Arithmetic instructions | | | | | |
| ADD, SUB | r,r | 1 | 1 | 0.25 | |
| ADD, SUB | r,i | 1 | 1 | 0.25 | |
| ADD, SUB | r,m | 1 | | 0.5 | |
| ADD, SUB | m,r | 1 | 6 | 1 | |
| ADD, SUB | m,i | 1 | 6 | 1 | |
| ADC, SBB | r,r | 1 | 1 | | |
| ADC, SBB | r,i | 1 | 1 | | |
| ADC, SBB | r,m | 1 | 1 | 1 | |
| ADC, SBB | m,r | 1 | 6 | 1 | |
| ADC, SBB | m,i | 1 | 6 | 1 | |

Ryzen

| | | | | | |
|--------------------|-----------|----|-------|-------|------------------------------|
| ADCX ADOX | r,r | 1 | 1 | 1 | depends on operand values |
| CMP | r,r | 1 | 1 | 0.25 | |
| CMP | r,i | 1 | 1 | 0.25 | |
| CMP | r,m | 1 | | 0.5 | |
| CMP | m,i | 1 | | 0.5 | |
| INC, DEC, NEG | r | 1 | 1 | 0.25 | |
| INC, DEC, NEG | m | 1 | 6 | 1 | |
| AAA, AAS | | 10 | 6 | | |
| DAA | | 16 | 8 | | |
| DAS | | 20 | 9 | | |
| AAD | | 4 | 5 | | |
| AAM | | 4 | 13 | | |
| MUL, IMUL | r8/m8 | 1 | 3 | 1 | |
| MUL, IMUL | r16/m16 | 3 | 3 | 2 | |
| MUL, IMUL | r32/m32 | 2 | 3 | 2 | |
| MUL, IMUL | r64/m64 | 2 | 3 | 2 | |
| IMUL | r,r | 1 | 3 | 1 | |
| IMUL | r,m | 1 | | 1 | |
| IMUL | r16,r16,i | 2 | 4 | 1 | |
| IMUL | r32,r32,i | 1 | 3 | 1 | |
| IMUL | r64,r64,i | 1 | 3 | | |
| IMUL | r16,m16,i | 2 | | 1 | |
| IMUL | r32,m32,i | 1 | | 1 | |
| IMUL | r64,m64,i | 1 | | 1 | |
| MULX | r,r,r | 2 | 4 | 2 | |
| DIV | r8/m8 | 1 | 13-16 | 13-16 | |
| DIV | r16/m16 | 2 | 14-21 | 14-21 | |
| DIV | r32/m32 | 2 | 14-30 | 14-30 | |
| DIV | r64/m64 | 2 | 14-46 | 14-45 | |
| IDIV | r8/m8 | 1 | 13-16 | 13-16 | |
| IDIV | r16/m16 | 2 | 13-21 | 14-22 | |
| IDIV | r32/m32 | 2 | 14-30 | 14-30 | |
| IDIV | r64/m64 | 2 | 14-47 | 14-45 | |
| CBW | | 1 | 1 | 1 | |
| CWDE, CDQE | | 1 | 1 | 0.5 | |
| CDQ, CQO | | 1 | 1 | 0.25 | |
| CWD | | 2 | 1 | 1 | |
| Logic instructions | | | | | |
| AND, OR, XOR | r,r | 1 | 1 | 0.25 | |
| AND, OR, XOR | r,i | 1 | 1 | 0.25 | |
| AND, OR, XOR | r,m | 1 | | 0.5 | |
| AND, OR, XOR | m,r | 1 | 6 | 1 | |
| AND, OR, XOR | m,i | 1 | 6 | 1 | |
| TEST | r,r | 1 | 1 | 0.25 | |
| TEST | r,i | 1 | 1 | 0.25 | |
| TEST | m,r | 1 | | 0.5 | |
| TEST | m,i | 1 | | 0.5 | |
| NOT | r | 1 | 1 | 0.25 | |
| NOT | m | 1 | 6 | 1 | |
| ANDN | r,r,r | 1 | 1 | 0.25 | |
| SHL, SHR, SAR | r,i/CL | 1 | 1 | 0.25 | |
| ROL, ROR | r,i/CL | 1 | 1 | 0.25 | |
| BMI1 | | | | | |

Ryzen

| | | | | | |
|--------------------------------------|------------|---|----|-------|--------------|
| RCL | r,1 | 1 | 1 | 1 | |
| RCL | r,i | 9 | 4 | 4 | |
| RCL | r,cl | 9 | 4 | 4 | |
| RCR | r,1 | 1 | 1 | 1 | |
| RCR | r,i | 7 | 3 | 3 | |
| RCR | r,cl | 7 | 3 | 3 | |
| SHLD, SHRD | r,r,i | 6 | 3 | 3 | |
| SHLD, SHRD | r,r,cl | 7 | 3 | 3 | |
| SHLD, SHRD | m,r,i/CL | 8 | | 3 | |
| SARX | r,r,r | 1 | 1 | 0.25 | BMI2 |
| SHLX | r,r,r | 1 | 1 | 0.25 | BMI2 |
| SHRX | r,r,r | 1 | 1 | 0.25 | BMI2 |
| RORX | r,r,i | 1 | 1 | 0.25 | BMI2 |
| BT | r,r/i | 1 | 1 | 0.25 | |
| BT | m,i | 1 | | 0.5 | |
| BT | m,r | 5 | | 3 | |
| BTC, BTR, BTS | r,r/i | 2 | 2 | 0.5 | |
| BTC, BTR, BTS | m,i | 4 | | 2 | |
| BTC, BTR, BTS | m,r | 8 | | 3 | |
| BSF | r,r | 6 | 3 | 3 | |
| BSF | r,m | 8 | 4 | 4 | |
| BSR | r,r | 6 | 4 | 4 | |
| BSR | r,m | 8 | 4 | 4 | |
| SETcc | r | 1 | 1 | 0.5 | |
| SETcc | m | 1 | | 2 | |
| CLC, STC | | 1 | | 0.25 | |
| CMC | | 1 | 1 | | |
| CLD | | 2 | | 3 | |
| STD | | 2 | | 4 | |
| POPCNT | r,r | 1 | 1 | 0.25 | SSE4.2 |
| LZCNT | r,r | 1 | 1 | 0.25 | LZCNT |
| TZCNT | r,r | 2 | 2 | 0.5 | BMI1 |
| BEXTR | r,r,r | 1 | 1 | 0.25 | BMI1 |
| BLSI | r,r | 2 | 2 | 0.5 | BMI1 |
| BLSMSK | r,r | 2 | 2 | 0.5 | BMI1 |
| BLSR | r,r | 2 | 2 | 0.5 | BMI1 |
| PDEP | r,r,r | 6 | 18 | 18 | BMI2 |
| PEXT | r,r,r | 7 | 18 | 18 | BMI2 |
| BZHI | r,r,r | 1 | 1 | 0.25 | BMI2 |
| Control transfer instructions | | | | | |
| JMP | short/near | 1 | | 2 | |
| JMP | r | 1 | | 2 | |
| JMP | m | 1 | | 2 | |
| Jcc | short/near | 1 | | 0.5-2 | 2 if jumping |
| fused CMP+Jcc | short/near | 1 | | 0.5-2 | 2 if jumping |
| J(E/R)CXZ | short | 1 | | 0.5-2 | 2 if jumping |
| LOOP | short | 1 | | 2 | 2 if jumping |
| LOOPE LOOPNE | short | 1 | | 2 | 2 if jumping |
| CALL | near | 2 | | 2 | |
| CALL | r | 2 | | 2 | |
| CALL | m | 6 | | 2 | |
| RET | | 1 | | 2 | |

Ryzen

| | | | | | | |
|----------------------------|----------|-----------|-----|-----------|--|-------------|
| RET | i | 2 | | 2 | | |
| BOUND | m | 11 | | 3 | | for no jump |
| INTO | | 4 | | 2 | | for no jump |
| String instructions | | | | | | |
| LODS | | 3 | | 3 | | |
| REP LODS | m | 6n | | 2n | | |
| STOS | | 3 | | 3 | | |
| REP STOS | | 1n | | ~1n | | small n |
| REP STOS | | 3 per 16B | | 1 per 16B | | best case |
| MOVS | | 5 | | 3 | | |
| REP MOVS | | ~1n | | ~1n | | small n |
| REP MOVS | | 4 pr 16B | | 1 per 16B | | best case |
| SCAS | | 3 | | 3 | | |
| REP SCAS | | 7n | | 2n | | |
| CMPS | | 6 | | 3 | | |
| REP CMPS | | 9n | | 3n | | |
| Synchronization | | | | | | |
| LOCK ADD | m,r | 1 | ~17 | | | |
| XADD | m,r | 4 | 7 | | | |
| LOCK XADD | m,r | 4 | ~23 | | | |
| CMPXCHG | m,r8 | 5 | 8 | | | |
| CMPXCHG | m,r16 | 6 | 8 | | | |
| CMPXCHG | m,r32/64 | 6 | 8 | | | |
| LOCK CMPXCHG | m8,r8 | 5 | ~22 | | | |
| LOCK CMPXCHG | m16,r16 | 6 | ~22 | | | |
| LOCK CMPXCHG | m,r32/64 | 6 | ~22 | | | |
| CMPXCHG8B | m64 | 18 | 8 | | | |
| LOCK CMPXCHG8B | m64 | 18 | ~22 | | | |
| CMPXCHG16B | m128 | 27 | 13 | | | |
| LOCK CMPXCHG16B | m128 | 27 | ~21 | | | |
| Other | | | | | | |
| NOP (90) | | 1 | | 0.2 | | |
| Long NOP (0F 1F) | | 1 | | 0.2 | | |
| PAUSE | | 8 | | 3 | | |
| ENTER | a,0 | 12 | | 16 | | |
| ENTER | a,b | 11+3b | | ~18+b | | |
| LEAVE | | 2 | | 3 | | |
| CPUID | | 37-50 | | 125-133 | | |
| XGETBV | | | | 42 | | |
| RDTSC | | 37 | | 36 | | |
| RDTSCP | | 64 | | 64 | | rdtscp |
| RDPNC | | 20 | | 20 | | |
| CRC32 | r32,r8 | 3 | 3 | 3 | | |
| CRC32 | r32,r16 | 3 | 3 | 3 | | |
| CRC32 | r32,r32 | 3 | 3 | 3 | | |
| RDRAND RDSEED | r16/32 | 13 | | ~1200 | | |
| RDRAND RDSEED | r64 | 19 | | ~2500 | | |

Floating point x87 instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Domain, notes |
|--------------------------------|----------|--------|---------|-----------------------|-----------------|---------------|
| Move instructions | | | | | | |
| FLD | r | 1 | 1 | 1 | P3 | |
| FLD | m32/64 | 1 | 6 | 1 | P1 | |
| FLD | m80 | 8 | 7 | 4 | | |
| FBLD | m80 | 24 | | 24 | P2 P3 | |
| FST(P) | r | 1 | 1 | 1 | P3 | |
| FST(P) | m32/64 | 2 | 5 | 1 | P2 P3 | |
| FSTP | m80 | 15 | 8 | | | |
| FBSTP | m80 | 274 | | ~145 | P2 P3 | |
| FXCH | r | 1 | 0 | 0.25 | | |
| FILD | m | 1 | 8 | 1 | P3 | |
| FIST(T)(P) | m | 2 | 5 | 1 | P2 P3 | |
| FLDZ, FLD1 | | 1 | | 1 | P3 | |
| FCMOVcc | st0,r | 7 | 3 | 3 | P0 P1 P2 P3 | |
| FFREE | r | 1 | 0 | 0.25 | | |
| FINCSTP, FDECSTP | | 1 | 0 | 0.25 | | |
| FNSTSW | AX | 3 | | 16 | P2 P3 | |
| FNSTSW | m16 | 2 | | 14 | P2 P3 | |
| FLDCW | m16 | 1 | | 2 | P3 | |
| FNSTCW | m16 | 2 | | 2 | P2 P3 | |
| Arithmetic instructions | | | | | | |
| FADD(P),FSUB(R)(P) | r/m | 1 | 5 | 1 | P0 | |
| FIADD,FISUB(R) | m | 2 | | 1 | P0 P3 | |
| FMUL(P) | r/m | 1 | 5 | 1 | P0 | |
| FIMUL | m | 2 | | 1 | P0 P3 | |
| FDIV(R)(P) | r | 1 | 8-15 | 4-6 | P3 | |
| FDIV(R) | m | 1 | | | P3 | |
| FIDIV(R) | m | 2 | | | P3 | |
| FABS, FCHS | | 1 | 1 | 1 | P3 | |
| FCOM(P), FUCOM(P) | r/m | 1 | | 1 | P0 | |
| FCOMPP, FUCOMPP | | 1 | | 1 | P0 | |
| FCOMI(P) | r | 2 | | 1 | P2 | |
| FICOM(P) | m | 2 | | 1 | P0 P3 | |
| FTST | | 1 | | 1 | P0 | |
| FXAM | | 1 | | 1 | P3 | |
| FRNDINT | | 1 | 4 | 3 | | |
| FPREM FPREM1 | | 2 | | 12-50 | | |
| Math | | | | | | |
| FSQRT | | 1 | 8-21 | 4-10 | P3 | |
| FLDPI, etc. | | 1 | | 1 | P3 | |
| FSIN | | 11-60 | 50-170 | | P0 P3 | |
| FCOS | | 55 | 50-115 | | | |
| FSINCOS | | 80-140 | 60-120 | | P0 P3 | |
| FPTAN | | 11-52 | ~90 | 50-80 | | |
| FPATAN | | 11-82 | 50-160 | 45-150 | | |
| FSCALE | | 8 | 9 | 4 | P0 P2 P3 | |
| FEXTRACT | | 13 | 10 | 7 | P0 P2 P3 | |
| F2XM1 | | 10 | ~50 | | P2 P3 | |

Ryzen

| | | | | | | |
|------------------|------|-------------|-------------|-------------|----------------------|--|
| FYL2X FYL2XP1 | | 10-25 69 | ~50 ~135 | ~50 ~135 | P0 P2 P3 P0 P2 P3 | |
| Other | | | | | | |
| FNOP | | 1 | | 0.25 | | |
| (F)WAIT | | 1 | | 0.25 | | |
| FNCLEX | | 20 | | 45 | | |
| FNINIT | | 34 | | 85 | | |
| FNSAVE | m864 | 99 | | ~160 | | |
| FRSTOR | m864 | 77 | | ~130 | | |

Integer vector instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Notes |
|--------------------------|-----------|-----|---------|-----------------------|-----------------|--------------------|
| Move instructions | | | | | | |
| MOVD | r32, mm | 1 | 3 | 1 | | |
| MOVD | mm, r32 | 2 | 3 | 1 | | |
| MOVD | r32/64, x | 1 | 3 | 1 | | |
| MOVD | x, r32/64 | 1 | 3 | 1 | | |
| MOVD | mm/x,m32 | 1 | 4 | 0.5 | | |
| MOVD | m32,mm/x | 1 | 4 | 1 | P2 | |
| MOVQ | mm/x,mm/x | 1 | 1 | 0.25 | P0123 | |
| MOVQ | mm/x,m64 | 1 | 4 | 0.5 | | |
| MOVQ | m64,mm/x | 1 | 4 | 1 | P2 | |
| MOVDQA | x,x | 1 | 0 | 0.25 | | renaming |
| MOVDQA | x,m | 1 | 3 | 0.5 | | |
| MOVDQA | m,x | 1 | 4 | 1 | P2 | |
| VMOVDQA | y,y | 2 | 1 | 0.5 | | lower half renamed |
| VMOVDQA | y,m256 | 2 | 3 | 1 | | |
| VMOVDQA | m256,y | 2 | 4 | 2 | P2 | |
| MOVDQU | x,x | 1 | 0 | 0.25 | | renaming |
| MOVDQU | x,m | 1 | 3 | 0.5 | | |
| MOVDQU | m,x | 1 | 4 | 1 | P2 | |
| LDDQU | x,m | 1 | 3 | 0.5 | | |
| VMOVDQU | ymm,m256 | 2 | 3 | 1 | | |
| VMOVDQU | m256,ymm | 2 | 4 | 2 | P2 | |
| MOVDQ2Q | mm,xmm | 1 | 1 | 0.25 | P0123 | |
| MOVQ2DQ | xmm,mm | 1 | 1 | 0.25 | P0123 | |
| MOVNTQ | m,mm | 1 | ~900 | 1 | P2 | |
| MOVNTDQ | m,xmm | 1 | ~900 | 1 | P2 | |
| MOVNTDQA | xmm,m | 1 | 3 | 0.5 | | |
| PACKSSWB/DW | mm/x,r/m | 1 | 1 | 0.5 | P12 | |
| PACKUSWB | mm/x,r/m | 1 | 1 | 0.5 | P12 | |
| VPACKSSWB/DW | y,r/m | 2 | 1 | 1 | P12 | |
| VPACKUSWB | y,r/m | 2 | 1 | 1 | P12 | |
| PUNPCKH/LBW/WD/DQ | mm/x,r/m | 1 | 1 | 0.5 | P12 | ivec |
| PUNPCKL/HQDQ | xmm,r/m | 1 | 1 | 0.5 | P12 | |
| VPUNPCKL/HQDQ | y,r/m | 2 | 1 | 1 | P12 | |
| PSHUFB | mm/x,r/m | 1 | 1 | 0.5 | P12 | |
| VPSHUFB | y,r/m | 2 | 1 | 1 | P12 | |

Ryzen

| | | | | | | |
|--------------------------------|-------------|----|-------|------|-------------|-----------|
| PSHUFD | x,x,i | 1 | 1 | 0.5 | P12 | |
| VPSHUFD | y,y,i | 2 | 1 | 1 | P12 | |
| PSHUFW | mm,mm,i | 1 | 1 | 0.5 | P12 | |
| PSHUFL/HW | x,x,i | 1 | 1 | 0.5 | P12 | |
| VPSHUFL/HW | y,y,i | 2 | 1 | 1 | P12 | |
| PALIGNR | mm/x,r/m,i | 1 | 1 | 0.5 | P12 | |
| VPALIGNR | y,r/m,i | 2 | 1 | 1 | P12 | |
| PBLENDW | x,r/m,i | 1 | 1 | 0.33 | P013 | SSE4.1 |
| PBLENDW | y,r/m,i | 2 | 1 | 0.67 | P013 | |
| MASKMOVQ | mm,mm | 30 | ~3000 | ~9 | | |
| MASKMOVDQU | x,x | 60 | ~3000 | ~18 | | |
| PMOVMASKB | r32,mm/x | 1 | 3 | 1 | P2 | |
| VPMOVMASKB | r32,y | 2 | 3 | 2 | P2 | |
| PEXTRB/W/D/Q | r,x/mm,i | 2 | 3 | 1 | P1 P2 | SSE4.1 |
| PINSRB/W/D/Q | x/mm,r,i | 2 | 3 | 1 | P12 | |
| EXTRQ | x,i,i | 2 | 6 | 4 | | AMD SSE4A |
| EXTRQ | x,x | 1 | 4 | 4 | P1 | AMD SSE4A |
| INSERTQ | x,x,i,i | 2 | 4 | 4 | P0 P1 | AMD SSE4A |
| INSERTQ | x,x | 1 | 4 | 4 | P1 | AMD SSE4A |
| PMOVSXBW/BD/BQ/ WD/WQ/DQ | x,x | 1 | 1 | 0.5 | P12 | SSE4.1 |
| PMOVZXBW/BD/BQ/ WD/WQ/DQ | x,x | 1 | 1 | 0.5 | P12 | SSE4.1 |
| VINSERTI128 | y,y,x,i | 2 | 1 | .67 | P013 | AVX2 |
| VINSERTI128 | y,y,m,i | 2 | 4 | 1 | P013 | AVX2 |
| VPBROADCAST B/W/D/Q | x,x | 1 | 1 | 0.5 | P12 | AVX2 |
| VPBROADCAST B/W/D/Q | x,m | 1 | 4 | 1 | | AVX2 |
| VPBROADCAST B/W/D/Q | y,x | 2 | 1 | 1 | P12 | AVX2 |
| VPBROADCAST B/W/D/Q | y,m | 2 | | 0.5 | | AVX2 |
| VBROADCASTI128 | y,m128 | 2 | 3 | 0.5 | | AVX2 |
| VPGATHERDD | x,[r+s*x],x | 38 | | 13 | P0 P1 P2 | AVX2 |
| VPGATHERDD | y,[r+s*y],y | 66 | | 20 | P0 P1 P2 | AVX2 |
| VPGATHERQD | x,[r+s*x],x | 24 | | 9 | P0 P1 P2 | AVX2 |
| VPGATHERQD | x,[r+s*y],x | 36 | | 12 | P0 P1 P2 | AVX2 |
| VPGATHERDQ | x,[r+s*x],x | 23 | | 9 | P0 P1 P2 | AVX2 |
| VPGATHERDQ | y,[r+s*x],y | 35 | | 12 | P0 P1 P2 | AVX2 |
| VPGATHERQQ | x,[r+s*x],x | 23 | | 9 | P0 P1 P2 | AVX2 |
| VPGATHERQQ | y,[r+s*y],y | 35 | | 12 | P0 P1 P2 | AVX2 |
| Arithmetic instructions | | | | | | |
| PADDB/W/D/Q/SB/SW/ USB/USW | mm/x,r/m | 1 | 1 | 0.33 | | ivec |
| VPADD.. | y,y,r/m | 2 | 1 | 0.67 | | |
| PSUBB/W/D/Q/SB/SW/ USB/USW | mm/x,r/m | 1 | 1 | 0.33 | | |
| VPSUB.. | y,y,r/m | 2 | 1 | 0.67 | | |
| PHADD/SUB(S)W/D | x,x | 4 | 2 | 2 | P0 P1 P2 P3 | SSSE3 |
| VPHADD/SUB(S)W/D | y,y,y | 8 | 3 | 3 | P0 P1 P2 P3 | |
| PCMPEQ B/W/D | mm/x,r/m | 1 | 1 | 0.33 | P013 | |
| VPCMPEQ B/W/D | y,y,r/m | 2 | 2 | 0.67 | P013 | |
| PCMPEQQ | mm/x,r/m | 1 | 1 | 0.5 | P03 | |

Ryzen

| | | | | | | |
|--------------------|----------|---|---|------|----------|--------|
| VPCMPEQQ | y,y,r/m | 2 | 2 | 1 | P03 | |
| PCMPGT B/W/D | mm/x,r/m | 1 | 1 | 0.33 | P013 | |
| VPCMPGT B/W/D | y,y,r/m | 2 | 2 | 0.67 | P013 | |
| PCMPGTQ | mm/x,r/m | 1 | 1 | 1 | P0 | |
| VPCMPGTQ | y,y,r/m | 2 | | 2 | P0 | |
| PMULLW PMULHW | | | | | | |
| PMULHUW PMULDQ | | | | | | |
| PMULUDQ | mm/x,r/m | 1 | 3 | 1 | P0 | |
| VPMULLW VPMULHW | | | | | | |
| VPMULHUW | | | | | | |
| VPMULDQ | | | | | | |
| VPMULUDQ | y,y,r/m | 2 | 3 | 2 | P0 | |
| PMULLD | x,r/m | 1 | 4 | 2 | P0 | SSE4.1 |
| VPMULLD | y,y,r/m | 2 | 4 | 4 | P0 | |
| PMULHRSW | mm/x,r/m | 1 | 4 | 1 | P0 | SSSE3 |
| VPMULHRSW | y,y,r/m | 2 | 4 | 2 | P0 | |
| PMADDWD | mm/x,r/m | 1 | 3 | 1 | P0 | |
| VPMADDWD | y,y,r/m | 2 | 3 | 2 | P0 | |
| PMADDUBSW | mm/x,r/m | 1 | 4 | 1 | P0 | |
| VPMADDUBSW | y,y,r/m | 2 | 4 | 2 | P0 | |
| PAVGB/W | mm/x,r/m | 1 | 1 | 0.5 | P03 | |
| VPAVGB/W | mm/x,r/m | 2 | 1 | 1 | P03 | |
| PMIN/MAX SB/SW/ SD | | | | | | |
| UB/UW/UD | mm/x,r/m | 1 | 1 | 0.33 | P013 | |
| VPMIN/MAX SB/SW/ | | | | | | |
| SD UB/UW/UD | y,y,r/m | 2 | 1 | 0.67 | P013 | |
| PHMINPOSUW | x,r/m | 1 | 3 | 2 | | SSE4.1 |
| PABSB/W/D | mm/x,r/m | 1 | 1 | 0.5 | P03 | SSSE3 |
| VPABSB/W/D | y,r/m | 2 | 1 | 1 | P03 | |
| PSIGNB/W/D | mm/x,r/m | 1 | 1 | 0.5 | P03 | SSSE3 |
| VPSIGNB/W/D | y,r/m | 2 | 1 | 1 | P03 | SSSE3 |
| PSADBW | mm/x,r/m | 1 | 3 | 1 | P0 | |
| VPSADBW | y,y,r/m | 1 | 3 | 2 | P0 | |
| MPSADBW | x,x,i | 4 | 4 | 2 | P0 P1 P2 | SSE4.1 |
| VMPSADBW | y,y,y,i | 8 | 4 | 3 | | |
| Logic | | | | | | |
| PAND PANDN POR | | | | | | |
| PXOR | mm/x,r/m | 1 | 1 | 0.25 | P0123 | ivec |
| VPAND VPANDN | | | | | | |
| VPOR VPXOR | y,y,r/m | 2 | 1 | 0.5 | P0123 | |
| PSLL/RL W/D/Q | | | | | | |
| PSRAW/D | mm/x,r/m | 1 | 1 | 1 | P2 | |
| VPSLL/RL W/D/Q VP- | | | | | | |
| SRAW/D | y,y,x/m | 2 | 1 | 2 | P2 | |
| PSLL/RL W/D/Q | | | | | | |
| PSRAW/D | mm/x,i | 1 | 1 | 1 | P2 | |
| VPSLL/RL W/D/Q VP- | | | | | | |
| SRAW/D | y,y,i | 2 | 1 | 2 | P2 | |
| PSLLDQ, PSRLDQ | x,i | 1 | 1 | 0.5 | P12 | |
| VPSLLDQ VPSRLDQ | y,y,i | 2 | 1 | 1 | P12 | |
| VPSLLVD/Q | | | | | | |
| VPSRAVD | | | | | | |
| VPSRLVD/Q | x,x,x | 1 | 3 | 2 | P1 | AVX2 |

Ryzen

| | | | | | | |
|----------------------------|---------|----|----|------|----------|--------|
| VPSLLVD/Q | | | | | | |
| VPSRAVD | | | | | | |
| VPSRLVD/Q | y,y,y | 2 | 3 | 4 | P1 | AVX2 |
| PTEST | x,r/m | 1 | 2 | 1 | P2 | SSE4.1 |
| VPTEST | y,y/m | 3 | 4 | 2 | P1 P2 | |
| String instructions | | | | | | |
| PCMPESTRI | x,x,i | 6 | 8 | 3 | P1 P2 | SSE4.2 |
| PCMPESTRI | x,m,i | 12 | | 4 | | SSE4.2 |
| PCMPESTRM | x,x,i | 7 | 8 | 3 | P0 P1 P2 | SSE4.2 |
| PCMPESTRM | x,m,i | 12 | | 4 | | SSE4.2 |
| PCMPISTRI | x,x,i | 2 | 11 | 2 | P1 P2 | SSE4.2 |
| PCMPISTRI | x,m,i | 3 | | 2 | | SSE4.2 |
| PCMPISTRM | x,x,i | 3 | 7 | 2 | P1 P2 | SSE4.2 |
| PCMPISTRM | x,m,i | 4 | | 2 | | SSE4.2 |
| Encryption | | | | | | |
| PCLMULQDQ | x,x/m,i | 4 | 4 | 2 | | pclmul |
| AESDEC | x,x | 1 | 4 | 0.5 | P01 | aes |
| AESDECLAST | x,x | 1 | 4 | 0.5 | P01 | aes |
| AESENC | x,x | 1 | 4 | 0.5 | P01 | aes |
| AESENCCLAST | x,x | 1 | 4 | 0.5 | P01 | aes |
| AESIMC | x,x | 1 | 4 | 0.5 | P01 | aes |
| AESKEYGENASSIST | x,x,i | 1 | 4 | 0.5 | P01 | aes |
| SHA1RND4S | x,x,i | 1 | 6 | 4 | P1 | sha |
| SHA1NEXTE | x,x | 1 | 1 | 1 | P1 | sha |
| SHA1MSG1 | x,x | 2 | 2 | 1 | multi | sha |
| SHA1MSG2 | x,x | 1 | 1 | 0.5 | P12 | sha |
| SHA256RND4S | x,x | 1 | 4 | 2 | P1 | sha |
| SHA256MSG1 | x,x | 2 | 2 | 0.5 | P0123 | sha |
| SHA256MSG2 | x,x | 4 | 3 | 2 | P0123 | sha |
| Other | | | | | | |
| EMMS | | 1 | | 0.25 | | |

Floating point XMM and YMM instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipes | Domain, notes |
|--------------------------|----------|-----|---------|-----------------------|-----------------|---------------|
| Move instructions | | | | | | |
| MOVAPS/D MOVUPS/D | x,x | 1 | 0 | 0.25 | none | inherit |
| VMOVAPS/D | y,y | 2 | 1 | 0.5 | | inherit |
| MOVAPS/D MOVUPS/D | x,m128 | 1 | 3 | 0.5 | | |
| VMOVAPS/D | y,m256 | 2 | 5 | 1 | | |
| MOVAPS/D MOVUPS/D | m128,x | 1 | 4 | 1 | P2 | |
| VMOVAPS/D | m256,y | 2 | 3 | 2 | P2 | |
| MOVSS/D | x,x | 1 | 1 | 0.25 | P0123 | |
| MOVSS/D | x,m32/64 | 1 | 4 | 0.5 | | |

Ryzen

| | | | | | | |
|-----------------|-------------|----|------|------|-------------|-----------|
| MOVSS/D | m32/64,x | 1 | 4 | 1 | P2 | |
| MOVHPS/D | x,m64 | 1 | 4 | 0.5 | P12 | |
| MOVLPS/D | x,m64 | 1 | 4 | 0.5 | P12 | |
| MOVHPS/D | m64,x | 2 | 5 | 1 | P1 P2 | |
| MOVLPS/D | m64,x | 1 | 4 | 1 | P1 P2 | |
| MOVLHPS MOVHLPS | x,x | 1 | 1 | 0.5 | P12 | ivec |
| MOVMSKPS/D | r32,x | 1 | 3 | 1 | P2 | |
| VMOVMSKPS/D | r32,y | 1 | 3 | 1 | P2 | |
| MOVNTPS/D | m128,x | 1 | ~950 | 1 | P2 | |
| VMOVNTPS/D | m256,y | 2 | ~950 | 2 | P2 | |
| MOVNTSS/SD | m,x | 1 | | 4 | P2 | AMD SSE4A |
| SHUFPS/D | x,x/m,i | 1 | 1 | 0.5 | P12 | ivec |
| VSHUFPS/D | y,y,y/m,i | 2 | 1 | 1 | P2 | ivec |
| VPERMILPS/PD | x,x,x/m | 1 | 3 | 2 | P1 | |
| VPERMILPS/PD | y,y,y/m | 2 | 4 | 4 | P1 | |
| VPERMILPS/PD | x,x/m,i | 1 | 1 | 0.5 | P12 | |
| VPERMILPS/PD | y,y/m,i | 2 | 1 | 1 | P12 | |
| VPERM2F128 | y,y,y,i | 8 | 3 | 3 | P0 P1 P3 | |
| VPERM2F128 | y,y,m,i | 12 | | 4 | P0 P1 P3 | |
| VPERMPS | y,y,y | 3 | 5 | 4 | P0 P1 P2 | AVX2 |
| VPERMPS | y,y,m | 4 | | 4 | | AVX2 |
| VPERMPD | y,y,i | 3 | 2 | 2 | P0 P1 P2 P3 | AVX2 |
| VPERMPD | y,m,i | 4 | | 2 | P0 P1 P2 P3 | AVX2 |
| BLENDPS/PD | x,x/m,i | 1 | 1 | 0.5 | P01 | fp |
| VBLENDPS/PD | y,y,y/m,i | 2 | 1 | 1 | P01 | |
| BLENDVPS/PD | x,x/m,xmm0 | 1 | 1 | 0.5 | P01 | |
| VBLENDVPS/PD | y,y,y/m,y | 2 | 1 | 1 | P01 | |
| MOVDDUP | x,x | 1 | 1 | 0.5 | P12 | ivec |
| MOVDDUP | x,m64 | 1 | | 0.5 | | |
| VMOVDDUP | y,y | 2 | 1 | 1 | P12 | |
| VMOVDDUP | y,m256 | 2 | | 1 | | |
| VBROADCASTSS/D | x,x | 1 | 1 | 0.5 | P12 | |
| VBROADCASTSS/D | y,x | 2 | 1 | 1 | P12 | |
| VBROADCASTSS/D | x,m | 2 | 3 | 0.5 | | |
| VBROADCASTSS | y,m32 | 2 | 3 | 1 | | |
| VBROADCASTF128 | y,m128 | 2 | 4 | 5 | | |
| MOVSH/LDUP | x,x/m | 1 | 1 | 0.5 | P12 | ivec |
| VMOVSH/LDUP | y,y/m | 2 | | 1 | P12 | |
| UNPCKH/LPS/D | x,x/m | 1 | 1 | 0.5 | P12 | |
| VUNPCKH/LPS/D | y,y,y/m | 2 | 1 | 1 | P12 | |
| EXTRACTPS | r32,x,i | 2 | 3 | 1 | P1 P2 | |
| EXTRACTPS | m32,x,i | 2 | 6 | 1 | P1 P2 | |
| VEXTRACTF128 | x,y,i | 1 | 1 | 0.33 | P013 | |
| VEXTRACTF128 | m128,y,i | 2 | 7 | 1 | P01 P2 | |
| INSERTPS | x,x,i | 1 | 1 | 0.5 | P12 | ivec |
| INSERTPS | x,m32,i | 1 | 4 | 1 | P12 | |
| VINSERTF128 | y,y,x,i | 2 | 1 | 0.5 | P013 | |
| VINSERTF128 | y,y,m128,i | 2 | 5 | 1 | P013 | |
| VMASKMOVPS/D | x,x,m128 | 1 | 4 | 1 | P01 | |
| VMASKMOVPS/D | y,y,m256 | 2 | 4 | 1 | P01 | |
| VMASKMOVPS/D | m128,x,x | 19 | ~50 | 5 | P1 P2 | |
| VMASKMOVPS/D | m256,y,y | 42 | ~50 | 11 | P1 P2 | |
| VGATHERDPS | x,[r+s*x],x | 38 | | 13 | P0 P1 P2 | AVX2 |

Ryzen

| | | | | | | |
|-------------------|-------------|----|----|-----|----------|--------------|
| VGATHERDPS | y,[r+s*y],y | 66 | | 20 | P0 P1 P2 | AVX2 |
| VGATHERQPS | x,[r+s*x],x | 24 | | 9 | P0 P1 P2 | AVX2 |
| VGATHERQPS | x,[r+s*y],x | 36 | | 12 | P0 P1 P2 | AVX2 |
| VGATHERDPD | x,[r+s*x],x | 23 | | 9 | P0 P1 P2 | AVX2 |
| VGATHERDPD | y,[r+s*x],y | 35 | | 12 | P0 P1 P2 | AVX2 |
| VGATHERQPD | x,[r+s*x],x | 23 | | 9 | P0 P1 P2 | AVX2 |
| VGATHERQPD | y,[r+s*y],y | 35 | | 12 | P0 P1 P2 | AVX2 |
| Conversion | | | | | | |
| CVTPD2PS | x,x | 1 | 3 | 1 | P3 | fp |
| VCVTPD2PS | x,y | 2 | 5 | 2 | P3 | |
| CVTPS2PD | x,x | 1 | 3 | 1 | P3 | |
| VCVTPS2PD | y,x | 2 | 5 | 2 | P3 | |
| CVTSD2SS | x,x | 1 | 3 | 1 | P3 | |
| CVTSS2SD | x,x | 1 | 3 | 1 | P3 | |
| CVTDQ2PS | x,x | 1 | 4 | 1 | P3 | |
| VCVTDQ2PS | y,y | 2 | 4 | 2 | P3 | |
| CVT(T) PS2DQ | x,x | 1 | 4 | 1 | P3 | mixed domain |
| VCVT(T) PS2DQ | y,y | 2 | 4 | 2 | P3 | |
| CVTDQ2PD | x,x | 2 | 6 | 1 | P12 P3 | |
| VCVTDQ2PD | y,x | 4 | 6 | 2 | P12 P3 | |
| CVT(T)PD2DQ | x,x | 2 | 6 | 1 | P12 P3 | |
| VCVT(T)PD2DQ | x,y | 4 | 6 | 2 | P12 P3 | |
| CVTPI2PS | x,mm | 2 | 6 | 1 | P12 P3 | |
| CVT(T)PS2PI | mm,x | 2 | 6 | 1 | P12 P3 | |
| CVTPI2PD | x,mm | 2 | 6 | 1 | P12 P3 | |
| CVT(T) PD2PI | mm,x | 2 | 6 | 1 | P12 P3 | |
| CVTSI2SS | x,r32 | 2 | 8 | 1 | | |
| CVT(T)SS2SI | r32,x | 2 | 7 | 1 | P2 P3 | |
| CVTSI2SD | x,r32/64 | 2 | 8 | 1 | | |
| CVT(T)SD2SI | r32/64,x | 2 | 7 | 1 | P2 P3 | |
| VCVTPS2PH | x/m,x,i | 2 | 6 | 2 | P1 P3 | |
| VCVTPS2PH | x/m,y,i | 4 | 6 | 2 | P12 P3 | |
| VCVTPH2PS | x,x/m | 2 | 6 | 2 | P1 P3 | |
| VCVTPH2PS | y,x/m | 4 | 6 | 2 | P12 P3 | |
| Arithmetic | | | | | | |
| ADDSS/D SUBSS/D | x,x/m | 1 | 3 | 0.5 | P23 | fp |
| ADDPs/D SUBPS/D | x,x/m | 1 | 3 | 0.5 | P23 | fp |
| VADDPs/D VSUBPS/D | y,y,y/m | 2 | 3 | 1 | P23 | fp |
| ADDSUBPS/D | x,x/m | 1 | 3 | 0.5 | P23 | fp |
| VADDSUBPS/D | y,y,y/m | 2 | 3 | 1 | P23 | |
| HADDPs/D HSUBPS/D | x,x | 4 | 7 | 2 | P1 P2 P3 | |
| VHADDPs/D | | | | | | |
| VHSUBPS/D | y,y,y/m | 8 | 7 | 3 | P1 P2 P3 | mixed domain |
| MULSS MULPS | x,x/m | 1 | 3 | 0.5 | P01 | fp |
| MULSD MULPD | x,x/m | 1 | 4 | 0.5 | P01 | fp |
| VMULPS | y,y,y/m | 2 | 3 | 1 | P01 | fp |
| VMULPD | y,y,y/m | 2 | 4 | 1 | P01 | fp |
| DIVSS DIVPS | x,x/m | 1 | 10 | 3 | P3 | |
| VDIVPS | y,y,y/m | 2 | 10 | 6 | P3 | |

Ryzen

| | | | | | | |
|--|-----------|-----|-------|------|-------------|---|
| DIVSD DIVPD | x,x/m | 1 | 8-13 | 4-5 | P3 | |
| VDIVPD | y,y,y/m | 2 | 8-13 | 8-9 | P3 | |
| RCPSS/PS | x,x/m | 1 | 5 | 1 | P01 | |
| VRCPPS | y,y/m | 2 | 5 | 2 | P01 | |
| CMPSS/D | | | | | | |
| CMPPS/D | x,x/m | 1 | 1 | 0.5 | P01 | |
| VCMPSS/D | y,y,y/m | 2 | 1 | 1 | P01 | |
| COMISS/D UCOMISS/D | x,x/m | 2 | 4 | 1 | P012 | |
| MAXSS/SD/PS/PD | | | | | | |
| MINSS/SD/PS/PD | x,x/m | 1 | 1 | 0.5 | P01 | fp |
| VMAXPS/D VMINPS/D | y,y,y/m | 2 | 1 | 1 | P01 | |
| ROUNDSS/SD/PS/PD | x,x/m,i | 1 | 4 | 1 | P3 | fp |
| VROUNDSS/SD/PS/PD | y,y/m,i | 2 | 4 | 2 | P3 | fp |
| DPPS | x,x,i | 8 | 15 | 4 | P0 P1 P2 P3 | SSE4.1 |
| DPPS | x,m,i | 10 | | 5 | P0 P1 P2 P3 | SSE4.1 |
| VDPPS | y,y,y,i | 13 | 16 | 5 | P0 P1 P2 P3 | SSE4.1 |
| VDPPS | y,m,i | 14 | | 5 | P0 P1 P2 P3 | SSE4.1 |
| DPPD | x,x,i | 3 | 10 | 3 | P0 P1 P2 P3 | SSE4.1 |
| DPPD | x,m,i | 5 | | 4 | P0 P1 P2 P3 | SSE4.1 |
| VFMADD132SS/SD | x,x,x/m | 1 | 5 | 0.5 | P01 | FMA3 |
| VFMADD132PS/PD | x,x,x/m | 1 | 5 | 0.5 | P01 | FMA3 |
| VFMADD132PS/PD | y,y,y/m | 2 | 5 | 1 | P01 | FMA3 |
| All other FMA3 instructions: same as above | | | | | P01 | FMA3 |
| VFMADDSS/SD | x,x,x,x/m | 1 | 5 | 0.5 | P01 | Not officially supported. Don't use! |
| VFMADDPS/PD | x,x,x,x/m | 1 | 5 | 0.5 | P01 | |
| VFMADDPS/PD | y,y,y,y/m | 2 | 5 | 1 | P01 | |
| All other FMA4 instructions: same as above | | | | | | |
| Math | | | | | | |
| SQRTSS/PS | x,x/m | 1 | 9-10 | 4-5 | P3 | fp |
| VSQRTPS | y,y/m | 2 | 9-10 | 8-10 | | |
| SQRTSD/PD | x,x/m | 1 | 14-15 | 4-8 | | |
| VSQRTPD | y,y/m | 2 | 14-15 | 8-16 | | |
| RSQRTSS/PS | x,x/m | 1 | 5 | 1 | P01 | |
| VRSQRTPS | y,y/m | 2 | 5 | 2 | P01 | |
| Logic | | | | | | |
| AND/ANDN/OR/XORPS/PD | x,x/m | 1 | 1 | 0.25 | P0123 | fp |
| VAND/ANDN/OR/XORPS/PD | y,y,y/m | 2 | 1 | 0.5 | P0123 | |
| Other | | | | | | |
| VZEROUPPER | | 10 | | 4 | | 32 bit mode |
| VZEROUPPER | | 17 | | 6 | | 64 bit mode |
| VZEROALL | | 18 | | 6 | | 32 bit mode |
| VZEROALL | | 33 | | 11 | | 64 bit mode |
| LDMXCSR | m32 | 1 | | 16 | | |
| STMXCSR | m32 | 2 | | 14 | | |
| FXSAVE | m4096 | 87 | | 90 | | |
| FXRSTOR | m4096 | 121 | | 140 | | |

Ryzen

| | | | | | | |
|----------|---|-----|--|-----|--|--|
| XSAVE | m | 160 | | 166 | | |
| XSAVEOPT | m | 97 | | 130 | | |
| XRSTOR | m | 213 | | 340 | | |
| XSAVEC | m | 111 | | 150 | | |

AMD Bobcat

List of instruction timings and macro-operation breakdown

Explanation of column headings:

| | |
|-------------------------------|--|
| Instruction: | Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc. |
| Operands: | i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, xmm = 128 bit xmm register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc. |
| Ops: | Number of micro-operations issued from instruction decoder to schedulers. Instructions with more than 2 micro-operations are micro-coded. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latencies listed do not include memory operands where the operand is listed as register or memory (r/m). The clock frequency varies dynamically, which makes it difficult to measure latencies. The values listed are measured after the execution of millions of similar instructions, assuming that this will make the processor boost the clock frequency to the highest possible value. |
| Reciprocal throughput: | This is also called issue latency. This value indicates the average number of clock cycles from the execution of an instruction begins to a subsequent independent instruction of the same kind can begin to execute. A value of 1/2 indicates that the execution units can handle 2 instructions per clock cycle in one thread. However, the throughput may be limited by other bottlenecks in the pipeline. |
| Execution pipe: | Indicates which execution pipe is used for the micro-operations. I0 means integer pipe 0. I0/1 means integer pipe 0 or 1. FP0 means floating point pipe 0 (ADD). FP1 means floating point pipe 1 (MUL). FP0/1 means either one of the two floating point pipes. Two micro-operations can execute simultaneously if they go to different execution pipes. |

Integer instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipe | Notes |
|--------------------------|----------|-----|---------|-----------------------|----------------|--|
| Move instructions | | | | | | |
| MOV | r,r | 1 | 1 | 0.5 | I0/1 | Any addr. mode Any addr. mode AH, BH, CH, DH |
| MOV | r,i | 1 | | 0.5 | I0/1 | |
| MOV | r,m | 1 | 4 | 1 | AGU | |
| MOV | m,r | 1 | 4 | 1 | AGU | |
| MOV | m8,r8H | 1 | 7 | 1 | AGU | |
| MOV | m,i | 1 | | 1 | AGU | |
| MOVNTI | m,r | 1 | 6 | 1 | AGU | |
| MOVZX, MOVZX | r,r | 1 | 1 | 0.5 | I0/1 | |
| MOVZX, MOVZX | r,m | 1 | 5 | 1 | | |
| MOVSLD | r64,r32 | 1 | 1 | 0.5 | | |
| MOVSLD | r64,m32 | 1 | 5 | 1 | | |
| CMOVcc | r,r | 1 | 1 | 0.5 | I0/1 | Timing dep. on hw |
| CMOVcc | r,m | 1 | | 1 | | |
| XCHG | r,r | 2 | 1 | 1 | I0/1 | |
| XCHG | r,m | 3 | 20 | | | |
| XLAT | | 2 | 5 | | | |
| PUSH | r | 1 | | 1 | | |

Bobcat

| | | | | | | |
|--------------------------------|-------------|----|-----|-----|------|----------------------|
| PUSH | i | 1 | | 1 | | |
| PUSH | m | 3 | | 2 | | |
| PUSHF(D/Q) | | 9 | | 6 | | |
| PUSHA(D) | | 9 | | 9 | | |
| POP | r | 1 | | 1 | | |
| POP | m | 4 | | 4 | | |
| POPF(D/Q) | | 29 | | 22 | | |
| POPA(D) | | 9 | | 8 | | |
| LEA | r16,[m] | 2 | 3 | 2 | I0 | Any address size |
| LEA | r32/64,[m] | 1 | 1 | 0.5 | I0/1 | no scale, no offset |
| LEA | r32/64,[m] | 1 | 2-4 | 1 | I0 | w. scale or offset |
| LEA | r64,[m] | 1 | | 0.5 | I0/1 | RIP relative |
| LAHF | | 4 | 4 | 2 | | |
| SAHF | | 1 | 1 | 0.5 | I0/1 | |
| SALC | | 1 | 1 | | | |
| BSWAP | r | 1 | 1 | 0.5 | I0/1 | |
| PREFETCHNTA | m | 1 | | 1 | AGU | |
| PREFETCHT0/1/2 | m | 1 | | 1 | AGU | |
| PREFETCH | m | 1 | | 1 | AGU | AMD only |
| SFENCE | | 4 | | ~45 | AGU | |
| LFENCE | | 1 | | 1 | AGU | |
| MFENCE | | 4 | | ~45 | AGU | |
| Arithmetic instructions | | | | | | |
| ADD, SUB | r,r/i | 1 | 1 | 0.5 | I0/1 | |
| ADD, SUB | r,m | 1 | | 1 | | |
| ADD, SUB | m,r | 1 | | 1 | | |
| ADC, SBB | r,r/i | 1 | 1 | 1 | I0/1 | |
| ADC, SBB | r,m | 1 | | 1 | | |
| ADC, SBB | m,r/i | 1 | 6-7 | | | |
| CMP | r,r/i | 1 | 1 | 0.5 | I0/1 | |
| CMP | r,m | 1 | | 1 | | |
| INC, DEC, NEG | r | 1 | 1 | 0.5 | I0/1 | |
| INC, DEC, NEG | m | 1 | 6 | | | |
| AAA | | 9 | 5 | | | |
| AAS | | 9 | 10 | | | |
| DAA | | 12 | 7 | | | |
| DAS | | 16 | 8 | | | |
| AAD | | 4 | 5 | | | |
| AAM | | 33 | 23 | 23 | | |
| MUL, IMUL | r8/m8 | 1 | 3 | 1 | I0 | |
| MUL, IMUL | r16/m16 | 3 | 3-5 | | I0 | latency ax=3, dx=5 |
| MUL, IMUL | r32/m32 | 2 | 3-4 | 2 | I0 | latency eax=3, edx=4 |
| MUL, IMUL | r64/m64 | 2 | 6-7 | | I0 | latency rax=6, rdx=7 |
| IMUL | r16,r16/m16 | 1 | 3 | 1 | I0 | |
| IMUL | r32,r32/m32 | 1 | 3 | 1 | I0 | |
| IMUL | r64,r64/m64 | 1 | 6 | 4 | I0 | |
| IMUL | r16,(r16),i | 2 | 4 | 3 | I0 | |
| IMUL | r32,(r32),i | 1 | 3 | 1 | I0 | |
| IMUL | r64,(r64),i | 1 | 7 | 4 | I0 | |
| DIV | r8/m8 | 1 | 27 | 27 | I0 | |
| DIV | r16/m16 | 1 | 33 | 33 | I0 | |
| DIV | r32/m32 | 1 | 49 | 49 | I0 | |

Bobcat

| | | | | | | |
|--------------------------------------|----------|-----|----|-----|-------|------------------|
| DIV | r64/m64 | 1 | 81 | 81 | I0 | |
| IDIV | r8/m8 | 1 | 29 | 29 | I0 | |
| IDIV | r16/m16 | 1 | 37 | 37 | I0 | |
| IDIV | r32/m32 | 1 | 55 | 55 | I0 | |
| IDIV | r64/m64 | 1 | 81 | 81 | I0 | |
| CBW, CWDE, CDQE | | 1 | 1 | | I0/1 | |
| CWD, CDQ, CQO | | 1 | 1 | | I0/1 | |
| Logic instructions | | | | | | |
| AND, OR, XOR | r,r | 1 | 1 | 0.5 | I0/1 | |
| AND, OR, XOR | r,m | 1 | | 1 | | |
| AND, OR, XOR | m,r | 1 | | 1 | | |
| TEST | r,r | 1 | 1 | 0.5 | I0/1 | |
| TEST | r,m | 1 | | 1 | | |
| NOT | r | 1 | 1 | 0.5 | I0/1 | |
| NOT | m | 1 | | 1 | | |
| SHL, SHR, SAR | r,i/CL | 1 | 1 | 0.5 | I0/1 | |
| ROL, ROR | r,i/CL | 1 | 1 | 0.5 | I0/1 | |
| RCL, RCR | r,1 | 1 | 1 | 1 | I0/1 | |
| RCL | r,i | 9 | 5 | 5 | | |
| RCR | r,i | 7 | 4 | 4 | | |
| RCL | r,CL | 9 | 6 | 5 | | |
| RCR | r,CL | 9 | 5 | 4 | | |
| SHL,SHR,SAR,ROL, ROR | m,i /CL | 1 | 7 | 1 | | |
| RCL, RCR | m,1 | 1 | 7 | 1 | | |
| RCL | m,i | 10 | | ~15 | | |
| RCR | m,i | 9 | 18 | ~14 | | |
| RCL | m,CL | 9 | | 15 | | |
| RCR | m,CL | 8 | | 15 | | |
| SHLD, SHRD | r,r,i | 6 | 3 | 3 | | |
| SHLD, SHRD | r,r,cl | 7 | 4 | 4 | | |
| SHLD, SHRD | m,r,i/CL | 8 | 18 | 15 | | |
| BT | r,r/i | 1 | | 0.5 | | |
| BT | m,i | 1 | | 1 | | |
| BT | m,r | 5 | | 3 | | |
| BTC, BTR, BTS | r,r/i | 2 | 2 | 1 | | |
| BTC | m,i | 5 | | 15 | | |
| BTR, BTS | m,i | 4-5 | | 15 | | |
| BTC | m,r | 8 | 16 | 13 | | |
| BTR, BTS | m,r | 8 | 15 | 15 | | |
| BSF, BSR | r,r | 11 | 6 | 6 | | |
| BSF, BSR | r,m | 11 | | 6 | | |
| POPCNT | r,r/m | 9 | 12 | 5 | | SSE4.A/SSE4.2 |
| LZCNT | r,r/m | 8 | 5 | | | SSE4.A, AMD only |
| SETcc | r | 1 | 1 | 0.5 | | |
| SETcc | m | 1 | | 1 | | |
| CLC, STC | | 1 | | 0.5 | I0/1 | |
| CMC | | 1 | 1 | 0.5 | I0/1 | |
| CLD | | 1 | | 1 | I0 | |
| STD | | 2 | | 2 | I0,I1 | |
| Control transfer instructions | | | | | | |

Bobcat

| | | | | | | |
|----------------------------|------------|-------|--------|---------|-------|-----------------------|
| JMP | short/near | 1 | | 2 | | |
| JMP | r | 1 | | 2 | | |
| JMP | m(near) | 1 | | 2 | | |
| Jcc | short/near | 1 | | 1/2 - 2 | | recip. t. = 2 if jump |
| J(E/R)CXZ | short | 2 | | 1 - 2 | | recip. t. = 2 if jump |
| LOOP | short | 8 | | 4 | | |
| CALL | near | 2 | | 2 | | |
| CALL | r | 2 | | 2 | | |
| CALL | m(near) | 5 | | 2 | | |
| RET | | 1 | | ~3 | | |
| RET | i | 4 | | ~4 | | |
| BOUND | m | 8 | | 4 | | values for no jump |
| INTO | | 4 | | 2 | | values for no jump |
| String instructions | | | | | | |
| LODS | | 4 | | ~3 | | |
| REP LODS | | 5 | | ~3 | | values are per count |
| STOS | | 4 | | 2 | | |
| REP STOS | | 2 | | | | best case 6-7 B/clock |
| MOVS | | 7 | | 5 | | |
| REP MOVS | | 2 | | | | best case 5 B/clock |
| SCAS | | 5 | | 3 | | |
| REP SCAS | | 6 | | 3 | | values are per count |
| CMPS | | 7 | | 4 | | |
| REP CMPS | | 6 | | 3 | | values are per count |
| Other | | | | | | |
| NOP (90) | | 1 | 0 | 0.5 | I0/1 | |
| Long NOP (0F 1F) | | 1 | 0 | 0.5 | I0/1 | |
| PAUSE | | 6 | | 6 | | |
| ENTER | i,0 | | 12 | | 36 | |
| ENTER | a,b | | 10+6b | | 34+6b | |
| LEAVE | | 2 | | 3 | | 32 bit mode |
| CPUID | | 30-52 | 70-830 | | | |
| RDTSC | | 26 | | 87 | | |
| RDPMS | | 14 | | 8 | | |

Floating point x87 instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipe | Notes |
|--------------------------|----------|-----|---------|-----------------------|----------------|-------|
| Move instructions | | | | | | |
| FLD | r | 1 | 2 | 0.5 | FP0/1 | |
| FLD | m32/64 | 1 | 6 | 1 | FP0/1 | |
| FLD | m80 | 7 | 14 | 5 | | |
| FBLD | m80 | 21 | 30 | 35 | | |
| FST(P) | r | 1 | 2 | 0.5 | FP0/1 | |
| FST(P) | m32/64 | 1 | 6 | 1 | FP1 | |
| FSTP | m80 | 16 | 19 | 9 | | |
| FBSTP | m80 | 217 | 177 | 180 | | |
| FXCH | r | 1 | 0 | 1 | FP1 | |
| FILD | m | 1 | 9 | 1 | FP1 | |
| FIST(T)(P) | m | 1 | 6 | 1 | | |

Bobcat

| | | | | | |
|--------------------------------|-------|-------|--------|--------|----------|
| FLDZ, FLD1 | | 1 | | 1 | FP1 |
| FCMOVcc | st0,r | 12 | 7 | 7 | FP0/1 |
| FFREE | r | 1 | | 1 | FP1 |
| FINCSTP, FDECSTP | | 1 | 1 | 1 | FP1 |
| FNSTSW | AX | 2 | ~20 | 10 | FP1 |
| FNSTSW | m16 | 2 | ~20 | 10 | FP1 |
| FNSTCW | m16 | 3 | | 2 | FP0 |
| FLDCW | m16 | 12 | | 10 | FP1 |
| Arithmetic instructions | | | | | |
| FADD(P),FSUB(R)(P) | r | 1 | 3 | 1 | FP0 |
| FADD(P),FSUB(R)(P) | m | 1 | 3 | 1 | FP0 |
| FIADD,FISUB(R) | m | 2 | | 3 | FP0,FP1 |
| FMUL(P) | r | 1 | 5 | 3 | FP1 |
| FMUL(P) | m | 1 | 5 | 3 | FP1 |
| FIMUL | m | 2 | | | FP1 |
| FDIV(R)(P) | r | 1 | 19 | 19 | FP1 |
| FDIV(R)(P) | m | 1 | | 19 | FP1 |
| FIDIV(R) | m | 2 | | 19 | FP1 |
| FABS, FCHS | | 1 | 2 | 2 | FP1 |
| FCOM(P), FUCOM(P) | r | 1 | | 1 | FP0 |
| FCOM(P), FUCOM(P) | m | 1 | | 1 | FP0 |
| FCOMPP, FUCOMPP | | 1 | | 1 | FP0 |
| FCOMI(P) | r | 1 | 2 | 2 | FP0 |
| FICOM(P) | m | 2 | | 1 | FP0, FP1 |
| FTST | | 1 | | 1 | FP0 |
| FXAM | | 2 | | 2 | FP1 |
| FRNDINT | | 5 | 11 | | FP0, FP1 |
| FPREM | | 1 | 11-16 | | FP1 |
| FPREM1 | | 1 | 11-19 | | FP1 |
| Math | | | | | |
| FSQRT | | 1 | 31 | | FP1 |
| FLDPI, etc. | | 1 | | 1 | FP0 |
| FSIN | | 4-44 | 27-105 | 27-105 | FP0, FP1 |
| FCOS | | 11-51 | 51-94 | 51-94 | FP0, FP1 |
| FSINCOS | | 11-75 | 48-110 | 48-110 | FP0, FP1 |
| FPTAN | | ~45 | ~113 | ~113 | FP0, FP1 |
| FPATAN | | 9-75 | 49-163 | 49-163 | FP0, FP1 |
| FSCALE | | 5 | 8 | | FP0, FP1 |
| FXTRACT | | 7 | 9 | | FP0, FP1 |
| F2XM1 | | 30-56 | ~60 | | FP0, FP1 |
| FYL2X | | 8 | 29 | | FP0, FP1 |
| FYL2XP1 | | 12 | 44 | | FP0, FP1 |
| Other | | | | | |
| FNOP | | 1 | 0 | 0.5 | FP0, FP1 |
| (F)WAIT | | 1 | 0 | 0.5 | ALU |
| FNCLEX | | 9 | | 30 | FP0, FP1 |
| FNINIT | | 26 | | 78 | FP0, FP1 |
| FNSAVE | m | 85 | | 163 | FP0, FP1 |
| FRSTOR | m | 80 | | 123 | FP0, FP1 |
| FXSAVE | m | 71 | | 105 | FP0, FP1 |

| | | | | | | |
|---------|---|-----|--|-----|----------|--|
| FXRSTOR | m | 111 | | 118 | FP0, FP1 | |
|---------|---|-----|--|-----|----------|--|

Integer MMX and XMM instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipe | Notes |
|-------------------|------------|-----|----------|-----------------------|----------------|--|
| Move instructions | | | | | | |
| MOVD | r32, mm | 1 | 7 | 1 | FP0 | Moves 64 bits. Name differs do. do. |
| MOVD | mm, r32 | 1 | 7 | 3 | FP0/1 | |
| MOVD | mm,m32 | 1 | 5 | 1 | FP0/1 | |
| MOVD | r32, xmm | 1 | 6 | 1 | FP0 | |
| MOVD | xmm, r32 | 3 | 6 | 3 | FP1 | |
| MOVD | xmm,m32 | 2 | 5 | 1 | FP1 | |
| MOVD | m32,mm/x | 1 | 6 | 2 | FP1 | |
| MOVD (MOVQ) | r64,mm/x | 1 | 7 | 1 | FP0 | |
| MOVD (MOVQ) | mm,r64 | 2 | 7 | 3 | FP0/1 | |
| MOVD (MOVQ) | xmm,r64 | 3 | 7 | 3 | FP0/1 | |
| MOVQ | mm,mm | 1 | 1 | 0.5 | FP0/1 | |
| MOVQ | xmm,xmm | 2 | 1 | 1 | FP0/1 | |
| MOVQ | mm,m64 | 1 | 5 | 1 | FP0/1 | |
| MOVQ | xmm,m64 | 2 | 5 | 1 | FP1 | |
| MOVQ | m64,mm/x | 1 | 6 | 2 | FP1 | |
| MOVDQA | xmm,xmm | 2 | 1 | 1 | FP0/1 | |
| MOVDQA | xmm,m | 2 | 6 | 2 | AGU | |
| MOVDQA | m,xmm | 2 | 6 | 3 | FP1 | |
| MOVDQU, LDDQU | xmm,m | 2 | 6-9 | 2-5.5 | AGU | |
| MOVDQU | m,xmm | 2 | 6-9 | 3-6 | FP1 | |
| MOVDQ2Q | mm,xmm | 1 | 1 | 0.5 | FP0/1 | |
| MOVQ2DQ | xmm,mm | 2 | 1 | 1 | FP0/1 | |
| MOVNTQ | m,mm | 1 | 13 | 1.5 | FP1 | |
| MOVNTDQ | m,xmm | 2 | 13 | 3 | FP1 | |
| PACKSSWB/DW | mm,r/m | 1 | 1 | 0.5 | FP0/1 | |
| PACKUSWB | | | | | | |
| PACKSSWB/DW | xmm,r/m | 3 | 2 | 2 | FP0/1 | |
| PACKUSWB | | | | | | |
| PUNPCKH/LBW/WD/DQ | mm,r/m | 1 | 1 | 0.5 | | |
| PUNPCKH/LBW/WD/DQ | | | | | | |
| PUNPCKHQDQ | xmm,r/m | 2 | 1 | 1 | FP0, FP1 | |
| PUNPCKLQDQ | xmm,r/m | 1 | 1 | 0.5 | FP0/1 | |
| PSHUFB | mm,mm | 1 | 2 | 1 | FP0/1 | |
| PSHUFB | xmm,xmm | 6 | 3 | 3 | FP0/1 | |
| PSHUFD | xmm,xmm,i | 3 | 2 | 2 | FP0/1 | |
| PSHUFW | mm,mm,i | 1 | 1 | 0.5 | FP0/1 | |
| PSHUFL/HW | xmm,xmm,i | 2 | 2 | 2 | FP0/1 | |
| PALIGNR | xmm,xmm,i | 20 | 19 | 12 | FP0/1 | |
| MASKMOVQ | mm,mm | 32 | 146-1400 | 130-1170 | FP0, FP1 | |
| MASKMOVDQU | xmm,xmm | 64 | 279-3000 | 260-2300 | FP0, FP1 | |
| PMOVBMSKB | r32,mm/x | 1 | 8 | 2 | FP0 | |
| PEXTRW | r32,mm/x,i | 2 | 12 | 2 | FP0, FP1 | |
| PINSRW | mm,r32,i | 2 | 10 | 6 | FP0/1 | |
| PINSRW | xmm,r32,i | 3 | 10 | | FP0/1 | |

Bobcat

| | | | | | | |
|--------------------------------|-------------|---|-----|-----|----------|------------------|
| INSERTQ | xmm,xmm | 3 | 3-4 | 3 | FP0, FP1 | SSE4.A, AMD only |
| INSERTQ | xmm,xmm,i,i | 3 | 3-4 | 3 | FP0, FP1 | SSE4.A, AMD only |
| EXTRQ | xmm,xmm | 1 | 1 | 1 | FP0/1 | SSE4.A, AMD only |
| EXTRQ | xmm,xmm,i,i | 1 | 2 | 2 | FP0/1 | SSE4.A, AMD only |
| Arithmetic instructions | | | | | | |
| PADDB/W/D/Q | mm,r/m | 1 | 1 | 0.5 | FP0/1 | |
| PADDSB/W | | | | | | |
| PADDUSB/W PSUBB/W | | | | | | |
| W/D/Q PSUBSB/W | | | | | | |
| PSUBUSB/W | xmm,r/m | 2 | 1 | 1 | FP0/1 | |
| PADDB/W/D/Q | | | | | | |
| PADDSB/W ADDUSB/W | | | | | | |
| PSUBB/W/D/Q PSUBSB/W | | | | | | |
| PSUBUSB/W | mm,r/m | 1 | 1 | 0.5 | FP0/1 | Suppl. SSE3 |
| PHADD/SUBW/SW/D | xmm,r/m | 2 | 4 | 1 | FP0/1 | Suppl. SSE3 |
| PCMPEQ/GT B/W/D | mm,r/m | 1 | 1 | 0.5 | FP0/1 | |
| PCMPEQ/GT B/W/D | xmm,r/m | 2 | 1 | 1 | FP0/1 | |
| PMULLW PMULHW | mm,r/m | 1 | 2 | 1 | FP0 | |
| PMULHUW | | | | | | |
| PMULUDQ | | | | | | |
| PMULLW PMULHW | | | | | | |
| PMULHUW | xmm,r/m | 2 | 2 | 2 | FP0 | |
| PMULUDQ | | | | | | |
| PMULHSW | | | | | | |
| PMULHSW | | | | | | |
| PMADDWD | mm,r/m | 1 | 2 | 1 | FP0 | |
| PMADDWD | xmm,r/m | 2 | 2 | 2 | FP0 | |
| PMADDUBSW | mm,r/m | 1 | 2 | 1 | FP0 | Suppl. SSE3 |
| PMADDUBSW | xmm,r/m | 2 | 2 | 2 | FP0 | Suppl. SSE3 |
| PAVGB/W | mm,r/m | 1 | 1 | 0.5 | FP0/1 | |
| PAVGB/W | xmm,r/m | 2 | 1 | 1 | FP0/1 | |
| PMIN/MAX SW/UB | mm,r/m | 1 | 1 | 0.5 | FP0/1 | |
| PMIN/MAX SW/UB | xmm,r/m | 2 | 1 | 1 | FP0/1 | |
| PABSB/W/D | mm,r/m | 1 | 1 | 0.5 | FP0/1 | Suppl. SSE3 |
| PABSB/W/D | xmm,r/m | 2 | 1 | 1 | FP0/1 | Suppl. SSE3 |
| PSIGNB/W/D | mm,r/m | 1 | 1 | 0.5 | FP0/1 | Suppl. SSE3 |
| PSIGNB/W/D | xmm,r/m | 2 | 1 | 1 | FP0/1 | Suppl. SSE3 |
| PSADBW | mm,r/m | 1 | 2 | 2 | FP0 | |
| PSADBW | xmm,r/m | 2 | 2 | 2 | FP0, FP1 | |
| Logic | | | | | | |
| PAND PANDN POR | mm,r/m | 1 | 1 | 0.5 | FP0/1 | |
| PXOR | | | | | | |
| PAND PANDN POR | xmm,r/m | 2 | 1 | 1 | FP0/1 | |
| PXOR | | | | | | |
| PSLL/RL W/D/Q | mm,i/mm/m | 1 | 1 | 1 | FP0/1 | |
| PSRAW/D | | | | | | |
| PSLL/RL W/D/Q | xmm,i/xmm/m | 2 | 1 | 1 | FP0/1 | |
| PSRAW/D | | | | | | |
| PSLLDQ, PSRLDQ | xmm,i | 2 | 1 | 1 | FP0/1 | |
| Other | | | | | | |

| | | | | | | |
|------|--|---|--|-----|-------|--|
| EMMS | | 1 | | 0.5 | FP0/1 | |
|------|--|---|--|-----|-------|--|

Floating point XMM instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipe | Notes |
|--------------------------|----------|-----|---------|-----------------------|----------------|------------------|
| Move instructions | | | | | | |
| MOVAPS/D | r,r | 2 | 1 | 1 | FP0/1 | |
| MOVAPS/D | r,m | 2 | 6 | 2 | AGU | |
| MOVAPS/D | m,r | 2 | 6 | 3 | FP1 | |
| MOVUPS/D | r,r | 2 | 1 | 1 | FP0/1 | |
| MOVUPS/D | r,m | 2 | 6-9 | 2-6 | AGU | |
| MOVUPS/D | m,r | 2 | 6-9 | 3-6 | FP1 | |
| MOVSS/D | r,r | 1 | 1 | 0.5 | FP0/1 | |
| MOVSS/D | r,m | 2 | 6 | 2 | FP1 | |
| MOVSS/D | m,r | 1 | 5 | 2 | FP1 | |
| MOVHLPs, MOVLHPs | r,r | 1 | 1 | 0.5 | FP0/1 | |
| MOVHPS/D, MOVLPS/D | r,m | 1 | 6 | 2 | AGU | |
| MOVHPS/D, MOVLPS/D | m,r | 1 | 5 | 3 | FP1 | |
| MOVNTPS/D | m,r | 2 | 12 | 3 | FP1 | |
| MOVNTSS/D | m,r | 1 | 12 | 2 | FP1 | SSE4.A, AMD only |
| MOVDDUP | r,r | 2 | 2 | 1 | FP0/1 | SSE3 |
| MOVDDUP | r,m64 | 2 | 7 | 2 | FP0/1 | SSE3 |
| MOVSHDUP, MOVSLDUP | r,r | 2 | 1 | 1 | FP0/1 | |
| MOVSHDUP, MOVSLDUP | r,m | 2 | 12 | 3 | AGU | |
| MOVMSKPS/D | r32,r | 1 | ~6 | 2 | FP0 | |
| SHUFPS/D | r,r/m,i | 3 | 2 | 2 | FP0/1 | |
| UNPCK H/L PS/D | r,r/m | 2 | 1 | 1 | FP0/1 | |
| Conversion | | | | | | |
| CVTTPS2PD | r,r/m | 2 | 5 | 2 | FP1 | |
| CVTPD2PS | r,r/m | 4 | 5 | 3 | FP0, FP1 | |
| CVTSD2SS | r,r/m | 3 | 5 | 3 | FP0, FP1 | |
| CVTSS2SD | r,r/m | 1 | 4 | 1 | FP1 | |
| CVTDQ2PS | r,r/m | 2 | 4 | 4 | FP1 | |
| CVTDQ2PD | r,r/m | 2 | 5 | 2 | FP1 | |
| CVT(T)PS2DQ | r,r/m | 2 | 4 | 4 | FP1 | |
| CVT(T)PD2DQ | r,r/m | 4 | 6 | 3 | FP0, FP1 | |
| CVTPI2PS | xmm,mm | 1 | 4 | 2 | FP1 | |
| CVTPI2PD | xmm,mm | 2 | 5 | 2 | FP1 | |
| CVT(T)PS2PI | mm,xmm | 1 | 4 | 1 | FP1 | |
| CVT(T)PD2PI | mm,xmm | 3 | 6 | 2 | FP0, FP1 | |
| CVTSI2SS | xmm,r32 | 3 | 12 | 3 | FP0, FP1 | |
| CVTSI2SD | xmm,r32 | 2 | 11 | 3 | FP1 | |
| CVT(T)SS2SI | r32,xmm | 2 | 12 | 1 | FP0, FP1 | |
| CVT(T)SD2SI | r32,xmm | 2 | 11 | 1 | FP0, FP1 | |
| Arithmetic | | | | | | |
| ADDSS/D SUBSS/D | r,r/m | 1 | 3 | 1 | FP0 | |

Bobcat

| | | | | | | |
|------------------------------------|-------|----|----|----|----------|------|
| ADDPS/D SUBPS/D | r,r/m | 2 | 3 | 2 | FP0 | SSE3 |
| ADDSUBPS/D | r,r/m | 2 | 3 | 2 | FP0 | |
| HADDPS/D HSUBPS/D | r,r/m | 2 | 3 | 2 | FP0 | SSE3 |
| MULSS | r,r/m | 1 | 2 | 1 | FP1 | |
| MULSD | r,r/m | 1 | 4 | 2 | FP1 | |
| MULPS | r,r/m | 2 | 2 | 2 | FP1 | |
| MULPD | r,r/m | 2 | 4 | 4 | FP1 | |
| DIVSS | r,r/m | 1 | 13 | 13 | FP1 | |
| DIVPS | r,r/m | 2 | 38 | 38 | FP1 | |
| DIVSD | r,r/m | 1 | 17 | 17 | FP1 | |
| DIVPD | r,r/m | 2 | 34 | 34 | FP1 | |
| RCPSS | r,r/m | 1 | 3 | 1 | FP1 | |
| RCPPS | r,r/m | 2 | 3 | 2 | FP1 | |
| MAXSS/D MINSS/D | r,r/m | 1 | 2 | 1 | FP0 | |
| MAXPS/D MINPS/D | r,r/m | 2 | 2 | 2 | FP0 | |
| CMPccSS/D | r,r/m | 1 | 2 | 1 | FP0 | |
| CMPccPS/D | r,r/m | 2 | 2 | 2 | FP0 | |
| COMISS/D UCOMISS/D | r,r/m | 1 | | 1 | FP0 | |
| Logic | | | | | | |
| ANDPS/D ANDNPS/D ORPS/D XORPS/D | r,r/m | 2 | 1 | 1 | FP0/1 | |
| Math | | | | | | |
| SQRTSS | r,r/m | 1 | 14 | 14 | FP1 | |
| SQRTPS | r,r/m | 2 | 48 | 48 | FP1 | |
| SQRTSD | r,r/m | 1 | 24 | 24 | FP1 | |
| SQRTPD | r,r/m | 2 | 48 | 48 | FP1 | |
| RSQRTSS | r,r/m | 1 | 3 | 1 | FP1 | |
| RSQRTPS | r,r/m | 2 | 3 | 2 | FP1 | |
| Other | | | | | | |
| LDMXCSR | m | 12 | | 10 | FP0, FP1 | |
| STMXCSR | m | 3 | | 11 | FP0, FP1 | |

AMD Jaguar

List of instruction timings and macro-operation breakdown

Explanation of column headings:

| | |
|-------------------------------|--|
| Instruction: | Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc. |
| Operands: | i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, xmm = 128 bit xmm register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc. |
| Ops: | Number of micro-operations issued from instruction decoder to schedulers. Instructions with more than 2 micro-operations are micro-coded. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latencies listed do not include memory operands where the operand is listed as register or memory (r/m). The clock frequency varies dynamically, which makes it difficult to measure latencies. The values listed are measured after the execution of millions of similar instructions, assuming that this will make the processor boost the clock frequency to the highest possible value. |
| Reciprocal throughput: | This is also called issue latency. This value indicates the average number of clock cycles from the execution of an instruction begins to a subsequent independent instruction of the same kind can begin to execute. A value of 1/2 indicates that the execution units can handle 2 instructions per clock cycle in one thread. However, the throughput may be limited by other bottlenecks in the pipeline. |
| Execution pipe: | Indicates which execution pipe is used for the micro-operations. I0 means integer pipe 0. I0/1 means integer pipe 0 or 1. FP0 means floating point pipe 0 (ADD). FP1 means floating point pipe 1 (MUL). FP0/1 means either one of the two floating point pipes. Two micro-operations can execute simultaneously if they go to different execution pipes. |

Integer instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipe | Notes |
|--------------------------|----------|-----|---------|-----------------------|----------------|---|
| Move instructions | | | | | | |
| MOV | r,r | 1 | 1 | 0.5 | I0/1 | Any addressing mode Any addressing mode Any addressing mode Any addressing mode Any addressing mode Any addressing mode Any addressing mode Any addressing mode Any addressing mode Any addressing mode Any addressing mode Any addressing mode Any addressing mode Any addressing mode Any addressing mode |
| MOV | r,i | 1 | | 0.5 | I0/1 | |
| MOV | r8/16,m | 1 | 4 | 1 | AGU | |
| MOV | m,r8/16 | 1 | 4 | 1 | AGU | |
| MOV | r32/64,m | 1 | 3 | 1 | AGU | |
| MOV | m,r32/64 | 1 | 0 | 1 | AGU | |
| MOV | m,i | 1 | | 1 | AGU | |
| MOVNTI | m,r | 1 | 6 | 1 | AGU | |
| MOVZX, MOVZX | r,r | 1 | 1 | 0.5 | I0/1 | |
| MOVZX, MOVZX | r,m | 1 | 4 | 1 | | |
| MOVSLD | r64,r32 | 1 | 1 | 0.5 | | |
| MOVSLD | r64,m32 | 1 | 3 | 1 | | |
| CMOVcc | r,r | 1 | 1 | 0.5 | I0/1 | |
| CMOVcc | r,m | 1 | | 1 | | |

| Jaguar | | | | | | |
|--------------------------------|------------|----|----|------|------|---|
| XCHG | r8,r8 | 3 | 2 | 2 | I0/1 | Timing depends on hw |
| XCHG | r,r | 2 | 1 | 1 | I0/1 | |
| XCHG | r,m | 3 | 16 | | | |
| XLAT | | 2 | 5 | 3 | | |
| PUSH | r | 1 | | 1 | | |
| PUSH | i | 1 | | 1 | | |
| PUSH | m | 2 | | 1 | | |
| PUSH | SP | 2 | | 1 | | |
| PUSHF(D/Q) | | 9 | | 6 | | |
| PUSHA(D) | | 9 | | 8 | | |
| POP | r | 1 | | 1 | | |
| POP | m | 3 | | 2 | | |
| POP | SP | 1 | | 2 | | |
| POPF(D/Q) | | 29 | | 18 | | |
| POPA(D) | | 9 | | 8 | | |
| LEA | r16,[m] | 2 | 3 | 2 | I0 | Any address size 1-2 comp., no scale 3 comp. or scale RIP relative |
| LEA | r32/64,[m] | 1 | 1 | 0.5 | I0/1 | |
| LEA | r32/64,[m] | 1 | 2 | 1 | I0 | |
| LEA | r64,[m] | 1 | | 0.5 | I0/1 | |
| LAHF | | 4 | 3 | 2 | | |
| SAHF | | 1 | 1 | 0.5 | I0/1 | MOVBE MOVBE |
| SALC | | 1 | 1 | 1 | | |
| BSWAP | r | 1 | 1 | 0.5 | I0/1 | |
| MOVBE | r,m | 1 | | 1 | | |
| MOVBE | m,r | 1 | | 1 | | |
| PREFETCHNTA | m | 1 | | ~100 | AGU | |
| PREFETCHT0/1/2 | m | 1 | | ~100 | AGU | |
| PREFETCHW | m | 1 | | ~100 | AGU | |
| LFENCE | | 1 | | 0.5 | AGU | |
| MFENCE | | 4 | | ~45 | AGU | |
| SFENCE | | 4 | | ~45 | AGU | |
| Arithmetic instructions | | | | | | |
| ADD, SUB | r,r/i | 1 | 1 | 0.5 | I0/1 | |
| ADD, SUB | r,m | 1 | | 1 | | |
| ADD, SUB | m,r | 1 | 6 | 1 | | |
| ADC, SBB | r,r/i | 1 | 1 | 1 | I0/1 | |
| ADC, SBB | r,m | 1 | | 1 | | |
| ADC, SBB | m,r/i | 1 | 8 | | | |
| CMP | r,r/i | 1 | 1 | 0.5 | I0/1 | |
| CMP | r,m | 1 | | 1 | | |
| INC, DEC, NEG | r | 1 | 1 | 0.5 | I0/1 | |
| INC, DEC, NEG | m | 1 | 6 | 1 | | |
| AAA | | 9 | 5 | | | |
| AAS | | 9 | 8 | | | |
| DAA | | 12 | 6 | | | |
| DAS | | 16 | 8 | | | |
| AAD | | 4 | 5 | | | |
| AAM | | 8 | 14 | 13 | | |
| MUL, IMUL | r8/m8 | 1 | 3 | 1 | I0 | |
| MUL, IMUL | r16/m16 | 3 | 3 | 3 | I0 | |
| MUL, IMUL | r32/m32 | 2 | 3 | 2 | I0 | |

Jaguar

| | | | | | | |
|---------------------------|-------------|----|-------|-------|------|------|
| MUL, IMUL | r64/m64 | 2 | 6 | 5 | I0 | |
| IMUL | r16,r16/m16 | 1 | 3 | 1 | I0 | |
| IMUL | r32,r32/m32 | 1 | 3 | 1 | I0 | |
| IMUL | r64,r64/m64 | 1 | 6 | 4 | I0 | |
| IMUL | r16,(r16),i | 2 | 4 | 1 | I0 | |
| IMUL | r32,(r32),i | 1 | 3 | 1 | I0 | |
| IMUL | r64,(r64),i | 1 | 6 | 4 | I0 | |
| DIV | r8/m8 | 1 | 11-14 | 11-14 | I0 | |
| DIV | r16/m16 | 2 | 12-19 | 12-19 | I0 | |
| DIV | r32/m32 | 2 | 12-27 | 12-27 | I0 | |
| DIV | r64/m64 | 2 | 12-43 | 12-43 | I0 | |
| IDIV | r8/m8 | 1 | 11-14 | 11-14 | I0 | |
| IDIV | r16/m16 | 2 | 12-19 | 12-19 | I0 | |
| IDIV | r32/m32 | 2 | 12-27 | 12-27 | I0 | |
| IDIV | r64/m64 | 2 | 12-43 | 12-43 | I0 | |
| CBW, CWDE, CDQE | | 1 | 1 | | I0/1 | |
| CWD, CDQ, CQO | | 1 | 1 | | I0/1 | |
| Logic instructions | | | | | | |
| AND, OR, XOR | r,i | 1 | 1 | 0.5 | I0/1 | |
| AND, OR, XOR | r,r | 1 | 1 | 0.5 | I0/1 | |
| AND, OR, XOR | r,m | 1 | | 1 | | |
| AND, OR, XOR | m,r | 1 | 6 | 1 | | |
| ANDN | r,r,r | 1 | 1 | 0.5 | | BMI1 |
| ANDN | r,r,m | 2 | | 1 | | BMI1 |
| TEST | r,i | 1 | 1 | 0.5 | I0/1 | |
| TEST | r,r | 1 | 1 | 0.5 | I0/1 | |
| TEST | r,m | 1 | | 1 | | |
| NOT | r | 1 | 1 | 0.5 | I0/1 | |
| NOT | m | 1 | 6 | 1 | | |
| SHL, SHR, SAR | r,i/CL | 1 | 1 | 0.5 | I0/1 | |
| ROL, ROR | r,i/CL | 1 | 1 | 0.5 | I0/1 | |
| RCL, RCR | r,1 | 1 | 1 | 1 | I0/1 | |
| RCL | r,i | 9 | 5 | 5 | | |
| RCR | r,i | 7 | 4 | 4 | | |
| RCL | r,CL | 9 | 5 | 5 | | |
| RCR | r,CL | 7 | 4 | 4 | | |
| SHL,SHR,SAR,ROL, ROR | m,i /CL | 1 | 6 | 1 | | |
| RCL, RCR | m,1 | 1 | | 1 | | |
| RCL | m,i | 10 | | 11 | | |
| RCR | m,i | 9 | | 11 | | |
| RCL | m,CL | 9 | | 11 | | |
| RCR | m,CL | 8 | | 11 | | |
| SHLD, SHRD | r,r,i | 6 | 3 | 3 | | |
| SHLD, SHRD | r,r,cl | 7 | 4 | 4 | | |
| SHLD, SHRD | m,r,i/CL | 8 | | 11 | | |
| BT | r,r/i | 1 | | 0.5 | | |
| BT | m,i | 1 | | 1 | | |
| BT | m,r | 5 | | 3 | | |
| BTC, BTR, BTS | r,r/i | 2 | 2 | 1 | | |
| BTC | m,i | 5 | | 11 | | |
| BTR, BTS | m,i | 4 | | 11 | | |

Jaguar

| | | | | | | |
|--------------------------------------|------------|-------|---|---------|-------|------------------------|
| BTC, BTR, BTS | m,r | 8 | | 11 | | |
| BSF | r,r | 7 | 4 | 4 | | |
| BSR | r,r | 8 | 4 | 4 | | |
| BSF, BSR | r,m | 8 | | 4 | | |
| POPCNT | r,r/m | 1 | 1 | 0.5 | | SSE4A/SSE4.2 |
| LZCNT | r,r | 1 | 1 | 0.5 | | SSE4A/LZCNT |
| TZCNT | r,r | 2 | 2 | 1 | | BMI1 |
| BLSI BLSR | r,r | 2 | 2 | 1 | | BMI1 |
| BLSI BLSR | r,m | 3 | | 2 | | BMI1 |
| BLSMSK | r,r | 2 | 2 | 1 | | BMI1 |
| BLSMSK | r,m | 3 | | 2 | | BMI1 |
| BEXTR | r,r,r | 1 | 1 | 0.5 | | BMI1 |
| BEXTR | r,m,r | 2 | | 1 | | BMI1 |
| SETcc | r | 1 | 1 | 0.5 | | |
| SETcc | m | 1 | | 1 | | |
| CLC, STC | | 1 | | 0.5 | I0/1 | |
| CMC | | 1 | 1 | | I0/1 | |
| CLD | | 1 | | 1 | I0 | |
| STD | | 2 | | 2 | I0,I1 | |
| Control transfer instructions | | | | | | |
| JMP | short/near | 1 | | 2 | | |
| JMP | r | 1 | | 2 | | |
| JMP | m(near) | 1 | | 2 | | |
| Jcc | short/near | 1 | | 0.5 - 2 | | 2 if jumping |
| J(E/R)CXZ | short | 2 | | 1 - 2 | | 2 if jumping |
| LOOP | short | 8 | | 5 | | |
| LOOPE LOOPNE | short | 10 | | 6 | | |
| CALL | near | 2 | | 2 | | |
| CALL | r | 2 | | 2 | | |
| CALL | m(near) | 5 | | 2 | | |
| RET | | 1 | | 3 | | |
| RET | i | 4 | | 3 | | |
| BOUND | m | 8 | | 4 | | values are for no jump |
| INTO | | 4 | | 2 | | values are for no jump |
| String instructions | | | | | | |
| LODS | | 4 | | 2 | | |
| REP LODS | | ~5n | | ~3n | | |
| STOS | | 4 | | 2 | | |
| REP STOS | | ~2n | | ~n | | for small n |
| REP STOS | | 2/16B | | 1/16B | | best case |
| MOVS | | 7 | | 4 | | |
| REP MOVS | | ~2n | | ~1.5n | | for small n |
| REP MOVS | | 2/16B | | 1/16B | | best case |
| SCAS | | 5 | | 3 | | |
| REP SCAS | | ~6n | | ~3n | | |
| CMPS | | 7 | | 4 | | |
| REP CMPS | | ~6n | | ~3n | | |
| Synchronization | | | | | | |

Jaguar

| | | | | | | |
|------------------|-------------|-------|--------|-------|------|-------------|
| LOCK ADD | m,r | 1 | 19 | | | |
| XADD | m,r | 4 | 11 | | | |
| LOCK XADD | m,r | 4 | 16 | | | |
| CMPXCHG | m,r8 | 5 | 11 | | | |
| LOCK CMPXCHG | m,r8 | 5 | 16 | | | |
| CMPXCHG | m,r16/32/64 | 6 | 11 | | | |
| LOCK CMPXCHG | m,r16/32/64 | 6 | 17 | | | |
| CMPXCHG8B | m64 | 18 | 11 | | | |
| LOCK CMPXCHG8B | m64 | 18 | 19 | | | |
| CMPXCHG16B | m128 | 28 | 32 | | | |
| LOCK CMPXCHG16B | m128 | 28 | 38 | | | |
| Other | | | | | | |
| NOP (90) | | 1 | | 0.5 | I0/1 | |
| Long NOP (0F 1F) | | 1 | | 0.5 | I0/1 | |
| PAUSE | | 37 | | 46 | | |
| ENTER | | i,0 | 12 | | 18 | |
| ENTER | | a,b | 10+6b | 17+3b | | |
| LEAVE | | 2 | | 3 | | 32 bit mode |
| CPUID | | 30-59 | 70-230 | | | |
| XGETBV | | 5 | | 5 | | |
| RDTSC | | 34 | | 41 | | |
| RDTSCP | | 34 | | 42 | | rdtscp |
| RDPMC | | 30 | | 27 | | |
| CRC32 | r,r | 3 | 3 | 2 | | |
| CRC32 | r,m | 4 | | 2 | | |

Floating point x87 instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipe | Notes |
|--------------------------------|----------|-----|---------|-----------------------|----------------|-------|
| Move instructions | | | | | | |
| FLD | r | 1 | 2 | 0.5 | FP0/1 | |
| FLD | m32/64 | 1 | 4 | 1 | FP0/1 | |
| FLD | m80 | 7 | 9 | 5 | | |
| FBLD | m80 | 21 | 24 | 29 | | |
| FST(P) | r | 1 | 2 | 0.5 | FP0/1 | |
| FST(P) | m32/64 | 1 | 3 | 1 | FP1 | |
| FSTP | m80 | 10 | 9 | 7 | | |
| FBSTP | m80 | 217 | 167 | 168 | | |
| FXCH | r | 1 | 0 | 1 | FP1 | |
| FILD | m | 1 | 8 | 1 | FP1 | |
| FIST(T)(P) | m | 1 | 4 | 1 | FP1 | |
| FLDZ, FLD1 | | 1 | | 1 | FP1 | |
| FCMOVcc | st0,r | 12 | 7 | 7 | FP0/1 | |
| FFREE | r | 1 | | 1 | FP1 | |
| FINCSTP, FDECSTP | | 1 | 1 | 1 | FP1 | |
| FNSTSW | AX | 2 | | 11 | FP1 | |
| FNSTSW | m16 | 2 | | 11 | FP1 | |
| FNSTCW | m16 | 3 | | 2 | FP0 | |
| FLDCW | m16 | 12 | | 9 | FP1 | |
| Arithmetic instructions | | | | | | |

Jaguar

| | | | | | |
|--------------------|---|-------|---------|---------|----------|
| FADD(P),FSUB(R)(P) | r | 1 | 3 | 1 | FP0 |
| FADD(P),FSUB(R)(P) | m | 1 | | 1 | FP0 |
| FIADD,FISUB(R) | m | 2 | | 2 | FP0,FP1 |
| FMUL(P) | r | 1 | 5 | 3 | FP1 |
| FMUL(P) | m | 1 | | 3 | FP1 |
| FIMUL | m | 1 | | | FP1 |
| FDIV(R)(P) | r | 1 | 22 | 22 | FP1 |
| FDIV(R)(P) | m | 1 | | 22 | FP1 |
| FIDIV(R) | m | 2 | | 22 | FP1 |
| FABS, FCHS | | 1 | 2 | 2 | FP1 |
| FCOM(P), FUCOM(P) | r | 1 | | 1 | FP0 |
| FCOM(P), FUCOM(P) | m | 1 | | 1 | FP0 |
| FCOMPP, FUCOMPP | | 1 | | 1 | FP0 |
| FCOMI(P) | r | 1 | | 2 | FP0 |
| FICOM(P) | m | 2 | | 1 | FP0, FP1 |
| FTST | | 1 | | 1 | FP0 |
| FXAM | | 2 | | 2 | 1FP1 |
| FRNDINT | | 5 | 8 | 4 | FP0, FP1 |
| FPREM | | 1 | 11-54 | | FP1 |
| FPREM1 | | 1 | 11-56 | | FP1 |
| Math | | | | | |
| FSQRT | | 1 | 35 | 35 | FP1 |
| FLDPI, etc. | | 1 | | 1 | FP0 |
| FSIN | | 4-44 | 30-139 | 30-151 | FP0, FP1 |
| FCOS | | 11-51 | 38-93 | | FP0, FP1 |
| FSINCOS | | 11-76 | 55-122 | 55-180 | FP0, FP1 |
| FPTAN | | 11-45 | 55-177 | 55-177 | FP0, FP1 |
| FPATAN | | 9-75 | 44-167 | 44-167 | FP0, FP1 |
| FSCALE | | 5 | 27 | | FP0, FP1 |
| FXTRACT | | 7 | 9 | 6 | FP0, FP1 |
| F2XM1 | | 8 | 32-37 | | FP0, FP1 |
| FYL2X | | 8-51 | 30-120 | 30-120 | FP0, FP1 |
| FYL2XP1 | | 61 | ~160 | ~160 | FP0, FP1 |
| Other | | | | | |
| FNOP | | 1 | | 0.5 | FP0/1 |
| (F)WAIT | | 1 | 0 | 0.5 | ALU |
| FNCLEX | | 9 | | 32 | FP0, FP1 |
| FNINIT | | 27 | | 78 | FP0, FP1 |
| FNSAVE | m | 88 | 138-150 | 138-150 | FP0, FP1 |
| FRSTOR | m | 80 | 136 | 136 | FP0, FP1 |

Integer vector instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipe | Notes |
|--------------------------|----------|-----|---------|-----------------------|----------------|-------|
| Move instructions | | | | | | |
| MOVD | r32, mm | 1 | 4 | 1 | FP0 | |
| MOVD | mm, r32 | 2 | 6 | 1 | FP0/1 | |
| MOVD | mm,m32 | 1 | 4 | 1 | AGU | |
| MOVD | r32, x | 1 | 4 | 1 | FP0 | |
| MOVD | x, r32 | 2 | 6 | 1 | FP1 | |

Jaguar

| | | | | | | |
|-------------------------|------------|----|---------|-----|----------|---|
| MOVD | x,m32 | 1 | 4 | 1 | AGU | Moves 64 bits.Name of instruction differs |
| MOVD | m32,mm/x | 1 | 3 | 1 | FP1 | |
| MOVD / MOVQ | r64,mm/x | 1 | 4 | 1 | FP0 | |
| MOVQ | mm,r64 | 2 | 6 | 1 | FP0/1 | |
| MOVQ | x,r64 | 2 | 6 | 1 | FP0/1 | do. |
| MOVQ | mm,mm | 1 | 1 | 0.5 | FP0/1 | do. |
| MOVQ | x,x | 1 | 1 | 0.5 | FP0/1 | |
| MOVQ | mm/x,m64 | 1 | 4 | 1 | AGU | |
| MOVQ | m64,mm/x | 1 | 3 | 1 | FP1 | |
| MOVDQA | x,x | 1 | 1 | 0.5 | FP0/1 | |
| VMOVDQA | y,y | 2 | 1 | 1 | FP0/1 | AVX |
| MOVDQA | x,m | 1 | 4 | 1 | AGU | |
| VMOVDQA | y,m | 2 | 4 | 2 | AGU | AVX |
| MOVDQA | m,x | 1 | 3 | 1 | FP1 | |
| VMOVDQA | m,y | 2 | 3 | 2 | FP1 | AVX |
| MOVDQU, LDDQU | x,m | 1 | 4 | 1 | AGU | |
| MOVDQU | m,x | 1 | 3 | 1 | FP1 | |
| MOVDQ2Q | mm,x | 1 | 1 | 0.5 | FP0/1 | |
| MOVQ2DQ | x,mm | 1 | 1 | 0.5 | FP0/1 | |
| MOVNTQ | m,mm | 1 | 429 | 2 | FP1 | |
| MOVNTDQ | m,x | 1 | 429 | 2 | FP1 | |
| PACKSSWB/DW | | | | | | |
| PACKUSWB | mm,r/m | 1 | 1 | 0.5 | FP0/1 | |
| PACKSSWB/DW | | | | | | |
| PACKUSWB | x,r/m | 1 | 2 | 0.5 | FP0/1 | |
| PUNPCKH/LBW/WD/DQ | mm,r/m | 1 | 1 | 0.5 | FP0/1 | |
| PUNPCKH/LBW/WD/DQ | x,r/m | 1 | 2 | 0.5 | FP0/1 | |
| PUNPCKH/LQDQ | x,r/m | 1 | 2 | 0.5 | FP0/1 | |
| PSHUFB | mm,mm | 1 | 1 | 0.5 | FP0/1 | Suppl. SSE3 |
| PSHUFB | x,x | 3 | 4 | 2 | FP0/1 | Suppl. SSE3 |
| PSHUFD | x,x,i | 1 | 2 | 0.5 | FP0/1 | |
| PSHUFW | mm,mm,i | 1 | 1 | 0.5 | FP0/1 | |
| PSHUFL/HW | x,x,i | 1 | 1 | 0.5 | FP0/1 | |
| PALIGNR | x,x,i | 1 | 2 | 0.5 | FP0/1 | Suppl. SSE3 |
| PBLENDW | x,r/m | 1 | 1 | 0.5 | FP0/1 | SSE4.1 |
| MASKMOVQ | mm,mm | 32 | 432 | 17 | FP0, FP1 | |
| MASKMOVDQU | x,x | 64 | 43-2210 | 34 | FP0, FP1 | |
| PMOVMASKB | r32,mm/x | 1 | 3 | 1 | FP0 | |
| PEXTRW | r32,mm/x,i | 1 | 4 | 1 | FP0 | |
| PINSRW | mm,r32,i | 2 | 8 | 1 | FP0/1 | |
| PINSRB/W/D/Q | x,r,i | 2 | 7 | 1 | FP0/1 | |
| PINSRB/W/D/Q | x,m,i | 1 | | 1 | FP0/1 | |
| PEXTRB/W/D/Q | r,x,i | 1 | 3 | 1 | FP0 | SSE4.1 |
| PEXTRB/W/D/Q | m,x,i | 1 | | 1 | FP1 | SSE4.1 |
| INSERTQ | x,x | 3 | 2 | 2 | FP0, FP1 | SSE4A, AMD only |
| INSERTQ | x,x,i,i | 3 | 2 | 2 | FP0, FP1 | SSE4A, AMD only |
| EXTRQ | x,x | 1 | 1 | 0.5 | FP0/1 | SSE4A, AMD only |
| EXTRQ | x,x,i,i | 1 | 1 | 0.5 | FP0/1 | SSE4A, AMD only |
| PMOVSXBW/BD/BQ/WD/WQ/DQ | x,x | 1 | 2 | 0.5 | FP0/1 | SSE4.1 |

Jaguar

| | | | | | | |
|-----------------------------------|-----------|----|---|-----|---------|-------------|
| PMOVZXBW/BD/BQ/ WD/WQ/DQ | x,x | 1 | 2 | 0.5 | FP0/1 | SSE4.1 |
| Arithmetic instructions | | | | | | |
| PADDB/W/D/Q | | | | | | |
| PADDSB/W ADDUSB/ W PSUBB/W/D/Q | | | | | | |
| PSUBSB/W | | | | | | |
| PSUBUSB/W | mm/x,r/m | 1 | 1 | 0.5 | FP0/1 | |
| PHADD/SUBW/SW/D | mm,r/m | 1 | 1 | 0.5 | FP0/1 | Suppl. SSE3 |
| PHADD/SUBW/SW/D | x,r/m | 1 | 2 | 0.5 | FP0/1 | Suppl. SSE3 |
| PCMPEQ/GT B/W/D | mm,r/m | 1 | 1 | 0.5 | FP0/1 | |
| PCMPEQ/GT B/W/D | x,r/m | 1 | 1 | 0.5 | FP0/1 | |
| PCMPEQQ | mm/x,r/m | 1 | 1 | 0.5 | FP0/1 | SSE4.1 |
| PCMPGTQ | mm/x,r/m | 1 | 1 | 0.5 | FP0/1 | SSE4.2 |
| PMULLW PMULHW | | | | | | |
| PMULHUW | | | | | | |
| PMULUDQ | mm/x,r/m | 1 | 2 | 1 | FP0 | |
| PMULLD | x,r/m | 3 | 4 | 2 | FP0 FP1 | SSE4.1 |
| PMULDQ | x,r/m | 1 | 2 | 1 | FP0 | SSE4.1 |
| PMULHSW | mm/x,r/m | 1 | 2 | 1 | FP0 | Suppl. SSE3 |
| PMADDWD | mm/x,r/m | 1 | 2 | 1 | FP0 | |
| PMADDUBSW | mm/x,r/m | 1 | 2 | 1 | FP0 | Suppl. SSE3 |
| PAVGB/W | mm/x,r/m | 1 | 1 | 0.5 | FP0/1 | |
| PMIN/MAX SW/UB | mm/x,r/m | 1 | 1 | 0.5 | FP0/1 | |
| PASB/W/D | mm/x,r/m | 1 | 1 | 0.5 | FP0/1 | Suppl. SSE3 |
| PSIGNB/W/D | mm/x,r/m | 1 | 1 | 0.5 | FP0/1 | Suppl. SSE3 |
| PSADBW | mm/x,r/m | 1 | 2 | 0.5 | FP0/1 | |
| MPSADBW | x,x,i | 3 | 4 | 1 | FP0/1 | SSE4.1 |
| Logic | | | | | | |
| PAND PANDN POR PXOR | mm/x,r/m | 1 | 1 | 0.5 | FP0/1 | |
| PSLL/RL W/D/Q | | | | | | |
| PSRAW/D | mm,i/mm/m | 1 | 1 | 0.5 | FP0/1 | |
| PSLL/RL W/D/Q | | | | | | |
| PSRAW/D | x,x | 1 | 2 | 0.5 | FP0/1 | |
| PSLL/RL W/D/Q | | | | | | |
| PSRAW/D | x,i | 1 | 1 | 0.5 | FP0/1 | |
| PSLLDQ, PSRLDQ | x,i | 1 | 2 | 0.5 | FP0/1 | |
| PTEST | x,x/m | 1 | 3 | 1 | FP0 | SSE4.1 |
| String instructions | | | | | | |
| PCMPESTRI | x,x,i | 9 | 5 | 5 | FP0/1 | SSE4.2 |
| PCMPESTRI | x,m,i | 10 | | 5 | FP0/1 | SSE4.2 |
| PCMPESTRM | x,x,i | 9 | 9 | 9 | FP0/1 | SSE4.2 |
| PCMPESTRM | x,m,i | 10 | | 9 | FP0/1 | SSE4.2 |
| PCMPISTRI | x,x,i | 3 | 2 | 2 | FP0/1 | SSE4.2 |
| PCMPISTRI | x,m,i | 4 | | 2 | FP0/1 | SSE4.2 |
| PCMPISTRM | x,x,i | 3 | 8 | 8 | FP0/1 | SSE4.2 |
| PCMPISTRM | x,m,i | 4 | | 2 | FP0/1 | SSE4.2 |
| Encryption | | | | | | |
| PCLMULQDQ | x,x/m,i | 1 | 3 | 1 | FP0 | PCLMUL |
| AESDEC | x,x | 2 | 5 | 1 | FP0/1 | AES |

Jaguar

| | | | | | | |
|-----------------|-------|---|---|-----|-------|-----|
| AESDECLAST | x,x | 2 | 5 | 1 | FP0/1 | AES |
| AESENC | x,x | 2 | 5 | 1 | FP0/1 | AES |
| AESENCLAST | x,x | 2 | 5 | 1 | FP0/1 | AES |
| AESIMC | x,x | 1 | 2 | 1 | FP0/1 | AES |
| AESKEYGENASSIST | x,x,i | 1 | 2 | 1 | FP0/1 | AES |
| Other | | | | | | |
| EMMS | | 1 | | 0.5 | FP0/1 | |

Floating point XMM instructions

| Instruction | Operands | Ops | Latency | Reciprocal throughput | Execution pipe | Notes |
|--------------------------|----------|-----|---------|-----------------------|----------------|-----------------|
| Move instructions | | | | | | |
| MOVAPS/D | x,x | 1 | 1 | 0.5 | FP0/1 | |
| VMOVAPS/D | y,y | 2 | 1 | 1 | FP0/1 | |
| MOVAPS/D | x,m | 1 | 4 | 1 | AGU | |
| VMOVAPS/D | y,m | 2 | 4 | 2 | AGU | |
| MOVAPS/D | m,x | 1 | 3 | 1 | FP1 | |
| VMOVAPS/D | m,y | 2 | 3 | 2 | FP1 | |
| MOVUPS/D | x,x | 1 | 1 | 0.5 | FP0/1 | |
| VMOVUPS/D | y,y | 2 | 1 | 1 | FP0/1 | |
| MOVUPS/D | x,m | 1 | 4 | 1 | AGU | |
| VMOVUPS/D | y,m | 2 | 4 | 2 | AGU | |
| MOVUPS/D | m,x | 1 | 3 | 1 | FP1 | |
| VMOVUPS/D | m,y | 2 | 3 | 2 | FP1 | |
| MOVSS/D | x,x | 1 | 1 | 0.5 | FP0/1 | |
| MOVSS/D | x,m | 1 | 4 | 1 | AGU | |
| MOVSS/D | m,x | 1 | 3 | 1 | FP1 | |
| MOVHLPs, MOVLHPS | x,x | 1 | 2 | 2 | FP0/1 | |
| MOVHPS/D, MOVLPS/D | x,m | 1 | 5 | 1 | FP0/1 | |
| MOVHPS/D, MOVLPS/D | m,x | 1 | 4 | 1 | FP1 | |
| MOVNTPS/D | m,x | 1 | 429 | 1 | FP1 | |
| MOVNTSS/D | m,x | 1 | | 1 | FP1 | SSE4A, AMD only |
| MOVDDUP | x,x | 1 | 2 | 0.5 | FP0/1 | SSE3 |
| MOVDDUP | x,m64 | 1 | | 1 | AGU | SSE3 |
| VMOVDDUP | y,y | 2 | 2 | 1 | FP0/1 | AVX |
| VMOVDDUP | y,m | 2 | | 2 | AGU | AVX |
| MOVSH/LDUP | x,x | 1 | 1 | 0.5 | FP0/1 | |
| MOVSH/LDUP | x,m | 1 | | 1 | AGU | |
| VMOVSH/LDUP | y,y | 2 | 1 | 1 | FP0/1 | AVX |
| VMOVSH/LDUP | y,m | 2 | | 2 | AGU | AVX |
| MOVMSKPS/D | r32,x | 1 | 3 | 1 | FP0 | |
| VMOVMSKPS/D | r32,y | 1 | 3 | 1 | FP0 | AVX |
| SHUFPS/D | x,x/m,i | 1 | 2 | 0.5 | FP0/1 | |
| VSHUFPS/D | y,y,y,i | 2 | 2 | 1 | FP0/1 | AVX |
| UNPCK H/L PS/D | x,x/m | 1 | 2 | 0.5 | FP0/1 | |
| VUNPCK H/L PS/D | y,y,y | 2 | 2 | 1 | FP0/1 | AVX |
| EXTRACTPS | r32,x,i | 1 | 3 | 1 | FP0 | |
| EXTRACTPS | m32,x,i | 1 | 3 | 1 | FP1 | |

Jaguar

| | | | | | | |
|-------------------|------------|----|----|-----|----------|--------------------|
| VEXTRACTF128 | x,y,i | 1 | 1 | 0.5 | FP0/1 | AVX |
| VEXTRACTF128 | m128,y,i | 1 | 12 | 1 | FP1 | AVX |
| INSERTPS | x,x,i | 1 | | 1 | FP0/1 | |
| INSERTPS | x,m32,i | 1 | 6 | 1 | FP0/1 | |
| VINSERTF128 | y,y,x,i | 2 | 1 | 1 | FP0/1 | AVX |
| VINSERTF128 | y,y,m128,i | 2 | 13 | 2 | FP0/1 | AVX |
| VMASKMOVPS/D | x,x,m128 | 1 | 15 | 1 | FP0/1 | >300 clk if mask=0 |
| VMASKMOVPS/D | y,y,m256 | 2 | 15 | 2 | FP0/1 | >300 clk if mask=0 |
| VMASKMOVPS/D | m128,x,x | 19 | 21 | 16 | FP1 | AVX |
| VMASKMOVPS/D | m256,y,y | 36 | 32 | 22 | FP1 | AVX |
| Conversion | | | | | | |
| CVTPS2PD | x,x/m | 1 | 3 | 1 | FP1 | |
| VCVTPS2PD | y,x/m | 2 | 4 | 2 | FP1 | |
| CVTPD2PS | x,x/m | 1 | 4 | 1 | FP1 | |
| VCVTPD2PS | x,y | 3 | 6 | 2 | FP0, FP1 | |
| CVTSD2SS | x,x/m | 2 | 5 | 8 | FP1 | |
| CVTSS2SD | x,x/m | 2 | 4 | 7 | FP1 | |
| CVTDQ2PS/PD | x,x/m | 1 | 4 | 1 | FP1 | |
| VCVTDQ2PS/PD | y,y | 2 | 4 | 2 | FP1 | |
| CVT(T)PS2DQ | x,x/m | 1 | 4 | 1 | FP1 | |
| VCVT(T)PS2DQ | y,y | 2 | 4 | 2 | FP1 | |
| CVT(T)PD2DQ | x,x/m | 1 | 4 | 1 | FP1 | |
| VCVT(T)PD2DQ | y,y | 3 | 7 | 2 | FP1 | |
| CVTPI2PS | xmm,mm | 1 | 4 | 1 | FP1 | |
| CVTPI2PD | xmm,mm | 1 | 4 | 1 | FP1 | |
| CVT(T)PS2PI | mm,xmm | 1 | 4 | 1 | FP1 | |
| CVT(T)PD2PI | mm,xmm | 1 | 4 | 1 | FP1 | |
| CVTSI2SS | xmm,r32 | 2 | 9 | 1 | FP1 | |
| CVTSI2SD | xmm,r32 | 2 | 9 | 1 | FP1 | |
| CVT(T)SS2SI | r32,xmm | 2 | 8 | 1 | FP1 | |
| CVT(T)SD2SI | r32,xmm | 2 | 8 | 1 | FP1 | |
| VCVTPS2PH | x/m,x,i | 1 | 4 | 1 | FP1 | F16C |
| VCVTPS2PH | x/m,y,i | 3 | 6 | 2 | FP0, FP1 | F16C |
| VCVTPH2PS | x,x/m | 1 | 4 | 1 | FP1 | F16C |
| VCVTPH2PS | y,x/m | 2 | 5 | 2 | FP1 | F16C |
| Arithmetic | | | | | | |
| ADDSS/D SUBSS/D | x,x/m | 1 | 3 | 1 | FP0 | |
| ADDPS/D SUBPS/D | x,x/m | 1 | 3 | 1 | FP0 | |
| VADDPS/D VSUBPS/D | y,y/m | 2 | 3 | 2 | FP0 | |
| ADDSUBPS/D | x,x/m | 1 | 3 | 1 | FP0 | SSE3 |
| VADDSUBPS/D | y,y/m | 2 | 3 | 2 | FP0 | |
| HADD/SUBPS/D | x,x/m | 1 | 4 | 1 | FP0 | SSE3 |
| VHADD/SUBPS/D | y,y/m | 2 | 4 | 2 | FP0 | |
| MULSS/PS | x,x/m | 1 | 2 | 1 | FP1 | |
| VMULPS | y,y/m | 2 | 2 | 2 | FP1 | |
| MULSD/PD | x,x/m | 1 | 4 | 2 | FP1 | |
| VMULPD | y,y/m | 2 | 4 | 2 | FP1 | |
| DIVSS | x,x/m | 1 | 14 | 14 | FP1 | |
| DIVPS | x,x/m | 1 | 19 | 19 | FP1 | |
| VDIVPS | y,y/m | 2 | 38 | 38 | FP1 | |
| DIVSD | x,x/m | 1 | 19 | 19 | FP1 | |

Jaguar

| | | | | | | |
|-------------------------|---------|-----|-----|-----|----------|-------------|
| DIVPD | x,x/m | 1 | 19 | 19 | FP1 | |
| VDIVPD | y,y/m | 2 | 38 | 38 | FP1 | |
| RCPSS | x,x/m | 1 | 2 | 1 | FP1 | |
| RCPPS | x,x/m | 1 | 2 | 1 | FP1 | |
| VRCPSS | y,y/m | 2 | 2 | 2 | FP1 | |
| MAXSS/D MINSS/D | x,x/m | 1 | 2 | 1 | FP0 | |
| MAXPS/D MINPS/D | x,x/m | 1 | 2 | 1 | FP0 | |
| VMAXPS/D VMINPS/D | y,y/m | 2 | 2 | 2 | FP0 | |
| CMP _{cc} SS/D | x,x/m | 1 | 2 | 1 | FP0 | |
| CMP _{cc} PS/D | x,x/m | 1 | 2 | 1 | FP0 | |
| VCMP _{cc} PS/D | y,y/m | 2 | 2 | 2 | FP0 | |
| (U)COMISS/D | x,x/m | 1 | | 1 | FP0 | |
| ROUNDSS/SD/PS/PD | x,x/m,i | 1 | 4 | 1 | FP1 | |
| VROUNDSS/D/PS/D | y,y/m,i | 2 | 4 | 2 | FP1 | |
| DPPS | x,x,i | 5 | 11 | 4 | FP0, FP1 | SSE4.1 |
| DPPS | x,m,i | 6 | | 4 | FP0, FP1 | SSE4.1 |
| VDPPS | y,y,y,i | 10 | 12 | 7 | FP0, FP1 | SSE4.1 |
| VDPPS | y,m,i | 12 | | 7 | FP0, FP1 | SSE4.1 |
| DPPD | x,x,i | 3 | 9 | 3 | FP0, FP1 | SSE4.1 |
| DPPD | x,m,i | 4 | | 3 | FP0, FP1 | SSE4.1 |
| Logic | | | | | | |
| ANDPS/D ANDNPS/D | | | | | | |
| ORPS/D XORPS/D | x,x/m | 1 | 1 | 0.5 | FP0/1 | |
| VANDPS/D, etc. | y,y/m | 2 | 1 | 1 | FP0/1 | |
| Math | | | | | | |
| SQRTSS | x,x/m | 1 | 16 | 16 | FP1 | |
| SQRTPS | x,x/m | 2 | 21 | 21 | FP1 | |
| VSQRTPS | y,y/m | 2 | 42 | 42 | FP1 | |
| SQRTSD | x,x/m | 1 | 27 | 27 | FP1 | |
| SQRTPD | x,x/m | 2 | 27 | 27 | FP1 | |
| VSQRTPD | y,y/m | 2 | 54 | 54 | FP1 | |
| RSQRTSS/PS | x,x/m | 1 | 2 | 1 | FP1 | |
| VRSQRTPS | y,y/m | 2 | 2 | 2 | FP1 | |
| Other | | | | | | |
| LDMXCSR | m | 12 | 9 | 8 | FP0, FP1 | |
| STMXCSR | m | 3 | 13 | 12 | FP0, FP1 | |
| VZEROUPPER | | 21 | | 30 | | 32 bit mode |
| VZEROUPPER | | 37 | | 46 | | 64 bit mode |
| VZEROALL | | 41 | | 58 | | 32 bit mode |
| VZEROALL | | 73 | | 90 | | 64 bit mode |
| FXSAVE | | 66 | 66 | 66 | | 32 bit mode |
| FXSAVE | | 58 | 58 | 58 | | 64 bit mode |
| FXRSTOR | | 115 | 189 | 189 | | 32 bit mode |
| FXRSTOR | | 123 | 198 | 197 | | 64 bit mode |
| XSAVE | | 130 | 145 | 145 | | 32 bit mode |
| XSAVE | | 114 | 129 | 129 | | 64 bit mode |
| XRSTOR | | 219 | 342 | 342 | | 32 bit mode |
| XRSTOR | | 251 | 375 | 375 | | 64 bit mode |

Intel Pentium and Pentium MMX**List of instruction timings***Explanation of column headings:*

| | |
|---------------------|---|
| Operands | r = register, accum = al, ax or eax, m = memory, i = immediate data, sr = segment register, m32 = 32 bit memory operand, etc. |
| Clock cycles | The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. |
| Pairability | u = pairable in u-pipe, v = pairable in v-pipe, uv = pairable in either pipe, np = not pairable. |

Integer instructions (Pentium and Pentium MMX)

| Instruction | Operands | Clock cycles | Pairability |
|---------------------|-----------------|---------------------|--------------------|
| NOP | | 1 | uv |
| MOV | r/m, r/m/i | 1 | uv |
| MOV | r/m, sr | 1 | np |
| MOV | sr, r/m | >= 2 b) | np |
| MOV | m, accum | 1 | uv h) |
| XCHG | (E)AX, r | 2 | np |
| XCHG | r, r | 3 | np |
| XCHG | r, m | >15 | np |
| XLAT | | 4 | np |
| PUSH | r/i | 1 | uv |
| POP | r | 1 | uv |
| PUSH | m | 2 | np |
| POP | m | 3 | np |
| PUSH | sr | 1 b) | np |
| POP | sr | >= 3 b) | np |
| PUSHF | | 3-5 | np |
| POPF | | 4-6 | np |
| PUSHA POPA | | 5-9 i) | np |
| PUSHAD POPAD | | 5 | np |
| LAHF SAHF | | 2 | np |
| MOVSX MOVZX | r, r/m | 3 a) | np |
| LEA | r, m | 1 | uv |
| LDS LES LFS LGS LSS | m | 4 c) | np |
| ADD SUB AND OR XOR | r, r/i | 1 | uv |
| ADD SUB AND OR XOR | r, m | 2 | uv |
| ADD SUB AND OR XOR | m, r/i | 3 | uv |
| ADC SBB | r, r/i | 1 | u |
| ADC SBB | r, m | 2 | u |
| ADC SBB | m, r/i | 3 | u |
| CMP | r, r/i | 1 | uv |
| CMP | m, r/i | 2 | uv |
| TEST | r, r | 1 | uv |
| TEST | m, r | 2 | uv |
| TEST | r, i | 1 | f) |
| TEST | m, i | 2 | np |
| INC DEC | r | 1 | uv |
| INC DEC | m | 3 | uv |
| NEG NOT | r/m | 1/3 | np |
| MUL IMUL | r8/r16/m8/m16 | 11 | np |

Intel Pentium

| | | | |
|---------------------|--------------------|------------|----|
| MUL IMUL | all other versions | 9 d) | np |
| DIV | r8/m8 | 17 | np |
| DIV | r16/m16 | 25 | np |
| DIV | r32/m32 | 41 | np |
| IDIV | r8/m8 | 22 | np |
| IDIV | r16/m16 | 30 | np |
| IDIV | r32/m32 | 46 | np |
| CBW CWDE | | 3 | np |
| CWD CDQ | | 2 | np |
| SHR SHL SAR SAL | r, i | 1 | u |
| SHR SHL SAR SAL | m, i | 3 | u |
| SHR SHL SAR SAL | r/m, CL | 4/5 | np |
| ROR ROL RCR RCL | r/m, 1 | 1/3 | u |
| ROR ROL | r/m, i(><1) | 1/3 | np |
| ROR ROL | r/m, CL | 4/5 | np |
| RCR RCL | r/m, i(><1) | 8/10 | np |
| RCR RCL | r/m, CL | 7/9 | np |
| SHLD SHRD | r, i/CL | 4 a) | np |
| SHLD SHRD | m, i/CL | 5 a) | np |
| BT | r, r/i | 4 a) | np |
| BT | m, i | 4 a) | np |
| BT | m, i | 9 a) | np |
| BTR BTS BTC | r, r/i | 7 a) | np |
| BTR BTS BTC | m, i | 8 a) | np |
| BTR BTS BTC | m, r | 14 a) | np |
| BSF BSR | r, r/m | 7-73 a) | np |
| SETcc | r/m | 1/2 a) | np |
| JMP CALL | short/near | 1 e) | v |
| JMP CALL | far | >= 3 e) | np |
| conditional jump | short/near | 1/4/5/6 e) | v |
| CALL JMP | r/m | 2/5 e | np |
| RETN | | 2/5 e | np |
| RETN | i | 3/6 e) | np |
| RETF | | 4/7 e) | np |
| RETF | i | 5/8 e) | np |
| J(E)CXZ | short | 4-11 e) | np |
| LOOP | short | 5-10 e) | np |
| BOUND | r, m | 8 | np |
| CLC STC CMC CLD STD | | 2 | np |
| CLI STI | | 6-9 | np |
| LODS | | 2 | np |
| REP LODS | | 7+3*n g) | np |
| STOS | | 3 | np |
| REP STOS | | 10+n g) | np |
| MOVS | | 4 | np |
| REP MOVS | | 12+n g) | np |
| SCAS | | 4 | np |
| REP(N)E SCAS | | 9+4*n g) | np |
| CMPS | | 5 | np |
| REP(N)E CMPS | | 8+4*n g) | np |
| BSWAP | r | 1 a) | np |
| CPUID | | 13-16 a) | np |
| RDTSC | | 6-13 a) j) | np |

Notes:

| | |
|---|---|
| a | This instruction has a 0FH prefix which takes one clock cycle extra to decode on a P1 unless preceded by a multi-cycle instruction. |
| b | versions with FS and GS have a 0FH prefix. see note a. |
| c | versions with SS, FS, and GS have a 0FH prefix. see note a. |
| d | versions with two operands and no immediate have a 0FH prefix, see note a. |
| e | high values are for mispredicted jumps/branches. |
| f | only pairable if register is AL, AX or EAX. |
| g | add one clock cycle for decoding the repeat prefix unless preceded by a multi-cycle instruction (such as CLD). |
| h | pairs as if it were writing to the accumulator. |
| i | 9 if SP divisible by 4 (imperfect pairing). |
| j | on P1: 6 in privileged or real mode; 11 in non-privileged; error in virtual mode. On PMMX: 8 and 13 clocks respectively. |

Floating point instructions (Pentium and Pentium MMX)*Explanation of column headings*

| | |
|---------------------|--|
| Operands | r = register, m = memory, m32 = 32-bit memory operand, etc. |
| Clock cycles | The numbers are minimum values. Cache misses, misalignment, denormal operands, and exceptions may increase the clock counts considerably. |
| Pairability | + = pairable with FXCH, np = not pairable with FXCH. |
| i-ov | Overlap with integer instructions. i-ov = 4 means that the last four clock cycles can overlap with subsequent integer instructions. |
| fp-ov | Overlap with floating point instructions. fp-ov = 2 means that the last two clock cycles can overlap with subsequent floating point instructions. (WAIT is considered a floating point instruction here) |

| Instruction | Operand | Clock cycles | Pairability | i-ov | fp-ov |
|-------------------|-----------|--------------|-------------|-------|-------|
| FLD | r/m32/m64 | 1 | 0 | 0 | 0 |
| FLD | m80 | 3 | np | 0 | 0 |
| FBLD | m80 | 48-58 | np | 0 | 0 |
| FST(P) | r | 1 | np | 0 | 0 |
| FST(P) | m32/m64 | 2 m) | np | 0 | 0 |
| FST(P) | m80 | 3 m) | np | 0 | 0 |
| FBSTP | m80 | 148-154 | np | 0 | 0 |
| FILD | m | 3 | np | 2 | 2 |
| FIST(P) | m | 6 | np | 0 | 0 |
| FLDZ FLD1 | | 2 | np | 0 | 0 |
| FLDPI FLDL2E etc. | | 5 s) | np | 2 | 2 |
| FNSTSW | AX/m16 | 6 q) | np | 0 | 0 |
| FLDCW | m16 | 8 | np | 0 | 0 |
| FNSTCW | m16 | 2 | np | 0 | 0 |
| FADD(P) | r/m | 3 | 0 | 2 | 2 |
| FSUB(R)(P) | r/m | 3 | 0 | 2 | 2 |
| FMUL(P) | r/m | 3 | 0 | 2 | 2 n) |
| FDIV(R)(P) | r/m | 19/33/39 p) | 0 | 38 o) | 2 |
| FCHS FABS | | 1 | 0 | 0 | 0 |
| FCOM(P)(P) FUCOM | r/m | 1 | 0 | 0 | 0 |
| FIADD FISUB(R) | m | 6 | np | 2 | 2 |
| FIMUL | m | 6 | np | 2 | 2 |

Intel Pentium

| | | | | | |
|-----------------|---|-------------|----|-------|---|
| FIDIV(R) | m | 22/36/42 p) | np | 38 o) | 2 |
| FICOM | m | 4 | np | 0 | 0 |
| FTST | | 1 | np | 0 | 0 |
| FXAM | | 17-21 | np | 4 | 0 |
| FPREM | | 16-64 | np | 2 | 2 |
| FPREM1 | | 20-70 | np | 2 | 2 |
| FRNDINT | | 9-20 | np | 0 | 0 |
| FSCALE | | 20-32 | np | 5 | 0 |
| FXTRACT | | 12-66 | np | 0 | 0 |
| FSQRT | | 70 | np | 69 o) | 2 |
| FSIN FCOS | | 65-100 r) | np | 2 | 2 |
| FSINCOS | | 89-112 r) | np | 2 | 2 |
| F2XM1 | | 53-59 r) | np | 2 | 2 |
| FYL2X | | 103 r) | np | 2 | 2 |
| FYL2XP1 | | 105 r) | np | 2 | 2 |
| FPTAN | | 120-147 r) | np | 36 o) | 0 |
| FPATAN | | 112-134 r) | np | 2 | 2 |
| FNOP | | 1 | np | 0 | 0 |
| FXCH | r | 1 | np | 0 | 0 |
| FINCSTP FDECSTP | | 2 | np | 0 | 0 |
| FFREE | r | 2 | np | 0 | 0 |
| FNCLEX | | 6-9 | np | 0 | 0 |
| FNINIT | | 12-22 | np | 0 | 0 |
| FNSAVE | m | 124-300 | np | 0 | 0 |
| FRSTOR | m | 70-95 | np | 0 | 0 |
| WAIT | | 1 | np | 0 | 0 |

Notes:

| | |
|---|---|
| m | The value to store is needed one clock cycle in advance. |
| n | 1 if the overlapping instruction is also an FMUL. |
| o | Cannot overlap integer multiplication instructions. |
| p | FDIV takes 19, 33, or 39 clock cycles for 24, 53, and 64 bit precision respectively. FIDIV takes 3 clocks more. The precision is defined by bit 8-9 of the floating point control word. |
| q | The first 4 clock cycles can overlap with preceding integer instructions. |
| r | Clock counts are typical. Trivial cases may be faster, extreme cases may be slower. |
| s | May be up to 3 clocks more when output needed for FST, FCHS, or FABS. |

MMX instructions (Pentium MMX)

A list of MMX instruction timings is not needed because they all take one clock cycle, except the MMX multiply instructions which take 3. MMX multiply instructions can be pipelined to yield a throughput of one multiplication per clock cycle.

The EMMS instruction takes only one clock cycle, but the first floating point instruction after an EMMS takes approximately 58 clocks extra, and the first MMX instruction after a floating point instruction takes approximately 38 clocks extra. There is no penalty for an MMX instruction after EMMS on the PMMX.

There is no penalty for using a memory operand in an MMX instruction because the MMX arithmetic unit is one step later in the pipeline than the load unit. But the penalty comes when you store data from an MMX register to memory or to a 32-bit register: The data have to be ready one clock cycle in advance. This is analogous to the floating point store instructions.

All MMX instructions except EMMS are pairable in either pipe. Pairing rules for MMX instructions are described in manual 3: "The microarchitecture of Intel, AMD and VIA CPUs".

Intel Pentium II and Pentium III

List of instruction timings and μ op breakdown

Explanation of column headings:

Operands: i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc.

μ ops: The number of μ ops that the instruction generates for each execution port.

p0: Port 0: ALU, etc.

p1: Port 1: ALU, jumps

p01: Instructions that can go to either port 0 or 1, whichever is vacant first.

p2: Port 2: load data, etc.

p3: Port 3: address generation for store

p4: Port 4: store data

Latency: This is the delay that the instruction generates in a dependency chain. (This is not the same as the time spent in the execution unit. Values may be inaccurate in situations where they cannot be measured exactly, especially with memory operands). The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays by 50-150 clocks, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay.

Reciprocal throughput: The average number of clock cycles per instruction for a series of independent instructions of the same kind.

Integer instructions (Pentium Pro, Pentium II and Pentium III)

| Instruction | Operands | μ ops | | | | | | Latency | Reciprocal throughput |
|-------------|----------|-----------|----|-----|----|----|----|---------|-----------------------|
| | | p0 | p1 | p01 | p2 | p3 | p4 | | |
| MOV | r,r/i | | | 1 | | | | 5 8 | |
| MOV | r,m | | | | 1 | | | | |
| MOV | m,r/i | | | | | 1 | 1 | | |
| MOV | r,sr | | | 1 | | | | | |
| MOV | m,sr | | | 1 | | 1 | 1 | | |
| MOV | sr,r | 8 | | | | | | | |
| MOV | sr,m | 7 | | | 1 | | | | |
| MOVSX MOVZX | r,r | | | 1 | | | | | |
| MOVSX MOVZX | r,m | | | | 1 | | | | |
| CMOVcc | r,r | 1 | | 1 | | | | | |
| CMOVcc | r,m | 1 | | 1 | 1 | | | high b) | |
| XCHG | r,r | | | 3 | | | | | |
| XCHG | r,m | | | 4 | 1 | 1 | 1 | | |
| XLAT | | | | 1 | 1 | | | | |
| PUSH | r/i | | | 1 | | 1 | 1 | | |
| POP | r | | | 1 | 1 | | | | |
| POP | (E)SP | | | 2 | 1 | | | | |
| PUSH | m | | | 1 | 1 | 1 | 1 | | |
| POP | m | | | 5 | 1 | 1 | 1 | | |
| PUSH | sr | | | 2 | | 1 | 1 | | |
| POP | sr | | | 8 | 1 | | | | |
| PUSHF(D) | | 3 | | 11 | | 1 | 1 | | |
| POPF(D) | | 10 | | 6 | 1 | | | | |
| PUSHA(D) | | | | 2 | | 8 | 8 | | |
| POPA(D) | | | | 2 | 8 | | | | |

Pentium II and III

| | | | | | | | | | |
|--------------------|-------------|----|---|---|---|---|---|------|----|
| LAHF SAHF | | | | 1 | | | | | |
| LEA | r,m | 1 | | | | | | 1 c) | |
| LDS LES LFS LGS | | | | | | | | | |
| LSS | m | | | 8 | 3 | | | | |
| ADD SUB AND OR XOR | r,r/i | | | 1 | | | | | |
| ADD SUB AND OR XOR | r,m | | | 1 | 1 | | | | |
| ADD SUB AND OR XOR | m,r/i | | | 1 | 1 | 1 | 1 | | |
| ADC SBB | r,r/i | | | 2 | | | | | |
| ADC SBB | r,m | | | 2 | 1 | | | | |
| ADC SBB | m,r/i | | | 3 | 1 | 1 | 1 | | |
| CMP TEST | r,r/i | | | 1 | | | | | |
| CMP TEST | m,r/i | | | 1 | 1 | | | | |
| INC DEC NEG NOT | r | | | 1 | | | | | |
| INC DEC NEG NOT | m | | | 1 | 1 | 1 | 1 | | |
| AAA AAS DAA DAS | | | 1 | | | | | | |
| AAD | | 1 | | 2 | | | | 4 | |
| AAM | | 1 | 1 | 2 | | | | 15 | |
| IMUL | r,(r),(i) | 1 | | | | | | 4 | 1 |
| IMUL | (r),m | 1 | | | 1 | | | 4 | 1 |
| DIV IDIV | r8 | 2 | | 1 | | | | 19 | 12 |
| DIV IDIV | r16 | 3 | | 1 | | | | 23 | 21 |
| DIV IDIV | r32 | 3 | | 1 | | | | 39 | 37 |
| DIV IDIV | m8 | 2 | | 1 | 1 | | | 19 | 12 |
| DIV IDIV | m16 | 2 | | 1 | 1 | | | 23 | 21 |
| DIV IDIV | m32 | 2 | | 1 | 1 | | | 39 | 37 |
| CBW CWDE | | | | 1 | | | | | |
| CWD CDQ | | 1 | | | | | | | |
| SHR SHL SAR ROR | | | | | | | | | |
| ROL | r,i/CL | 1 | | | | | | | |
| SHR SHL SAR ROR | | | | | | | | | |
| ROL | m,i/CL | 1 | | | 1 | 1 | 1 | | |
| RCR RCL | r,1 | 1 | | 1 | | | | | |
| RCR RCL | r8,i/CL | 4 | | 4 | | | | | |
| RCR RCL | r16/32,i/CL | 3 | | 3 | | | | | |
| RCR RCL | m,1 | 1 | | 2 | 1 | 1 | 1 | | |
| RCR RCL | m8,i/CL | 4 | | 3 | 1 | 1 | 1 | | |
| RCR RCL | m16/32,i/CL | 4 | | 2 | 1 | 1 | 1 | | |
| SHLD SHRD | r,r,i/CL | 2 | | | | | | | |
| SHLD SHRD | m,r,i/CL | 2 | | 1 | 1 | 1 | 1 | | |
| BT | r,r/i | | | 1 | | | | | |
| BT | m,r/i | 1 | | 6 | 1 | | | | |
| BTR BTS BTC | r,r/i | | | 1 | | | | | |
| BTR BTS BTC | m,r/i | 1 | | 6 | 1 | 1 | 1 | | |
| BSF BSR | r,r | | 1 | 1 | | | | | |
| BSF BSR | r,m | | 1 | 1 | 1 | | | | |
| SETcc | r | | | 1 | | | | | |
| SETcc | m | | | 1 | | 1 | 1 | | |
| JMP | short/near | | 1 | | | | | | 2 |
| JMP | far | 21 | | | 1 | | | | |
| JMP | r | | 1 | | | | | | 2 |
| JMP | m(near) | | 1 | | 1 | | | | 2 |
| JMP | m(far) | 21 | | | 2 | | | | |
| conditional jump | short/near | | 1 | | | | | | 2 |

Pentium II and III

| | | | | | | | | | |
|-------------------|---------|-------|--------|-----|----|-----|----|------|-----|
| CALL | near | | 1 | 1 | | 1 | 1 | | 2 |
| CALL | far | 28 | | | 1 | 2 | 2 | | |
| CALL | r | | 1 | 2 | | 1 | 1 | | 2 |
| CALL | m(near) | | 1 | 4 | 1 | 1 | 1 | | 2 |
| CALL | m(far) | 28 | | | 2 | 2 | 2 | | |
| RETN | | | 1 | 2 | 1 | | | | 2 |
| RETN | i | | 1 | 3 | 1 | | | | 2 |
| RETF | | 23 | | | 3 | | | | |
| RETF | i | 23 | | | 3 | | | | |
| J(E)CXZ | short | | 1 | 1 | | | | | |
| LOOP | short | 2 | 1 | 8 | | | | | |
| LOOP(N)E | short | 2 | 1 | 8 | | | | | |
| ENTER | i,0 | | | 12 | | 1 | 1 | | |
| ENTER | a,b | ca. | 18 | +4b | | b-1 | 2b | | |
| LEAVE | | | | 2 | 1 | | | | |
| BOUND | r,m | 7 | | 6 | 2 | | | | |
| CLC STC CMC | | | | 1 | | | | | |
| CLD STD | | | | 4 | | | | | |
| CLI | | 9 | | | | | | | |
| STI | | 17 | | | | | | | |
| INTO | | | | 5 | | | | | |
| LODS | | | | | 2 | | | | |
| REP LODS | | | 10+6n | | | | | | |
| STOS | | | | | 1 | 1 | 1 | | |
| REP STOS | | | ca. 5n | | a) | | | | |
| MOVS | | | | 1 | 3 | 1 | 1 | | |
| REP MOVS | | | ca. 6n | | a) | | | | |
| SCAS | | | | 1 | 2 | | | | |
| REP(N)E SCAS | | | 12+7n | | | | | | |
| CMPS | | | | 4 | 2 | | | | |
| REP(N)E CMPS | | | 12+9n | | | | | | |
| BSWAP | r | 1 | | 1 | | | | | |
| NOP (90) | | | | 1 | | | | | 0.5 |
| Long NOP (0F 1F) | | | | 1 | | | | | 1 |
| CPUID | | 23-48 | | | | | | | |
| RDTSC | | 31 | | | | | | | |
| IN | | 18 | | | | | | >300 | |
| OUT | | 18 | | | | | | >300 | |
| PREFETCHNTA d) | m | | | | 1 | | | | |
| PREFETCHT0/1/2 d) | m | | | | 1 | | | | |
| SFENCE d) | | | | | | 1 | 1 | | 6 |

Notes

- a) Faster under certain conditions: see manual 3: "The microarchitecture of Intel, AMD and VIA CPUs".
- b) Has an implicit LOCK prefix.
- c) 3 if constant without base or index register
- d) P3 only.

Floating point x87 instructions (Pentium Pro, II and III)

| Instruction | Operands | μops | | | | | | Latency | Reciprocal throughput |
|-------------|----------|------|----|-----|----|----|----|---------|-----------------------|
| | | p0 | p1 | p01 | p2 | p3 | p4 | | |
| FLD | r | 1 | | | | | | | |

Pentium II and III

| | | | | | | | | | |
|------------------------|---------|--------|---|---|---|---|--------|-------|--------|
| FLD | m32/64 | | | | 1 | | | 1 | |
| FLD | m80 | 2 | | | 2 | | | | |
| FBLD | m80 | 38 | | | 2 | | | | |
| FST(P) | r | 1 | | | | | | | |
| FST(P) | m32/m64 | | | | | 1 | 1 | 1 | |
| FSTP | m80 | 2 | | | | 2 | 2 | | |
| FBSTP | m80 | 165 | | | | 2 | 2 | | |
| FXCH | r | | | | | | | 0 | 1/3 f) |
| FILD | m | 3 | | | 1 | | | 5 | |
| FIST(P) | m | 2 | | | | 1 | 1 | 5 | |
| FLDZ | | 1 | | | | | | | |
| FLD1 FLDPI FLDL2E etc. | | 2 | | | | | | | |
| FCMOVcc | r | 2 | | | | | | 2 | |
| FNSTSW | AX | 3 | | | | | | 7 | |
| FNSTSW | m16 | 1 | | | | 1 | 1 | | |
| FLDCW | m16 | 1 | 1 | 1 | | | | 10 | |
| FNSTCW | m16 | 1 | | | | 1 | 1 | | |
| FADD(P) FSUB(R)(P) | r | 1 | | | | | | 3 | 1 |
| FADD(P) FSUB(R)(P) | m | 1 | | | 1 | | | 3-4 | 1 |
| FMUL(P) | r | 1 | | | | | | 5 | 2 g) |
| FMUL(P) | m | 1 | | | 1 | | | 5-6 | 2 g) |
| FDIV(R)(P) | r | 1 | | | | | | 38 h) | 37 |
| FDIV(R)(P) | m | 1 | | | 1 | | | 38 h) | 37 |
| FABS | | 1 | | | | | | | |
| FCHS | | 3 | | | | | | 2 | |
| FCOM(P) FUCOM | r | 1 | | | | | | 1 | |
| FCOM(P) FUCOM | m | 1 | | | 1 | | | 1 | |
| FCOMPP FUCOMPP | | 1 | 1 | | | | | 1 | |
| FCOMI(P) FUCOMI(P) | r | 1 | | | | | | 1 | |
| FCOMI(P) FUCOMI(P) | m | 1 | | | 1 | | | 1 | |
| FIADD FISUB(R) | m | 6 | | | 1 | | | | |
| FIMUL | m | 6 | | | 1 | | | | |
| FIDIV(R) | m | 6 | | | 1 | | | | |
| FICOM(P) | m | 6 | | | 1 | | | | |
| FTST | | 1 | | | | | | 1 | |
| FXAM | | 1 | | | | | | 2 | |
| FPREM | | 23 | | | | | | | |
| FPREM1 | | 33 | | | | | | | |
| FRNDINT | | 30 | | | | | | | |
| FSCALE | | 56 | | | | | | | |
| FXTRACT | | 15 | | | | | | | |
| FSQRT | | 1 | | | | | | 69 | e,i) |
| FSIN FCOS | | 17-97 | | | | | 27-103 | e) | |
| FSINCOS | | 18-110 | | | | | 29-130 | e) | |
| F2XM1 | | 17-48 | | | | | 66 | e) | |
| FYL2X | | 36-54 | | | | | 103 | e) | |
| FYL2XP1 | | 31-53 | | | | | 98-107 | e) | |
| FPTAN | | 21-102 | | | | | 13-143 | e) | |
| FPATAN | | 25-86 | | | | | 44-143 | e) | |
| FNOP | | 1 | | | | | | | |
| FINCSTP FDECSTP | | 1 | | | | | | | |
| FFREE | r | 1 | | | | | | | |
| FFREEP | r | 2 | | | | | | | |

Pentium II and III

| | | | | | | | |
|--------|-----|---|---|--|--|--|--|
| FNCLEX | | | 3 | | | | |
| FNINIT | 13 | | | | | | |
| FNSAVE | 141 | | | | | | |
| FRSTOR | 72 | | | | | | |
| WAIT | | 2 | | | | | |

Notes:

- e) Not pipelined
- f) FXCH generates 1 μ op that is resolved by register renaming without going to any port.
- g) FMUL uses the same circuitry as integer multiplication. Therefore, the combined throughput of mixed floating point and integer multiplications is 1 FMUL + 1 IMUL per 3 clock cycles.
- h) FDIV latency depends on precision specified in control word: 64 bits precision gives latency 38, 53 bits precision gives latency 32, 24 bits precision gives latency 18. Division by a power of 2 takes 9 clocks. Reciprocal throughput is 1/(latency-1).
- i) Faster for lower precision.

Integer MMX instructions (Pentium II and Pentium III)

| Instruction | Operands | μ ops | | | | | | Latency | Reciprocal throughput |
|------------------|-----------|-----------|----|-----|----|----|----|---------|-----------------------|
| | | p0 | p1 | p01 | p2 | p3 | p4 | | |
| MOVD MOVQ | r,r | | | 1 | | | | 1 | 0.5 |
| MOVD MOVQ | mm,m32/64 | | | | 1 | | | | 1 |
| MOVD MOVQ | m32/64,mm | | | | | 1 | 1 | | 1 |
| PADD PSUB PCMP | mm,mm | | | 1 | | | | 1 | 0.5 |
| PADD PSUB PCMP | mm,m64 | | | 1 | 1 | | | | 1 |
| PMUL PMADD | mm,mm | 1 | | | | | | 3 | 1 |
| PMUL PMADD | mm,m64 | 1 | | | 1 | | | 3 | 1 |
| PAND(N) POR PXOR | mm,mm | | | 1 | | | | 1 | 0.5 |
| PAND(N) POR PXOR | mm,m64 | | | 1 | 1 | | | | 1 |
| PSRA PSRL PSLL | mm,mm/i | | 1 | | | | | 1 | 1 |
| PSRA PSRL PSLL | mm,m64 | | 1 | | 1 | | | | 1 |
| PACK PUNPCK | mm,mm | | 1 | | | | | 1 | 1 |
| PACK PUNPCK | mm,m64 | | 1 | | 1 | | | | 1 |
| EMMS | | 11 | | | | | | 6 k) | |
| MASKMOVQ d) | mm,mm | | | 1 | | 1 | 1 | 2-8 | 2 - 30 |
| PMOVMASKB d) | r32,mm | | 1 | | | | | 1 | 1 |
| MOVNTQ d) | m64,mm | | | | | 1 | 1 | | 1 - 30 |
| PSHUFW d) | mm,mm,i | | 1 | | | | | 1 | 1 |
| PSHUFW d) | mm,m64,i | | 1 | | 1 | | | 2 | 1 |
| PEXTRW d) | r32,mm,i | | 1 | 1 | | | | 2 | 1 |
| PINSRW d) | mm,r32,i | | 1 | | | | | 1 | 1 |
| PINSRW d) | mm,m16,i | | 1 | | 1 | | | 2 | 1 |
| PAVGB PAVGW d) | mm,mm | | | 1 | | | | 1 | 0.5 |
| PAVGB PAVGW d) | mm,m64 | | | 1 | 1 | | | 2 | 1 |
| PMIN/MAXUB/SW d) | mm,mm | | | 1 | | | | 1 | 0.5 |
| PMIN/MAXUB/SW d) | mm,m64 | | | 1 | 1 | | | 2 | 1 |
| PMULHUW d) | mm,mm | 1 | | | | | | 3 | 1 |
| PMULHUW d) | mm,m64 | 1 | | | 1 | | | 4 | 1 |
| PSADBW d) | mm,mm | 2 | | 1 | | | | 5 | 2 |
| PSADBW d) | mm,m64 | 2 | | 1 | 1 | | | 6 | 2 |

Notes:

- d) P3 only.

Pentium II and III

- k) The delay can be hidden by inserting other instructions between EMMS and any subsequent floating point instruction.

Floating point XMM instructions (Pentium III)

| Instruction | Operands | μ ops | | | | | | Latency | Reciprocal throughput |
|---------------------|----------|-----------|----|-----|----|----|----|---------|-----------------------|
| | | p0 | p1 | p01 | p2 | p3 | p4 | | |
| MOVAPS | xmm,xmm | | | 2 | | | | 1 | 1 |
| MOVAPS | xmm,m128 | | | | 2 | | | 2 | 2 |
| MOVAPS | m128,xmm | | | | | 2 | 2 | 3 | 2 |
| MOVUPS | xmm,m128 | | | | 4 | | | 2 | 4 |
| MOVUPS | m128,xmm | | 1 | | | 4 | 4 | 3 | 4 |
| MOVSS | xmm,xmm | | | 1 | | | | 1 | 1 |
| MOVSS | xmm,m32 | | | 1 | 1 | | | 1 | 1 |
| MOVSS | m32,xmm | | | | | 1 | 1 | 1 | 1 |
| MOVHPS MOVLPSS | xmm,m64 | | | 1 | | | | 1 | 1 |
| MOVHPS MOVLPSS | m64,xmm | | | | | 1 | 1 | 1 | 1 |
| MOVLHPS MOVHLPS | xmm,xmm | | | 1 | | | | 1 | 1 |
| MOVMSKPS | r32,xmm | 1 | | | | | | 1 | 1 |
| MOVNTPS | m128,xmm | | | | | 2 | 2 | | 2 - 15 |
| CVTPI2PS | xmm,mm | | 2 | | | | | 3 | 1 |
| CVTPI2PS | xmm,m64 | | 2 | | 1 | | | 4 | 2 |
| CVT(T)PS2PI | mm,xmm | | 2 | | | | | 3 | 1 |
| CVTPS2PI | mm,m128 | | 1 | | 2 | | | 4 | 1 |
| CVTSI2SS | xmm,r32 | | 2 | | 1 | | | 4 | 2 |
| CVTSI2SS | xmm,m32 | | 2 | | 2 | | | 5 | 2 |
| CVT(T)SS2SI | r32,xmm | | 1 | | 1 | | | 3 | 1 |
| CVTSS2SI | r32,m128 | | 1 | | 2 | | | 4 | 2 |
| ADDPS SUBPS | xmm,xmm | | 2 | | | | | 3 | 2 |
| ADDPS SUBPS | xmm,m128 | | 2 | | 2 | | | 3 | 2 |
| ADDSS SUBSS | xmm,xmm | | 1 | | | | | 3 | 1 |
| ADDSS SUBSS | xmm,m32 | | 1 | | 1 | | | 3 | 1 |
| MULPS | xmm,xmm | 2 | | | | | | 4 | 2 |
| MULPS | xmm,m128 | 2 | | | 2 | | | 4 | 2 |
| MULSS | xmm,xmm | 1 | | | | | | 4 | 1 |
| MULSS | xmm,m32 | 1 | | | 1 | | | 4 | 1 |
| DIVPS | xmm,xmm | 2 | | | | | | 48 | 34 |
| DIVPS | xmm,m128 | 2 | | | 2 | | | 48 | 34 |
| DIVSS | xmm,xmm | 1 | | | | | | 18 | 17 |
| DIVSS | xmm,m32 | 1 | | | 1 | | | 18 | 17 |
| AND(N)PS ORPS XORPS | xmm,xmm | | 2 | | | | | 2 | 2 |
| AND(N)PS ORPS XORPS | xmm,m128 | | 2 | | 2 | | | 2 | 2 |
| MAXPS MINPS | xmm,xmm | | 2 | | | | | 3 | 2 |
| MAXPS MINPS | xmm,m128 | | 2 | | 2 | | | 3 | 2 |
| MAXSS MINSS | xmm,xmm | | 1 | | | | | 3 | 1 |
| MAXSS MINSS | xmm,m32 | | 1 | | 1 | | | 3 | 1 |
| CMPccPS | xmm,xmm | | 2 | | | | | 3 | 2 |
| CMPccPS | xmm,m128 | | 2 | | 2 | | | 3 | 2 |
| CMPccSS | xmm,xmm | | 1 | | | | | 3 | 1 |
| CMPccSS | xmm,m32 | | 1 | | 1 | | | 3 | 1 |
| COMISS UCOMISS | xmm,xmm | | 1 | | | | | 1 | 1 |
| COMISS UCOMISS | xmm,m32 | | 1 | | 1 | | | 1 | 1 |

Pentium II and III

| | | | | | | | | |
|-------------------|------------|-----|---|---|---|--|----|----|
| SQRTPS | xmm,xmm | 2 | | | | | 56 | 56 |
| SQRTPS | xmm,m128 | 2 | | | 2 | | 57 | 56 |
| SQRTSS | xmm,xmm | 2 | | | | | 30 | 28 |
| SQRTSS | xmm,m32 | 2 | | | 1 | | 31 | 28 |
| RSQRTPS | xmm,xmm | 2 | | | | | 2 | 2 |
| RSQRTPS | xmm,m128 | 2 | | | 2 | | 3 | 2 |
| RSQRTSS | xmm,xmm | 1 | | | | | 1 | 1 |
| RSQRTSS | xmm,m32 | 1 | | | 1 | | 2 | 1 |
| RCPPS | xmm,xmm | 2 | | | | | 2 | 2 |
| RCPPS | xmm,m128 | 2 | | | 2 | | 3 | 2 |
| RCPSS | xmm,xmm | 1 | | | | | 1 | 1 |
| RCPSS | xmm,m32 | 1 | | | 1 | | 2 | 1 |
| SHUFPS | xmm,xmm,i | | 2 | 1 | | | 2 | 2 |
| SHUFPS | xmm,m128,i | | 2 | | 2 | | 2 | 2 |
| UNPCKHPS UNPCKLPS | xmm,xmm | | 2 | 2 | | | 3 | 2 |
| UNPCKHPS UNPCKLPS | xmm,m128 | | 2 | | 2 | | 3 | 2 |
| LDMXCSR | m32 | 11 | | | | | 15 | 15 |
| STMXCSR | m32 | 6 | | | | | 7 | 9 |
| FXSAVE | m4096 | 116 | | | | | 62 | |
| FXRSTOR | m4096 | 89 | | | | | 68 | |

Intel Pentium M, Core Solo and Core Duo

List of instruction timings and μ op breakdown

Explanation of column headings:

| | |
|--|---|
| Operands: | i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc. |
| μops fused domain: | The number of μ ops at the decode, rename, allocate and retirement stages in the pipeline. Fused μ ops count as one. |
| μops unfused domain: | The number of μ ops for each execution port. Fused μ ops count as two. |
| p0: | Port 0: ALU, etc. |
| p1: | Port 1: ALU, jumps |
| p01: | Instructions that can go to either port 0 or 1, whichever is vacant first. |
| p2: | Port 2: load data, etc. |
| p3: | Port 3: address generation for store |
| p4: | Port 4: store data |
| Latency: | This is the delay that the instruction generates in a dependency chain. (This is not the same as the time spent in the execution unit. Values may be inaccurate in situations where they cannot be measured exactly, especially with memory operands). The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays by 50-150 clocks, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay. |
| Reciprocal throughput: | The average number of clock cycles per instruction for a series of independent instructions of the same kind. |

Integer instructions

| Instruction | Operands | μ ops fused domain | μ ops unfused domain | | | | | | Latency | Reciprocal throughput |
|-------------------|----------|------------------------|--------------------------|----|-----|----|----|----|---------|-----------------------|
| | | | p0 | p1 | p01 | p2 | p3 | p4 | | |
| Move instructions | | | | | | | | | | |
| MOV | r,r/i | 1 | | | 1 | | | | | 0.5 |
| MOV | r,m | 1 | | | | 1 | | | | 1 |
| MOV | m,r | 1 | | | | | 1 | 1 | | 1 |
| MOV | m,i | 2 | | | | | 1 | 1 | | 1 |
| MOV | r,sr | 1 | | | 1 | | | | | |
| MOV | m,sr | 2 | | | 1 | | 1 | 1 | | |
| MOV | sr,r | 8 | 8 | | | | | | 5 | |
| MOV | sr,m | 8 | 7 | | | 1 | | | 8 | |
| MOVNTI | m,r32 | 2 | | | | | 1 | 1 | | 2 |
| MOVSX MOVZX | r,r | 1 | | | 1 | | | | 1 | 0.5 |
| MOVSX MOVZX | r,m | 1 | | | | 1 | | | | 1 |
| CMOVcc | r,r | 2 | 1 | | 1 | | | | 2 | 1.5 |
| CMOVcc | r,m | 2 | 1 | | 1 | 1 | | | | |
| XCHG | r,r | 3 | | | 3 | | | | 2 | 1.5 |
| XCHG | r,m | 7 | | | 4 | 1 | 1 | 1 | high b) | |
| XLAT | | 2 | | | 1 | 1 | | | | 1 |

Pentium M

| | | | | | | | | | | |
|--------------------------------|---------|----|----|---|----|---|---|---|----------|----------|
| PUSH | r | 1 | | | | | 1 | 1 | 1 | 1 |
| PUSH | i | 2 | | | | | 1 | 1 | 1 | 1 |
| PUSH | m | 2 | | | | 1 | 1 | 1 | 2 | 1 |
| PUSH | sr | 2 | | | 1 | | 1 | 1 | | |
| PUSHF(D) | | 16 | 3 | | 11 | | 1 | 1 | | 6 |
| PUSHA(D) | | 18 | | | 2 | | 8 | 8 | 8 | 8 |
| POP | r | 1 | | | | 1 | | | | |
| POP | (E)SP | 3 | | | 2 | 1 | | | | |
| POP | m | 2 | | | | 1 | 1 | 1 | 2 | 1 |
| POP | sr | 10 | | | 9 | 1 | | | | |
| POPF(D) | | 17 | 10 | | 6 | 1 | | | | 16 |
| POPA(D) | | 10 | | | 2 | 8 | | | 7 | 7 |
| LAHF SAHF | | 1 | | | 1 | | | | 1 | 1 |
| SALC | | 2 | 1 | 1 | | | | | | 1 |
| LEA | r,m | 1 | 1 | | | | | | 1 | 1 |
| BSWAP | r | 2 | 1 | | 1 | | | | | |
| LDS LES LFS LGS LSS | m | 11 | | | 8 | 3 | | | | |
| PREFETCHNTA | m | 1 | | | | 1 | | | | 1 |
| PREFETCH0/1/2 | m | 1 | | | | 1 | | | | 1 |
| SFENCE/LFENCE/MFENCE | | 2 | | | | | 1 | 1 | | 6 |
| IN | | | 18 | | | | | | >300 | |
| OUT | | | 18 | | | | | | >300 | |
| Arithmetic instructions | | | | | | | | | | |
| ADD SUB | r,r/i | 1 | | | 1 | | | | 1 | 0.5 |
| ADD SUB | r,m | 1 | | | 1 | 1 | | | 2 | 1 |
| ADD SUB | m,r/i | 3 | | | 1 | 1 | 1 | 1 | | 1 |
| ADC SBB | r,r/i | 2 | | 1 | 1 | | | | 2 | 2 |
| ADC SBB | r,m | 2 | | 1 | 1 | 1 | | | | |
| ADC SBB | m,r/i | 7 | | | 4 | 1 | 1 | 1 | | |
| CMP | r,r/i | 1 | | | 1 | | | | 1 | 0.5 |
| CMP | m,r | 1 | | | 1 | 1 | | | 1 | 1 |
| CMP | m,i | 2 | | | 1 | 1 | | | | 1 |
| INC DEC NEG NOT | r | 1 | | | 1 | | | | 1 | 0.5 |
| INC DEC NEG NOT | m | 3 | | | 1 | 1 | 1 | 1 | | |
| AAA AAS DAA DAS | | 1 | | 1 | | | | | | |
| AAD | | 3 | 1 | | 2 | | | | 2 | |
| AAM | | 4 | 1 | 1 | 2 | | | | 15 | |
| MUL IMUL | r8 | 1 | 1 | | | | | | 4 | 1 |
| MUL IMUL | r16/r32 | 3 | 3 | | | | | | 5 | 1 |
| IMUL | r,r | 1 | 1 | | | | | | 4 | 1 |
| IMUL | r,r,i | 1 | 1 | | | | | | 4 | 1 |
| MUL IMUL | m8 | 1 | 1 | | | 1 | | | 4 | 1 |
| MUL IMUL | m16/m32 | 3 | 3 | | | 1 | | | 5 | 1 |
| IMUL | r,m | 1 | 1 | | | 1 | | | 4 | 1 |
| IMUL | r,m,i | 2 | 1 | | | 1 | | | 4 | 1 |
| DIV IDIV | r8 | 5 | 4 | | 1 | | | | 15-16 c) | 12 |
| DIV IDIV | r16 | 4 | 3 | | 1 | | | | 15-24 c) | 12-20 c) |
| DIV IDIV | r32 | 4 | 3 | | 1 | | | | 15-39 c) | 12-20 c) |
| DIV IDIV | m8 | 6 | 4 | | 1 | 1 | | | 15-16 c) | 12 |
| DIV IDIV | m16 | 5 | 3 | | 1 | 1 | | | 15-24 c) | 12-20 c) |
| DIV IDIV | m32 | 5 | 3 | | 1 | 1 | | | 15-39 c) | 12-20 c) |
| CBW CWDE | | 1 | | 1 | | | | | 1 | 1 |

Pentium M

| | | | | | | | | | | |
|--------------------------------------|-------------|----|----|---|---|---|---|---|----|-----|
| CWD CDQ | | 1 | | 1 | | | | | 1 | 1 |
| Logic instructions | | | | | | | | | | |
| AND OR XOR | r,r/i | 1 | | | 1 | | | | 1 | 0.5 |
| AND OR XOR | r,m | 1 | | | 1 | 1 | | | 2 | 1 |
| AND OR XOR | m,r/i | 3 | | | 1 | 1 | 1 | 1 | | 1 |
| TEST | r,r/i | 1 | | | 1 | | | | 1 | 0.5 |
| TEST | m,r | 1 | | | 1 | 1 | | | 1 | 1 |
| TEST | m,i | 2 | | | 1 | 1 | | | | 1 |
| SHR SHL SAR ROR ROL | r,i/CL | 1 | 1 | | | | | | 1 | 1 |
| SHR SHL SAR ROR ROL | m,i/CL | 3 | 1 | | | 1 | 1 | 1 | | |
| RCR RCL | r,1 | 2 | 1 | | 1 | | | | 2 | 2 |
| RCR | r8,i/CL | 9 | 5 | | 4 | | | | 11 | |
| RCL | r8,i/CL | 8 | 4 | | 4 | | | | 10 | |
| RCR RCL | r16/32,i/CL | 6 | 3 | | 3 | | | | 9 | 9 |
| RCR RCL | m,1 | 7 | 2 | | 2 | 1 | 1 | 1 | | |
| RCR | m8,i/CL | 12 | 6 | | 3 | 1 | 1 | 1 | | |
| RCL | m8,i/CL | 11 | 5 | | 3 | 1 | 1 | 1 | | |
| RCR RCL | m16/32,i/CL | 10 | 5 | | 2 | 1 | 1 | 1 | | |
| SHLD SHRD | r,r,i/CL | 2 | 2 | | | | | | 2 | 2 |
| SHLD SHRD | m,r,i/CL | 4 | 1 | | 1 | 1 | 1 | 1 | | |
| BT | r,r/i | 1 | | 1 | | | | | 1 | 1 |
| BT | m,r | 8 | | | 7 | 1 | | | | |
| BT | m,i | 2 | | 1 | | 1 | | | | |
| BTR BTS BTC | r,r/i | 1 | | 1 | | | | | | |
| BTR BTS BTC | m,r | 10 | | | 7 | 1 | 1 | 1 | 6 | |
| BTR BTS BTC | m,i | 3 | | 1 | | 1 | 1 | 1 | | |
| BSF BSR | r,r | 2 | | 1 | 1 | | | | | |
| BSF BSR | r,m | 2 | | 1 | 1 | 1 | | | | |
| SETcc | r | 1 | | 1 | | | | | | |
| SETcc | m | 2 | | 1 | | | 1 | 1 | | |
| CLC STC CMC | | 1 | | 1 | | | | | | 1 |
| CLD STD | | 4 | | | 4 | | | | | 7 |
| Control transfer instructions | | | | | | | | | | |
| JMP | short/near | 1 | | 1 | | | | | | 1 |
| JMP | far | 22 | 21 | | | 1 | | | | 28 |
| JMP | r | 1 | | 1 | | | | | | 1 |
| JMP | m(near) | 2 | | 1 | | 1 | | | | 2 |
| JMP | m(far) | 25 | 23 | | | 2 | | | | 31 |
| conditional jump | short/near | 1 | | 1 | | | | | | 1 |
| J(E)CXZ | short | 2 | | 1 | 1 | | | | | 1 |
| LOOP | short | 11 | 2 | 1 | 8 | | | | | 6 |
| LOOP(N)E | short | 11 | 2 | 1 | 8 | | | | | 6 |
| CALL | near | 4 | | 1 | 1 | | 1 | 1 | | 2 |
| CALL | far | 32 | 27 | | | 1 | 2 | 2 | | 27 |
| CALL | r | 4 | | 1 | 2 | | 1 | 1 | | 9 |
| CALL | m(near) | 4 | | 1 | | 1 | 1 | 1 | | 2 |
| CALL | m(far) | 35 | 29 | | | 2 | 2 | 2 | | 30 |
| RETN | | 2 | | 1 | 2 | 1 | | | | 2 |
| RETN | i | 3 | | 1 | 1 | 1 | | | | 2 |
| RETF | | 27 | 24 | | | 3 | | | | 30 |
| RETF | i | 27 | 24 | | | 3 | | | | 30 |

Pentium M

| | | | | | | | | | | |
|----------------------------|-----|-------|-------|----|--------|----|-----|----|---------|-----|
| BOUND | r,m | 15 | 7 | | 6 | 2 | | | | 8 |
| INTO | | 5 | | | 5 | | | | | 4 |
| String instructions | | | | | | | | | | |
| LODS | | 2 | | | | 2 | | | | 4 |
| REP LODS | | 6n | | | 10+6n | | | | | 0.5 |
| STOS | | 3 | | | | 1 | 1 | 1 | | 1 |
| REP STOS | | 5n | | | ca. 5r | a) | | | | 0.7 |
| MOVS | | 6 | | | 1 | 3 | 1 | 1 | | 0.7 |
| REP MOVS | | 6n | | | ca. 6r | a) | | | | 0.5 |
| SCAS | | 3 | | | 1 | 2 | | | | 1.3 |
| REP(N)E SCAS | | 7n | | | 12+7n | | | | | 0.6 |
| CMPS | | 6 | | | 4 | 2 | | | | 0.7 |
| REP(N)E CMPS | | 9n | | | 12+9n | | | | | 0.5 |
| Other | | | | | | | | | | |
| NOP (90) | | 1 | | | 1 | | | | | 0.5 |
| Long NOP (0F 1F) | | 1 | | | 1 | | | | | 1 |
| PAUSE | | 2 | | | 2 | | | | | |
| CLI | | | 9 | | | | | | | |
| STI | | | 17 | | | | | | | |
| ENTER | i,0 | 12 | | | 10 | | 1 | 1 | | |
| ENTER | a,b | | ca. | 18 | +4b | | b-1 | 2b | | |
| LEAVE | | 3 | | | 2 | 1 | | | | |
| CPUID | | 38-59 | 38-59 | | | | | | ca. 130 | |
| RDTSC | | 13 | 13 | | | | | | | 42 |

Notes:

- a) Faster under certain conditions: see manual 3: "The microarchitecture of Intel, AMD and VIA CPUs".
- b) Has an implicit LOCK prefix.
- c) High values are typical, low values are for round divisors. Core Solo/Duo is more efficient than Pentium M in cases with round values that allow an early-out algorithm.

Floating point x87 instructions

| Instruction | Operands | µops fused domain | µops unfused domain | | | | | | Latency | Reciprocal throughput |
|-------------------|----------|-------------------|---------------------|----|-----|----|----|----|---------|-----------------------|
| | | | p0 | p1 | p01 | p2 | p3 | p4 | | |
| Move instructions | | | | | | | | | | |
| FLD | r | 1 | 1 | | | | | | 1 | |
| FLD | m32/64 | 1 | | | | 1 | | | 1 | |
| FLD | m80 | 4 | 2 | | | 2 | | | | |
| FBLD | m80 | 40 | 38 | | | 2 | | | | |
| FST(P) | r | 1 | 1 | | | | | | | |
| FST(P) | m32/m64 | 1 | | | | | 1 | 1 | 1 | |
| FSTP | m80 | 6 | 2 | | | | 2 | 2 | | 3 |
| FBSTP | m80 | 169 | 165 | | | | 2 | 2 | | 167 |
| FXCH | r | 1 | | | | | | | 0 | 0.33 f) |
| FILD | m | 4 | 3 | | | 1 | | | 5 | 2 |
| FIST(P) | m | 4 | 2 | | | | 1 | 1 | 5 | 2 |
| FISTTP g) | m | 4 | 2 | | | | 1 | 1 | 5 | 2 |

Pentium M

| | | | | | | | | | | |
|--------------------------------|-----|--------|--------|----|---|---|---|---------|---------|--|
| FLDZ | | 1 | 1 | | | | | | | |
| FLD1 FLDPI FLDL2E etc. | | 2 | 2 | | | | | | | |
| FCMOVcc | r | 2 | 2 | | | | | 2 | | |
| FNSTSW | AX | 3 | 3 | | | | | 7 | 3 | |
| FNSTSW | m16 | 2 | 1 | | | 1 | 1 | | | |
| FLDCW | m16 | 3 | 1 | 1 | 1 | | | | 19 | |
| FNSTCW | m16 | 3 | 1 | | | 1 | 1 | | 3 | |
| FINCSTP FDECSTP | | 1 | 1 | | | | | 1 | | |
| FFREE | r | 1 | 1 | | | | | | 1 | |
| FFREEP | r | 2 | 2 | | | | | | 2 | |
| FNSAVE | | 142 | 142 | | | | | | 131 | |
| FRSTOR | | 72 | 72 | | | | | | 91 | |
| Arithmetic instructions | | | | | | | | | | |
| FADD(P) FSUB(R)(P) | r | 1 | | 1 | | | | 3 | 1 | |
| FADD(P) FSUB(R)(P) | m | 1 | | 1 | 1 | | | 3 | 1 | |
| FMUL(P) | r | 1 | 1 | | | | | 5 | 2 | |
| FMUL(P) | m | 1 | 1 | | 1 | | | 5 | 2 | |
| FDIV(R)(P) | r | 1 | 1 | | | | | 9-38 c) | 8-37 c) | |
| FDIV(R)(P) | m | 1 | 1 | | 1 | | | 9-38 c) | 8-37 c) | |
| FABS | | 1 | 1 | | | | | 1 | 1 | |
| FCHS | | 1 | 1 | | | | | 1 | 1 | |
| FCOM(P) FUCOM | r | 1 | | 1 | | | | 1 | 1 | |
| FCOM(P) FUCOM | m | 1 | | 1 | 1 | | | 1 | 1 | |
| FCOMPP FUCOMPP | | 2 | | 1 | 1 | | | 1 | 1 | |
| FCOMI(P) FUCOMI(P) | r | 1 | | 1 | | | | 1 | 1 | |
| FIADD FISUB(R) | m | 6 | 3 | 1 | 1 | 1 | | 3 | 3 | |
| FIMUL | m | 6 | 5 | | | 1 | | 5 | 3 | |
| FIDIV(R) | m | 6 | 5 | | | 1 | | 9-38 c) | 8-37 c) | |
| FICOM(P) | m | 6 | 3 | 2 | | 1 | | | 4 | |
| FTST | | 1 | | 1 | | | | | 1 | |
| FXAM | | 1 | | 1 | | | | | 1 | |
| FPREM FPREM1 | | 26 | 26 | | | | | 37 | | |
| FRNDINT | | 15 | 15 | | | | | 19 | | |
| Math | | | | | | | | | | |
| FSCALE | | 28 | 28 | | | | | 43 | | |
| FXTRACT | | 15 | | 15 | | | | 9 | | |
| FSQRT | | 1 | 1 | | | | | 9 h) | 8 | |
| FSIN FCOS | | 80-100 | 80-100 | | | | | 80-110 | | |
| FSINCOS | | 90-110 | 90-110 | | | | | 100-130 | | |
| F2XM1 | | ~ 20 | ~20 | | | | | ~45 | | |
| FYL2X | | ~ 40 | ~40 | | | | | ~60 | | |
| FYL2XP1 | | ~ 55 | ~55 | | | | | ~65 | | |
| FPTAN | | ~ 100 | ~100 | | | | | ~140 | | |
| FPATAN | | ~ 85 | ~85 | | | | | ~140 | | |
| Other | | | | | | | | | | |
| FNOP | | 1 | 1 | | | | | | 1 | |
| WAIT | | 2 | | 1 | 1 | | | | 1 | |
| FNCLEX | | 3 | 3 | | | | | | 13 | |
| FNINIT | | 14 | 14 | | | | | | 27 | |

Notes:

Pentium M

- c) High values are typical, low values are for low precision or round divisors.
- f) FXCH generates 1 μ op that is resolved by register renaming without going to any port.
- g) SSE3 instruction only available on Core Solo and Core Duo.

Integer MMX and XMM instructions

| Instruction | Operands | μops fused domain | μops unfused domain | | | | | | Latency | Reciprocal throughput |
|-------------------|-----------|-------------------|---------------------|----|-----|----|-----|-----|---------|-----------------------|
| | | | p0 | p1 | p01 | p2 | p3 | p4 | | |
| Move instructions | | | | | | | | | | |
| MOVD | r32,mm | 1 | | | 1 | | | | 1 | 0.5 |
| MOVD | mm,r32 | 1 | | | 1 | | | | 1 | 0.5 |
| MOVD | mm,m32 | 1 | | | | 1 | | | | 1 |
| MOVD | m32,mm | 1 | | | | | 1 | 1 | | 1 |
| MOVD | r32,xmm | 1 | | 1 | | | | | 1 | 1 |
| MOVD | xmm,r32 | 2 | | | 2 | | | | | 1 |
| MOVD | xmm,m32 | 2 | | | 1 | 1 | | | | 1 |
| MOVD | m32, xmm | 1 | | | | | 1 | 1 | | 1 |
| MOVQ | mm,mm | 1 | | | 1 | | | | | 0.5 |
| MOVQ | mm,m64 | 1 | | | | 1 | | | | 1 |
| MOVQ | m64,mm | 1 | | | | | 1 | 1 | | 1 |
| MOVQ | xmm,xmm | 2 | | | 2 | | | | 1 | 1 |
| MOVQ | xmm,m64 | 2 | | | 1 | 1 | | | | 1 |
| MOVQ | m64, xmm | 1 | | | | | 1 | 1 | | 1 |
| MOVDQA | xmm, xmm | 2 | | | 2 | | | | 1 | 1 |
| MOVDQA | xmm, m128 | 2 | | | | 2 | | | | 2 |
| MOVDQA | m128, xmm | 2 | | | | | 2 | 2 | | 2 |
| MOVDQU | xmm, m128 | 4 | | | 2 | 2 | | | | 2-10 |
| MOVDQU | m128, xmm | 8 | | | 5-6 | | 2-3 | 2-3 | | 4-20 |
| LDDQU g) | xmm, m128 | 4 | | | | | | | | 2 |
| MOVDQ2Q | mm, xmm | 1 | | 1 | | | | | 1 | 1 |
| MOVQ2DQ | xmm,mm | 2 | | 1 | 1 | | | | 1 | 1 |
| MOVNTQ | m64,mm | 1 | | | | | 1 | 1 | | 2 |
| MOVNTDQ | m128,xmm | 4 | | | | | 2 | 2 | | 3 |
| PACKSSWB/DW | | | | | | | | | | |
| PACKUSWB | mm,mm | 1 | 1 | | | | | | 1 | 1 |
| PACKSSWB/DW | | | | | | | | | | |
| PACKUSWB | mm,m64 | 1 | 1 | | | 1 | | | 1 | 1 |
| PACKSSWB/DW | | | | | | | | | | |
| PACKUSWB | xmm,xmm | 3 | 2 | 1 | | | | | 2 | 2 |
| PACKSSWB/DW | | | | | | | | | | |
| PACKUSWB | xmm,m128 | 4 | 1 | 1 | | 2 | | | 2 | 2 |
| PUNPCKH/LBW/WD/DQ | mm,mm | 1 | 1 | | | | | | 1 | 1 |
| PUNPCKH/LBW/WD/DQ | mm,m64 | 1 | 1 | | | 1 | | | | 1 |
| PUNPCKH/LBW/WD/DQ | xmm,xmm | 2 | 2 | | | | | | 2 | 2 |
| PUNPCKH/LBW/WD/DQ | xmm,m128 | 3 | 1 | | | 2 | | | | 2 |
| PUNPCKHQDQ | xmm,xmm | 2 | | 1 | 1 | | | | 1 | 1 |
| PUNPCKHQDQ | xmm, m128 | 3 | | 1 | | 2 | | | | 1 |
| PUNPCKLQDQ | xmm,xmm | 1 | | 1 | | | | | 1 | 1 |
| PUNPCKLQDQ | xmm, m128 | 1 | | | | 1 | | | | 1 |
| PSHUFW | mm,mm,i | 1 | 1 | | | | | | 1 | 1 |

Pentium M

| | | | | | | | | | | |
|--------------------------------|-------------|---|---|----|---|---|---|---|---|-----|
| PSHUFW | mm,m64,i | 2 | 1 | | | 1 | | | | 1 |
| PSHUFD | xmm,xmm,i | 3 | 2 | 1 | | | | | 2 | 2 |
| PSHUFD | xmm,m128,i | 4 | 1 | 1 | | 2 | | | | 2 |
| PSHUFL/HW | xmm,xmm,i | 2 | 1 | 1 | | | | | | 1 |
| PSHUFL/HW | xmm, m128,i | 3 | | 1 | | 2 | | | | 1 |
| MASKMOVQ | mm,mm | 3 | | | 1 | | 1 | 1 | | |
| MASKMOVDQU | xmm,xmm | 8 | | 1 | | | 2 | 2 | | |
| PMOVMSKB | r32,mm | 1 | 1 | | | | | | 1 | 1 |
| PMOVMSKB | r32,xmm | 1 | 1 | j) | | | | | 1 | 1 |
| PEXTRW | r32,mm,i | 2 | 1 | 1 | | | | | 2 | 1 |
| PEXTRW | r32,xmm,i | 4 | 2 | 2 | | | | | 3 | 2 |
| PINSRW | mm,r32,i | 1 | 1 | | | | | | 1 | 1 |
| PINSRW | xmm,r32,i | 2 | 2 | | | | | | 1 | 2 |
| Arithmetic instructions | | | | | | | | | | |
| PADD/SUB(U)(S)B/W/D | mm,mm | 1 | | | | 1 | | | 1 | 0.5 |
| PADD/SUB(U)(S)B/W/D | mm,m64 | 1 | | | | 1 | 1 | | | 1 |
| PADD/SUB(U)(S)B/W/D | xmm,xmm | 2 | | | | 2 | | | 1 | 1 |
| PADD/SUB(U)(S)B/W/D | xmm,m128 | 4 | | | | 2 | 2 | | | 2 |
| PADDQ PSUBQ | mm,mm | 2 | | | | 2 | | | 2 | 1 |
| PADDQ PSUBQ | mm,m64 | 2 | | | | 2 | 1 | | | 1 |
| PADDQ PSUBQ | xmm,xmm | 4 | | | | 4 | | | 2 | 2 |
| PADDQ PSUBQ | xmm,m128 | 6 | | | | 4 | 2 | | | 2 |
| PCMPEQ/GTB/W/D | mm,mm | 1 | | | | 1 | | | 1 | 0.5 |
| PCMPEQ/GTB/W/D | mm,m64 | 1 | | | | 1 | 1 | | | 1 |
| PCMPEQ/GTB/W/D | xmm,xmm | 2 | | | | 2 | | | 1 | 1 |
| PCMPEQ/GTB/W/D | xmm,m128 | 2 | | | | 2 | 2 | | | 2 |
| PMULL/HW PMULHUW | mm,mm | 1 | | | | 1 | | | 3 | 1 |
| PMULL/HW PMULHUW | mm,m64 | 1 | | | | 1 | 1 | | 3 | 1 |
| PMULL/HW PMULHUW | xmm,xmm | 2 | | | | 2 | | | 3 | 2 |
| PMULL/HW PMULHUW | xmm,m128 | 4 | | | | 2 | 2 | | 3 | 2 |
| PMULUDQ | mm,mm | 1 | 1 | | | | | | 4 | 1 |
| PMULUDQ | mm,m64 | 1 | 1 | | | | 1 | | 4 | 1 |
| PMULUDQ | xmm,xmm | 2 | 2 | | | | | | 4 | 2 |
| PMULUDQ | xmm,m128 | 4 | 2 | | | | 2 | | 4 | 2 |
| PMADDWD | mm,mm | 1 | | | | 1 | | | 3 | 1 |
| PMADDWD | mm,m64 | 1 | | | | 1 | 1 | | 3 | 1 |
| PMADDWD | xmm,xmm | 2 | | | | 2 | | | 3 | 2 |
| PMADDWD | xmm,m128 | 4 | | | | 2 | 2 | | 3 | 2 |
| PAVGB/W | mm,mm | 1 | | | | 1 | | | 1 | 0.5 |
| PAVGB/W | mm,m64 | 1 | | | | 1 | 1 | | | 1 |
| PAVGB/W | xmm,xmm | 2 | | | | 2 | | | 1 | 1 |
| PAVGB/W | xmm,m128 | 4 | | | | 2 | 2 | | | 2 |
| PMIN/MAXUB/SW | mm,mm | 1 | | | | 1 | | | 1 | 0.5 |
| PMIN/MAXUB/SW | mm,m64 | 1 | | | | 1 | 1 | | | 1 |
| PMIN/MAXUB/SW | xmm,xmm | 2 | | | | 2 | | | 1 | 1 |
| PMIN/MAXUB/SW | xmm,m128 | 4 | | | | 2 | 2 | | | 2 |
| PSADBW | mm,mm | 2 | | | | 2 | | | 4 | 1 |
| PSADBW | mm,m64 | 2 | | | | 2 | 1 | | 4 | 1 |
| PSADBW | xmm,xmm | 4 | | | | 4 | | | 4 | 2 |
| PSADBW | xmm,m128 | 6 | | | | 4 | 2 | | 4 | 2 |
| Logic instructions | | | | | | | | | | |

Pentium M

| | | | | | | | | | |
|------------------|----------|----|---|---|----|---|--|------|-----|
| PAND(N) POR PXOR | mm,mm | 1 | | | 1 | | | 1 | 0.5 |
| PAND(N) POR PXOR | mm,m64 | 1 | | | 1 | 1 | | 1 | 1 |
| PAND(N) POR PXOR | xmm,xmm | 2 | | | 2 | | | 1 | 1 |
| PAND(N) POR PXOR | xmm,m128 | 4 | | | 2 | 2 | | | 2 |
| PSLL/RL/RAW/D/Q | mm,mm/i | 1 | 1 | | | | | 1 | 1 |
| PSLL/RL/RAW/D/Q | mm,m64 | 1 | 1 | | | 1 | | | 1 |
| PSLL/RL/RAW/D/Q | xmm,i | 2 | 2 | | | | | 2 | 2 |
| PSLL/RL/RAW/D/Q | xmm,xmm | 3 | 2 | 1 | | | | 2 | 2 |
| PSLL/RL/RAW/D/Q | xmm,m128 | 3 | | 1 | | 2 | | | 2 |
| PSLL/RLDQ | xmm,i | 4 | 3 | 1 | | | | 3 | 3 |
| Other | | | | | | | | | |
| EMMS | | 11 | | | 11 | | | 6 k) | 6 |

Notes:

- g) SSE3 instruction only available on Core Solo and Core Duo.
- j) Also uses some execution units under port 1.
- k) You may hide the delay by inserting other instructions between EMMS and any subsequent floating point instruction.

Floating point XMM instructions

| Instruction | Operands | μops fused domain | μops unfused domain | | | | | | Latency | Reciprocal throughput |
|-------------------|------------|-------------------|---------------------|----|-----|----|----|----|---------|-----------------------|
| | | | p0 | p1 | p01 | p2 | p3 | p4 | | |
| Move instructions | | | | | | | | | | |
| MOVAPS/D | xmm,xmm | 2 | | | 2 | | | | 1 | 1 |
| MOVAPS/D | xmm,m128 | 2 | | | | 2 | | | 2 | 2 |
| MOVAPS/D | m128,xmm | 2 | | | | | 2 | 2 | 3 | 2 |
| MOVUPS/D | xmm,m128 | 4 | | | | 4 | | | 2 | 2 |
| MOVUPS/D | m128,xmm | 8 | | | 4 | | 2 | 2 | 3 | 4 |
| MOVSS/D | xmm,xmm | 1 | | 1 | | | | | 1 | 1 |
| MOVSS/D | xmm,m32/64 | 2 | | 1 | | 1 | | | 1 | 1 |
| MOVSS/D | m32/64,xmm | 1 | | | | | 1 | 1 | 1 | 1 |
| MOVHPS/D MOVLPS/D | xmm,m64 | 1 | | 1 | | 1 | | | 1 | 1 |
| MOVHPS/D MOVLPS/D | m64,xmm | 1 | | | | | 1 | 1 | 1 | 1 |
| MOVLHPS MOVHLPS | xmm,xmm | 1 | | 1 | | | | | 1 | 1 |
| MOVMSKPS/D | r32,xmm | 1 | 1 | j) | | | | | 2 | 1 |
| MOVNTPS/D | m128,xmm | 2 | | | | | 2 | 2 | | 3 |
| SHUFPS/D | xmm,xmm,i | 3 | 2 | 1 | | | | | 2 | 2 |
| SHUFPS/D | xmm,m128,i | 4 | 1 | 1 | | 2 | | | | 2 |
| MOVDDUP g) | xmm,xmm | 2 | | | | | | | 1 | 1 |
| MOVSH/LDUP g) | xmm,xmm | 2 | | | | | | | 2 | 2 |
| MOVSH/LDUP g) | xmm,m128 | 4 | | | | | | | | |
| UNPCKH/LPS | xmm,xmm | 4 | 2 | 2 | | | | | 3-4 | 5 |
| UNPCKH/LPS | xmm,m128 | 4 | | 2 | | 2 | | | | 5 |
| UNPCKH/LPD | xmm,xmm | 2 | | 1 | 1 | | | | 1 | 1 |
| UNPCKH/LPD | xmm,m128 | 3 | | 1 | 1 | 1 | | | | 1 |
| Conversion | | | | | | | | | | |
| CVTPS2PD | xmm,xmm | 4 | 2 | 2 | | | | | 3 | 3 |
| CVTPS2PD | xmm,m64 | 4 | 1 | 2 | | 1 | | | | 3 |
| CVTPD2PS | xmm,xmm | 4 | 3 | 1 | | | | | 4 | 3 |

Pentium M

| | | | | | | | | |
|-------------------|------------|----|---|---|---|---|----------|----------|
| CVTPD2PS | xmm,m128 | 6 | 3 | 1 | 2 | | | 3 |
| CVTSD2SS | xmm,xmm | 2 | | | 2 | | 4 | 2 |
| CVTSD2SS | xmm,m64 | 3 | | | 2 | 1 | | 2 |
| CVTSS2SD | xmm,xmm | 2 | 2 | | | | 2 | 2 |
| CVTSS2SD | xmm,m64 | 3 | 2 | | | 1 | | 2 |
| CVTDQ2PS | xmm,xmm | 2 | | | 2 | | 3 | 2 |
| CVTDQ2PS | xmm,m128 | 4 | | | 2 | 2 | | 2 |
| CVT(T) PS2DQ | xmm,xmm | 2 | | | 2 | | 3 | 2 |
| CVT(T) PS2DQ | xmm,m128 | 4 | | | 2 | 2 | | 2 |
| CVTDQ2PD | xmm,xmm | 4 | | | 4 | | 4 | 2 |
| CVTDQ2PD | xmm,m64 | 5 | | | 4 | 1 | | 2 |
| CVT(T)PD2DQ | xmm,xmm | 4 | | | 4 | | 4 | 3 |
| CVT(T)PD2DQ | xmm,m128 | 6 | | | 4 | 2 | | 3 |
| CVTPI2PS | xmm,mm | 1 | | 1 | | | 3 | 1 |
| CVTPI2PS | xmm,m64 | 2 | | 1 | | 1 | | 1 |
| CVT(T)PS2PI | mm,xmm | 1 | | 1 | | | 3 | 1 |
| CVT(T)PS2PI | mm,m128 | 2 | | 1 | | 1 | | 1 |
| CVTPI2PD | xmm,mm | 4 | 2 | 2 | | | 5 | 2 |
| CVTPI2PD | xmm,m64 | 5 | 2 | 2 | | 1 | | 2 |
| CVT(T) PD2PI | mm,xmm | 3 | | | 3 | | 4 | 2 |
| CVT(T) PD2PI | mm,m128 | 5 | | | 3 | 2 | | 2 |
| CVTSI2SS | xmm,r32 | 2 | 1 | 1 | | | 4 | 1 |
| CVT(T)SS2SI | r32,xmm | 2 | | 1 | 1 | | 4 | 1 |
| CVT(T)SS2SI | r32,m32 | 3 | | 1 | 1 | 1 | | 1 |
| CVTSI2SD | xmm,r32 | 2 | 1 | 1 | | | 4 | 1 |
| CVTSI2SD | xmm,m32 | 3 | 1 | 1 | | 1 | | 1 |
| CVT(T)SD2SI | r32,xmm | 2 | | 1 | 1 | | 4 | 1 |
| CVT(T)SD2SI | r32,m64 | 3 | | 1 | 1 | 1 | | 1 |
| Arithmetic | | | | | | | | |
| ADDSS/D SUBSS/D | xmm,xmm | 1 | | | 1 | | 3 | 1 |
| ADDSS/D SUBSS/D | xmm,m32/64 | 2 | | | 1 | 1 | 3 | 1 |
| ADDPS/D SUBPS/D | xmm,xmm | 2 | | | 2 | | 3 | 2 |
| ADDPS/D SUBPS/D | xmm,m128 | 4 | | | 2 | 2 | 3 | 2 |
| ADDSUBPS/D g) | xmm,xmm | 2 | | | 2 | | 3 | 2 |
| HADDPS HSUBPS g) | xmm,xmm | 6? | | | ? | | 7 | 4 |
| HADDPD HSUBPD g) | xmm,xmm | 3 | | | 3 | | 4 | 2 |
| MULSS | xmm,xmm | 1 | 1 | | | | 4 | 1 |
| MULSD | xmm,xmm | 1 | 1 | | | | 5 | 2 |
| MULSS | xmm,m32 | 2 | 1 | | | 1 | 4 | 1 |
| MULSD | xmm,m64 | 2 | 1 | | | 1 | 5 | 2 |
| MULPS | xmm,xmm | 2 | 2 | | | | 4 | 2 |
| MULPD | xmm,xmm | 2 | 2 | | | | 5 | 4 |
| MULPS | xmm,m128 | 4 | 2 | | | 2 | 4 | 2 |
| MULPD | xmm,m128 | 4 | 2 | | | 2 | 5 | 4 |
| DIVSS | xmm,xmm | 1 | 1 | | | | 9-18 c) | 8-17 c) |
| DIVSD | xmm,xmm | 1 | 1 | | | | 9-32 c) | 8-31 c) |
| DIVSS | xmm,m32 | 2 | 1 | | | 1 | 9-18 c) | 8-17 c) |
| DIVSD | xmm,m64 | 2 | 1 | | | 1 | 9-32 c) | 8-31 c) |
| DIVPS | xmm,xmm | 2 | 2 | | | | 16-34 c) | 16-34 c) |
| DIVPD | xmm,xmm | 2 | 2 | | | | 16-62 c) | 16-62 c) |
| DIVPS | xmm,m128 | 4 | 2 | | | 2 | 16-34 c) | 16-34 c) |
| DIVPD | xmm,m128 | 4 | 2 | | | 2 | 16-62 c) | 16-62 c) |

Pentium M

| | | | | | | | | | | |
|---------------------|------------|-----|----|---|---|----|----|----|--------|--------|
| CMPccSS/D | xmm,xmm | 1 | | | 1 | | | | 3 | 1 |
| CMPccSS/D | xmm,m32/64 | 2 | | | 1 | 1 | | | | 1 |
| CMPccPS/D | xmm,xmm | 2 | | | 2 | | | | 3 | 2 |
| CMPccPS/D | xmm,m128 | 4 | | | 2 | 2 | | | | 2 |
| COMISS/D UCOMISS/D | xmm,xmm | 1 | | 1 | | | | | | 1 |
| COMISS/D UCOMISS/D | xmm,m32/64 | 2 | | 1 | | 1 | | | | 1 |
| MAXSS/D MINSS/D | xmm,xmm | 1 | | | 1 | | | | 3 | 1 |
| MAXSS/D MINSS/D | xmm,m32/64 | 2 | | | 1 | 1 | | | 3 | 1 |
| MAXPS/D MINPS/D | xmm,xmm | 2 | | | 2 | | | | 3 | 2 |
| MAXPS/D MINPS/D | xmm,m128 | 4 | | | 2 | 2 | | | 3 | 2 |
| RCPSS | xmm,xmm | 1 | | 1 | | | | | 3 | 1 |
| RCPSS | xmm,m32 | 2 | | 1 | | 1 | | | | 1 |
| RCPPS | xmm,xmm | 2 | | 2 | | | | | 3 | 2 |
| RCPPS | xmm,m128 | 4 | | 2 | | 2 | | | | 2 |
| Math | | | | | | | | | | |
| SQRTSS | xmm,xmm | 2 | 2 | | | | | | 6-30 | 4-28 |
| SQRTSS | xmm,m32 | 3 | 2 | | | 1 | | | | 4-28 |
| SQRTSD | xmm,xmm | 1 | 1 | | | | | | 5-58 | 4-57 |
| SQRTSD | xmm,m64 | 2 | 1 | | | 1 | | | | 4-57 |
| SQRTPS | xmm,xmm | 2 | 2 | | | | | | 8-56 | 16-55 |
| SQRTPD | xmm,xmm | 2 | 2 | | | | | | 16-114 | 16-114 |
| SQRTPS | xmm,m128 | 4 | 2 | | | 2 | | | | 16-55 |
| SQRTPD | xmm,m128 | 4 | 2 | | | 2 | | | | 16-114 |
| RSQRTSS | xmm,xmm | 1 | | 1 | | | | | 3 | 1 |
| RSQRTSS | xmm,m32 | 2 | | 1 | | 1 | | | | 1 |
| RSQRTPS | xmm,xmm | 2 | | 3 | | | | | 3 | 2 |
| RSQRTPS | xmm,m128 | 4 | | 2 | | 2 | | | | 2 |
| Logic | | | | | | | | | | |
| AND/ANDN/OR/XORPS/D | xmm,xmm | 2 | | | 2 | | | | 1 | 1 |
| AND/ANDN/OR/XORPS/D | xmm,m128 | 4 | | | 2 | 2 | | | | 1 |
| Other | | | | | | | | | | |
| LDMXCSR | m32 | 9 | 9 | | | | | | | 20 |
| STMXCSR | m32 | 6 | 6 | | | | | | | 12 |
| FXSAVE | m4096 | 118 | 32 | | | | 43 | 43 | | 63 |
| FXRSTOR | m4096 | 87 | 43 | | | 44 | | | | 72 |

Notes:

- c) High values are typical, low values are for round divisors.
- g) SSE3 instruction only available on Core Solo and Core Duo.
- j) Also uses some execution units under port 1.

Intel Core 2 (Merom, 65nm)

List of instruction timings and μ op breakdown

Explanation of column headings:

| | |
|--|---|
| Operands: | i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, mm/x = mmx or xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc. |
| μops fused domain: | The number of μ ops at the decode, rename, allocate and retirement stages in the pipeline. Fused μ ops count as one. |
| μops unfused domain: | The number of μ ops for each execution port. Fused μ ops count as two. Fused macro-ops count as one. The instruction has μ op fusion if the sum of the numbers listed under p015 + p2 + p3 + p4 exceeds the number listed under μ ops fused domain. An x under p0, p1 or p5 means that at least one of the μ ops listed under p015 can optionally go to this port. For example, a 1 under p015 and an x under p0 and p5 means one μ op which can go to either port 0 or port 5, whichever is vacant first. A value listed under p015 but nothing under p0, p1 and p5 means that it is not known which of the three ports these μ ops go to. |
| p015: | The total number of μ ops going to port 0, 1 and 5. |
| p0: | The number of μ ops going to port 0 (execution units). |
| p1: | The number of μ ops going to port 1 (execution units). |
| p5: | The number of μ ops going to port 5 (execution units). |
| p2: | The number of μ ops going to port 2 (memory read). |
| p3: | The number of μ ops going to port 3 (memory write address). |
| p4: | The number of μ ops going to port 4 (memory write data). |
| Unit: | Tells which execution unit cluster is used. An additional delay of 1 clock cycle is generated if a register written by a μ op in the integer unit (int) is read by a μ op in the floating point unit (float) or vice versa. flt→int means that an instruction with multiple μ ops receive the input in the float unit and delivers the output in the int unit. Delays for moving data between different units are included under latency when they are unavoidable. For example, movd eax,xmm0 has an extra 1 clock delay for moving from the XMM-integer unit to the general purpose integer unit. This is included under latency because it occurs regardless of which instruction comes next. Nothing listed under unit means that additional delays are either unlikely to occur or unavoidable and therefore included in the latency figure. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay. The time unit used is core clock cycles, not the reference clock cycles given by the time stamp counter. |
| Reciprocal throughput: | The average number of core clock cycles per instruction for a series of independent instructions of the same kind in the same thread. |

Integer instructions

| Instruction | Operands | μ ops fused domain | μ ops unfused domain | | | | | | | Unit | Latency | Reciprocal throughput |
|--------------------------|----------|------------------------|--------------------------|----|----|----|----|----|----|------|---------|-----------------------|
| | | | p015 | p0 | p1 | p5 | p2 | p3 | p4 | | | |
| Move instructions | | | | | | | | | | | | |
| MOV | r,r/i | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |

Merom

| | | | | | | | | | | | | |
|--------------------------------|---------|----|----|---|---|---|---|---|---|-----|---------|------|
| MOV a) | r,m | 1 | | | | | 1 | | | int | 2 | 1 |
| MOV a) | m,r | 1 | | | | | | 1 | 1 | int | 3 | 1 |
| MOV | m,i | 1 | | | | | | 1 | 1 | int | 3 | 1 |
| MOV | r,sr | 1 | | | | | 1 | | | int | | 1 |
| MOV | m,sr | 2 | | | | | 1 | 1 | 1 | int | | 1 |
| MOV | sr,r | 8 | 4 | x | x | x | 4 | | | int | | 16 |
| MOV | sr,m | 8 | 3 | x | | x | 5 | | | int | | 16 |
| MOVNTI | m,r | 2 | | | | | | 1 | 1 | int | | 2 |
| MOVSB MOVZX | | | | | | | | | | | | |
| MOVSD | r,r | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| MOVSB MOVZX | r,m | 1 | | | | | 1 | | | int | | 1 |
| CMOVcc | r,r | 2 | 2 | x | x | x | | | | int | 2 | 1 |
| CMOVcc | r,m | 2 | 2 | x | x | x | 1 | | | int | | |
| XCHG | r,r | 3 | 3 | x | x | x | | | | int | 2 | 2 |
| XCHG | r,m | 7 | x | | | | 1 | 1 | 1 | int | high b) | |
| XLAT | | 2 | 1 | | | | 1 | | | int | 4 | 1 |
| PUSH | r | 1 | | | | | | 1 | 1 | int | 3 | 1 |
| PUSH | i | 1 | | | | | | 1 | 1 | int | | 1 |
| PUSH | m | 2 | | | | | 1 | 1 | 1 | int | | 1 |
| PUSH | sr | 2 | 1 | | | | | 1 | 1 | int | | 1 |
| PUSHF(D/Q) | | 17 | 15 | x | x | x | | 1 | 1 | int | | 7 |
| PUSHA(D) i) | | 18 | 9 | | | | | 1 | 8 | int | | 8 |
| POP | r | 1 | | | | | 1 | | | int | 2 | 1 |
| POP | (E/R)SP | 4 | 3 | | | | 1 | | | int | | |
| POP | m | 2 | | | | | 1 | 1 | 1 | int | | 1.5 |
| POP | sr | 10 | 9 | | | | 1 | | | int | | 17 |
| POPF(D/Q) | | 24 | 23 | x | x | x | 1 | | | int | 20 | |
| POPA(D) i) | | 10 | 2 | | | | 8 | | | int | | 7 |
| LAHF SAHF | | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| SALC i) | | 2 | 2 | x | x | x | | | | int | 4 | 1 |
| LEA a) | r,m | 1 | 1 | 1 | | | | | | int | 1 | 1 |
| BSWAP | r | 2 | 2 | 1 | | 1 | | | | int | 4 | 1 |
| LDS LES LFS LGS LSS | m | 11 | 11 | | | | 1 | | | int | | 17 |
| PREFETCHNTA | m | 1 | | | | | 1 | | | int | | 1 |
| PREFETCHT0/1/2 | m | 1 | | | | | 1 | | | int | | 1 |
| LFENCE | | 2 | | | | | | 1 | 1 | int | | 8 |
| MFENCE | | 2 | | | | | | 1 | 1 | int | | 9 |
| SFENCE | | 2 | | | | | | 1 | 1 | int | | 9 |
| CLFLUSH | m8 | 4 | 2 | x | x | x | | 1 | 1 | int | 240 | 117 |
| IN | | | | | | | | | | int | | |
| OUT | | | | | | | | | | int | | |
| Arithmetic instructions | | | | | | | | | | | | |
| ADD SUB | r,r/i | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| ADD SUB | r,m | 1 | 1 | x | x | x | 1 | | | int | | 1 |
| ADD SUB | m,r/i | 2 | 1 | x | x | x | 1 | 1 | 1 | int | 6 | 1 |
| ADC SBB | r,r/i | 2 | 2 | x | x | x | | | | int | 2 | 2 |
| ADC SBB | r,m | 2 | 2 | x | x | x | 1 | | | int | 2 | 2 |
| ADC SBB | m,r/i | 4 | 3 | x | x | x | 1 | 1 | 1 | int | 7 | |
| CMP | r,r/i | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| CMP | m,r/i | 1 | 1 | x | x | x | 1 | | | int | 1 | 1 |
| INC DEC NEG NOT | r | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| INC DEC NEG NOT | m | 3 | 1 | x | x | x | 1 | 1 | 1 | int | 6 | 1 |

Merom

| | | | | | | | | | | | | |
|---------------------------|----------------|----|----|---|---|---|---|---|---|-----|-------|----------|
| AAA AAS DAA DAS i) | | 1 | 1 | | 1 | | | | | int | | 1 |
| AAD i) | | 3 | 3 | x | x | x | | | | int | | 1 |
| AAM i) | | 4 | 4 | | | | | | | int | 17 | |
| MUL IMUL | r8 | 1 | 1 | | 1 | | | | | int | 3 | 1 |
| MUL IMUL | r16 | 3 | 3 | x | x | x | | | | int | 5 | 1.5 |
| MUL IMUL | r32 | 3 | 3 | x | x | x | | | | int | 5 | 1.5 |
| MUL IMUL | r64 | 3 | 3 | x | x | x | | | | int | 7 | 4 |
| IMUL | r16,r16 | 1 | 1 | | 1 | | | | | int | 3 | 1 |
| IMUL | r32,r32 | 1 | 1 | | 1 | | | | | int | 3 | 1 |
| IMUL | r64,r64 | 1 | 1 | 1 | | | | | | int | 5 | 2 |
| IMUL | r16,r16,i | 1 | 1 | | 1 | | | | | int | 3 | 1 |
| IMUL | r32,r32,i | 1 | 1 | | 1 | | | | | int | 3 | 1 |
| IMUL | r64,r64,i | 1 | 1 | 1 | | | | | | int | 5 | 2 |
| MUL IMUL | m8 | 1 | 1 | | 1 | | 1 | | | int | 3 | 1 |
| MUL IMUL | m16 | 3 | 3 | x | x | x | 1 | | | int | 5 | 1.5 |
| MUL IMUL | m32 | 3 | 3 | x | x | x | 1 | | | int | 5 | 1.5 |
| MUL IMUL | m64 | 3 | 2 | 2 | | | 1 | | | int | 7 | 4 |
| IMUL | r16,m16 | 1 | 1 | | 1 | | 1 | | | int | 3 | 1 |
| IMUL | r32,m32 | 1 | 1 | | 1 | | 1 | | | int | 3 | 1 |
| IMUL | r64,m64 | 1 | 1 | 1 | | | 1 | | | int | 5 | 2 |
| IMUL | r16,m16,i | 1 | 1 | | 1 | | 1 | | | int | | 2 |
| IMUL | r32,m32,i | 1 | 1 | | 1 | | 1 | | | int | | 1 |
| IMUL | r64,m64,i | 1 | 1 | 1 | | | 1 | | | int | | 2 |
| DIV IDIV | r8 | 3 | 3 | | | | | | | int | 18 | 12 |
| DIV IDIV | r16 | 5 | 5 | | | | | | | int | 18-26 | 12-20 c) |
| DIV IDIV | r32 | 4 | 4 | | | | | | | int | 18-42 | 12-36 c) |
| DIV | r64 | 32 | 32 | | | | | | | int | 29-61 | 18-37 c) |
| IDIV | r64 | 56 | 56 | | | | | | | int | 39-72 | 28-40 c) |
| DIV IDIV | m8 | 4 | 3 | | | | 1 | | | int | 18 | 12 |
| DIV IDIV | m16 | 6 | 5 | | | | 1 | | | int | 18-26 | 12-20 c) |
| DIV IDIV | m32 | 5 | 4 | | | | 1 | | | int | 18-42 | 12-36 c) |
| DIV | m64 | 32 | 31 | | | | 1 | | | int | 29-61 | 18-37 c) |
| IDIV | m64 | 56 | 55 | | | | 1 | | | int | 39-72 | 28-40 c) |
| CBW CWDE CDQE | | 1 | 1 | x | x | x | | | | int | 1 | |
| CWD CDQ CQO | | 1 | 1 | x | | x | | | | int | 1 | |
| Logic instructions | | | | | | | | | | | | |
| AND OR XOR | r,r/i | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| AND OR XOR | r,m | 1 | 1 | x | x | x | 1 | | | int | | 1 |
| AND OR XOR | m,r/i | 2 | 1 | x | x | x | 1 | 1 | 1 | int | 6 | 1 |
| TEST | r,r/i | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| TEST | m,r/i | 1 | 1 | x | x | x | 1 | | | int | | 1 |
| SHR SHL SAR | r,i/cl | 1 | 1 | x | | x | | | | int | 1 | 0.5 |
| SHR SHL SAR | m,i/cl | 3 | 2 | x | | x | 1 | 1 | 1 | int | 6 | 1 |
| ROR ROL | r,i/cl | 1 | 1 | x | | x | | | | int | 1 | 1 |
| ROR ROL | m,i/cl | 3 | 2 | x | | x | 1 | 1 | 1 | int | 6 | 1 |
| RCR RCL | r,1 | 2 | 2 | x | x | x | | | | int | 2 | 2 |
| RCR | r8,i/cl | 9 | 9 | x | x | x | | | | int | 12 | |
| RCL | r8,i/cl | 8 | 8 | x | x | x | | | | int | 11 | |
| RCR RCL | r16/32/64,i/cl | 6 | 6 | x | x | x | | | | int | 11 | |
| RCR RCL | m,1 | 4 | 3 | x | x | x | 1 | 1 | 1 | int | 7 | |
| RCR | m8,i/cl | 12 | 9 | x | x | x | 1 | 1 | 1 | int | 14 | |
| RCL | m8,i/cl | 11 | 8 | x | x | x | 1 | 1 | 1 | int | 13 | |

Merom

| | | | | | | | | | | | | |
|--------------------------------------|----------------|----------------|----|---|---|---|-----|-------|-----|-----|--------------|------|
| RCR RCL | m16/32/64,i/cl | 10 | 7 | x | x | x | 1 | 1 | 1 | int | 13 | |
| SHLD SHRD | r,r,i/cl | 2 | 2 | x | x | x | | | | int | 2 | 1 |
| SHLD SHRD | m,r,i/cl | 3 | 2 | x | x | x | 1 | 1 | 1 | int | 7 | |
| BT | r,r/i | 1 | 1 | x | x | x | | | | int | 1 | 1 |
| BT | m,r | 10 | 9 | x | x | x | 1 | | | int | | 5 |
| BT | m,i | 2 | 1 | x | x | x | 1 | | | int | | 1 |
| BTR BTS BTC | r,r/i | 1 | 1 | x | x | x | | | | int | 1 | |
| BTR BTS BTC | m,r | 11 | 8 | x | x | x | 1 | 1 | 1 | int | 5 | |
| BTR BTS BTC | m,i | 3 | 1 | x | x | x | 1 | 1 | 1 | int | 6 | |
| BSF BSR | r,r | 2 | 2 | x | 1 | x | | | | int | 2 | 1 |
| BSF BSR | r,m | 2 | 2 | x | 1 | x | 1 | | | int | | 2 |
| SETcc | r | 1 | 1 | x | x | x | | | | int | 1 | 1 |
| SETcc | m | 2 | 1 | x | x | x | | 1 | 1 | int | | 1 |
| CLC STC CMC | | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| CLD | | 7 | 7 | x | x | x | | | | int | | 4 |
| STD | | 6 | 6 | x | x | x | | | | int | | 14 |
| Control transfer instructions | | | | | | | | | | | | |
| JMP | short/near | 1 | 1 | | | | 1 | | | int | 0 | 1-2 |
| JMP i) | far | 30 | 30 | | | | | | | int | | 76 |
| JMP | r | 1 | 1 | | | | 1 | | | int | 0 | 1-2 |
| JMP | m(near) | 1 | 1 | | | | 1 | 1 | | int | 0 | 1-2 |
| JMP | m(far) | 31 | 29 | | | | | 2 | | int | | 68 |
| Conditional jump | short/near | 1 | 1 | | | | 1 | | | int | 0 | 1 |
| Fused compare/test and branch e,i) | | 1 | 1 | | | | 1 | | | int | 0 | 1 |
| J(E/R)CXZ | short | 2 | 2 | x | x | 1 | | | | int | | 1-2 |
| LOOP | short | 11 | 11 | x | x | x | | | | int | | 5 |
| LOOP(N)E | short | 11 | 11 | x | x | x | | | | int | | 5 |
| CALL | near | 3 | 2 | x | x | x | | | 1 1 | int | | 2 |
| CALL i) | far | 43 | 43 | | | | | | | int | | 75 |
| CALL | r | 3 | 2 | | | | | | 1 1 | int | | 2 |
| CALL | m(near) | 4 | 3 | | | | | 1 1 1 | | int | | 2 |
| CALL | m(far) | 44 | 42 | | | | | 2 | | int | | 75 |
| RETN | | 1 | 1 | | | | 1 1 | | | int | | 2 |
| RETN | i | 3 | x | | | | 1 1 | | | int | | 2 |
| RETF | | 32 | 30 | | | | | 2 | | int | | 78 |
| RETF | i | 32 | 30 | | | | | 2 | | int | | 78 |
| BOUND i) | r,m | 15 | 13 | | | | | 2 | | int | | 8 |
| INTO i) | | 5 | 5 | | | | | | | int | | 3 |
| String instructions | | | | | | | | | | | | |
| LODS | | 3 | 2 | | | | 1 | | | int | | 1 |
| REP LODS | | 4+7n - 14+6n | | | | | | | | int | 1+5n - 21+3n | |
| STOS | | 4 | 2 | | | | | | 1 1 | int | | 1 |
| REP STOS | | 8+5n - 20+1.2n | | | | | | | | int | 7+2n - 0.55n | |
| MOVS | | 8 | 5 | | | | | | | int | | |
| | | 1 | 1 | 1 | | 5 | | | | int | | |
| REP MOVS | | 7+7n - 13+n | | | | | | | | int | 1+3n - 0.63n | |
| SCAS | | 4 | 3 | | | | | 1 | | int | | 1 |
| REP(N)E SCAS | | 7+8n - 17+7n | | | | | | | | int | 3+8n - 23+6n | |
| CMPS | | 7 | 5 | | | | | 2 | | int | | 3 |
| REP(N)E CMPS | | 7+10n - 7+9n | | | | | | | | int | 2+7n - 22+5n | |

Merom

| | | | | | | | | | | | | |
|------------------|-----|--------|----|---|---|---|---|---|---|--|-----|---------|
| Other | | | | | | | | | | | | |
| NOP (90) | | 1 | 1 | x | x | x | | | | | int | 0.33 |
| Long NOP (0F 1F) | | 1 | 1 | x | x | x | | | | | int | 1 |
| PAUSE | | 3 | 3 | x | x | x | | | | | int | 8 |
| ENTER | i,0 | 12 | 10 | | | | | 1 | 1 | | int | 8 |
| ENTER | a,b | | | | | | | | | | int | |
| LEAVE | | 3 | 2 | | | | 1 | | | | int | |
| CPUID | | 46-100 | | | | | | | | | int | 180-215 |
| RDTSC | | 29 | | | | | | | | | int | 64 |
| RDPNC | | 23 | | | | | | | | | int | 54 |

Notes:

- a) Applies to all addressing modes
- b) Has an implicit LOCK prefix.
- c) Low values are for small results, high values for high results.
- e) See manual 3: "The microarchitecture of Intel, AMD and VIA CPUs" for restrictions on macro-op fusion.
- i) Not available in 64 bit mode.

Floating point x87 instructions

| Instruction | Operands | μops fused domain | μops unfused domain | | | | | | | | Unit | Latency | Reciprocal throughput |
|-------------------------|----------|-------------------|---------------------|----|----|----|----|----|----|---|-------|---------|-----------------------|
| | | | p015 | p0 | p1 | p5 | p2 | p3 | p4 | | | | |
| Move instructions | | | | | | | | | | | | | |
| FLD | r | 1 | 1 | 1 | | | | | | | float | 1 | 1 |
| FLD | m32/64 | 1 | 1 | | | | | 1 | | | float | 3 | 1 |
| FLD | m80 | 4 | 2 | 2 | | | | 2 | | | float | 4 | 3 |
| FBLD | m80 | 40 | 38 | | | | | 2 | | | float | 45 | 20 |
| FST(P) | r | 1 | 1 | 1 | | | | | | | float | 1 | 1 |
| FST(P) | m32/m64 | 1 | | | | | | | 1 | 1 | float | 3 | 1 |
| FSTP | m80 | 7 | 3 | x | x | x | | | 2 | 2 | float | 4 | 5 |
| FBSTP | m80 | 170 | 166 | x | x | x | | | 2 | 2 | float | 164 | 166 |
| FXCH | r | 1 | 0 f) | | | | | | | | float | 0 | 1 |
| FILD | m | 1 | 1 | 1 | | | | 1 | | | float | 6 | 1 |
| FIST | m | 2 | 1 | | 1 | | | | 1 | 1 | float | 6 | 1 |
| FISTP | m | 3 | 1 | | 1 | | | | 1 | 1 | float | 6 | 1 |
| FISTTP g) | m | 3 | 1 | | 1 | | | | 1 | 1 | float | 6 | 1 |
| FLDZ | | 1 | 1 | 1 | | | | | | | float | | 1 |
| FLD1 | | 2 | 2 | 1 | 1 | | | | | | float | | 2 |
| FLDPI FLDL2E etc. | | 2 | 2 | | 2 | | | | | | float | | 2 |
| FCMOVcc | r | 2 | 2 | 2 | | | | | | | float | 2 | 2 |
| FNSTSW | AX | 1 | 1 | 1 | | | | | | | float | | 1 |
| FNSTSW | m16 | 2 | 1 | 1 | | | | | 1 | 1 | float | | 2 |
| FLDCW | m16 | 2 | 1 | | | | | 1 | | | float | | 10 |
| FNSTCW | m16 | 3 | 1 | | | | | | 1 | 1 | float | | 8 |
| FINCSTP FDECSTP | | 1 | 1 | 1 | | | | | | | float | 1 | 1 |
| FFREE(P) | r | 2 | 2 | 2 | | | | | | | float | | 2 |
| FNSAVE | m | 142 | | | | | | | | | float | 184 | 192 |
| FRSTOR | m | 78 | | | | | | | | | float | 169 | 177 |
| Arithmetic instructions | | | | | | | | | | | | | |
| FADD(P) FSUB(R)(P) | r | 1 | 1 | | 1 | | | | | | float | 3 | 1 |

Merom

| | | | | | | | | | | |
|--------------------|---|-------|-------|---|---|---|---|-------|---------|---------|
| FADD(P) FSUB(R)(P) | m | 1 | 1 | 1 | 1 | 1 | 1 | float | | 1 |
| FMUL(P) | r | 1 | 1 | 1 | | | | float | 5 | 2 |
| FMUL(P) | m | 1 | 1 | 1 | | 1 | | float | | 2 |
| FDIV(R)(P) | r | 1 | 1 | 1 | | | | float | 6-38 d) | 5-37 d) |
| FDIV(R)(P) | m | 1 | 1 | 1 | | 1 | | float | | 5-37 d) |
| FABS | | 1 | 1 | 1 | | | | float | 1 | 1 |
| FCHS | | 1 | 1 | 1 | | | | float | 1 | 1 |
| FCOM(P) FUCOM | r | 1 | 1 | | 1 | | | float | | 1 |
| FCOM(P) FUCOM | m | 1 | 1 | | 1 | 1 | | float | | 1 |
| FCOMPP FUCOMPP | | 2 | 2 | 1 | 1 | | | float | | |
| FCOMI(P) FUCOMI(P) | r | 1 | 1 | | 1 | | | float | | 1 |
| FIADD FISUB(R) | m | 2 | 2 | 1 | 1 | 1 | | float | | 2 |
| FIMUL | m | 2 | 2 | 2 | | 1 | | float | | 2 |
| FIDIV(R) | m | 2 | 2 | 2 | | 1 | | float | | 5-37 d) |
| FICOM(P) | m | 2 | 2 | 1 | 1 | 1 | | float | | 2 |
| FTST | | 1 | 1 | | 1 | | | float | | 1 |
| FXAM | | 1 | 1 | | 1 | | | float | | 1 |
| FPREM FPREM1 | | 21-27 | 21-27 | | | | | float | 16-56 | |
| FRNDINT | | 7-15 | 7-15 | | | | | float | 22-29 | |
| Math | | | | | | | | | | |
| FSCALE | | 27 | 27 | | | | | float | 41 | |
| FXTRACT | | 82 | 82 | | | | | float | 170 | |
| FSQRT | | 1 | 1 | | | | | float | 6-69 | |
| FSIN FCOS | | ~96 | ~96 | | | | | float | ~96 | |
| FSINCOS | | ~100 | ~100 | | | | | float | ~115 | |
| F2XM1 | | ~19 | ~19 | | | | | float | ~45 | |
| FYL2X FYL2XP1 | | ~53 | ~53 | | | | | float | ~96 | |
| FPTAN | | ~98 | ~98 | | | | | float | ~136 | |
| FPATAN | | ~70 | ~70 | | | | | float | ~119 | |
| Other | | | | | | | | | | |
| FNOP | | 1 | 1 | 1 | | | | float | | 1 |
| WAIT | | 2 | 2 | | | | | float | | 1 |
| FNCLEX | | 4 | 4 | | | | | float | | 15 |
| FNINIT | | 15 | 15 | | | | | float | | 63 |

Notes:

- d) Round divisors or low precision give low values.
f) Resolved by register renaming. Generates no μ ops in the unfused domain.
g) SSE3 instruction set.

Integer MMX and XMM instructions

| Instruction | Operands | μ ops fused domain | μ ops unfused domain | | | | | | | | Unit | Latency | Reciprocal throughput |
|-------------------|-------------|------------------------|--------------------------|----|----|----|----|----|----|-----|------|---------|-----------------------|
| | | | p015 | p0 | p1 | p5 | p2 | p3 | p4 | | | | |
| Move instructions | | | | | | | | | | | | | |
| MOVD k) | r32/64,mm/x | 1 | 1 | x | x | x | | | | int | 2 | 0.33 | |
| MOVD k) | m32/64,mm/x | 1 | | | | | | 1 | 1 | | 3 | 1 | |
| MOVD k) | mm/x,r32/64 | 1 | 1 | x | | x | | | | int | 2 | 0.5 | |
| MOVD k) | mm/x,m32/64 | 1 | | | | | 1 | | | int | 2 | 1 | |
| MOVQ | mm/x, mm/x | 1 | 1 | x | x | x | | | | int | 1 | 0.33 | |
| MOVQ | mm/x,m64 | 1 | | | | | 1 | | | int | 2 | 1 | |

Merom

| | | | | | | | | | | | | |
|--------------------------------|-------------|----|---|---|---|---|---|---|---|---------|-----|------|
| MOVQ | m64, mm/x | 1 | | | | | | 1 | 1 | | 3 | 1 |
| MOVDQA | xmm, xmm | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| MOVDQA | xmm, m128 | 1 | | | | | 1 | | | int | 2 | 1 |
| MOVDQA | m128, xmm | 1 | | | | | | 1 | 1 | | 3 | 1 |
| MOVDQU | m128, xmm | 9 | 4 | x | x | x | 1 | 2 | 2 | | 3-8 | 4 |
| MOVDQU | xmm, m128 | 4 | 2 | x | | x | 2 | | | int | 2-8 | 2 |
| LDDQU g) | xmm, m128 | 4 | 2 | x | | x | 2 | | | int | 2-8 | 2 |
| MOVDQ2Q | mm, xmm | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| MOVQ2DQ | xmm,mm | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| MOVNTQ | m64,mm | 1 | | | | | | 1 | 1 | | | 2 |
| MOVNTDQ | m128,xmm | 1 | | | | | | 1 | 1 | | | 2 |
| PACKSSWB/DW | mm,mm | 1 | 1 | 1 | | | | | | int | 1 | 1 |
| PACKUSWB | mm,m64 | 1 | 1 | 1 | | | 1 | | | int | | 1 |
| PACKSSWB/DW | xmm,xmm | 3 | 3 | | | | | | | flt→int | 3 | 2 |
| PACKUSWB | xmm,m128 | 4 | 3 | | | | 1 | | | int | | 2 |
| PUNPCKH/LBW/WD/DQ | mm,mm | 1 | 1 | 1 | | | | | | int | 1 | 1 |
| PUNPCKH/LBW/WD/DQ | mm,m64 | 1 | 1 | 1 | | | 1 | | | int | | 1 |
| PUNPCKH/LBW/WD/DQ | xmm,xmm | 3 | 3 | | | | | | | flt→int | 3 | 2 |
| PUNPCKH/LBW/WD/DQ | xmm,m128 | 4 | 3 | | | | 1 | | | int | | 2 |
| PUNPCKH/LQDQ | xmm,xmm | 1 | 1 | | | | | | | int | 1 | 1 |
| PUNPCKH/LQDQ | xmm, m128 | 2 | 1 | | | | 1 | | | int | | 1 |
| PSHUFB h) | mm,mm | 1 | 1 | | | 1 | | | | int | 1 | 1 |
| PSHUFB h) | mm,m64 | 2 | 1 | | | 1 | 1 | | | int | | 1 |
| PSHUFB h) | xmm,xmm | 4 | 4 | | | | | | | int | 3 | 2 |
| PSHUFB h) | xmm,m128 | 5 | 4 | | | | 1 | | | int | | 2 |
| PSHUFW | mm,mm,i | 1 | 1 | | | 1 | | | | int | 1 | 1 |
| PSHUFW | mm,m64,i | 2 | 1 | | | 1 | 1 | | | int | | 1 |
| PSHUFD | xmm,xmm,i | 2 | 2 | x | x | 1 | | | | flt→int | 3 | 1 |
| PSHUFD | xmm,m128,i | 3 | 2 | x | x | 1 | 1 | | | int | | 1 |
| PSHUFL/HW | xmm,xmm,i | 1 | 1 | | | 1 | | | | int | 1 | 1 |
| PSHUFL/HW | xmm, m128,i | 2 | 1 | | | 1 | 1 | | | int | | 1 |
| PALIGNR h) | mm,mm,i | 2 | 2 | x | x | x | | | | int | 2 | 1 |
| PALIGNR h) | mm,m64,i | 2 | 2 | x | x | x | 1 | | | int | | 1 |
| PALIGNR h) | xmm,xmm,i | 2 | 2 | x | x | x | | | | int | 2 | 1 |
| PALIGNR h) | xmm,m128,i | 2 | 2 | x | x | x | 1 | | | int | | 1 |
| MASKMOVQ | mm,mm | 4 | | | | | | | | int | | 2-5 |
| MASKMOVDQU | xmm,xmm | 10 | | | | | | | | int | | 6-10 |
| PMOVBMSKB | r32,mm/x | 1 | 1 | 1 | | | | | | int | 2 | 1 |
| PEXTRW | r32,mm,i | 2 | 2 | | | | | | | int | 3 | 1 |
| PEXTRW | r32,xmm,i | 3 | 3 | | | | | | | int | 5 | 1 |
| PINSRW | mm,r32,i | 1 | 1 | | | 1 | | | | int | 2 | 1 |
| PINSRW | mm,m16,i | 2 | 1 | | | 1 | 1 | | | int | | 1 |
| PINSRW | xmm,r32,i | 3 | 3 | x | x | x | | | | int | 6 | 1.5 |
| PINSRW | xmm,m16,i | 4 | 3 | x | x | x | 1 | | | int | | 1.5 |
| Arithmetic instructions | | | | | | | | | | | | |
| PADD/SUB(U)(S)B/W/D | mm/x, mm/x | 1 | 1 | x | | x | | | | int | 1 | 0.5 |
| PADD/SUB(U)(S)B/W/D | mm/x,m | 1 | 1 | x | | x | 1 | | | int | | 1 |
| PADDQ PSUBQ | mm/x, mm/x | 2 | 2 | x | | x | | | | int | 2 | 1 |
| PADDQ PSUBQ | mm/x,m | 2 | 2 | x | | x | 1 | | | int | | 1 |
| PHADD(S)W | | | | | | | | | | | | |
| PHSUB(S)W h) | mm,mm | 5 | 5 | | | | | | | int | 5 | 4 |

Merom

[illegible]

Notes:

- g) SSE3 instruction set.
- h) Supplementary SSE3 instruction set.
- k) MASM uses the name MOVD rather than MOVQ for this instruction even when moving 64 bits.

Floating point XMM instructions

Merom

| Instruction | Operands | μops fused do- main | μops unfused domain | | | | | | | | Unit | Laten- cy | Reci- procal through- put |
|-------------------|------------|------------------------------|---------------------|----|----|----|----|----|----|---------|------|--------------|------------------------------------|
| | | | p015 | p0 | p1 | p5 | p2 | p3 | p4 | | | | |
| Move instructions | | | | | | | | | | | | | |
| MOVAPS/D | xmm,xmm | 1 | 1 | x | x | x | | | | int | 1 | 0.33 | |
| MOVAPS/D | xmm,m128 | 1 | | | | | 1 | | | int | 2 | 1 | |
| MOVAPS/D | m128,xmm | 1 | | | | | | 1 | 1 | | 3 | 1 | |
| MOVUPS/D | xmm,m128 | 4 | 2 | 1 | | 1 | 2 | | | int | 2-4 | 2 | |
| MOVUPS/D | m128,xmm | 9 | 4 | x | x | x | 1 | 2 | 2 | | 3-4 | 4 | |
| MOVSS/D | xmm,xmm | 1 | 1 | x | x | x | | | | int | 1 | 0.33 | |
| MOVSS/D | xmm,m32/64 | 1 | | | | | 1 | | | int | 2 | 1 | |
| MOVSS/D | m32/64,xmm | 1 | | | | | | 1 | 1 | | 3 | 1 | |
| MOVHPS/D MOVLPS/D | xmm,m64 | 2 | 1 | | | 1 | 1 | | | int | 3 | 1 | |
| MOVHPS/D | m64,xmm | 2 | 1 | 1 | | | | 1 | 1 | | 5 | 1 | |
| MOVLPS/D | m64,xmm | 1 | | | | | | 1 | 1 | | 3 | 1 | |
| MOVLHPS MOVHLPS | xmm,xmm | 1 | 1 | 1 | | | | | | float | 1 | 1 | |
| MOVMSKPS/D | r32,xmm | 1 | 1 | 1 | | | | | | float | 1 | 1 | |
| MOVNTPS/D | m128,xmm | 1 | | | | | | 1 | 1 | | | 2-3 | |
| SHUFPS | xmm,xmm,i | 3 | 3 | | 3 | | | | | flt→int | 3 | 2 | |
| SHUFPS | xmm,m128,i | 4 | 3 | | 3 | | 1 | | | flt→int | | 2 | |
| SHUFPD | xmm,xmm,i | 1 | 1 | 1 | | | | | | float | 1 | 1 | |
| SHUFPD | xmm,m128,i | 2 | 1 | 1 | | | 1 | | | float | | 1 | |
| MOVDDUP g) | xmm,xmm | 1 | 1 | 1 | | | | | | int | 1 | 1 | |
| MOVDDUP g) | xmm,m64 | 2 | 1 | 1 | | | 1 | | | int | | 1 | |
| MOVSH/LDUP g) | xmm,xmm | 1 | 1 | | | 1 | | | | int | 1 | 1 | |
| MOVSH/LDUP g) | xmm,m128 | 2 | 1 | | | 1 | 1 | | | int | | 1 | |
| UNPCKH/LPS | xmm,xmm | 3 | 3 | | 3 | | | | | flt→int | 3 | 2 | |
| UNPCKH/LPS | xmm,m128 | 4 | 3 | | 3 | | 1 | | | int | | 2 | |
| UNPCKH/LPD | xmm,xmm | 1 | 1 | 1 | | | | | | float | 1 | 1 | |
| UNPCKH/LPD | xmm,m128 | 2 | 1 | 1 | | | 1 | | | float | | 1 | |
| Conversion | | | | | | | | | | | | | |
| CVTPD2PS | xmm,xmm | 2 | 2 | | | | | | | float | 4 | 1 | |
| CVTPD2PS | xmm,m128 | 2 | 2 | | | | 1 | | | float | | 1 | |
| CVTSD2SS | xmm,xmm | 2 | 2 | | | | | | | float | 4 | 1 | |
| CVTSD2SS | xmm,m64 | 2 | 2 | | | | 1 | | | float | | 1 | |
| CVTPS2PD | xmm,xmm | 2 | 2 | 2 | | | | | | float | 2 | 2 | |
| CVTPS2PD | xmm,m64 | 2 | 2 | 2 | | | 1 | | | float | | 2 | |
| CVTSS2SD | xmm,xmm | 2 | 2 | | | | | | | float | 2 | 2 | |
| CVTSS2SD | xmm,m32 | 2 | 2 | 2 | | | 1 | | | float | | 2 | |
| CVTDQ2PS | xmm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 1 | |
| CVTDQ2PS | xmm,m128 | 1 | 1 | | 1 | | 1 | | | float | | 1 | |
| CVT(T) PS2DQ | xmm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 1 | |
| CVT(T) PS2DQ | xmm,m128 | 1 | 1 | | 1 | | 1 | | | float | | 1 | |
| CVTDQ2PD | xmm,xmm | 2 | 2 | 1 | 1 | | | | | float | 4 | 1 | |
| CVTDQ2PD | xmm,m64 | 3 | 2 | | | | 1 | | | float | | 1 | |
| CVT(T)PD2DQ | xmm,xmm | 2 | 2 | | | | | | | float | 4 | 1 | |
| CVT(T)PD2DQ | xmm,m128 | 2 | 2 | | | | 1 | | | float | | 1 | |
| CVTPI2PS | xmm,mm | 1 | 1 | | 1 | | | | | float | 3 | 3 | |
| CVTPI2PS | xmm,m64 | 1 | 1 | | 1 | | 1 | | | float | | 3 | |
| CVT(T)PS2PI | mm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 1 | |
| CVT(T)PS2PI | mm,m128 | 1 | 1 | | 1 | | 1 | | | float | | 1 | |

Merom

| | | | | | | | | | | | |
|--------------------|------------|---|---|---|---|---|--|--|-------|---------|---------|
| CVTPI2PD | xmm,mm | 2 | 2 | 1 | 1 | | | | float | 4 | 1 |
| CVTPI2PD | xmm,m64 | 2 | 2 | 1 | 1 | 1 | | | float | | 1 |
| CVT(T) PD2PI | mm,xmm | 2 | 2 | 1 | 1 | | | | float | 4 | 1 |
| CVT(T) PD2PI | mm,m128 | 2 | 2 | 1 | 1 | 1 | | | float | | 1 |
| CVTSI2SS | xmm,r32 | 1 | 1 | | 1 | | | | float | 4 | 3 |
| CVTSI2SS | xmm,m32 | 1 | 1 | | 1 | 1 | | | float | | 3 |
| CVT(T)SS2SI | r32,xmm | 1 | 1 | | 1 | | | | float | 3 | 1 |
| CVT(T)SS2SI | r32,m32 | 1 | 1 | | 1 | 1 | | | float | | 1 |
| CVTSI2SD | xmm,r32 | 2 | 2 | 1 | 1 | | | | float | 4 | 3 |
| CVTSI2SD | xmm,m32 | 2 | 1 | | 1 | 1 | | | float | | 3 |
| CVT(T)SD2SI | r32,xmm | 1 | 1 | | 1 | | | | float | 3 | 1 |
| CVT(T)SD2SI | r32,m64 | 1 | 1 | | 1 | 1 | | | float | | 1 |
| Arithmetic | | | | | | | | | | | |
| ADDSS/D SUBSS/D | xmm,xmm | 1 | 1 | | 1 | | | | float | 3 | 1 |
| ADDSS/D SUBSS/D | xmm,m32/64 | 1 | 1 | | 1 | 1 | | | float | | 1 |
| ADDPS/D SUBPS/D | xmm,xmm | 1 | 1 | | 1 | | | | float | 3 | 1 |
| ADDPS/D SUBPS/D | xmm,m128 | 1 | 1 | | 1 | 1 | | | float | | 1 |
| ADDSUBPS/D g) | xmm,xmm | 1 | 1 | | 1 | | | | float | 3 | 1 |
| ADDSUBPS/D g) | xmm,m128 | 1 | 1 | | 1 | 1 | | | float | | 1 |
| HADDPS HSUBPS g) | xmm,xmm | 6 | 6 | | | | | | float | 9 | 3 |
| HADDPS HSUBPS g) | xmm,m128 | 7 | 6 | | | 1 | | | float | | 3 |
| HADDPD HSUBPD g) | xmm,xmm | 3 | 3 | | | | | | float | 5 | 2 |
| HADDPD HSUBPD g) | xmm,m128 | 4 | 3 | | | 1 | | | float | | 2 |
| MULSS | xmm,xmm | 1 | 1 | 1 | | | | | float | 4 | 1 |
| MULSS | xmm,m32 | 1 | 1 | 1 | | 1 | | | float | | 1 |
| MULSD | xmm,xmm | 1 | 1 | 1 | | | | | float | 5 | 1 |
| MULSD | xmm,m64 | 1 | 1 | 1 | | 1 | | | float | | 1 |
| MULPS | xmm,xmm | 1 | 1 | 1 | | | | | float | 4 | 1 |
| MULPS | xmm,m128 | 1 | 1 | 1 | | 1 | | | float | | 1 |
| MULPD | xmm,xmm | 1 | 1 | 1 | | | | | float | 5 | 1 |
| MULPD | xmm,m128 | 1 | 1 | 1 | | 1 | | | float | | 1 |
| DIVSS | xmm,xmm | 1 | 1 | 1 | | | | | float | 6-18 d) | 5-17 d) |
| DIVSS | xmm,m32 | 1 | 1 | 1 | | 1 | | | float | | 5-17 d) |
| DIVSD | xmm,xmm | 1 | 1 | 1 | | | | | float | 6-32 d) | 5-31 d) |
| DIVSD | xmm,m64 | 1 | 1 | 1 | | 1 | | | float | | 5-31 d) |
| DIVPS | xmm,xmm | 1 | 1 | 1 | | | | | float | 6-18 d) | 5-17 d) |
| DIVPS | xmm,m128 | 1 | 1 | 1 | | 1 | | | float | | 5-17 d) |
| DIVPD | xmm,xmm | 1 | 1 | 1 | | | | | float | 6-32 d) | 5-31 d) |
| DIVPD | xmm,m128 | 1 | 1 | 1 | | 1 | | | float | | 5-31 d) |
| RCPSS/PS | xmm,xmm | 1 | 1 | | 1 | | | | float | 3 | 2 |
| RCPSS/PS | xmm,m | 1 | 1 | | 1 | 1 | | | float | | 2 |
| CMPccSS/D | xmm,xmm | 1 | 1 | | 1 | | | | float | 3 | 1 |
| CMPccSS/D | xmm,m32/64 | 1 | 1 | | 1 | 1 | | | float | | 1 |
| CMPccPS/D | xmm,xmm | 1 | 1 | | 1 | | | | float | 3 | 1 |
| CMPccPS/D | xmm,m128 | 1 | 1 | | 1 | 1 | | | float | | 1 |
| COMISS/D UCOMISS/D | xmm,xmm | 1 | 1 | | 1 | | | | float | 3 | 1 |
| COMISS/D UCOMISS/D | xmm,m32/64 | 1 | 1 | | 1 | 1 | | | float | | 1 |
| MAXSS/D MINSS/D | xmm,xmm | 1 | 1 | | 1 | | | | float | 3 | 1 |
| MAXSS/D MINSS/D | xmm,m32/64 | 1 | 1 | | 1 | 1 | | | float | | 1 |
| MAXPS/D MINPS/D | xmm,xmm | 1 | 1 | | 1 | | | | float | 3 | 1 |
| MAXPS/D MINPS/D | xmm,m128 | 1 | 1 | | 1 | 1 | | | float | | 1 |

Merom

| Math | | | | | | | | | | | | | |
|---------------------|----------|-----|----|---|---|---|---|---|---|--|-------|------|------|
| SQRTSS/PS | xmm,xmm | 1 | 1 | 1 | | | | | | | float | 6-29 | 6-29 |
| SQRTSS/PS | xmm,m | 2 | 1 | 1 | | | 1 | | | | float | | 6-29 |
| SQRTSD/PD | xmm,xmm | 1 | 1 | 1 | | | | | | | float | 6-58 | 6-58 |
| SQRTSD/PD | xmm,m | 2 | 1 | 1 | | | 1 | | | | float | | 6-58 |
| RSQRTSS/PS | xmm,xmm | 1 | 1 | | 1 | | | | | | float | 3 | 2 |
| RSQRTSS/PS | xmm,m | 1 | 1 | | 1 | | 1 | | | | float | | 2 |
| Logic | | | | | | | | | | | | | |
| AND/ANDN/OR/XORPS/D | xmm,xmm | 1 | 1 | x | x | x | | | | | int | 1 | 0.33 |
| AND/ANDN/OR/XORPS/D | xmm,m128 | 1 | 1 | x | x | x | 1 | | | | int | | 1 |
| Other | | | | | | | | | | | | | |
| LDMXCSR | m32 | 14 | 13 | | | | 1 | | | | | | 42 |
| STMXCSR | m32 | 6 | 4 | | | | | 1 | 1 | | | | 19 |
| FXSAVE | m4096 | 141 | | | | | | | | | | 145 | 145 |
| FXRSTOR | m4096 | 119 | | | | | | | | | | 164 | 164 |

Notes:

- d) Round divisors give low values.
g) SSE3 instruction set.

Intel Core 2 (Wolfdale, 45nm)

List of instruction timings and μ op breakdown

Explanation of column headings:

| | |
|--|---|
| Operands: | i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, mm/x = mmx or xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc. |
| μops fused domain: | The number of μ ops at the decode, rename, allocate and retirement stages in the pipeline. Fused μ ops count as one. |
| μops unfused domain: | The number of μ ops for each execution port. Fused μ ops count as two. Fused macro-ops count as one. The instruction has μ op fusion if the sum of the numbers listed under p015 + p2 + p3 + p4 exceeds the number listed under μ ops fused domain. An x under p0, p1 or p5 means that at least one of the μ ops listed under p015 can optionally go to this port. For example, a 1 under p015 and an x under p0 and p5 means one μ op which can go to either port 0 or port 5, whichever is vacant first. A value listed under p015 but nothing under p0, p1 and p5 means that it is not known which of the three ports these μ ops go to. |
| p015: | The total number of μ ops going to port 0, 1 and 5. |
| p0: | The number of μ ops going to port 0 (execution units). |
| p1: | The number of μ ops going to port 1 (execution units). |
| p5: | The number of μ ops going to port 5 (execution units). |
| p2: | The number of μ ops going to port 2 (memory read). |
| p3: | The number of μ ops going to port 3 (memory write address). |
| p4: | The number of μ ops going to port 4 (memory write data). |
| Unit: | Tells which execution unit cluster is used. An additional delay of 1 clock cycle is generated if a register written by a μ op in the integer unit (int) is read by a μ op in the floating point unit (float) or vice versa. flt→int means that an instruction with multiple μ ops receive the input in the float unit and delivers the output in the int unit. Delays for moving data between different units are included under latency when they are unavoidable. For example, movd eax,xmm0 has an extra 1 clock delay for moving from the XMM-integer unit to the general purpose integer unit. This is included under latency because it occurs regardless of which instruction comes next. Nothing listed under unit means that additional delays are either unlikely to occur or unavoidable and therefore included in the latency figure. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay. The time unit used is core clock cycles, not the reference clock cycles given by the time stamp counter. |
| Reciprocal throughput: | The average number of core clock cycles per instruction for a series of independent instructions of the same kind in the same thread. |

Integer instructions

| Instruction | Operands | μ ops fused domain | μ ops unfused domain | | | | | | | Unit | Latency | Reciprocal throughput |
|--------------------------|----------|------------------------|--------------------------|----|----|----|----|----|----|------|---------|-----------------------|
| | | | p015 | p0 | p1 | p5 | p2 | p3 | p4 | | | |
| Move instructions | | | | | | | | | | | | |
| MOV | r,r/i | 1 | 1 | x | x | x | | | | | 1 | 0.33 |
| MOV a) | r,m | 1 | | | | | 1 | | | | 2 | 1 |

Wolfdale

| | | | | | | | | | | | |
|--------------------------------|----------|----|----|---|---|---|---|---|---|---------|------|
| MOV a) | m,r | 1 | | | | | 1 | 1 | | 3 | 1 |
| MOV | m,i | 1 | | | | | 1 | 1 | | 3 | 1 |
| MOV | r,sr | 1 | | | | | 1 | | | | 1 |
| MOV | m,sr | 2 | | | | | 1 | 1 | 1 | | 1 |
| MOV | sr,r | 8 | 4 | x | x | x | 4 | | | | 16 |
| MOV | sr,m | 8 | 3 | x | | x | 5 | | | | 16 |
| MOVNTI | m,r | 2 | | | | | | 1 | 1 | | 2 |
| MOVSX MOVZX | | | | | | | | | | | |
| MOVSXD | r,r | 1 | 1 | x | x | x | | | | 1 | 0.33 |
| MOVSX MOVZX | r16/32,m | 1 | | | | | 1 | | | | 1 |
| MOVSX MOVSXD | r64,m | 2 | 1 | x | x | x | 1 | | | | 1 |
| CMOVcc | r,r | 2 | 2 | x | x | x | | | | 2 | 1 |
| CMOVcc | r,m | 2 | 2 | x | x | x | 1 | | | | |
| XCHG | r,r | 3 | 3 | x | x | x | | | | 2 | 2 |
| XCHG | r,m | 7 | x | | | | 1 | 1 | 1 | high b) | |
| XLAT | | 2 | 1 | | | | 1 | | | 4 | 1 |
| PUSH | r | 1 | | | | | | 1 | 1 | 3 | 1 |
| PUSH | i | 1 | | | | | | 1 | 1 | | 1 |
| PUSH | m | 2 | | | | | 1 | 1 | 1 | | 1 |
| PUSH | sr | 2 | 1 | | | | | 1 | 1 | | 1 |
| PUSHF(D/Q) | | 17 | 15 | x | x | x | | 1 | 1 | | 7 |
| PUSHA(D) i) | | 18 | 9 | | | | | 1 | 8 | | 8 |
| POP | r | 1 | | | | | 1 | | | 2 | 1 |
| POP | (E/R)SP | 4 | 3 | | | | 1 | | | | |
| POP | m | 2 | | | | | 1 | 1 | 1 | | 1.5 |
| POP | sr | 10 | 9 | | | | 1 | | | | 17 |
| POPF(D/Q) | | 24 | 23 | x | x | x | 1 | | | 20 | |
| POPA(D) i) | | 10 | 2 | | | | 8 | | | | 7 |
| LAHF SAHF | | 1 | 1 | x | x | x | | | | 1 | 0.33 |
| SALC i) | | 2 | 2 | x | x | x | | | | 4 | 1 |
| LEA a) | r,m | 1 | 1 | 1 | | | | | | 1 | 1 |
| BSWAP | r | 2 | 2 | 1 | | 1 | | | | 4 | 1 |
| LDS LES LFS LGS LSS | m | 11 | 11 | | | | 1 | | | | 17 |
| PREFETCHNTA | m | 1 | | | | | 1 | | | | 1 |
| PREFETCHT0/1/2 | m | 1 | | | | | 1 | | | | 1 |
| LFENCE | | 2 | | | | | | 1 | 1 | | 8 |
| MFENCE | | 2 | | | | | | 1 | 1 | | 6 |
| SFENCE | | 2 | | | | | | 1 | 1 | | 9 |
| CLFLUSH | m8 | 4 | 2 | 1 | | 1 | | 1 | 1 | 120 | 90 |
| IN | | | | | | | | | | | |
| OUT | | | | | | | | | | | |
| Arithmetic instructions | | | | | | | | | | | |
| ADD SUB | r,r/i | 1 | 1 | x | x | x | | | | 1 | 0.33 |
| ADD SUB | r,m | 1 | 1 | x | x | x | 1 | | | | 1 |
| ADD SUB | m,r/i | 2 | 1 | x | x | x | 1 | 1 | 1 | 6 | 1 |
| ADC SBB | r,r/i | 2 | 2 | x | x | x | | | | 2 | 2 |
| ADC SBB | r,m | 2 | 2 | x | x | x | 1 | | | 2 | 2 |
| ADC SBB | m,r/i | 4 | 3 | x | x | x | 1 | 1 | 1 | 7 | |
| CMP | r,r/i | 1 | 1 | x | x | x | | | | 1 | 0.33 |
| CMP | m,r/i | 1 | 1 | x | x | x | 1 | | | 1 | 1 |
| INC DEC NEG NOT | r | 1 | 1 | x | x | x | | | | 1 | 0.33 |
| INC DEC NEG NOT | m | 3 | 1 | x | x | x | 1 | 1 | 1 | 6 | 1 |

Wolfdale

| | | | | | | | | | | | | |
|---------------------------|-----------|-------|-------|---|----|----|---|---|---|--|----------|------|
| AAA AAS DAA DAS i) | | 1 | 1 | 1 | | | | | | | | 1 |
| AAD i) | | 3 | 3 | x | x | x | | | | | | 1 |
| AAM i) | | 5 | 5 | x | x | x | | | | | 17 | |
| MUL IMUL | r8 | 1 | 1 | | 1 | | | | | | 3 | 1 |
| MUL IMUL | r16 | 3 | 3 | x | x | x | | | | | 5 | 1.5 |
| MUL IMUL | r32 | 3 | 3 | x | x | x | | | | | 5 | 1.5 |
| MUL IMUL | r64 | 3 | 3 | x | x | x | | | | | 7 | 4 |
| IMUL | r16,r16 | 1 | 1 | | 1 | | | | | | 3 | 1 |
| IMUL | r32,r32 | 1 | 1 | | 1 | | | | | | 3 | 1 |
| IMUL | r64,r64 | 1 | 1 | 1 | | | | | | | 5 | 2 |
| IMUL | r16,r16,i | 1 | 1 | | 1 | | | | | | 3 | 1 |
| IMUL | r32,r32,i | 1 | 1 | | 1 | | | | | | 3 | 1 |
| IMUL | r64,r64,i | 1 | 1 | 1 | | | | | | | 5 | 2 |
| MUL IMUL | m8 | 1 | 1 | | 1 | | 1 | | | | 3 | 1 |
| MUL IMUL | m16 | 3 | 3 | x | x | x | 1 | | | | 5 | 1.5 |
| MUL IMUL | m32 | 3 | 3 | x | x | x | 1 | | | | 5 | 1.5 |
| MUL IMUL | m64 | 3 | 2 | 2 | | | 1 | | | | 7 | 4 |
| IMUL | r16,m16 | 1 | 1 | | 1 | | 1 | | | | 3 | 1 |
| IMUL | r32,m32 | 1 | 1 | | 1 | | 1 | | | | 3 | 1 |
| IMUL | r64,m64 | 1 | 1 | 1 | | | 1 | | | | 5 | 2 |
| IMUL | r16,m16,i | 1 | 1 | | 1 | | 1 | | | | | 2 |
| IMUL | r32,m32,i | 1 | 1 | | 1 | | 1 | | | | | 1 |
| IMUL | r64,m64,i | 1 | 1 | 1 | | | 1 | | | | | 2 |
| DIV IDIV | r8 | 4 | 4 | 1 | 2 | 1 | | | | | 9-18 c) | |
| DIV IDIV | r16 | 7 | 7 | x | x | x | | | | | 14-22 c) | |
| DIV IDIV | r32 | 7 | 7 | 2 | 3 | 2 | | | | | 14-23 c) | |
| DIV | r64 | 32-38 | 32-38 | 9 | 10 | 13 | | | | | 18-57 c) | |
| IDIV | r64 | 56-62 | 56-62 | x | x | x | | | | | 34-88 c) | |
| DIV IDIV | m8 | 4 | 3 | 1 | 2 | | 1 | | | | 9-18 | |
| DIV IDIV | m16 | 7 | 7 | 2 | 3 | 2 | 1 | | | | 14-22 c) | |
| DIV IDIV | m32 | 7 | 6 | x | x | x | 1 | | | | 14-23 c) | |
| DIV | m64 | 32 | 31 | x | x | x | 1 | | | | 34-88 c) | |
| IDIV | m64 | 56 | 55 | x | x | x | 1 | | | | 39-72 c) | |
| CBW CWDE CDQE | | 1 | 1 | x | x | x | | | | | 1 | |
| CWD CDQ CQO | | 1 | 1 | x | | x | | | | | 1 | |
| Logic instructions | | | | | | | | | | | | |
| AND OR XOR | r,r/i | 1 | 1 | x | x | x | | | | | 1 | 0.33 |
| AND OR XOR | r,m | 1 | 1 | x | x | x | 1 | | | | | 1 |
| AND OR XOR | m,r/i | 2 | 1 | x | x | x | 1 | 1 | 1 | | 6 | 1 |
| TEST | r,r/i | 1 | 1 | x | x | x | | | | | 1 | 0.33 |
| TEST | m,r/i | 1 | 1 | x | x | x | 1 | | | | | 1 |
| SHR SHL SAR | r,i/cl | 1 | 1 | x | | x | | | | | 1 | 0.5 |
| SHR SHL SAR | m,i/cl | 3 | 2 | x | | x | 1 | 1 | 1 | | 6 | 1 |
| ROR ROL | r,i/cl | 1 | 1 | x | | x | | | | | 1 | 1 |
| ROR ROL | m,i/cl | 3 | 2 | x | | x | 1 | 1 | 1 | | 6 | 1 |
| RCR RCL | r,1 | 2 | 2 | x | x | x | | | | | 2 | 2 |
| RCR | r8,i/cl | 9 | 9 | x | x | x | | | | | 12 | |
| RCL | r8,i/cl | 8 | 8 | x | x | x | | | | | 11 | |
| RCR RCL | r,i/cl | 6 | 6 | x | x | x | | | | | 11 | |
| RCR RCL | m,1 | 4 | 3 | x | x | x | 1 | 1 | 1 | | 7 | |
| RCR | m8,i/cl | 12 | 9 | x | x | x | 1 | 1 | 1 | | 14 | |
| RCL | m8,i/cl | 11 | 8 | x | x | x | 1 | 1 | 1 | | 13 | |

Wolfdale

| | | | | | | | | | | | |
|--------------------------------------|------------|--------------|----|---|---|---|---|---|---|------------|------|
| RCR RCL | m,i/cl | 10 | 7 | x | x | x | 1 | 1 | 1 | 13 | |
| SHLD SHRD | r,r,i/cl | 2 | 2 | x | x | x | | | | 2 | 1 |
| SHLD SHRD | m,r,i/cl | 3 | 2 | x | x | x | 1 | 1 | 1 | 7 | |
| BT | r,r/i | 1 | 1 | x | x | x | | | | 1 | 1 |
| BT | m,r | 9 | 8 | x | x | x | 1 | | | | 4 |
| BT | m,i | 3 | 2 | x | x | x | 1 | | | | 1 |
| BTR BTS BTC | r,r/i | 1 | 1 | x | x | x | | | | 1 | |
| BTR BTS BTC | m,r | 10 | 7 | x | x | x | 1 | 1 | 1 | 5 | |
| BTR BTS BTC | m,i | 3 | 1 | x | x | x | 1 | 1 | 1 | 6 | |
| BSF BSR | r,r | 2 | 2 | x | 1 | x | | | | 2 | 1 |
| BSF BSR | r,m | 2 | 2 | x | 1 | x | 1 | | | | 1 |
| SETcc | r | 1 | 1 | x | x | x | | | | 1 | 1 |
| SETcc | m | 2 | 1 | x | x | x | | 1 | 1 | | 1 |
| CLC STC CMC | | 1 | 1 | x | x | x | | | | 1 | 0.33 |
| CLD | | 6 | 6 | x | x | x | | | | | 3 |
| STD | | 6 | 6 | x | x | x | | | | | 14 |
| Control transfer instructions | | | | | | | | | | | |
| JMP | short/near | 1 | 1 | | | | 1 | | | 0 | 1-2 |
| JMP i) | far | 30 | 30 | | | | | | | | 76 |
| JMP | r | 1 | 1 | | | | 1 | | | 0 | 1-2 |
| JMP | m(near) | 1 | 1 | | | | 1 | 1 | | 0 | 1-2 |
| JMP | m(far) | 31 | 29 | | | | | 2 | | | 68 |
| Conditional jump | short/near | 1 | 1 | | | | 1 | | | 0 | 1 |
| Fused compare/test and branch e,i) | | 1 | 1 | | | | 1 | | | 0 | 1 |
| J(E/R)CXZ | short | 2 | 2 | x | x | 1 | | | | | 1-2 |
| LOOP | short | 11 | 11 | x | x | x | | | | | 5 |
| LOOP(N)E | short | 11 | 11 | x | x | x | | | | | 5 |
| CALL | near | 3 | 2 | x | x | x | | | 1 | 1 | 2 |
| CALL i) | far | 43 | 43 | | | | | | | | 75 |
| CALL | r | 3 | 2 | | | | | | 1 | 1 | 2 |
| CALL | m(near) | 4 | 3 | | | | | 1 | 1 | 1 | 2 |
| CALL | m(far) | 44 | 42 | | | | | 2 | | | 75 |
| RETN | | 1 | 1 | | | | 1 | 1 | | | 2 |
| RETN | i | 3 | 1 | | | | 1 | 1 | | | 2 |
| RETF | | 32 | 30 | | | | | 2 | | | 78 |
| RETF | i | 32 | 30 | | | | | 2 | | | 78 |
| BOUND i) | r,m | 15 | 13 | | | | | 2 | | | 8 |
| INTO i) | | 5 | 5 | | | | | | | | 3 |
| String instructions | | | | | | | | | | | |
| LODS | | 3 | 2 | | | | 1 | | | | 1 |
| REP LODS | | 4+7n-14+6n | | | | | | | | 1+5n-21+3n | |
| STOS | | 4 | 2 | | | | | | 1 | 1 | 1 |
| REP STOS | | 8+5n-20+1.2n | | | | | | | | 7+2n-0.55n | |
| MOVS | | 8 | 5 | | | | | | | | |
| | | 1 | 1 | 1 | | 5 | | | | | |
| REP MOVS | | 7+7n-13+n | | | | | | | | 1+3n-0.63n | |
| SCAS | | 4 | 3 | | | | | 1 | | | 1 |
| REP(N)E SCAS | | 7+8n-17+7n | | | | | | | | 3+8n-23+6n | |
| CMPS | | 7 | 5 | | | | | 2 | | | 3 |
| REP(N)E CMPS | | 7+10n-7+9n | | | | | | | | 2+7n-22+5n | |

Wolfdale

| Other | | | | | | | | | | | | |
|------------------|-----|--------|----|---|---|---|---|---|---|--|--|--------|
| NOP (90) | | 1 | 1 | x | x | x | | | | | | 0.33 |
| Long NOP (0F 1F) | | 1 | 1 | x | x | x | | | | | | 1 |
| PAUSE | | 3 | 3 | x | x | x | | | | | | 8 |
| ENTER | i,0 | 12 | 10 | | | | | 1 | 1 | | | 8 |
| ENTER | a,b | | | | | | | | | | | |
| LEAVE | | 3 | 2 | | | | 1 | | | | | |
| CPUID | | 53-117 | | | | | | | | | | 53-211 |
| RDTSC | | 13 | | | | | | | | | | 32 |
| RDPMSR | | 23 | | | | | | | | | | 54 |

Notes:

- a) Applies to all addressing modes
- b) Has an implicit LOCK prefix.
- c) Low values are for small results, high values for high results. The reciprocal throughput is only slightly less than the latency.
- e) See manual 3: "The microarchitecture of Intel, AMD and VIA CPUs" for restrictions on macro-op fusion.
- i) Not available in 64 bit mode.

Floating point x87 instructions

| Instruction | Operands | µops fused do- main | µops unfused domain | | | | | | | | Unit | Laten- cy | Reci- procal through- put |
|-------------------------|----------|---------------------------|---------------------|----|----|----|----|----|----|---|-------|--------------|------------------------------------|
| | | | p015 | p0 | p1 | p5 | p2 | p3 | p4 | | | | |
| Move instructions | | | | | | | | | | | | | |
| FLD | r | 1 | 1 | 1 | | | | | | | float | 1 | 1 |
| FLD | m32/64 | 1 | 1 | | | | | 1 | | | float | 3 | 1 |
| FLD | m80 | 4 | 2 | 2 | | | | 2 | | | float | 4 | 3 |
| FBLD | m80 | 40 | 38 | x | x | x | 2 | | | | float | 45 | 20 |
| FST(P) | r | 1 | 1 | 1 | | | | | | | float | 1 | 1 |
| FST(P) | m32/m64 | 1 | | | | | | | 1 | 1 | float | 3 | 1 |
| FSTP | m80 | 7 | 3 | x | x | x | | 2 | 2 | | float | 4 | 5 |
| FBSTP | m80 | 171 | 167 | x | x | x | | 2 | 2 | | float | 164 | 166 |
| FXCH | r | 1 | 0 f) | | | | | | | | float | 0 | 1 |
| FILD | m | 1 | 1 | | 1 | | 1 | | | | float | 6 | 1 |
| FIST | m | 2 | 1 | | 1 | | | 1 | 1 | | float | 6 | 1 |
| FISTP | m | 3 | 1 | | 1 | | | 1 | 1 | | float | 6 | 1 |
| FISTTP g) | m | 3 | 1 | | 1 | | | 1 | 1 | | float | 6 | 1 |
| FLDZ | | 1 | 1 | 1 | | | | | | | float | | 1 |
| FLD1 | | 2 | 2 | 1 | 1 | | | | | | float | | 2 |
| FLDPI FLDL2E etc. | | 2 | 2 | | 2 | | | | | | float | | 2 |
| FCMOVcc | r | 2 | 2 | 2 | | | | | | | float | 2 | 2 |
| FNSTSW | AX | 1 | 1 | 1 | | | | | | | float | | 1 |
| FNSTSW | m16 | 2 | 1 | 1 | | | | 1 | 1 | | float | | 2 |
| FLDCW | m16 | 2 | 1 | | | | 1 | | | | float | | 10 |
| FNSTCW | m16 | 3 | 1 | | | 1 | | 1 | 1 | | float | | 8 |
| FINCSTP FDECSTP | | 1 | 1 | 1 | | | | | | | float | 1 | 1 |
| FFREE(P) | r | 2 | 2 | x | x | x | | | | | float | | 2 |
| FNSAVE | m | 141 | 95 | x | x | x | 7 | 23 | 23 | | float | | 142 |
| FRSTOR | m | 78 | 51 | x | x | x | 27 | | | | float | | 177 |
| Arithmetic instructions | | | | | | | | | | | | | |

Wolfdale

| | | | | | | | | | | | | |
|--------------------|---|--------|----|---|---|---|---|--|--|-------|---------|---------|
| FADD(P) FSUB(R)(P) | r | 1 | 1 | | 1 | | | | | float | 3 | 1 |
| FADD(P) FSUB(R)(P) | m | 1 | 1 | | 1 | | 1 | | | float | | 1 |
| FMUL(P) | r | 1 | 1 | 1 | | | | | | float | 5 | 2 |
| FMUL(P) | m | 1 | 1 | 1 | | | 1 | | | float | | 2 |
| FDIV(R)(P) | r | 1 | 1 | 1 | | | | | | float | 6-21 d) | 5-20 d) |
| FDIV(R)(P) | m | 1 | 1 | 1 | | | 1 | | | float | 6-21 d) | 5-20 d) |
| FABS | | 1 | 1 | 1 | | | | | | float | 1 | 1 |
| FCHS | | 1 | 1 | 1 | | | | | | float | 1 | 1 |
| FCOM(P) FUCOM | r | 1 | 1 | | 1 | | | | | float | | 1 |
| FCOM(P) FUCOM | m | 1 | 1 | | 1 | | 1 | | | float | | 1 |
| FCOMPP FUCOMPP | | 2 | 2 | 1 | 1 | | | | | float | | |
| FCOMI(P) FUCOMI(P) | r | 1 | 1 | | 1 | | | | | float | | 1 |
| FIADD FISUB(R) | m | 2 | 2 | | 2 | | 1 | | | float | 3 | 2 |
| FIMUL | m | 2 | 2 | 1 | 1 | | 1 | | | float | 5 | 2 |
| FIDIV(R) | m | 2 | 2 | 1 | 1 | | 1 | | | float | 6-21 | 5-20 d) |
| FICOM(P) | m | 2 | 2 | | 2 | | 1 | | | float | | 2 |
| FTST | | 1 | 1 | | 1 | | | | | float | | 1 |
| FXAM | | 1 | 1 | | 1 | | | | | float | | 1 |
| FPREM | | 26-29 | | | x | x | x | | | float | 13-40 | |
| FPREM1 | | 28-35 | | | x | x | x | | | float | 18-41 | |
| FRNDINT | | 17-19 | | | x | x | x | | | float | 10-22 | |
| Math | | | | | | | | | | | | |
| FSCALE | | 28 | 28 | | x | x | x | | | float | 43 | |
| FXTRACT | | 53-84 | | | x | x | x | | | float | ~170 | |
| FSQRT | | 1 | 1 | 1 | | | | | | float | 6-20 | |
| FSIN | | 18-85 | | | x | x | x | | | float | 32-85 | |
| FCOS | | 76-100 | | | x | x | x | | | float | 70-100 | |
| FSINCOS | | 18-105 | | | x | x | x | | | float | 38-107 | |
| F2XM1 | | 19 | 19 | | x | x | x | | | float | 45 | |
| FYL2X FYL2XP1 | | 57-65 | | | x | x | x | | | float | 50-100 | |
| FPTAN | | 19-100 | | | x | x | x | | | float | 40-130 | |
| FPATAN | | 23-87 | | | x | x | x | | | float | 55-130 | |
| Other | | | | | | | | | | | | |
| FNOP | | 1 | 1 | 1 | | | | | | float | | 1 |
| WAIT | | 2 | 2 | | x | x | x | | | float | | 1 |
| FNCLEX | | 4 | 4 | | | x | x | | | float | | 15 |
| FNINIT | | 15 | 15 | | x | x | x | | | float | | 63 |

Notes:

- d) Round divisors or low precision give low values.
- f) Resolved by register renaming. Generates no μ ops in the unfused domain.
- g) SSE3 instruction set.

Integer MMX and XMM instructions

| Instruction | Operands | μ ops fused domain | μ ops unfused domain | | | | | | | Unit | Latency | Reciprocal throughput |
|--------------------------|----------|------------------------|--------------------------|----|----|----|----|----|----|------|---------|-----------------------|
| | | | p015 | p0 | p1 | p5 | p2 | p3 | p4 | | | |
| Move instructions | | | | | | | | | | | | |
| MOVD k) | r,mm/x | 1 | 1 | x | x | x | | | | int | 2 | 0.33 |

Wolfdale

| | | | | | | | | | | | |
|-------------------|------------|---|---|---|---|---|---|---|-----|-----|------|
| MOVD k) | m,mm/x | 1 | | | | | 1 | 1 | | 3 | 1 |
| MOVD k) | mm/x,r | 1 | 1 | x | | x | | | int | 2 | 0.5 |
| MOVD k) | mm/x,m | 1 | | | | | 1 | | int | 2 | 1 |
| MOVQ | v,v | 1 | 1 | x | x | x | | | int | 1 | 0.33 |
| MOVQ | mm/x,m64 | 1 | | | | | 1 | | int | 2 | 1 |
| MOVQ | m64, mm/x | 1 | | | | | | 1 | 1 | 3 | 1 |
| MOVDQA | xmm, xmm | 1 | 1 | x | x | x | | | int | 1 | 0.33 |
| MOVDQA | xmm, m128 | 1 | | | | | 1 | | int | 2 | 1 |
| MOVDQA | m128, xmm | 1 | | | | | | 1 | 1 | 3 | 1 |
| MOVDQU | m128, xmm | 9 | 4 | x | x | x | 1 | 2 | 2 | 3-8 | 4 |
| MOVDQU | xmm, m128 | 4 | 2 | x | | x | 2 | | int | 2-8 | 2 |
| LDDQU g) | xmm, m128 | 4 | 2 | x | | x | 2 | | int | 2-8 | 2 |
| MOVDQ2Q | mm, xmm | 1 | 1 | x | x | x | | | int | 1 | 0.33 |
| MOVQ2DQ | xmm,mm | 1 | 1 | x | x | x | | | int | 1 | 0.33 |
| MOVNTQ | m64,mm | 1 | | | | | | 1 | 1 | | 2 |
| MOVNTDQ | m128,xmm | 1 | | | | | | 1 | 1 | | 2 |
| MOVNTDQA j) | xmm, m128 | 1 | | | | | 1 | | | 2 | 1 |
| PACKSSWB/DW | | | | | | | | | | | |
| PACKUSWB | mm,mm | 1 | 1 | 1 | | | | | int | 1 | 1 |
| PACKSSWB/DW | | | | | | | | | | | |
| PACKUSWB | mm,m64 | 1 | 1 | 1 | | | 1 | | int | | 1 |
| PACKSSWB/DW | | | | | | | | | | | |
| PACKUSWB | xmm,xmm | 1 | 1 | | | 1 | | | int | 1 | 1 |
| PACKSSWB/DW | | | | | | | | | | | |
| PACKUSWB | xmm,m128 | 1 | 1 | | | 1 | 1 | | int | | 1 |
| PACKUSDW j) | xmm,xmm | 1 | 1 | | | 1 | | | int | 1 | 1 |
| PACKUSDW j) | xmm,m | 1 | 1 | | | 1 | 1 | | int | | 1 |
| PUNPCKH/LBW/WD/DQ | mm,mm | 1 | 1 | 1 | | | | | int | 1 | 1 |
| PUNPCKH/LBW/WD/DQ | mm,m64 | 1 | 1 | 1 | | | 1 | | int | | 1 |
| PUNPCKH/LBW/WD/DQ | xmm,xmm | 1 | 1 | | | 1 | | | int | 1 | 1 |
| PUNPCKH/LBW/WD/DQ | xmm,m128 | 1 | 1 | | | 1 | 1 | | int | | 1 |
| PUNPCKH/LQDQ | xmm,xmm | 1 | 1 | | | 1 | | | int | 1 | 1 |
| PUNPCKH/LQDQ | xmm, m128 | 2 | 1 | | | 1 | 1 | | int | | 1 |
| PMOVSX/ZXBW j) | xmm,xmm | 1 | 1 | | | 1 | | | int | 1 | 1 |
| PMOVSX/ZXBW j) | xmm,m64 | 1 | 1 | | | 1 | 1 | | int | | 1 |
| PMOVSX/ZXBD j) | xmm,xmm | 1 | 1 | | | 1 | | | int | 1 | 1 |
| PMOVSX/ZXBD j) | xmm,m32 | 1 | 1 | | | 1 | 1 | | int | | 1 |
| PMOVSX/ZXBQ j) | xmm,xmm | 1 | 1 | | | 1 | | | int | 1 | 1 |
| PMOVSX/ZXBQ j) | xmm,m16 | 1 | 1 | | | 1 | 1 | | int | | 1 |
| PMOVSX/ZXWD j) | xmm,xmm | 1 | 1 | | | 1 | | | int | 1 | 1 |
| PMOVSX/ZXWD j) | xmm,m64 | 1 | 1 | | | 1 | 1 | | int | | 1 |
| PMOVSX/ZXWQ j) | xmm,xmm | 1 | 1 | | | 1 | | | int | 1 | 1 |
| PMOVSX/ZXWQ j) | xmm,m32 | 1 | 1 | | | 1 | 1 | | int | | 1 |
| PMOVSX/ZXDQ j) | xmm,xmm | 1 | 1 | | | 1 | | | int | 1 | 1 |
| PMOVSX/ZXDQ j) | xmm,m64 | 1 | 1 | | | 1 | 1 | | int | | 1 |
| PSHUFB h) | mm,mm | 1 | 1 | | | 1 | | | int | 1 | 1 |
| PSHUFB h) | mm,m64 | 2 | 1 | | | 1 | 1 | | int | | 1 |
| PSHUFB h) | xmm,xmm | 1 | 1 | | | 1 | | | int | 1 | 1 |
| PSHUFB h) | xmm,m128 | 1 | 1 | | | 1 | 1 | | int | | 1 |
| PSHUFW | mm,mm,i | 1 | 1 | | | 1 | | | int | 1 | 1 |
| PSHUFW | mm,m64,i | 2 | 1 | | | 1 | 1 | | int | | 1 |
| PSHUFD | xmm,xmm,i | 1 | 1 | | | 1 | | | int | 1 | 1 |
| PSHUFD | xmm,m128,i | 2 | 1 | | | 1 | 1 | | int | | 1 |
| PSHUFL/HW | xmm,xmm,i | 1 | 1 | | | 1 | | | int | 1 | 1 |

Wolfdale

| | | | | | | | | | | | | |
|--------------------------------|------------|----|---|---|---|---|---|---|---|-----|---|------|
| PSHUFL/HW | x, m128,i | 2 | 1 | | | 1 | 1 | | | int | | 1 |
| PALIGNR h) | mm,mm,i | 2 | 2 | | | 2 | | | | int | 2 | 1 |
| PALIGNR h) | mm,m64,i | 3 | 3 | | | 3 | 1 | | | int | | 1 |
| PALIGNR h) | xmm,xmm,i | 1 | 1 | | | 1 | | | | int | 1 | 1 |
| PALIGNR h) | xmm,m128,i | 1 | 1 | | | 1 | 1 | | | int | | 1 |
| PBLENDVB j) | x,x,xmm0 | 2 | 2 | | | 2 | | | | int | 2 | 2 |
| PBLENDVB j) | x,m,xmm0 | 2 | 2 | | | 2 | 1 | | | int | | 2 |
| PBLENDW j) | xmm,xmm,i | 1 | 1 | | | 1 | | | | int | 1 | 1 |
| PBLENDW j) | xmm,m,i | 1 | 1 | | | 1 | 1 | | | int | | 1 |
| MASKMOVQ | mm,mm | 4 | 1 | 1 | | | 1 | 1 | 1 | int | | 2-5 |
| MASKMOVDQU | xmm,xmm | 10 | 4 | 1 | | 3 | 2 | 2 | 3 | int | | 6-10 |
| PMOVMASKB | r32,mm/x | 1 | 1 | 1 | | | | | | int | 2 | 1 |
| PEXTRB j) | r32,xmm,i | 2 | 2 | x | x | x | | | | int | 3 | 1 |
| PEXTRB j) | m8,xmm,i | 2 | 2 | x | x | x | | | | int | 3 | 1 |
| PEXTRW | r32,mm/x,i | 2 | 2 | x | x | x | 1 | | | int | 3 | 1 |
| PEXTRW j) | m16,mm/x,i | 2 | 2 | ? | ? | 1 | | 1 | 1 | int | | 1 |
| PEXTRD j) | r32,xmm,i | 2 | 2 | x | x | x | | | | int | 3 | 1 |
| PEXTRD j) | m32,xmm,i | 2 | 1 | | | 1 | | 1 | 1 | int | | 1 |
| PEXTRQ j,m) | r64,xmm,i | 2 | 2 | x | x | x | | | | int | 3 | 1 |
| PEXTRQ j,m) | m64,xmm,i | 2 | 1 | | | 1 | | 1 | 1 | int | | 1 |
| PINSRB j) | xmm,r32,i | 1 | 1 | | | 1 | | | | int | 1 | 1 |
| PINSRB j) | xmm,m8,i | 2 | 1 | | | 1 | 1 | | | int | | 1 |
| PINSRW | mm/x,r32,i | 1 | 1 | | | 1 | | | | int | 2 | 1 |
| PINSRW | mm/x,m16,i | 2 | 1 | | | 1 | 1 | | | int | | 1 |
| PINSRD j) | xmm,r32,i | 1 | 1 | | | 1 | | | | int | 1 | 1 |
| PINSRD j) | xmm,m32,i | 2 | 1 | | | 1 | 1 | | | int | | 1 |
| PINSRQ j,m) | xmm,r64,i | 1 | 1 | | | 1 | | | | int | 1 | 1 |
| PINSRQ j,m) | xmm,m64,i | 2 | 1 | | | 1 | 1 | | | int | | 1 |
| Arithmetic instructions | | | | | | | | | | | | |
| PADD/SUB(U)(S)B/W/D | v,v | 1 | 1 | x | | x | | | | int | 1 | 0.5 |
| PADD/SUB(U)(S)B/W/D | mm/x,m | 1 | 1 | x | | x | 1 | | | int | | 1 |
| PADDQ PSUBQ | v,v | 2 | 2 | x | | x | | | | int | 2 | 1 |
| PADDQ PSUBQ | mm/x,m | 2 | 2 | x | | x | 1 | | | int | | 1 |
| PHADD(S)W | | | | | | | | | | | | |
| PHSUB(S)W h) | v,v | 3 | 3 | 1 | | 2 | | | | int | 3 | 2 |
| PHADD(S)W | | | | | | | | | | | | |
| PHSUB(S)W h) | mm/x,m64 | 4 | 3 | 1 | | 2 | 1 | | | int | | 2 |
| PHADDD PHSUBD h) | v,v | 3 | 3 | 1 | | 2 | | | | int | 3 | 2 |
| PHADDD PHSUBD h) | mm/x,m64 | 4 | 3 | 1 | | 2 | 1 | | | int | | 2 |
| PCMPEQ/GTB/W/D | v,v | 1 | 1 | x | | x | | | | int | 1 | 0.5 |
| PCMPEQ/GTB/W/D | mm/x,m | 1 | 1 | x | | x | 1 | | | int | | 1 |
| PCMPEQQ j) | xmm,xmm | 1 | 1 | | | 1 | | | | int | 1 | 1 |
| PCMPEQQ j) | xmm,m128 | 1 | 1 | | | 1 | 1 | | | int | | 1 |
| PMULL/HW PMULHUW | v,v | 1 | 1 | | 1 | | | | | int | 3 | 1 |
| PMULL/HW PMULHUW | mm/x,m | 1 | 1 | | 1 | | 1 | | | int | | 1 |
| PMULHRW h) | v,v | 1 | 1 | | 1 | | | | | int | 3 | 1 |
| PMULHRW h) | mm/x,m | 1 | 1 | | 1 | | 1 | | | int | | 1 |
| PMULLD j) | xmm,xmm | 4 | 4 | | 2 | 2 | | | | int | 5 | 2 |
| PMULLD j) | xmm,m128 | 6 | 5 | 1 | 2 | 2 | 1 | | | int | 5 | 4 |
| PMULDQ j) | xmm,xmm | 1 | 1 | | 1 | | | | | int | 3 | 1 |
| PMULDQ j) | xmm,m128 | 1 | 1 | | 1 | | 1 | | | int | | 1 |
| PMULUDQ | v,v | 1 | 1 | | 1 | | | | | int | 3 | 1 |

Wolfdale

| | | | | | | | | | | | | |
|---------------------------|-----------|----|----|---|---|---|---|--|--|-------|---|------|
| PMULUDQ | mm/x,m | 1 | 1 | | 1 | | 1 | | | int | | 1 |
| PMADDWD | v,v | 1 | 1 | | 1 | | | | | int | 3 | 1 |
| PMADDWD | mm/x,m | 1 | 1 | | 1 | | 1 | | | int | | 1 |
| PMADDUBSW h) | v,v | 1 | 1 | | 1 | | | | | int | 3 | 1 |
| PMADDUBSW h) | mm/x,m | 1 | 1 | | 1 | | 1 | | | int | | 1 |
| PAVGB/W | v,v | 1 | 1 | x | | x | | | | int | 1 | 0.5 |
| PAVGB/W | mm/x,m | 1 | 1 | x | | x | 1 | | | int | | 1 |
| PMIN/MAXSB j) | xmm,xmm | 1 | 1 | 1 | | | | | | int | 1 | 1 |
| PMIN/MAXSB j) | xmm,m128 | 1 | 1 | 1 | | | 1 | | | int | | 1 |
| PMIN/MAXUB | v,v | 1 | 1 | x | | x | | | | int | 1 | 0.5 |
| PMIN/MAXUB | mm/x,m | 1 | 1 | x | | x | 1 | | | int | | 1 |
| PMIN/MAXSW | v,v | 1 | 1 | x | | x | | | | int | 1 | 0.5 |
| PMIN/MAXSW | mm/x,m | 1 | 1 | x | | x | 1 | | | int | | 1 |
| PMIN/MAXUW j) | xmm,xmm | 1 | 1 | 1 | | | | | | int | 1 | 1 |
| PMIN/MAXUW j) | xmm,m | 1 | 1 | | | | 1 | | | int | | 1 |
| PMIN/MAXSD j) | xmm,xmm | 1 | 1 | 1 | | | | | | int | 1 | 1 |
| PMIN/MAXSD j) | xmm,m128 | 1 | 1 | 1 | | | 1 | | | int | | 1 |
| PMIN/MAXUD j) | xmm,xmm | 1 | 1 | 1 | | | | | | int | 1 | 1 |
| PMIN/MAXUD j) | xmm,m128 | 1 | 1 | 1 | | | 1 | | | int | | 1 |
| PHMINPOSUW j) | xmm,xmm | 4 | 4 | | | 4 | | | | int | 4 | 4 |
| PHMINPOSUW j) | xmm,m128 | 4 | 4 | | | 4 | 1 | | | int | | 4 |
| PABSB PABSW PABSD h) | v,v | 1 | 1 | x | | x | | | | int | 1 | 0.5 |
| PABSB PABSW PABSD h) | mm/x,m | 1 | 1 | x | | x | 1 | | | int | | 1 |
| PSIGNB PSIGNW PSIGND h) | v,v | 1 | 1 | x | | x | | | | int | 1 | 0.5 |
| PSIGNB PSIGNW PSIGND h) | mm/x,m | 1 | 1 | x | | x | 1 | | | int | | 1 |
| PSADBW | v,v | 1 | 1 | | 1 | | | | | int | 3 | 1 |
| PSADBW | mm/x,m | 1 | 1 | | 1 | | 1 | | | int | | 1 |
| MPSADBW j) | xmm,xmm,i | 3 | 3 | | 1 | 2 | | | | int | 5 | 2 |
| MPSADBW j) | xmm,m,i | 4 | 3 | | 1 | 2 | 1 | | | int | | 2 |
| Logic instructions | | | | | | | | | | | | |
| PAND(N) POR PXOR | v,v | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| PAND(N) POR PXOR | mm/x,m | 1 | 1 | x | x | x | 1 | | | int | | 1 |
| PTEST j) | xmm,xmm | 2 | 2 | 1 | x | x | | | | int | 1 | 1 |
| PTEST j) | xmm,m128 | 2 | 2 | 1 | x | x | 1 | | | int | | 1 |
| PSLL/RL/RAW/D/Q | mm,mm/i | 1 | 1 | 1 | | | | | | int | 1 | 1 |
| PSLL/RL/RAW/D/Q | mm,m64 | 1 | 1 | 1 | | | 1 | | | int | | 1 |
| PSLL/RL/RAW/D/Q | xmm,i | 1 | 1 | 1 | | | | | | int | 1 | 1 |
| PSLL/RL/RAW/D/Q | xmm,xmm | 2 | 2 | x | | x | | | | int | 2 | 1 |
| PSLL/RL/RAW/D/Q | xmm,m128 | 3 | 2 | x | | x | 1 | | | int | | 1 |
| PSLL/RDQ | xmm,i | 1 | 1 | x | | x | | | | int | 1 | 1 |
| Other | | | | | | | | | | | | |
| EMMS | | 11 | 11 | x | x | x | | | | float | | 6 |

Notes:

- g) SSE3 instruction set.
- h) Supplementary SSE3 instruction set.
- j) SSE4.1 instruction set
- k) MASM uses the name MOVD rather than MOVQ for this instruction even when moving 64 bits
- m) Only available in 64 bit mode

Floating point XMM instructions

| Instruction | Operands | μops fused do-main | μops unfused domain | | | | | | | Unit | Laten-cy | Reci-procal through-put |
|-------------------|------------|--------------------|---------------------|----|----|----|----|----|----|-------|----------|-------------------------|
| | | | p015 | p0 | p1 | p5 | p2 | p3 | p4 | | | |
| Move instructions | | | | | | | | | | | | |
| MOVAPS/D | xmm,xmm | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| MOVAPS/D | xmm,m128 | 1 | | | | | 1 | | | int | 2 | 1 |
| MOVAPS/D | m128,xmm | 1 | | | | | | 1 | 1 | | 3 | 1 |
| MOVUPS/D | xmm,m128 | 4 | 2 | 1 | | 1 | 2 | | | int | 2-4 | 2 |
| MOVUPS/D | m128,xmm | 9 | 4 | x | x | x | 1 | 2 | 2 | | 3-4 | 4 |
| MOVSS/D | xmm,xmm | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| MOVSS/D | x,m32/64 | 1 | | | | | 1 | | | int | 2 | 1 |
| MOVSS/D | m32/64,x | 1 | | | | | | 1 | 1 | | 3 | 1 |
| MOVHPS/D MOVLPS/D | xmm,m64 | 2 | 1 | | | 1 | 1 | | | int | 3 | 1 |
| MOVHPS/D | m64,xmm | 2 | 1 | 1 | | | | 1 | 1 | | 5 | 1 |
| MOVLPS/D | m64,xmm | 1 | | | | | | 1 | 1 | | 3 | 1 |
| MOVLHPS MOVHLPS | xmm,xmm | 1 | 1 | 1 | | | | | | float | 1 | 1 |
| MOVMSKPS/D | r32,xmm | 1 | 1 | 1 | | | | | | float | 1 | 1 |
| MOVNTPS/D | m128,xmm | 1 | | | | | | 1 | 1 | | | 2-3 |
| SHUFPS | xmm,xmm,i | 1 | 1 | | | 1 | | | | int | 1 | 1 |
| SHUFPS | xmm,m128,i | 2 | 1 | | | 1 | 1 | | | int | | 1 |
| SHUFPD | xmm,xmm,i | 1 | 1 | 1 | | | | | | float | 1 | 1 |
| SHUFPD | xmm,m128,i | 2 | 1 | 1 | | | 1 | | | float | | 1 |
| BLENDPS/PD j) | xmm,xmm,i | 1 | 1 | | | 1 | | | | int | 1 | 1 |
| BLENDPS/PD j) | xmm,m128,i | 1 | 1 | | | 1 | 1 | | | int | | 1 |
| BLENDVPS/PD j) | x,x,xmm0 | 2 | 2 | | | 2 | | | | int | 2 | 2 |
| BLENDVPS/PD j) | x,m,xmm0 | 2 | 2 | | | 2 | 1 | | | int | | 2 |
| MOVDDUP g) | xmm,xmm | 1 | 1 | 1 | | | | | | int | 1 | 1 |
| MOVDDUP g) | xmm,m64 | 2 | 1 | 1 | | | 1 | | | int | | 1 |
| MOVSH/LDUP g) | xmm,xmm | 1 | 1 | | | 1 | | | | int | 1 | 1 |
| MOVSH/LDUP g) | xmm,m128 | 2 | 1 | | | 1 | 1 | | | int | | 1 |
| UNPCKH/LPS | xmm,xmm | 1 | 1 | | | 1 | | | | int | 1 | 1 |
| UNPCKH/LPS | xmm,m128 | 1 | 1 | | | 1 | 1 | | | int | | 1 |
| UNPCKH/LPD | xmm,xmm | 1 | 1 | 1 | | | | | | float | 1 | 1 |
| UNPCKH/LPD | xmm,m128 | 2 | 1 | 1 | | | 1 | | | float | | 1 |
| EXTRACTPS j) | r32,xmm,i | 2 | 2 | x | x | x | | | | int | 4 | 1 |
| EXTRACTPS j) | m32,xmm,i | 2 | 1 | | | 1 | | 1 | 1 | int | | 1 |
| INSERTPS j) | xmm,xmm,i | 1 | 1 | | | 1 | | | | int | 1 | 1 |
| INSERTPS j) | xmm,m32,i | 2 | 1 | | | 1 | 1 | | | int | | 1 |
| Conversion | | | | | | | | | | | | |
| CVTPD2PS | xmm,xmm | 2 | 2 | 1 | 1 | | | | | float | 4 | 1 |
| CVTPD2PS | xmm,m128 | 2 | 2 | 1 | 1 | | 1 | | | float | | 1 |
| CVTSD2SS | xmm,xmm | 2 | 2 | 1 | 1 | | | | | float | 4 | 1 |
| CVTSD2SS | xmm,m64 | 2 | 2 | 1 | 1 | | 1 | | | float | | 1 |
| CVTPS2PD | xmm,xmm | 2 | 2 | 2 | | | | | | float | 2 | 2 |
| CVTPS2PD | xmm,m64 | 2 | 2 | 2 | | | 1 | | | float | | 2 |
| CVTSS2SD | xmm,xmm | 2 | 2 | 2 | | | | | | float | 2 | 2 |
| CVTSS2SD | xmm,m32 | 2 | 2 | 2 | | | 1 | | | float | | 2 |
| CVTDQ2PS | xmm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 1 |

Wolfdale

| | | | | | | | | | | |
|-------------------|----------|---|---|---|---|---|---|-------|---------|---------|
| CVTDQ2PS | xmm,m128 | 1 | 1 | | 1 | | | float | | 1 |
| CVT(T) PS2DQ | xmm,xmm | 1 | 1 | | 1 | | | float | 3 | 1 |
| CVT(T) PS2DQ | xmm,m128 | 1 | 1 | | 1 | | 1 | float | | 1 |
| CVTDQ2PD | xmm,xmm | 2 | 2 | 1 | 1 | | | float | 4 | 1 |
| CVTDQ2PD | xmm,m64 | 2 | 2 | 1 | 1 | | 1 | float | | 1 |
| CVT(T)PD2DQ | xmm,xmm | 2 | 2 | 1 | 1 | | | float | 4 | 1 |
| CVT(T)PD2DQ | xmm,m128 | 2 | 2 | 1 | 1 | | 1 | float | | 1 |
| CVTPI2PS | xmm,mm | 1 | 1 | | 1 | | | float | 3 | 3 |
| CVTPI2PS | xmm,m64 | 1 | 1 | | 1 | | 1 | float | | 3 |
| CVT(T)PS2PI | mm,xmm | 1 | 1 | | 1 | | | float | 3 | 1 |
| CVT(T)PS2PI | mm,m128 | 1 | 1 | | 1 | | 1 | float | | 1 |
| CVTPI2PD | xmm,mm | 2 | 2 | 1 | 1 | | | float | 4 | 1 |
| CVTPI2PD | xmm,m64 | 2 | 2 | 1 | 1 | | 1 | float | | 1 |
| CVT(T) PD2PI | mm,xmm | 2 | 2 | 1 | 1 | | | float | 4 | 1 |
| CVT(T) PD2PI | mm,m128 | 2 | 2 | 1 | 1 | | 1 | float | | 1 |
| CVTSI2SS | xmm,r32 | 1 | 1 | | 1 | | | float | 4 | 3 |
| CVTSI2SS | xmm,m32 | 1 | 1 | | 1 | | 1 | float | | 3 |
| CVT(T)SS2SI | r32,xmm | 1 | 1 | | 1 | | | float | 3 | 1 |
| CVT(T)SS2SI | r32,m32 | 1 | 1 | | 1 | | 1 | float | | 1 |
| CVTSI2SD | xmm,r32 | 2 | 2 | 1 | 1 | | | float | 4 | 3 |
| CVTSI2SD | xmm,m32 | 2 | 1 | | 1 | | 1 | float | | 3 |
| CVT(T)SD2SI | r32,xmm | 1 | 1 | | 1 | | | float | 3 | 1 |
| CVT(T)SD2SI | r32,m64 | 1 | 1 | | 1 | | 1 | float | | 1 |
| Arithmetic | | | | | | | | | | |
| ADDSS/D SUBSS/D | xmm,xmm | 1 | 1 | | 1 | | | float | 3 | 1 |
| ADDSS/D SUBSS/D | x,m32/64 | 1 | 1 | | 1 | | 1 | float | | 1 |
| ADDPSS/D SUBPS/D | xmm,xmm | 1 | 1 | | 1 | | | float | 3 | 1 |
| ADDPSS/D SUBPS/D | xmm,m128 | 1 | 1 | | 1 | | 1 | float | | 1 |
| ADDSUBPS/D g) | xmm,xmm | 1 | 1 | | 1 | | | float | 3 | 1 |
| ADDSUBPS/D g) | xmm,m128 | 1 | 1 | | 1 | | 1 | float | | 1 |
| HADDPS HSUBPS g) | xmm,xmm | 3 | 3 | | 1 | 2 | | float | 7 | 3 |
| HADDPS HSUBPS g) | xmm,m128 | 4 | 3 | | 1 | 2 | 1 | float | | 3 |
| HADDPD HSUBPD g) | xmm,xmm | 3 | 3 | x | x | x | | float | 6 | 1.5 |
| HADDPD HSUBPD g) | xmm,m128 | 4 | 3 | x | x | x | 1 | float | | 1.5 |
| MULSS | xmm,xmm | 1 | 1 | 1 | | | | float | 4 | 1 |
| MULSS | xmm,m32 | 1 | 1 | 1 | | | 1 | float | | 1 |
| MULSD | xmm,xmm | 1 | 1 | 1 | | | | float | 5 | 1 |
| MULSD | xmm,m64 | 1 | 1 | 1 | | | 1 | float | | 1 |
| MULPS | xmm,xmm | 1 | 1 | 1 | | | | float | 4 | 1 |
| MULPS | xmm,m128 | 1 | 1 | 1 | | | 1 | float | | 1 |
| MULPD | xmm,xmm | 1 | 1 | 1 | | | | float | 5 | 1 |
| MULPD | xmm,m128 | 1 | 1 | 1 | | | 1 | float | | 1 |
| DIVSS | xmm,xmm | 1 | 1 | 1 | | | | float | 6-13 d) | 5-12 d) |
| DIVSS | xmm,m32 | 1 | 1 | 1 | | | 1 | float | | 5-12 d) |
| DIVSD | xmm,xmm | 1 | 1 | 1 | | | | float | 6-21 d) | 5-20 d) |
| DIVSD | xmm,m64 | 1 | 1 | 1 | | | 1 | float | | 5-20 d) |
| DIVPS | xmm,xmm | 1 | 1 | 1 | | | | float | 6-13 d) | 5-12 d) |
| DIVPS | xmm,m128 | 1 | 1 | 1 | | | 1 | float | | 5-12 d) |
| DIVPD | xmm,xmm | 1 | 1 | 1 | | | | float | 6-21 d) | 5-20 d) |
| DIVPD | xmm,m128 | 1 | 1 | 1 | | | 1 | float | | 5-20 d) |
| RCPSS/PS | xmm,xmm | 1 | 1 | | 1 | | | float | 3 | 2 |
| RCPSS/PS | xmm,m | 1 | 1 | | 1 | | 1 | float | | 2 |

Wolfdale

| | | | | | | | | | | | | |
|---------------------|------------|-----|----|---|---|---|----|----|----|-------|------|------|
| CMPccSS/D | xmm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 1 |
| CMPccSS/D | x,m32/64 | 1 | 1 | | 1 | | 1 | | | float | | 1 |
| CMPccPS/D | xmm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 1 |
| CMPccPS/D | xmm,m128 | 1 | 1 | | 1 | | 1 | | | float | | 1 |
| COMISS/D UCOMISS/D | xmm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 1 |
| COMISS/D UCOMISS/D | x,m32/64 | 1 | 1 | | 1 | | 1 | | | float | | 1 |
| MAXSS/D MINSS/D | xmm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 1 |
| MAXSS/D MINSS/D | x,m32/64 | 1 | 1 | | 1 | | 1 | | | float | | 1 |
| MAXPS/D MINPS/D | xmm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 1 |
| MAXPS/D MINPS/D | xmm,m128 | 1 | 1 | | 1 | | 1 | | | float | | 1 |
| ROUNDSS/D j) | xmm,xmm,i | 1 | 1 | | 1 | | | | | float | 3 | 1 |
| ROUNDSS/D j) | xmm,m128,i | 1 | 1 | | 1 | | 1 | | | float | | 1 |
| ROUNDPS/D j) | xmm,xmm,i | 1 | 1 | | 1 | | | | | float | 3 | 1 |
| ROUNDPS/D j) | xmm,m128,i | 1 | 1 | | 1 | | 1 | | | float | | 1 |
| DPPS j) | xmm,xmm,i | 4 | 4 | 2 | 2 | | | | | float | 11 | 3 |
| DPPS j) | xmm,m128,i | 4 | 4 | 2 | 2 | | 1 | | | float | | 3 |
| DPPD j) | xmm,xmm,i | 4 | 4 | x | x | x | | | | float | 9 | 3 |
| DPPD j) | xmm,m128,i | 4 | 4 | x | x | x | 1 | | | float | | 3 |
| Math | | | | | | | | | | | | |
| SQRTSS/PS | xmm,xmm | 1 | 1 | 1 | | | | | | float | 6-13 | 5-12 |
| SQRTSS/PS | xmm,m | 2 | 1 | 1 | | | 1 | | | float | | 5-12 |
| SQRTSD/PD | xmm,xmm | 1 | 1 | 1 | | | | | | float | 6-20 | 5-19 |
| SQRTSD/PD | xmm,m | 2 | 1 | 1 | | | 1 | | | float | | 5-19 |
| RSQRTSS/PS | xmm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 2 |
| RSQRTSS/PS | xmm,m | 1 | 1 | | 1 | | 1 | | | float | | 2 |
| Logic | | | | | | | | | | | | |
| AND/ANDN/OR/XORPS/D | xmm,xmm | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| AND/ANDN/OR/XORPS/D | xmm,m128 | 1 | 1 | x | x | x | 1 | | | int | | 1 |
| Other | | | | | | | | | | | | |
| LDMXCSR | m32 | 13 | 12 | x | x | x | 1 | | | | | 38 |
| STMXCSR | m32 | 10 | 8 | x | x | x | | 1 | 1 | | | 20 |
| FXSAVE | m4096 | 151 | 67 | x | x | x | 8 | 38 | 38 | | | 145 |
| FXRSTOR | m4096 | 121 | 74 | x | x | x | 47 | | | | | 150 |

Notes:

- d) Round divisors give low values.
g) SSE3 instruction set.

Intel Nehalem

List of instruction timings and μ op breakdown

Explanation of column headings:

| | |
|--|---|
| Operands: | i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, mm/x = mmx or xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc. |
| μops fused domain: | The number of μ ops at the decode, rename, allocate and retirement stages in the pipeline. Fused μ ops count as one. |
| μops unfused domain: | The number of μ ops for each execution port. Fused μ ops count as two. Fused macro-ops count as one. The instruction has μ op fusion if the sum of the numbers listed under p015 + p2 + p3 + p4 exceeds the number listed under μ ops fused domain. An x under p0, p1 or p5 means that at least one of the μ ops listed under p015 can optionally go to this port. For example, a 1 under p015 and an x under p0 and p5 means one μ op which can go to either port 0 or port 5, whichever is vacant first. A value listed under p015 but nothing under p0, p1 and p5 means that it is not known which of the three ports these μ ops go to. |
| p015: | The total number of μ ops going to port 0, 1 and 5. |
| p0: | The number of μ ops going to port 0 (execution units). |
| p1: | The number of μ ops going to port 1 (execution units). |
| p5: | The number of μ ops going to port 5 (execution units). |
| p2: | The number of μ ops going to port 2 (memory read). |
| p3: | The number of μ ops going to port 3 (memory write address). |
| p4: | The number of μ ops going to port 4 (memory write data). |
| Domain: | Tells which execution unit domain is used: "int" = integer unit (general purpose registers), "ivec" = integer vector unit (SIMD), "fp" = floating point unit (XMM and x87 floating point). An additional "bypass delay" is generated if a register written by a μ op in one domain is read by a μ op in another domain. The bypass delay is 1 clock cycle between the "int" and "ivec" units, and 2 clock cycles between the "int" and "fp", and between the "ivec" and "fp" units. |

The bypass delay is indicated under latency only where it is unavoidable because either the source operand or the destination operand is in an unnatural domain such as a general purpose register (e.g. eax) in the "ivec" domain. For example, the PEXTRW instruction executes in the "int" domain. The source operand is an xmm register and the destination operand is a general purpose register. The latency for this instruction is indicated as 2+1, where 2 is the latency of the instruction itself and 1 is the bypass delay, assuming that the xmm operand is most likely to come from the "ivec" domain. If the xmm operand comes from the "fp" domain then the bypass delay will be 2 rather than one. The flags register can also have a bypass delay. For example, the COMISS instruction (floating point compare) executes in the "fp" domain and returns the result in the integer flags. Almost all instructions that read these flags execute in the "int" domain. Here the latency is indicated as 1+2, where 1 is the latency of the instruction itself and 2 is the bypass delay from the "fp" domain to the "int" domain.

The bypass delay from the memory read unit to any other unit and from any unit to the memory write unit are included in the latency figures in the table. Where the domain is not listed, the bypass delays are either unlikely to occur or unavoidable and therefore included in the latency figure.

Nehalem

Latency: This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay. The time unit used is core clock cycles, not the reference clock cycles given by the time stamp counter.

Reciprocal throughput: The average number of core clock cycles per instruction for a series of independent instructions of the same kind in the same thread.

Integer instructions

| Instruction | Operands | μops fused do-main | μops unfused domain | | | | | | | Do-main | Laten-cy | Reci-procal through-put |
|---------------------|----------|--------------------|---------------------|----|----|----|----|----|----|---------|----------|-------------------------|
| | | | p015 | p0 | p1 | p5 | p2 | p3 | p4 | | | |
| Move instructions | | | | | | | | | | | | |
| MOV | r,r/i | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| MOV a) | r,m | 1 | | | | | 1 | | | int | 2 | 1 |
| MOV a) | m,r | 1 | | | | | | 1 | 1 | int | 3 | 1 |
| MOV | m,i | 1 | | | | | | 1 | 1 | int | 3 | 1 |
| MOV | r,sr | 1 | | | | | 1 | | | int | | 1 |
| MOV | m,sr | 2 | | | | | 1 | 1 | 1 | int | | 1 |
| MOV | sr,r | 6 | 3 | x | x | x | 3 | | | int | | 13 |
| MOV | sr,m | 6 | 2 | x | | x | 4 | | | int | | 14 |
| MOVNTI | m,r | 2 | | | | | | 1 | 1 | int | ~270 | 1 |
| MOVSX MOVZX | | | | | | | | | | | | |
| MOVSXD | r,r | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| MOVSX MOVZX | | | | | | | | | | | | |
| MOVSXD | r,m | 1 | | | | | 1 | | | int | | 1 |
| CMOVcc | r,r | 2 | 2 | x | x | x | | | | int | 2 | 1 |
| CMOVcc | r,m | 2 | 2 | x | x | x | 1 | | | int | | |
| XCHG | r,r | 3 | 3 | x | x | x | | | | int | 2 | 2 |
| XCHG | r,m | 7 | x | | | | 1 | 1 | 1 | int | 20 b) | |
| XLAT | | 2 | 1 | | | | 1 | | | int | 5 | 1 |
| PUSH | r | 1 | | | | | | 1 | 1 | int | 3 | 1 |
| PUSH | i | 1 | | | | | | 1 | 1 | int | | 1 |
| PUSH | m | 2 | | | | | 1 | 1 | 1 | int | | 1 |
| PUSH | sr | 2 | 1 | | | | | 1 | 1 | int | | 1 |
| PUSHF(D/Q) | | 3 | 2 | x | x | x | | 1 | 1 | int | | 1 |
| PUSHA(D) i) | | 18 | 2 | x | 1 | x | | 8 | 8 | int | | 8 |
| POP | r | 1 | | | | | 1 | | | int | 2 | 1 |
| POP | (E/R)SP | 3 | 2 | x | 1 | x | 1 | | | int | | 5 |
| POP | m | 2 | | | | | 1 | 1 | 1 | int | | 1 |
| POP | sr | 7 | 2 | | | | 5 | | | int | | 15 |
| POPF(D/Q) | | 8 | 7 | x | x | x | 1 | | | int | | 14 |
| POPA(D) i) | | 10 | 2 | | | | 8 | | | int | | 8 |
| LAHF SAHF | | 1 | 1 | x | x | x | | | | int | 1 | 0.33 |
| SALC i) | | 2 | 2 | x | x | x | | | | int | 4 | 1 |
| LEA a) | r,m | 1 | 1 | | 1 | | | | | int | 1 | 1 |
| BSWAP | r32 | 1 | 1 | | 1 | | | | | int | 1 | 1 |
| BSWAP | r64 | 1 | 1 | | 1 | | | | | int | 3 | 1 |
| LDS LES LFS LGS LSS | m | 9 | 3 | x | x | x | 6 | | | int | | 15 |
| PREFETCHNTA | m | 1 | | | | | 1 | | | int | | 1 |

Nehalem

| | | | | | | | | | | | | | |
|--------------------------------|-----------|-----|---|---|---|---|---|---|---|--|-----|--------|-------|
| PREFETCH0/1/2 | m | 1 | | | | | 1 | | | | int | | 1 |
| LFENCE | | 2 | | | | | | 1 | 1 | | int | | 9 |
| MFENCE | | 3 | 1 | x | x | x | | 1 | 1 | | int | | 23 |
| SFENCE | | 2 | | | | | | 1 | 1 | | int | | 5 |
| Arithmetic instructions | | | | | | | | | | | | | |
| ADD SUB | r,r/i | 1 | 1 | x | x | x | | | | | int | 1 | 0.33 |
| ADD SUB | r,m | 1 | 1 | x | x | x | 1 | | | | int | | 1 |
| ADD SUB | m,r/i | 2 | 1 | x | x | x | 1 | 1 | 1 | | int | 6 | 1 |
| ADC SBB | r,r/i | 2 | 2 | x | x | x | | | | | int | 2 | 2 |
| ADC SBB | r,m | 2 | 2 | x | x | x | 1 | | | | int | 2 | 2 |
| ADC SBB | m,r/i | 4 | 3 | x | x | x | 1 | 1 | 1 | | int | 7 | |
| CMP | r,r/i | 1 | 1 | x | x | x | | | | | int | 1 | 0.33 |
| CMP | m,r/i | 1 | 1 | x | x | x | 1 | | | | int | 1 | 1 |
| INC DEC NEG NOT | r | 1 | 1 | x | x | x | | | | | int | 1 | 0.33 |
| INC DEC NEG NOT | m | 3 | 1 | x | x | x | 1 | 1 | 1 | | int | 6 | 1 |
| AAA AAS DAA DAS i) | | 1 | 1 | | 1 | | | | | | int | 3 | 1 |
| AAD i) | | 3 | 3 | x | x | x | | | | | int | 15 | 2 |
| AAM i) | | 5 | 5 | x | x | x | | | | | int | 20 | 7 |
| MUL IMUL | r8 | 1 | 1 | | 1 | | | | | | int | 3 | 1 |
| MUL IMUL | r16 | 3 | 3 | x | x | x | | | | | int | 5 | 2 |
| MUL IMUL | r32 | 3 | 3 | x | x | x | | | | | int | 5 | 2 |
| MUL IMUL | r64 | 3 | 3 | x | x | x | | | | | int | 3 | 2 |
| IMUL | r16,r16 | 1 | 1 | | 1 | | | | | | int | 3 | 1 |
| IMUL | r32,r32 | 1 | 1 | | 1 | | | | | | int | 3 | 1 |
| IMUL | r64,r64 | 1 | 1 | 1 | | | | | | | int | 3 | 1 |
| IMUL | r16,r16,i | 1 | 1 | | 1 | | | | | | int | 3 | 1 |
| IMUL | r32,r32,i | 1 | 1 | | 1 | | | | | | int | 3 | 1 |
| IMUL | r64,r64,i | 1 | 1 | 1 | | | | | | | int | 3 | 2 |
| MUL IMUL | m8 | 1 | 1 | | 1 | | 1 | | | | int | 3 | 1 |
| MUL IMUL | m16 | 3 | 3 | x | x | x | 1 | | | | int | 5 | 2 |
| MUL IMUL | m32 | 3 | 3 | x | x | x | 1 | | | | int | 5 | 2 |
| MUL IMUL | m64 | 3 | 2 | 2 | | | 1 | | | | int | 3 | 2 |
| IMUL | r16,m16 | 1 | 1 | | 1 | | 1 | | | | int | 3 | 1 |
| IMUL | r32,m32 | 1 | 1 | | 1 | | 1 | | | | int | 3 | 1 |
| IMUL | r64,m64 | 1 | 1 | 1 | | | 1 | | | | int | 3 | 1 |
| IMUL | r16,m16,i | 1 | 1 | | 1 | | 1 | | | | int | | 1 |
| IMUL | r32,m32,i | 1 | 1 | | 1 | | 1 | | | | int | | 1 |
| IMUL | r64,m64,i | 1 | 1 | 1 | | | 1 | | | | int | | 1 |
| DIV c) | r8 | 4 | 4 | 1 | 2 | 1 | | | | | int | 11-21 | 7-11 |
| DIV c) | r16 | 6 | 6 | x | 4 | x | | | | | int | 17-22 | 7-12 |
| DIV c) | r32 | 6 | 6 | x | 3 | x | | | | | int | 17-28 | 7-17 |
| DIV c) | r64 | ~40 | x | x | x | x | | | | | int | 28-90 | 19-69 |
| IDIV c) | r8 | 4 | 4 | 1 | 2 | 1 | | | | | int | 10-22 | 7-11 |
| IDIV c) | r16 | 8 | 8 | x | 5 | x | | | | | int | 18-23 | 7-12 |
| IDIV c) | r32 | 7 | 7 | x | 3 | x | | | | | int | 17-28 | 7-17 |
| IDIV c) | r64 | ~60 | x | x | x | x | | | | | int | 37-100 | 26-86 |
| CBW CWDE CDQE | | 1 | 1 | x | x | x | | | | | int | 1 | 1 |
| CWD CDQ CQO | | 1 | 1 | x | | x | | | | | int | 1 | 1 |
| POPCNT ℓ) | r,r | 1 | 1 | | 1 | | | | | | int | 3 | 1 |
| POPCNT ℓ) | r,m | 1 | 1 | | 1 | | 1 | | | | int | | 1 |
| CRC32 ℓ) | r,r | 1 | 1 | | 1 | | | | | | int | 3 | 1 |
| CRC32 ℓ) | r,m | 1 | 1 | | 1 | | 1 | | | | int | | 1 |

Nehalem

| Logic instructions | | | | | | | | | | | | | |
|----------------------------------|----------------|----|----|---|---|---|----|---|---|--|-----|-------|-------|
| AND OR XOR | r,r/i | 1 | 1 | x | x | x | | | | | int | 1 | 0.33 |
| AND OR XOR | r,m | 1 | 1 | x | x | x | 1 | | | | int | | 1 |
| AND OR XOR | m,r/i | 2 | 1 | x | x | x | 1 | 1 | 1 | | int | 6 | 1 |
| TEST | r,r/i | 1 | 1 | x | x | x | | | | | int | 1 | 0.33 |
| TEST | m,r/i | 1 | 1 | x | x | x | 1 | | | | int | | 1 |
| SHR SHL SAR | r,i/cl | 1 | 1 | x | | x | | | | | int | 1 | 0.5 |
| SHR SHL SAR | m,i/cl | 3 | 2 | x | | x | 1 | 1 | 1 | | int | 6 | 1 |
| ROR ROL | r,i/cl | 1 | 1 | x | | x | | | | | int | 1 | 1 |
| ROR ROL | m,i/cl | 3 | 2 | x | | x | 1 | 1 | 1 | | int | 6 | 1 |
| RCR RCL | r,1 | 2 | 2 | x | x | x | | | | | int | 2 | 2 |
| RCR | r8,i/cl | 9 | 9 | x | x | x | | | | | int | 13 | |
| RCL | r8,i/cl | 8 | 8 | x | x | x | | | | | int | 11 | |
| RCR RCL | r16/32/64,i/cl | 6 | 6 | x | x | x | | | | | int | 12-13 | 12-13 |
| RCR RCL | m,1 | 4 | 3 | x | x | x | 1 | 1 | 1 | | int | 7 | |
| RCR | m8,i/cl | 12 | 9 | x | x | x | 1 | 1 | 1 | | int | 16 | |
| RCL | m8,i/cl | 11 | 8 | x | x | x | 1 | 1 | 1 | | int | 14 | |
| RCR RCL | m16/32/64,i/cl | 10 | 7 | x | x | x | 1 | 1 | 1 | | int | 15 | |
| SHLD | r,r,i/cl | 2 | 2 | x | x | x | | | | | int | 3 | 1 |
| SHLD | m,r,i/cl | 3 | 2 | x | x | x | 1 | 1 | 1 | | int | 8 | |
| SHRD | r,r,i/cl | 2 | 2 | x | x | x | | | | | int | 4 | 1 |
| SHRD | m,r,i/cl | 3 | 2 | x | x | x | 1 | 1 | 1 | | int | 9 | |
| BT | r,r/i | 1 | 1 | x | | x | | | | | int | 1 | 1 |
| BT | m,r | 9 | 8 | x | | x | 1 | | | | int | | 5 |
| BT | m,i | 2 | 2 | x | | x | 1 | | | | int | | 1 |
| BTR BTS BTC | r,r/i | 1 | 1 | x | | x | | | | | int | 1 | 1 |
| BTR BTS BTC | m,r | 10 | 7 | x | x | x | 1 | 1 | 1 | | int | 6 | |
| BTR BTS BTC | m,i | 3 | 3 | x | | x | 1 | 1 | 1 | | int | 6 | |
| BSF BSR | r,r | 1 | 1 | | 1 | | | | | | int | 3 | 1 |
| BSF BSR | r,m | 2 | 1 | | 1 | | 1 | | | | int | 3 | 1 |
| SETcc | r | 1 | 1 | x | | x | | | | | int | 1 | 1 |
| SETcc | m | 2 | 1 | x | x | x | | 1 | 1 | | int | | 1 |
| CLC STC CMC | | 1 | 1 | x | x | x | | | | | int | 1 | 0.33 |
| CLD | | 2 | 2 | x | x | x | | | | | int | | 4 |
| STD | | 2 | 2 | x | x | x | | | | | int | | 5 |
| Control transfer instructions | | | | | | | | | | | | | |
| JMP | short/near | 1 | 1 | | | 1 | | | | | int | 0 | 2 |
| JMP i) | far | 31 | 31 | | | | | | | | int | | 67 |
| JMP | r | 1 | 1 | | | 1 | | | | | int | 0 | 2 |
| JMP | m(near) | 1 | 1 | | | 1 | 1 | | | | int | 0 | 2 |
| JMP | m(far) | 31 | 31 | | | | 11 | | | | int | | 73 |
| Conditional jump | short/near | 1 | 1 | | | 1 | | | | | int | 0 | 2 |
| Fused compare/test and branch e) | | 1 | 1 | | | 1 | | | | | int | 0 | 2 |
| J(E/R)CXZ | short | 2 | 2 | x | x | 1 | | | | | int | | 2 |
| LOOP | short | 6 | 6 | x | x | x | | | | | int | | 4 |
| LOOP(N)E | short | 11 | 11 | x | x | x | | | | | int | | 7 |
| CALL | near | 2 | 2 | ? | ? | 1 | | 1 | 1 | | int | | 2 |
| CALL i) | far | 46 | 46 | | | | 9 | | | | int | | 74 |
| CALL | r | 3 | 2 | ? | ? | 1 | | 1 | 1 | | int | | 2 |
| CALL | m(near) | 4 | 3 | ? | ? | 1 | 1 | 1 | 1 | | int | | 2 |
| CALL | m(far) | 47 | 47 | | | | 1 | | | | int | | 79 |

Nehalem

| | | | | | | | | | | | | |
|----------------------------|---------|--------------|----|---|---|---|---|---|---|-----|------------------|------------------|
| RETN | | 1 | 1 | | | 1 | 1 | | | int | | 2 |
| RETN | i | 3 | 2 | | | 1 | 1 | | | int | | 2 |
| RETF | | 39 | 39 | | | | | | | int | | 120 |
| RETF | i | 40 | 40 | | | | | | | int | | 124 |
| BOUND i) | | 15 | 13 | | | | | 2 | | int | | 7 |
| INTO i) | r,m | 4 | 4 | | | | | | | int | | 5 |
| String instructions | | | | | | | | | | | | |
| LODS | | 2 | 1 | x | x | x | 1 | | | int | | 1 |
| REP LODS | | 11+4n | | | | | | | | int | 40+12n | |
| STOS | | 3 | 1 | x | x | x | | | 1 | 1 | int | 1 |
| REP STOS | small n | 60+n | | | | | | | | | int | 12+n |
| REP STOS | large n | 2.5/16 bytes | | | | | | | | | int | 1 clk / 16 bytes |
| MOVS | | 5 | 2 | x | x | x | 1 | 1 | 1 | int | | 4 |
| REP MOVS | small n | 13+6n | | | | | | | | int | 12+n | |
| REP MOVS | large n | 2/16 bytes | | | | | | | | int | 1 clk / 16 bytes | |
| SCAS | | 3 | 2 | x | x | x | 1 | | | int | | 1 |
| REP SCAS | | 37+6n | | | | | | | | int | 40+2n | |
| CMPS | | 5 | 3 | x | x | x | 2 | | | int | | 4 |
| REP CMPS | | 65+8n | | | | | | | | int | 42+2n | |
| Other | | | | | | | | | | | | |
| NOP (90) | | 1 | 1 | x | x | x | | | | int | | 0.33 |
| Long NOP (0F 1F) | | 1 | 1 | x | x | x | | | | int | | 1 |
| PAUSE | | 5 | 5 | x | x | x | | | | int | | 9 |
| ENTER | a,0 | 11 | 9 | x | x | x | 1 | 1 | 1 | int | | 8 |
| ENTER | a,b | 34+7b | | | | | | | | int | 79+5b | |
| LEAVE | | 3 | 3 | | | | 1 | | | int | | 5 |
| CPUID | | 25-100 | | | | | | | | int | ~200 | ~200 |
| RDTSC | | 22 | | | | | | | | int | | 24 |
| RDPMSR | | 28 | | | | | | | | int | | 40-60 |

Notes:

- a) Applies to all addressing modes
- b) Has an implicit LOCK prefix.
- c) Low values are for small results, high values for high results.
- e) See manual 3: "The microarchitecture of Intel, AMD and VIA CPUs" for restrictions on macro-op fusion.
- i) Not available in 64 bit mode.
- ℓ) SSE4.2 instruction set.

Floating point x87 instructions

| Instruction | Operands | μops fused do- main | μops unfused domain | | | | | | | | Do- main | Laten- cy | Reci- procal through- put |
|-------------------|----------|------------------------|---------------------|----|----|----|----|----|----|-------|-------------|--------------|------------------------------------|
| | | | p015 | p0 | p1 | p5 | p2 | p3 | p4 | | | | |
| Move instructions | | | | | | | | | | | | | |
| FLD | r | 1 | 1 | 1 | | | | | | float | 1 | 1 | |
| FLD | m32/64 | 1 | 1 | | | | 1 | | | float | 3 | 1 | |
| FLD | m80 | 4 | 2 | 1 | 1 | | 2 | | | float | 4 | 2 | |
| FBLD | m80 | 41 | 38 | x | x | x | 3 | | | float | 45 | 20 | |
| FST(P) | r | 1 | 1 | 1 | | | | | | float | 1 | 1 | |
| FST(P) | m32/m64 | 1 | | | | | | 1 | 1 | float | 4 | 1 | |

Nehalem

| | | | | | | | | | | | | |
|--------------------------------|-----|------|------|---|---|---|----|----|----|-------|---------|---------|
| FSTP | m80 | 7 | 3 | x | x | x | | 2 | 2 | float | 5 | 5 |
| FBSTP | m80 | 208 | 204 | x | x | x | | 2 | 2 | float | 242 | 245 |
| FXCH | r | 1 | 0 f) | | | | | | | float | 0 | 1 |
| FILD | m | 1 | 1 | | 1 | | 1 | | | float | 6 | 1 |
| FIST(P) | m | 3 | 1 | | 1 | | | 1 | 1 | float | 7 | 1 |
| FISTTP g) | m | 3 | 1 | | 1 | | | 1 | 1 | float | 7 | 1 |
| FLDZ | | 1 | 1 | 1 | | | | | | float | | 1 |
| FLD1 | | 2 | 2 | 1 | 1 | | | | | float | | 2 |
| FLDPI FLDL2E etc. | | 2 | 2 | | 2 | | | | | float | | 2 |
| FCMOVcc | r | 2 | 2 | 2 | | | | | | float | 2+2 | 2 |
| FNSTSW | AX | 2 | 2 | | | | | | | float | | 1 |
| FNSTSW | m16 | 3 | 2 | | | | | 1 | 1 | float | | 2 |
| FLDCW | m16 | 2 | 1 | | | | 1 | | | float | 7 | 31 |
| FNSTCW | m16 | 2 | 1 | 1 | | | | 1 | 1 | float | 5 | 1 |
| FINCSTP FDECSTP | | 1 | 1 | 1 | | | | | | float | 1 | 1 |
| FFREE(P) | r | 2 | 2 | x | x | x | | | | float | | 4 |
| FNSAVE | m | 143 | 89 | x | x | x | 8 | 23 | 23 | float | 178 | 178 |
| FRSTOR | m | 79 | 52 | x | x | x | 27 | | | float | 156 | 156 |
| Arithmetic instructions | | | | | | | | | | | | |
| FADD(P) FSUB(R)(P) | r | 1 | 1 | | 1 | | | | | float | 3 | 1 |
| FADD(P) FSUB(R)(P) | m | 1 | 1 | | 1 | | 1 | | | float | | 1 |
| FMUL(P) | r | 1 | 1 | 1 | | | | | | float | 5 | 1 |
| FMUL(P) | m | 1 | 1 | 1 | | | 1 | | | float | | 1 |
| FDIV(R)(P) | r | 1 | 1 | 1 | | | | | | float | 7-27 d) | 7-27 d) |
| FDIV(R)(P) | m | 1 | 1 | 1 | | | 1 | | | float | 7-27 d) | 7-27 d) |
| FABS | | 1 | 1 | 1 | | | | | | float | 1 | 1 |
| FCHS | | 1 | 1 | 1 | | | | | | float | 1 | 1 |
| FCOM(P) FUCOM | r | 1 | 1 | | 1 | | | | | float | | 1 |
| FCOM(P) FUCOM | m | 1 | 1 | | 1 | | 1 | | | float | | 1 |
| FCOMPP FUCOMPP | | 2 | 2 | 1 | 1 | | | | | float | | 1 |
| FCOMI(P) FUCOMI(P) | r | 1 | 1 | | 1 | | | | | float | | 1 |
| FIADD FISUB(R) | m | 2 | 2 | | 2 | | 1 | | | float | 3 | 2 |
| FIMUL | m | 2 | 2 | 1 | 1 | | 1 | | | float | 5 | 2 |
| FIDIV(R) | m | 2 | 2 | 1 | 1 | | 1 | | | float | 7-27 d) | 7-27 d) |
| FICOM(P) | m | 2 | 2 | | 2 | | 1 | | | float | | 1 |
| FTST | | 1 | 1 | | 1 | | | | | float | | 1 |
| FXAM | | 1 | 1 | | 1 | | | | | float | | 1 |
| FPREM | | 25 | 25 | x | x | x | | | | float | 14 | |
| FPREM1 | | 35 | 35 | x | x | x | | | | float | 19 | |
| FRNDINT | | 17 | 17 | x | x | x | | | | float | 22 | |
| Math | | | | | | | | | | | | |
| FSCALE | | 24 | 24 | x | x | x | | | | float | 12 | |
| FXTRACT | | 17 | 17 | x | x | x | | | | float | 13 | |
| FSQRT | | 1 | 1 | 1 | | | | | | float | ~27 | |
| FSIN | | ~100 | ~100 | x | x | x | | | | float | 40-100 | |
| FCOS | | ~100 | ~100 | x | x | x | | | | float | 40-100 | |
| FSINCOS | | ~100 | ~100 | x | x | x | | | | float | ~110 | |
| F2XM1 | | 19 | 19 | x | x | x | | | | float | 58 | |
| FYL2X FYL2XP1 | | ~55 | ~55 | x | x | x | | | | float | ~80 | |
| FPTAN | | ~100 | ~100 | x | x | x | | | | float | ~115 | |
| FPATAN | | ~82 | ~82 | x | x | x | | | | float | ~120 | |

Nehalem

| Other | | | | | | | | | | | | |
|--------|--|------|------|---|---|---|--|--|--|--|-------|----|
| FNOP | | 1 | 1 | 1 | | | | | | | float | 1 |
| WAIT | | 2 | 2 | x | x | x | | | | | float | 1 |
| FNCLEX | | 3 | 3 | | x | x | | | | | float | 17 |
| FNINIT | | ~190 | ~190 | x | x | x | | | | | float | 77 |

Notes:

- d) Round divisors or low precision give low values.
f) Resolved by register renaming. Generates no μ ops in the unfused domain.
g) SSE3 instruction set.

Integer MMX and XMM instructions

| Instruction | Operands | μops fused do-main | μops unfused domain | | | | | | | | Do-main | Laten-cy | Reci-procal through-put |
|-------------------|-------------|--------------------|---------------------|----|----|----|----|----|----|------|---------|----------|-------------------------|
| | | | p015 | p0 | p1 | p5 | p2 | p3 | p4 | | | | |
| Move instructions | | | | | | | | | | | | | |
| MOVD k) | r32/64,mm/x | 1 | 1 | x | x | x | | | | int | 1+1 | 0.33 | |
| MOVD k) | m32/64,mm/x | 1 | | | | | | 1 | 1 | | 3 | 1 | |
| MOVD k) | mm/x,r32/64 | 1 | 1 | x | x | x | | | | ivec | 1+1 | 0.33 | |
| MOVD k) | mm/x,m32/64 | 1 | | | | | 1 | | | | 2 | 1 | |
| MOVQ | mm/x, mm/x | 1 | 1 | x | x | x | | | | ivec | 1 | 0.33 | |
| MOVQ | mm/x,m64 | 1 | | | | | 1 | | | | 2 | 1 | |
| MOVQ | m64, mm/x | 1 | | | | | | 1 | 1 | | 3 | 1 | |
| MOVDQA | xmm, xmm | 1 | 1 | x | x | x | | | | ivec | 1 | 0.33 | |
| MOVDQA | xmm, m128 | 1 | | | | | 1 | | | | 2 | 1 | |
| MOVDQA | m128, xmm | 1 | | | | | | 1 | 1 | | 3 | 1 | |
| MOVDQU | xmm, m128 | 1 | 1 | | | | 1 | | | | 2 | 1 | |
| MOVDQU | m128, xmm | 1 | 1 | | | | | 1 | 1 | | 3 | 1 | |
| LDDQU g) | xmm, m128 | 1 | 1 | | | | 1 | | | | 2 | 1 | |
| MOVDQ2Q | mm, xmm | 1 | 1 | x | x | x | | | | ivec | 1 | 0.33 | |
| MOVQ2DQ | xmm,mm | 1 | 1 | x | x | x | | | | ivec | 1 | 0.33 | |
| MOVNTQ | m64,mm | 1 | | | | | | 1 | 1 | | ~270 | 2 | |
| MOVNTDQ | m128,xmm | 1 | | | | | | 1 | 1 | | ~270 | 2 | |
| MOVNTDQA j) | xmm, m128 | 1 | | | | | 1 | | | | 2 | 1 | |
| PACKSSWB/DW | | | | | | | | | | | | | |
| PACKUSWB | mm,mm | 1 | 1 | | 1 | | | | | ivec | 1 | 1 | |
| PACKSSWB/DW | | | | | | | | | | | | | |
| PACKUSWB | mm,m64 | 1 | 1 | | 1 | | 1 | | | | | 2 | |
| PACKSSWB/DW | | | | | | | | | | | | | |
| PACKUSWB | xmm,xmm | 1 | 1 | x | | x | | | | ivec | 1 | 0.5 | |
| PACKSSWB/DW | | | | | | | | | | | | | |
| PACKUSWB | xmm,m128 | 1 | 1 | x | | x | 1 | | | | | 2 | |
| PACKUSDW j) | xmm,xmm | 1 | 1 | x | | x | | | | ivec | 1 | 2 | |
| PACKUSDW j) | xmm,m | 1 | 1 | x | | x | 1 | | | | | 2 | |
| PUNPCKH/LBW/WD/DQ | mm/x, mm/x | 1 | 1 | x | | x | | | | ivec | 1 | 0.5 | |
| PUNPCKH/LBW/WD/DQ | mm/x,m | 1 | 1 | x | | x | 1 | | | | | 2 | |
| PUNPCKH/LQDQ | xmm,xmm | 1 | 1 | x | | x | | | | ivec | 1 | 0.5 | |
| PUNPCKH/LQDQ | xmm, m128 | 2 | 1 | x | | x | 1 | | | | | 1 | |
| PMOVSX/ZXBW j) | xmm,xmm | 1 | 1 | x | | x | | | | ivec | 1 | 1 | |
| PMOVSX/ZXBW j) | xmm,m64 | 1 | 1 | x | | x | 1 | | | | | 2 | |
| PMOVSX/ZXBD j) | xmm,xmm | 1 | 1 | x | | x | | | | ivec | 1 | 1 | |
| PMOVSX/ZXBD i) | xmm,m32 | 1 | 1 | x | | x | 1 | | | | | 2 | |

Nehalem

| | | | | | | | | | | | | |
|--------------------------------|-------------|----|---|---|---|---|---|---|--|-------|-----|-----|
| PMOVSX/ZXBQ j) | xmm,xmm | 1 | 1 | x | x | | | | | ivec | 1 | 1 |
| PMOVSX/ZXBQ j) | xmm,m16 | 1 | 1 | x | x | 1 | | | | | | 2 |
| PMOVSX/ZXWD j) | xmm,xmm | 1 | 1 | x | x | | | | | ivec | 1 | 1 |
| PMOVSX/ZXWD j) | xmm,m64 | 1 | 1 | x | x | 1 | | | | | | 2 |
| PMOVSX/ZXWQ j) | xmm,xmm | 1 | 1 | x | x | | | | | ivec | 1 | 1 |
| PMOVSX/ZXWQ j) | xmm,m32 | 1 | 1 | x | x | 1 | | | | | | 2 |
| PMOVSX/ZXDQ j) | xmm,xmm | 1 | 1 | x | x | | | | | ivec | 1 | 1 |
| PMOVSX/ZXDQ j) | xmm,m64 | 1 | 1 | x | x | 1 | | | | | | 2 |
| PSHUFB h) | mm/x, mm/x | 1 | 1 | x | x | | | | | ivec | 1 | 0.5 |
| PSHUFB h) | mm/x,m | 2 | 1 | x | x | 1 | | | | | | 1 |
| PSHUFW | mm,mm,i | 1 | 1 | x | x | | | | | ivec | 1 | 0.5 |
| PSHUFW | mm,m64,i | 2 | 1 | x | x | 1 | | | | | | 1 |
| PSHUFD | xmm,xmm,i | 1 | 1 | x | x | | | | | ivec | 1 | 0.5 |
| PSHUFD | xmm,m128,i | 2 | 1 | x | x | 1 | | | | | | 1 |
| PSHUFL/HW | xmm,xmm,i | 1 | 1 | x | x | | | | | ivec | 1 | 0.5 |
| PSHUFL/HW | xmm, m128,i | 2 | 1 | x | x | 1 | | | | | | 1 |
| PALIGNR h) | mm/x,mm/x,i | 1 | 1 | x | x | | | | | ivec | 1 | 1 |
| PALIGNR h) | mm/x,m,i | 2 | 1 | x | x | 1 | | | | | | 1 |
| PBLENDVB j) | x,x,xmm0 | 2 | 2 | 1 | 1 | | | | | ivec | 2 | 1 |
| PBLENDVB j) | xmm,m,xmm0 | 3 | 2 | 1 | 1 | 1 | | | | | | 1 |
| PBLENDW j) | xmm,xmm,i | 1 | 1 | x | x | | | | | ivec | 1 | 0.5 |
| PBLENDW j) | xmm,m,i | 2 | 1 | x | x | 1 | | | | | | 1 |
| MASKMOVQ | mm,mm | 4 | 1 | 1 | | 1 | 1 | 1 | | ivec | | 2 |
| MASKMOVDQU | xmm,xmm | 10 | 4 | x | x | 2 | 2 | x | | ivec | | 7 |
| PMOVMSKB | r32,mm/x | 1 | 1 | 1 | | | | | | float | 2+2 | 1 |
| PEXTRB j) | r32,xmm,i | 2 | 2 | x | x | x | | | | ivec | 2+1 | 1 |
| PEXTRB j) | m8,xmm,i | 2 | 2 | x | x | | | | | | | 1 |
| PEXTRW | r32,mm/x,i | 2 | 2 | x | x | x | | | | ivec | 2+1 | 1 |
| PEXTRW j) | m16,mm/x,i | 2 | 2 | x | x | | 1 | 1 | | | | 1 |
| PEXTRD j) | r32,xmm,i | 2 | 2 | x | x | x | | | | ivec | 2+1 | 1 |
| PEXTRD j) | m32,xmm,i | 2 | 1 | x | x | | 1 | 1 | | | | 1 |
| PEXTRQ j,m) | r64,xmm,i | 2 | 2 | x | x | x | | | | ivec | 2+1 | 1 |
| PEXTRQ j,m) | m64,xmm,i | 2 | 1 | x | x | | 1 | 1 | | | | 1 |
| PINSRB j) | xmm,r32,i | 1 | 1 | x | x | | | | | ivec | 1+1 | 1 |
| PINSRB j) | xmm,m8,i | 2 | 1 | x | x | 1 | | | | | | 1 |
| PINSRW | mm/x,r32,i | 1 | 1 | x | x | | | | | ivec | 1+1 | 1 |
| PINSRW | mm/x,m16,i | 2 | 1 | x | x | 1 | | | | | | 1 |
| PINSRD j) | xmm,r32,i | 1 | 1 | x | x | | | | | ivec | 1+1 | 1 |
| PINSRD j) | xmm,m32,i | 2 | 1 | x | x | 1 | | | | | | 1 |
| PINSRQ j,m) | xmm,r64,i | 1 | 1 | x | x | | | | | ivec | 1+1 | 1 |
| PINSRQ j,m) | xmm,m64,i | 2 | 1 | x | x | 1 | | | | | | 1 |
| Arithmetic instructions | | | | | | | | | | | | |
| PADD/SUB(U)(S)B/W/D/Q | mm/x, mm/x | 1 | 1 | x | x | | | | | ivec | 1 | 0.5 |
| PADD/SUB(U)(S)B/W/D/Q | mm/x,m | 1 | 1 | x | x | 1 | | | | | | 2 |
| PHADD/SUB(S)W/D h) | mm/x, mm/x | 3 | 3 | x | x | | | | | ivec | 3 | 1.5 |
| PHADD/SUB(S)W/D h) | mm/x,m64 | 4 | 3 | x | x | 1 | | | | | | 3 |
| PCMPEQ/GTB/W/D | mm/x,mm/x | 1 | 1 | x | x | | | | | ivec | 1 | 0.5 |
| PCMPEQ/GTB/W/D | mm/x,m | 1 | 1 | x | x | 1 | | | | | | 2 |
| PCMPEQQ j) | xmm,xmm | 1 | 1 | x | x | | | | | ivec | 1 | 0.5 |
| PCMPEQQ j) | xmm,m128 | 1 | 1 | x | x | 1 | | | | | | 2 |

Nehalem

| | | | | | | | | | | | |
|---|-----------|---|---|---|---|---|---|--|------|----|------|
| PCMPGTQ ℓ) | xmm,xmm | 1 | 1 | | 1 | | | | ivec | 3 | 1 |
| PCMPGTQ ℓ) | xmm,m128 | 1 | 1 | | 1 | | 1 | | | | 1 |
| PMULL/HW PMULHUW | mm/x,mm/x | 1 | 1 | | 1 | | | | ivec | 3 | 1 |
| PMULL/HW PMULHUW | mm/x,m | 1 | 1 | | 1 | | 1 | | | | 1 |
| PMULHRSW h) | mm/x,mm/x | 1 | 1 | | 1 | | | | ivec | 3 | 1 |
| PMULHRSW h) | mm/x,m | 1 | 1 | | 1 | | 1 | | | | 1 |
| PMULLD j) | xmm,xmm | 2 | 2 | | 2 | | | | ivec | 6 | 2 |
| PMULLD j) | xmm,m128 | 3 | 2 | | 2 | | 1 | | | | |
| PMULDQ j) | xmm,xmm | 1 | 1 | | 1 | | | | ivec | 3 | 1 |
| PMULDQ j) | xmm,m128 | 1 | 1 | | 1 | | 1 | | | | 1 |
| PMULUDQ | mm/x,mm/x | 1 | 1 | | 1 | | | | ivec | 3 | 1 |
| PMULUDQ | mm/x,m | 1 | 1 | | 1 | | 1 | | | | 1 |
| PMADDWD | mm/x,mm/x | 1 | 1 | | 1 | | | | ivec | 3 | 1 |
| PMADDWD | mm/x,m | 1 | 1 | | 1 | | 1 | | | | 1 |
| PMADDUBSW h) | mm/x,mm/x | 1 | 1 | | 1 | | | | ivec | 3 | 1 |
| PMADDUBSW h) | mm/x,m | 1 | 1 | | 1 | | 1 | | | | 1 |
| PAVGB/W | mm/x,mm/x | 1 | 1 | x | | x | | | ivec | 1 | 0.5 |
| PAVGB/W | mm/x,m | 1 | 1 | x | | x | 1 | | | | 1 |
| PMIN/MAXSB j) | xmm,xmm | 1 | 1 | x | | x | | | ivec | 1 | 1 |
| PMIN/MAXSB j) | xmm,m128 | 1 | 1 | x | | x | 1 | | | | 2 |
| PMIN/MAXUB | mm/x,mm/x | 1 | 1 | x | | x | | | ivec | 1 | 0.5 |
| PMIN/MAXUB | mm/x,m | 1 | 1 | x | | x | 1 | | | | 2 |
| PMIN/MAXSW | mm/x,mm/x | 1 | 1 | x | | x | | | ivec | 1 | 0.5 |
| PMIN/MAXSW | mm/x,m | 1 | 1 | x | | x | 1 | | | | 2 |
| PMIN/MAXUW j) | xmm,xmm | 1 | 1 | x | | x | | | ivec | 1 | 1 |
| PMIN/MAXUW j) | xmm,m | 1 | 1 | x | | x | 1 | | | | 2 |
| PMIN/MAXU/SD j) | xmm,xmm | 1 | 1 | x | | x | | | ivec | 1 | 1 |
| PMIN/MAXU/SD j) | xmm,m128 | 1 | 1 | x | | x | 1 | | | | 2 |
| PHMINPOSUW j) | xmm,xmm | 1 | 1 | | 1 | | | | ivec | 3 | 1 |
| PHMINPOSUW j) | xmm,m128 | 1 | 1 | | 1 | | 1 | | | | 3 |
| PABSB PABSW PABSD h) | mm/x,mm/x | 1 | 1 | x | | x | | | ivec | 1 | 0.5 |
| PABSB PABSW PABSD h) | mm/x,m | 1 | 1 | x | | x | 1 | | | | 1 |
| PSIGNB PSIGNW | | | | | | | | | | | |
| PSIGND h) | mm/x,mm/x | 1 | 1 | x | | x | | | ivec | 1 | 0.5 |
| PSIGNB PSIGNW | | | | | | | | | | | |
| PSIGND h) | mm/x,m | 1 | 1 | x | | x | 1 | | | | 2 |
| PSADBW | mm/x,mm/x | 1 | 1 | | 1 | | | | ivec | 3 | 1 |
| PSADBW | mm/x,m | 1 | 1 | | 1 | | 1 | | | | 3 |
| MPSADBW j) | xmm,xmm,i | 3 | 3 | x | x | x | | | ivec | 5 | 1 |
| MPSADBW j) | xmm,m,i | 4 | 3 | x | x | x | 1 | | | | 2 |
| PCLMULQDQ n) | xmm,xmm,i | | | | | | | | | 12 | 8 |
| AESDEC, AESDECLAST, AESENC, AESENCLAST n) | | | | | | | | | | | |
| | xmm,xmm | | | | | | | | | ~5 | ~2 |
| AESIMC n) | xmm,xmm | | | | | | | | | ~5 | ~2 |
| AESKEYGENASSIST n) | xmm,xmm,i | | | | | | | | | ~5 | ~2 |
| Logic instructions | | | | | | | | | | | |
| PAND(N) POR PXOR | mm/x,mm/x | 1 | 1 | x | x | x | | | ivec | 1 | 0.33 |
| PAND(N) POR PXOR | mm/x,m | 1 | 1 | x | x | x | 1 | | | | 1 |
| PTEST j) | xmm,xmm | 2 | 2 | x | x | x | | | ivec | 3 | 1 |

Nehalem

| | | | | | | | | | | | | |
|----------------------------|------------|----|----|---|---|---|---|--|--|-------|----|---|
| PTEST j) | xmm,m128 | 2 | 2 | x | x | x | 1 | | | | | 1 |
| PSLL/RL/RAW/D/Q | mm,mm/i | 1 | 1 | | 1 | | | | | ivec | 1 | 1 |
| PSLL/RL/RAW/D/Q | mm,m64 | 1 | 1 | | 1 | | 1 | | | | | 2 |
| PSLL/RL/RAW/D/Q | xmm,i | 1 | 1 | | 1 | | | | | ivec | 1 | 1 |
| PSLL/RL/RAW/D/Q | xmm,xmm | 2 | 2 | x | 1 | x | | | | ivec | 2 | 2 |
| PSLL/RL/RAW/D/Q | xmm,m128 | 3 | 2 | x | 1 | x | 1 | | | | | 1 |
| PSLL/RLDQ | xmm,i | 1 | 1 | x | | x | | | | ivec | 1 | 1 |
| String instructions | | | | | | | | | | | | |
| PCMPESTRI ℓ) | xmm,xmm,i | 8 | 8 | x | x | x | | | | ivec | 14 | 5 |
| PCMPESTRI ℓ) | xmm,m128,i | 9 | 8 | x | x | x | 1 | | | ivec | 14 | 6 |
| PCMPESTRM ℓ) | xmm,xmm,i | 9 | 9 | x | x | x | | | | ivec | 7 | 6 |
| PCMPESTRM ℓ) | xmm,m128,i | 10 | 10 | x | x | x | 1 | | | ivec | 7 | 6 |
| PCMPISTRI ℓ) | xmm,xmm,i | 3 | 3 | x | x | x | | | | ivec | 8 | 2 |
| PCMPISTRI ℓ) | xmm,m128,i | 4 | 4 | x | x | x | 1 | | | ivec | 8 | 2 |
| PCMPISTRM ℓ) | xmm,xmm,i | 4 | 4 | x | x | x | | | | ivec | 7 | 2 |
| PCMPISTRM ℓ) | xmm,m128,i | 6 | 5 | x | x | x | 1 | | | ivec | 7 | 5 |
| Other | | | | | | | | | | | | |
| EMMS | | 11 | 11 | x | x | x | | | | float | | 6 |

Notes:

- g) SSE3 instruction set.
- h) Supplementary SSE3 instruction set.
- j) SSE4.1 instruction set
- k) MASM uses the name MOVD rather than MOVQ for this instruction even when moving 64 bits
- ℓ) SSE4.2 instruction set
- m) Only available in 64 bit mode
- n) Only available on newer models

Floating point XMM instructions

| Instruction | Operands | μops fused do-main | μops unfused domain | | | | | | | Do-main | Laten-cy | Reci-procal through-put |
|-------------------|------------|--------------------|---------------------|----|----|----|----|----|----|---------|----------|-------------------------|
| | | | p015 | p0 | p1 | p5 | p2 | p3 | p4 | | | |
| Move instructions | | | | | | | | | | | | |
| MOVAPS/D | xmm,xmm | 1 | 1 | | | 1 | | | | float | 1 | 1 |
| MOVAPS/D | xmm,m128 | 1 | | | | | 1 | | | | 2 | 1 |
| MOVAPS/D | m128,xmm | 1 | | | | | | 1 | 1 | | 3 | 1 |
| MOVUPS/D | xmm,m128 | 1 | | | | | 1 | | | | 2 | 1-4 |
| MOVUPS/D | m128,xmm | 1 | | | | | | 1 | 1 | | 3 | 1-3 |
| MOVSS/D | xmm,xmm | 1 | 1 | | | 1 | | | | | 1 | 1 |
| MOVSS/D | xmm,m32/64 | 1 | | | | | 1 | | | | 2 | 1 |
| MOVSS/D | m32/64,xmm | 1 | | | | | | 1 | 1 | | 3 | 1 |
| MOVHPS/D MOVLPS/D | xmm,m64 | 2 | 1 | | | 1 | 1 | | | | 3 | 2 |
| MOVH/LPS/D | m64,xmm | 2 | 1 | | | 1 | | 1 | 1 | | 5 | 1 |
| MOVLHPS MOVHLPS | xmm,xmm | 1 | 1 | | | 1 | | | | float | 1 | 1 |
| MOVMSKPS/D | r32,xmm | 1 | 1 | 1 | | | | | | float | 1+2 | 1 |
| MOVNTPS/D | m128,xmm | 1 | | | | | | 1 | 1 | | ~270 | 2 |
| SHUFPS/D | xmm,xmm,i | 1 | 1 | | | 1 | | | | float | 1 | 1 |
| SHUFPS/D | xmm,m128,i | 2 | 1 | | | 1 | 1 | | | float | | 1 |
| BLENDPS/PD j) | xmm,xmm,i | 1 | 1 | | | 1 | | | | float | 1 | 1 |

Nehalem

| | | | | | | | | | | | | |
|-------------------|------------|---|---|---|---|---|---|---|---|------------|-----|---|
| BLENDPS/PD j) | xmm,m128,i | 2 | 1 | | | 1 | 1 | | | float | | 1 |
| BLENDVPS/PD j) | x,x,xmm0 | 2 | 2 | | | 2 | | | | float | 2 | 2 |
| BLENDVPS/PD j) | xmm,m,xmm0 | 3 | 2 | | | 2 | 1 | | | float | | 2 |
| MOVDDUP g) | xmm,xmm | 1 | 1 | | | 1 | | | | float | 1 | 1 |
| MOVDDUP g) | xmm,m64 | 1 | | | | | 1 | | | | 2 | 1 |
| MOVSH/LDUP g) | xmm,xmm | 1 | 1 | | | 1 | | | | float | 1 | 1 |
| MOVSH/LDUP g) | xmm,m128 | 1 | | | | | 1 | | | | | 1 |
| UNPCKH/LPS/D | xmm,xmm | 1 | 1 | | | 1 | | | | float | 1 | 1 |
| UNPCKH/LPS/D | xmm,m128 | 1 | 1 | | | 1 | 1 | | | float | | 1 |
| EXTRACTPS j) | r32,xmm,i | 1 | 1 | | | 1 | | | | float | 1+2 | 1 |
| EXTRACTPS j) | m32,xmm,i | 2 | 1 | | | 1 | | 1 | 1 | | | 1 |
| INSERTPS j) | xmm,xmm,i | 1 | 1 | | | 1 | | | | float | 1 | 1 |
| INSERTPS j) | xmm,m32,i | 3 | 2 | | | 2 | 1 | | | float | | 2 |
| Conversion | | | | | | | | | | | | |
| CVTPD2PS | xmm,xmm | 2 | 2 | | | 1 | 1 | | | float | 4 | 1 |
| CVTPD2PS | xmm,m128 | 2 | 2 | | | 1 | | 1 | | float | | 1 |
| CVTSD2SS | xmm,xmm | 2 | 2 | | | 1 | 1 | | | float | 4 | 1 |
| CVTSD2SS | xmm,m64 | 2 | 2 | ? | ? | ? | 1 | | | float | | 1 |
| CVTPS2PD | xmm,xmm | 2 | 2 | 1 | | 1 | | | | float | 2 | 1 |
| CVTPS2PD | xmm,m64 | 2 | 2 | 1 | | 1 | 1 | | | float | | 1 |
| CVTSS2SD | xmm,xmm | 1 | 1 | 1 | | | | | | float | 1 | 1 |
| CVTSS2SD | xmm,m32 | 1 | 1 | 1 | | | 1 | | | float | | 2 |
| CVTDQ2PS | xmm,xmm | 1 | 1 | | | 1 | | | | float | 3+2 | 1 |
| CVTDQ2PS | xmm,m128 | 1 | 1 | | | 1 | | 1 | | float | | 1 |
| CVT(T) PS2DQ | xmm,xmm | 1 | 1 | | | 1 | | | | float | 3+2 | 1 |
| CVT(T) PS2DQ | xmm,m128 | 1 | 1 | | | 1 | | 1 | | float | | 1 |
| CVTDQ2PD | xmm,xmm | 2 | 2 | | | 1 | 1 | | | float | 4+2 | 1 |
| CVTDQ2PD | xmm,m64 | 2 | 2 | | | 1 | 1 | 1 | | float | | 1 |
| CVT(T)PD2DQ | xmm,xmm | 2 | 2 | | | 1 | 1 | | | float | 4+2 | 1 |
| CVT(T)PD2DQ | xmm,m128 | 2 | 2 | | | 1 | 1 | 1 | | float | | 1 |
| CVTPI2PS | xmm,mm | 1 | 1 | | | 1 | | | | float | 3+2 | 3 |
| CVTPI2PS | xmm,m64 | 1 | 1 | | | 1 | | 1 | | float | | 3 |
| CVT(T)PS2PI | mm,xmm | 1 | 1 | | | 1 | | | | float | 3+2 | 1 |
| CVT(T)PS2PI | mm,m128 | 1 | 1 | | | 1 | | 1 | | float | | 1 |
| CVTPI2PD | xmm,mm | 2 | 2 | | | 1 | 1 | | | ivec/float | 6 | 1 |
| CVTPI2PD | xmm,m64 | 2 | 2 | | | 1 | 1 | 1 | | | | 1 |
| CVT(T) PD2PI | mm,xmm | 2 | 2 | x | | 1 | x | | | float/ivec | 6 | 1 |
| CVT(T) PD2PI | mm,m128 | 2 | 2 | x | | 1 | x | 1 | | | | 1 |
| CVTSI2SS | xmm,r32 | 1 | 1 | | | 1 | | | | float | 3+2 | 3 |
| CVTSI2SS | xmm,m32 | 1 | 1 | | | 1 | | 1 | | float | | 3 |
| CVT(T)SS2SI | r32,xmm | 1 | 1 | | | 1 | | | | float | 3+2 | 1 |
| CVT(T)SS2SI | r32,m32 | 1 | 1 | | | 1 | | 1 | | float | | 1 |
| CVTSI2SD | xmm,r32 | 2 | 2 | 1 | | 1 | | | | float | 4+2 | 3 |
| CVTSI2SD | xmm,m32 | 2 | 1 | | | 1 | | 1 | | float | | 3 |
| CVT(T)SD2SI | r32,xmm | 1 | 1 | | | 1 | | | | float | 3+2 | 1 |
| CVT(T)SD2SI | r32,m64 | 1 | 1 | | | 1 | | 1 | | float | | 1 |
| Arithmetic | | | | | | | | | | | | |
| ADDSS/D SUBSS/D | xmm,xmm | 1 | 1 | | | 1 | | | | float | 3 | 1 |
| ADDSS/D SUBSS/D | xmm,m32/64 | 1 | 1 | | | 1 | | 1 | | float | | 1 |
| ADDPS/D SUBPS/D | xmm,xmm | 1 | 1 | | | 1 | | | | float | 3 | 1 |
| ADDPS/D SUBPS/D | xmm,m128 | 1 | 1 | | | 1 | | 1 | | float | | 1 |

Nehalem

| | | | | | | | | | | | | |
|------------------------|------------|-----|-----|---|---|---|----|----|----|-------|------|------|
| ADDSUBPS/D g) | xmm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 1 |
| ADDSUBPS/D g) | xmm,m128 | 1 | 1 | | 1 | | 1 | | | float | | 1 |
| HADDPS HSUBPS g) | xmm,xmm | 3 | 3 | | 1 | 2 | | | | float | 5 | 2 |
| HADDPS HSUBPS g) | xmm,m128 | 4 | 3 | | 1 | 2 | 1 | | | float | | 2 |
| HADDPD HSUBPD g) | xmm,xmm | 3 | 3 | | 1 | 2 | | | | float | 3 | 2 |
| HADDPD HSUBPD g) | xmm,m128 | 4 | 3 | | 1 | 2 | 1 | | | float | | 2 |
| MULSS MULPS | xmm,xmm | 1 | 1 | 1 | | | | | | float | 4 | 1 |
| MULSS MULPS | xmm,m | 1 | 1 | 1 | | | 1 | | | float | | 1 |
| MULSD MULPD | xmm,xmm | 1 | 1 | 1 | | | | | | float | 5 | 1 |
| MULSD MULPD | xmm,m | 1 | 1 | 1 | | | 1 | | | float | | 1 |
| DIVSS DIVPS | xmm,xmm | 1 | 1 | 1 | | | | | | float | 7-14 | 7-14 |
| DIVSS DIVPS | xmm,m | 1 | 1 | 1 | | | 1 | | | float | | 7-14 |
| DIVSD DIVPD | xmm,xmm | 1 | 1 | 1 | | | | | | float | 7-22 | 7-22 |
| DIVSD DIVPD | xmm,m | 1 | 1 | 1 | | | 1 | | | float | | 7-22 |
| RCPSS/PS | xmm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 2 |
| RCPSS/PS | xmm,m | 1 | 1 | | 1 | | 1 | | | float | | 2 |
| CMPccSS/D CMPccPS/D | xmm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 1 |
| CMPccSS/D CMPccPS/D | xmm,m | 2 | 1 | | 1 | | 1 | | | float | | 1 |
| COMISS/D UCOMISS/D | xmm,xmm | 1 | 1 | | 1 | | | | | float | 1+2 | 1 |
| COMISS/D UCOMISS/D | xmm,m32/64 | 1 | 1 | | 1 | | 1 | | | float | | 1 |
| MAXSS/D MINSS/D | xmm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 1 |
| MAXSS/D MINSS/D | xmm,m32/64 | 1 | 1 | | 1 | | 1 | | | float | | 1 |
| MAXPS/D MINPS/D | xmm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 1 |
| MAXPS/D MINPS/D | xmm,m128 | 1 | 1 | | 1 | | 1 | | | float | | 1 |
| ROUNDSS/D ROUNDPS/D j) | xmm,xmm,i | 1 | 1 | | 1 | | | | | float | 3 | 1 |
| ROUNDSS/D ROUNDPS/D j) | xmm,m128,i | 2 | 1 | | 1 | | 1 | | | float | | 1 |
| DPPS j) | xmm,xmm,i | 4 | 4 | 1 | 2 | 1 | | | | float | 11 | 2 |
| DPPS j) | xmm,m128,i | 6 | 5 | x | x | x | 1 | | | float | | |
| DPPD j) | xmm,xmm,i | 3 | 3 | x | x | x | | | | float | 9 | 1 |
| DPPD j) | xmm,m128,i | 4 | 3 | x | x | x | 1 | | | float | | 3 |
| Math | | | | | | | | | | | | |
| SQRTSS/PS | xmm,xmm | 1 | 1 | 1 | | | | | | float | 7-18 | 7-18 |
| SQRTSS/PS | xmm,m | 2 | 1 | 1 | | | 1 | | | float | | 7-18 |
| SQRTSD/PD | xmm,xmm | 1 | 1 | 1 | | | | | | float | 7-32 | 7-32 |
| SQRTSD/PD | xmm,m | 2 | 1 | 1 | | | 1 | | | float | | 7-32 |
| RSQRTSS/PS | xmm,xmm | 1 | 1 | | 1 | | | | | float | 3 | 2 |
| RSQRTSS/PS | xmm,m | 1 | 1 | | 1 | | 1 | | | float | | 2 |
| Logic | | | | | | | | | | | | |
| AND/ANDN/OR/XORPS/D | xmm,xmm | 1 | 1 | | | 1 | | | | float | 1 | 1 |
| AND/ANDN/OR/XORPS/D | xmm,m128 | 1 | 1 | | | 1 | 1 | | | float | | 1 |
| Other | | | | | | | | | | | | |
| LDMXCSR | m32 | 6 | 6 | x | x | x | 1 | | | | | 5 |
| STMXCSR | m32 | 2 | 1 | | | 1 | | 1 | 1 | | | 1 |
| FXSAVE | m4096 | 141 | 141 | x | x | x | 5 | 38 | 38 | | 90 | 90 |
| FXRSTOR | m4096 | 112 | 90 | x | x | x | 42 | | | | | 100 |

Notes:

g)

SSE3 instruction set.

Intel Sandy Bridge

List of instruction timings and μ op breakdown

Explanation of column headings:

Operands: i = immediate data, r = register, mm = 64 bit mmx register, x = 128 bit xmm register, mm/x = mmx or xmm register, y = 256 bit ymm register, same = same register for both operands. m = memory operand, m32 = 32-bit memory operand, etc.

μ ops fused domain: The number of μ ops at the decode, rename, allocate and retirement stages in the pipeline. Fused μ ops count as one.

μ ops unfused domain: The number of μ ops for each execution port. Fused μ ops count as two. Fused macro-ops count as one. The instruction has μ op fusion if the sum of the numbers listed under p015 + p23 + p4 exceeds the number listed under μ ops fused domain. A number indicated as 1+ under a read or write port means a 256-bit read or write operation using two clock cycles for handling 128 bits each cycle. The port cannot receive another read or write μ op in the second clock cycle, but a read port can receive an address-calculation μ op in the second clock cycle. An x under p0, p1 or p5 means that at least one of the μ ops listed under p015 can optionally go to this port. For example, a 1 under p015 and an x under p0 and p5 means one μ op which can go to either port 0 or port 5, whichever is vacant first. A value listed under p015 but nothing under p0, p1 and p5 means that it is not known which of the three ports these μ ops go to.

p015: The total number of μ ops going to port 0, 1 and 5.

p0: The number of μ ops going to port 0 (execution units).

p1: The number of μ ops going to port 1 (execution units).

p5: The number of μ ops going to port 5 (execution units).

p23: The number of μ ops going to port 2 or 3 (memory read or address calculation).

p4: The number of μ ops going to port 4 (memory write data).

Latency: This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Where hyperthreading is enabled, the use of the same execution units in the other thread leads to inferior performance. Denormal numbers, NAN's and infinity do not increase the latency. The time unit used is core clock cycles, not the reference clock cycles given by the time stamp counter.

Reciprocal throughput: The average number of core clock cycles per instruction for a series of independent instructions of the same kind in the same thread.

The latencies and throughputs listed below for addition and multiplication using full size YMM registers are obtained only after a warm-up period of a thousand instructions or more. The latencies may be one or two clock cycles longer and the reciprocal throughputs double the values for shorter sequences of code. There is no warm-up effect when vectors are 128 bits wide or less.

Integer instructions

| Instruction | Operands | μ ops fused domain | μ ops unfused domain | | | | | | Latency | Reciprocal throughput | Comments |
|--------------------------|----------|------------------------|--------------------------|----|----|----|-----|----|---------|-----------------------|----------------------|
| | | | p015 | p0 | p1 | p5 | p23 | p4 | | | |
| Move instructions | | | | | | | | | | | |
| MOV | r,r/i | 1 | 1 | x | x | x | | | 1 | | |
| MOV | r,m | 1 | | | | | 1 | | 2 | 0.5 | all addressing modes |

Sandy Bridge

| | | | | | | | | | | | |
|--------------------------------|---------|----|----|---|---|---|---|---|------|------|-------------------------|
| MOV | m,r | 1 | | | | | 1 | 1 | 3 | 1 | |
| MOV | m,i | 1 | | | | | 1 | 1 | | 1 | |
| MOVNTI | m,r | 2 | | | | | 1 | 1 | ~350 | 1 | |
| MOVSX MOVZX | r,r | 1 | 1 | x | x | x | | | 1 | | |
| MOVSXD | | | | | | | | | | | |
| MOVSX MOVZX | r,m | 1 | | | | | 1 | | | 0.5 | |
| MOVSXD | | | | | | | | | | | |
| CMOVcc | r,r | 2 | 2 | x | x | x | | | 2 | 1 | |
| CMOVcc | r,m | 2 | 2 | x | x | x | 1 | | | 1 | |
| XCHG | r,r | 3 | 3 | x | x | x | | | 2 | 1 | |
| XCHG | r,m | 8 | x | | | | 2 | 1 | 25 | | implicit lock |
| XLAT | | 3 | 2 | | | | 1 | | 7 | 1 | |
| PUSH | r | 1 | | | | | 1 | 1 | 3 | 1 | |
| PUSH | i | 1 | | | | | 1 | 1 | | 1 | |
| PUSH | m | 2 | | | | | 2 | 1 | | 1 | |
| PUSHF(D/Q) | | 3 | 2 | x | x | x | 1 | 1 | | 1 | |
| PUSHA(D) | | 16 | 0 | | | | 8 | 8 | | 8 | not 64 bit |
| POP | r | 1 | | | | | 1 | | 2 | 0.5 | |
| POP | (E/R)SP | 1 | 0 | | | | 1 | | | 0.5 | |
| POP | m | 2 | | | | | 2 | 1 | | 1 | |
| POPF(D/Q) | | 9 | 8 | x | x | x | 1 | | | 18 | |
| POPA(D) | | 18 | 10 | | | | 8 | | | 9 | not 64 bit |
| LAHF SAHF | | 1 | 1 | | | | | | 1 | 1 | |
| SALC | | 3 | 3 | | | | | | 1 | 1 | not 64 bit |
| LEA | r,m | 1 | 1 | x | x | | | | 1 | 0.5 | simple |
| LEA | r,m | 1 | 1 | | 1 | | | | 3 | 1 | complex or rip relative |
| BSWAP | r32 | 1 | 1 | | 1 | | | | 1 | 1 | |
| BSWAP | r64 | 2 | 2 | | 2 | | | | 2 | 1 | |
| PREFETCHNTA | m | 1 | | | | | 1 | | | 0.5 | |
| PREFETCHT0/1/2 | m | 1 | | | | | 1 | | | 0.5 | |
| LFENCE | | 2 | | | | | 1 | 1 | | 4 | |
| MFENCE | | 3 | 1 | | | | 1 | 1 | | 33 | |
| SFENCE | | 2 | | | | | 1 | 1 | | 6 | |
| Arithmetic instructions | | | | | | | | | | | |
| ADD SUB | r,r/i | 1 | 1 | x | x | x | | | 1 | | |
| ADD SUB | r,m | 1 | 1 | x | x | x | 1 | | | 0.5 | |
| ADD SUB | m,r/i | 2 | 1 | x | x | x | 2 | 1 | 6 | 1 | |
| SUB | r,same | 1 | 0 | | | | | | 0 | 0.25 | |
| ADC SBB | r,r/i | 2 | 2 | x | x | x | | | 2 | 1 | |
| ADC SBB | r,m | 2 | 2 | x | x | x | 1 | | 2 | 1 | |
| ADC SBB | m,r/i | 4 | 3 | x | x | x | 2 | 1 | 7 | 1.5 | |
| CMP | r,r/i | 1 | 1 | x | x | x | | | 1 | | |
| CMP | m,r/i | 1 | 1 | x | x | x | 1 | | 1 | 0.5 | |
| INC DEC NEG NOT | r | 1 | 1 | x | x | x | | | 1 | | |
| INC DEC NEG NOT | m | 3 | 1 | x | x | x | 2 | 1 | 6 | 2 | |
| AAA AAS | | 2 | 2 | | | | | | 4 | | not 64 bit |
| DAA DAS | | 3 | 3 | | | | | | 4 | | not 64 bit |
| AAD | | 3 | 3 | | | | | | 2 | | not 64 bit |
| AAM | | 8 | 8 | | | | | | 20 | 11 | not 64 bit |
| MUL IMUL | r8 | 1 | 1 | | 1 | | | | 3 | 1 | |

Sandy Bridge

| | | | | | | | | | | | |
|---------------------------|-------------|--------|----|---|---|---|---|---|--------|-------|--------|
| MUL IMUL | r16 | 4 | 4 | | | | | | 4 | 2 | |
| MUL IMUL | r32 | 3 | 3 | | | | | | 4 | 2 | |
| MUL IMUL | r64 | 2 | 2 | | | | | | 3 | 1 | |
| IMUL | r,r | 1 | 1 | | 1 | | | | 3 | 1 | |
| IMUL | r16,r16,i | 2 | 2 | | | | | | 4 | 1 | |
| IMUL | r32,r32,i | 1 | 1 | | 1 | | | | 3 | 1 | |
| IMUL | r64,r64,i | 1 | 1 | | 1 | | | | 3 | 1 | |
| MUL IMUL | m8 | 1 | 1 | | 1 | | 1 | | 3 | 1 | |
| MUL IMUL | m16 | 4 | 3 | | | | 1 | | | 2 | |
| MUL IMUL | m32 | 3 | 2 | | | | 1 | | | 2 | |
| MUL IMUL | m64 | 2 | 1 | | | | 1 | | | 2 | |
| IMUL | r,m | 1 | 1 | | 1 | | 1 | | | 1 | |
| IMUL | r16,m16,i | 2 | 2 | | | | 1 | | | 1 | |
| IMUL | r32,m32,i | 1 | 1 | | 1 | | 1 | | | 1 | |
| IMUL | r64,m64,i | 1 | 1 | | 1 | | 1 | | | 1 | |
| DIV | r8 | 10 | 10 | | | | | | 20-24 | 11-14 | |
| DIV | r16 | 11 | 11 | | | | | | 21-25 | 11-14 | |
| DIV | r32 | 10 | 10 | | | | | | 20-28 | 11-18 | |
| DIV | r64 | 34-56 | x | | | | | | 30-94 | 22-76 | |
| IDIV | r8 | 10 | 10 | | | | | | 21-24 | 11-14 | |
| IDIV | r16 | 10 | 10 | | | | | | 21-25 | 11-14 | |
| IDIV | r32 | 9 | 9 | | | | | | 20-27 | 11-18 | |
| IDIV | r64 | 59-138 | x | | | | | | 40-103 | 25-84 | |
| CBW | | 1 | 1 | | | | | | 1 | 0.5 | |
| CWDE | | 1 | 1 | | | 1 | | | 1 | 1 | |
| CDQE | | 1 | 1 | | | | | | 1 | 0.5 | |
| CWD | | 2 | 2 | | | | | | 1 | 1 | |
| CDQ | | 1 | 1 | | | | | | 1 | 1 | |
| CQO | | 1 | 1 | | | | | | 1 | 0.5 | |
| POPCNT | r,r | 1 | 1 | | 1 | | | | 3 | 1 | SSE4.2 |
| POPCNT | r,m | 1 | 1 | | 1 | | 1 | | | 1 | SSE4.2 |
| CRC32 | r,r | 1 | 1 | | 1 | | | | 3 | 1 | SSE4.2 |
| CRC32 | r,m | 1 | 1 | | 1 | | 1 | | | 1 | SSE4.2 |
| Logic instructions | | | | | | | | | | | |
| AND OR XOR | r,r/i | 1 | 1 | x | x | x | | | 1 | | |
| AND OR XOR | r,m | 1 | 1 | x | x | x | 1 | | | 0.5 | |
| AND OR XOR | m,r/i | 2 | 1 | x | x | x | 2 | 1 | 6 | 1 | |
| XOR | r,same | 1 | 0 | | | | | | 0 | 0.25 | |
| TEST | r,r/i | 1 | 1 | x | x | x | | | 1 | | |
| TEST | m,r/i | 1 | 1 | x | x | x | 1 | | | 0.5 | |
| SHR SHL SAR | r,i | 1 | 1 | x | | x | | | 1 | 0.5 | |
| SHR SHL SAR | m,i | 3 | 1 | | | | 2 | 1 | | 2 | |
| SHR SHL SAR | r,cl | 3 | 3 | | | | | | 2 | 2 | |
| SHR SHL SAR | m,cl | 5 | 3 | | | | 2 | 1 | | 4 | |
| ROR ROL | r,i | 1 | 1 | | | | | | 1 | 1 | |
| ROR ROL | m,i | 4 | 3 | | | | 2 | 1 | | 2 | |
| ROR ROL | r,cl | 3 | 3 | | | | | | 2 | 2 | |
| ROR ROL | m,cl | 5 | 3 | | | | 2 | 1 | | 4 | |
| RCR | r8,1 | high | | | | | | | high | high | |
| RCR | r16/32/64,1 | 3 | 3 | | | | | | 2 | 2 | |
| RCR | r,i | 8 | 8 | | | | | | 5 | 5 | |

Sandy Bridge

| | | | | | | | | | | | |
|--------------------------------------|------------|---------|----|---|---|---|---|---|-------|------|---------------------|
| RCR | m,i | 11 | 7 | | | | x | x | | 6 | |
| RCR | r,cl | 8 | 8 | | | | | | 5 | 5 | |
| RCR | m,cl | 11 | 7 | | | | x | x | | 6 | |
| RCL | r,1 | 3 | 3 | | | | | | 2 | 2 | |
| RCL | r,i | 8 | 8 | | | | | | 6 | 6 | |
| RCL | m,i | 11 | 7 | | | | x | x | | 6 | |
| RCL | r,cl | 8 | 8 | | | | | | 6 | 6 | |
| RCL | m,cl | 11 | 7 | | | | x | x | | 6 | |
| SHRD SHLD | r,r,i | 1 | 1 | | | | | | | 0.5 | |
| SHRD SHLD | m,r,i | 3 | | | | | 2 | 1 | | 2 | |
| SHRD SHLD | r,r,cl | 4 | 4 | | | | | | 2 | 2 | |
| SHRD SHLD | m,r,cl | 5 | 3 | | | | 2 | 1 | | 4 | |
| BT | r,r/i | 1 | 1 | | | | | | 1 | 0.5 | |
| BT | m,r | 10 | 8 | | | | x | | | 5 | |
| BT | m,i | 2 | 1 | | | | 1 | | | 0.5 | |
| BTR BTS BTC | r,r/i | 1 | 1 | | | | | | 1 | 0.5 | |
| BTR BTS BTC | m,r | 11 | 7 | | | | x | x | | 5 | |
| BTR BTS BTC | m,i | 3 | 1 | | | | 2 | 1 | | 2 | |
| BSF BSR | r,r | 1 | 1 | | | | | | 3 | 1 | |
| BSF BSR | r,m | 1 | 1 | | 1 | | 1 | | | 1 | |
| SETcc | r | 1 | 1 | x | | x | | | 1 | 0.5 | |
| SETcc | m | 2 | 1 | x | | x | 1 | 1 | | 1 | |
| CLC | | 1 | 0 | | | | | | | 0.25 | |
| STC CMC | | 1 | 1 | x | x | x | | | 1 | | |
| CLD STD | | 3 | 3 | | | | | | | 4 | |
| Control transfer instructions | | | | | | | | | | | |
| JMP | short/near | 1 | 1 | | | 1 | | | 0 | 2 | |
| JMP | r | 1 | 1 | | | 1 | | | 0 | 2 | |
| JMP | m | 1 | 1 | | | 1 | 1 | | 0 | 2 | |
| Conditional jump | short/near | 1 | 1 | | | 1 | | | 0 | 1-2 | fast if not jumping |
| Fused arithmetic and branch | | 1 | 1 | | | 1 | | | 0 | 1-2 | |
| J(E/R)CXZ | short | 2 | 2 | x | x | 1 | | | | 2-4 | |
| LOOP | short | 7 | 7 | | | | | | | 5 | |
| LOOP(N)E | short | 11 | 11 | | | | | | | 5 | |
| CALL | near | 3 | 2 | | | 1 | 1 | 1 | | 2 | |
| CALL | r | 2 | 1 | | | 1 | 1 | 1 | | 2 | |
| CALL | m | 3 | 2 | | | 1 | 2 | 1 | | 2 | |
| RET | | 2 | 2 | | | 1 | 1 | | | 2 | |
| RET | i | 3 | 2 | | | 1 | 1 | | | 2 | |
| BOUND | r,m | 15 | 13 | | | | | | | 7 | not 64 bit |
| INTO | | 4 | 4 | | | | | | | 6 | not 64 bit |
| String instructions | | | | | | | | | | | |
| LODS | | 3 | 2 | | | 1 | | | | 1 | |
| REP LODS | | 5n+12 | | | | | | | ~2n | | |
| STOS | | 3 | 1 | | | 1 | 1 | | | 1 | |
| REP STOS | | 2n | | | | | | | n | | worst case |
| REP STOS | | 1.5/16B | | | | | | | 1/16B | | best case |

Sandy Bridge

| | | | | | | | | | | | |
|------------------|-----|-------|----|--|--|--|---|---|---------|------|-------------------------|
| MOVS | | 5 | | | | | | | 1.5 n | 4 | worst case best case |
| REP MOVS | | 2n | | | | | | | | | |
| REP MOVS | | 3/16B | | | | | | | 1/16B | | |
| SCAS | | 3 | | | | | | | | 1 | |
| REP SCAS | | 6n+47 | | | | | | | 2n+45 | | |
| CMPS | | 5 | | | | | | | | 4 | |
| REP CMPS | | 8n+80 | | | | | | | 2n+80 | | |
| Other | | | | | | | | | | | |
| NOP (90) | | 1 | 0 | | | | | | | 0.25 | decode only 1 per clk |
| Long NOP (0F 1F) | | 1 | 0 | | | | | | | 0.25 | |
| PAUSE | | 7 | 7 | | | | | | | 11 | |
| ENTER | a,0 | 12 | 10 | | | | 2 | 1 | | 8 | |
| ENTER | a,b | 49+6b | | | | | | | 84+3b | | |
| LEAVE | | 3 | 3 | | | | 1 | | | 7 | |
| CPUID | | 31-75 | | | | | | | 100-250 | | |
| RDTSC | | 21 | | | | | | | | 28 | |
| RDTSCP | | 23 | | | | | | | | 36 | |
| RDPMC | | 35 | | | | | | | | 42 | |

Floating point x87 instructions

| Instruction | Operands | μops fused domain | μops unfused domain | | | | | | Latency | Reciprocal throughput | Comments |
|-------------------|----------|-------------------|---------------------|----|----|----|-----|----|---------|-----------------------|----------|
| | | | p015 | p0 | p1 | p5 | p23 | p4 | | | |
| Move instructions | | | | | | | | | | | |
| FLD | r | 1 | 1 | 1 | | | | | 1 | 1 | SSE3 |
| FLD | m32/64 | 1 | 1 | | | | 1 | | 3 | 1 | |
| FLD | m80 | 4 | 2 | 1 | 1 | | 2 | | 4 | 2 | |
| FBLD | m80 | 43 | 40 | | | | 3 | | 45 | 21 | |
| FST(P) | r | 1 | 1 | 1 | | | | | 1 | 1 | |
| FST(P) | m32/m64 | 1 | | | | | 1 | 1 | 4 | 1 | |
| FSTP | m80 | 7 | 3 | | | | 2 | 2 | 5 | 5 | |
| FBSTP | m80 | 246 | | | | | | | | 252 | |
| FXCH | r | 1 | 0 | | | | | | 0 | 0.5 | |
| FILD | m | 1 | 1 | | 1 | | 1 | | 6 | 1 | |
| FIST(P) | m | 3 | 1 | | 1 | | 1 | 1 | 7 | 2 | |
| FISTTP | m | 3 | 1 | | 1 | | 1 | 1 | 7 | 2 | |
| FLDZ | | 1 | 1 | 1 | | | | | | 2 | |
| FLD1 | | 2 | 2 | 1 | 1 | | | | | 2 | |
| FLDPI FLDL2E etc. | | 2 | 2 | | 2 | | | | | 2 | |
| FCMOVcc | r | 3 | 3 | | | | | | 3 | 2 | |
| FNSTSW | AX | 2 | 2 | | | | | | 2 | 1 | |
| FNSTSW | m16 | 2 | 1 | | | | 1 | 1 | | 1 | |
| FLDCW | m16 | 3 | 2 | | | | 1 | | 8 | | |
| FNSTCW | m16 | 2 | 1 | 1 | | | 1 | 1 | 5 | 1 | |
| FINCSTP FDECSTP | | 1 | 1 | 1 | | | | | 1 | 1 | |
| FFREE(P) | r | 1 | 1 | | | | | | | 1 | |
| FNSAVE | m | 143 | | | | | | | | 166 | |

Sandy Bridge

| | | | | | | | | | | | |
|--------------------------------|---|--------|-----|---|---|---|---|--|--------|-------|--|
| FRSTOR | m | 90 | | | | | | | | 165 | |
| Arithmetic instructions | | | | | | | | | | | |
| FADD(P) FSUB(R)(P) | r | 1 | 1 | | 1 | | | | 3 | 1 | |
| FADD(P) FSUB(R)(P) | m | 2 | 2 | | 1 | | 1 | | | 1 | |
| FMUL(P) | r | 1 | 1 | 1 | | | | | 5 | 1 | |
| FMUL(P) | m | 1 | 1 | 1 | | | 1 | | | 1 | |
| FDIV(R)(P) | r | 1 | 1 | 1 | | | | | 10-24 | 10-24 | |
| FDIV(R)(P) | m | 1 | 1 | 1 | | | 1 | | | 10-24 | |
| FABS | | 1 | 1 | 1 | | | | | 1 | 1 | |
| FCHS | | 1 | 1 | 1 | | | | | 1 | 1 | |
| FCOM(P) FUCOM | r | 1 | 1 | | 1 | | | | 3 | 1 | |
| FCOM(P) FUCOM | m | 1 | 1 | | 1 | | 1 | | | 1 | |
| FCOMPP FUCOMPP | | 2 | 2 | 1 | 1 | | | | | 1 | |
| FCOMI(P) FUCOMI(P) | r | 3 | 3 | 1 | 1 | 1 | | | 4 | 1 | |
| FIADD FISUB(R) | m | 2 | 2 | | | | 1 | | | 1 | |
| FIMUL | m | 2 | 2 | 1 | 1 | | 1 | | | 1 | |
| FIDIV(R) | m | 2 | 2 | 1 | 1 | | 1 | | | | |
| FICOM(P) | m | 2 | 2 | | 2 | | 1 | | | 2 | |
| FTST | | 1 | 1 | | 1 | | | | | 1 | |
| FXAM | | 2 | 2 | | 1 | | | | | 2 | |
| FPREM | | 28 | 28 | | | | | | 21 | 21 | |
| FPREM1 | | 41-87 | | | | | | | 26-50 | 26-50 | |
| FRNDINT | | 17 | 17 | | | | | | 22 | | |
| Math | | | | | | | | | | | |
| FSCALE | | 27 | 27 | | | | | | 12 | | |
| FXTRACT | | 17 | 17 | | | | | | 10 | | |
| FSQRT | | 1 | 1 | 1 | | | | | 10-24 | | |
| FSIN | | 64-100 | x | | | | | | 47-100 | | |
| FCOS | | 20-110 | x | | | | | | 47-115 | | |
| FSINCOS | | 20-110 | x | | | | | | 43-123 | | |
| F2XM1 | | 53-118 | x | | | | | | 61-69 | | |
| FYL2X | | | | | | | | | | | |
| FYL2XP1 | | | | | | | | | | | |
| FPTAN | | 102 | 102 | | | | | | 130 | | |
| FPATAN | | 28-91 | x | | | | | | 93-146 | | |
| Other | | | | | | | | | | | |
| FNOP | | 1 | 1 | 1 | | | | | | 1 | |
| WAIT | | 2 | 2 | | | | | | | 1 | |
| FNCLEX | | 5 | 5 | | | | | | | 22 | |
| FNINIT | | 26 | 26 | | | | | | | 81 | |

Integer MMX and XMM instructions

| Instruction | Operands | μops fused domain | μops unfused domain | | | | | | Latency | Reciprocal throughput | Comments |
|-------------------|-------------|-------------------|---------------------|----|----|----|-----|----|---------|-----------------------|----------|
| | | | p015 | p0 | p1 | p5 | p23 | p4 | | | |
| Move instructions | | | | | | | | | | | |
| MOVD | r32/64,mm/x | 1 | 1 | x | x | x | | | 1 | | |
| MOVD | m32/64,mm/x | 1 | | | | | 1 | 1 | 3 | 1 | |

Sandy Bridge

| | | | | | | | | | | | |
|-------------------|-------------|---|---|---|---|---|---|---|------|-----|--------|
| MOVD | mm/x,r32/64 | 1 | 1 | x | x | x | | | 1 | | |
| MOVD | mm/x,m32/64 | 1 | | | | | 1 | | 3 | 0.5 | |
| MOVQ | mm/x,mm/x | 1 | 1 | x | x | x | | | 1 | | |
| MOVQ | mm/x,m64 | 1 | | | | | 1 | | 3 | 0.5 | |
| MOVQ | m64, mm/x | 1 | | | | | 1 | 1 | 3 | 1 | |
| MOVDQA | x,x | 1 | 1 | x | x | x | | | 1 | | |
| MOVDQA | x, m128 | 1 | | | | | 1 | | 3 | 0.5 | |
| MOVDQA | m128, x | 1 | | | | | 1 | 1 | 3 | 1 | |
| MOVDQU | x, m128 | 1 | 1 | | | | 1 | | 3 | 0.5 | |
| MOVDQU | m128, x | 1 | 1 | | | | 1 | 1 | 3 | 1 | |
| LDDQU | x, m128 | 1 | 1 | | | | 1 | | 3 | 0.5 | SSE3 |
| MOVDQ2Q | mm, x | 2 | 2 | | | | | | 1 | 1 | |
| MOVQ2DQ | x,mm | 1 | 1 | | | | | | 1 | | |
| MOVNTQ | m64,mm | 1 | | | | | 1 | 1 | ~300 | 1 | |
| MOVNTDQ | m128,x | 1 | | | | | 1 | 1 | ~300 | | |
| MOVNTDQA | x, m128 | 1 | | | | | 1 | | | 0.5 | SSE4.1 |
| PACKSSWB/DW | | | | | | | | | | | |
| PACKUSWB | mm,mm | 1 | 1 | 1 | | | | | 1 | 1 | |
| PACKSSWB/DW | | | | | | | | | | | |
| PACKUSWB | mm,m64 | 1 | 1 | 1 | | | 1 | | | | |
| PACKSSWB/DW | | | | | | | | | | | |
| PACKUSWB | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PACKSSWB/DW | | | | | | | | | | | |
| PACKUSWB | x,m128 | 1 | 1 | | x | x | 1 | | | 0.5 | |
| PACKUSDW | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PACKUSDW | x,m | 1 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PUNPCKH/LBW/WD/DQ | mm/x,mm/x | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PUNPCKH/LBW/WD/DQ | mm/x,m | 1 | 1 | | x | x | 1 | | | 0.5 | |
| PUNPCKH/LQDQ | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PUNPCKH/LQDQ | x, m128 | 2 | 1 | | x | x | 1 | | | 0.5 | |
| PMOVSX/ZXBW | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PMOVSX/ZXBW | x,m64 | 1 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PMOVSX/ZXBD | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PMOVSX/ZXBD | x,m32 | 1 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PMOVSX/ZXBQ | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PMOVSX/ZXBQ | x,m16 | 1 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PMOVSX/ZXWD | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PMOVSX/ZXWD | x,m64 | 1 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PMOVSX/ZXWQ | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PMOVSX/ZXWQ | x,m32 | 1 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PMOVSX/ZXDQ | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PMOVSX/ZXDQ | x,m64 | 1 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PSHUFB | mm/x,mm/x | 1 | 1 | | x | x | | | 1 | 0.5 | SSSE3 |
| PSHUFB | mm/x,m | 2 | 1 | | x | x | 1 | | | 0.5 | SSSE3 |
| PSHUFW | mm,mm,i | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PSHUFW | mm,m64,i | 2 | 1 | | x | x | 1 | | | 0.5 | |
| PSHUFD | x,x,i | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PSHUFD | x,m128,i | 2 | 1 | | x | x | 1 | | | 0.5 | |
| PSHUFL/HW | x,x,i | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PSHUFL/HW | x, m128,i | 2 | 1 | | x | x | 1 | | | 0.5 | |
| PALIGNR | mm/x,mm/x,i | 1 | 1 | | x | x | | | 1 | 0.5 | SSSE3 |
| PALIGNR | mm/x,m,i | 2 | 1 | | x | x | 1 | | | 0.5 | SSSE3 |
| PBLENDVB | x,x,xmm0 | 2 | 2 | | 1 | 1 | | | 2 | 1 | SSE4.1 |
| PBLENDVB | x,m,xmm0 | 3 | 2 | | 1 | 1 | 1 | | | 1 | SSE4.1 |

Sandy Bridge

| | | | | | | | | | | | |
|--------------------------------|------------|----|---|---|---|---|---|---|---|------|---------|
| PBLENDW | x,x,i | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PBLENDW | x,m,i | 2 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| MASKMOVQ | mm,mm | 4 | 1 | 1 | | | 2 | 1 | | 1 | |
| MASKMOVDQU | x,x | 10 | 4 | | | | 4 | x | | 6 | |
| PMOVMASKB | r32,mm/x | 1 | 1 | 1 | | | | | 2 | 1 | |
| PEXTRB | r32,x,i | 2 | 2 | 1 | x | x | | | 2 | 1 | SSE4.1 |
| PEXTRB | m8,x,i | 2 | 1 | | x | x | 1 | 1 | | 1 | SSE4.1 |
| PEXTRW | r32,mm/x,i | 2 | 2 | 1 | x | x | | | 2 | 1 | |
| PEXTRW | m16,mm/x,i | 2 | 1 | | x | x | 1 | 1 | | 2 | SSE4.1 |
| PEXTRD | r32,x,i | 2 | 2 | 1 | x | x | | | 2 | 1 | SSE4.1 |
| PEXTRD | m32,x,i | 3 | 2 | 1 | x | x | 1 | 1 | | 1 | SSE4.1 |
| PEXTRQ | r64,x,i | 2 | 2 | 1 | x | x | | | 2 | 1 | SSE4.1, |
| PEXTRQ | m64,x,i | 3 | 2 | 1 | x | x | 1 | 1 | | 1 | 64b |
| PINSRB | x,r32,i | 2 | 2 | | x | x | | | 2 | 1 | SSE4.1 |
| PINSRB | x,m8,i | 2 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PINSRW | mm/x,r32,i | 2 | 2 | | x | x | | | 2 | 1 | |
| PINSRW | mm/x,m16,i | 2 | 1 | | x | x | 1 | | | 0.5 | |
| PINSRD | x,r32,i | 2 | 2 | | x | x | | | 2 | 1 | SSE4.1 |
| PINSRD | x,m32,i | 2 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PINSRQ | x,r64,i | 2 | 2 | | x | x | | | 2 | 1 | SSE4.1, |
| PINSRQ | x,m64,i | 2 | 1 | | x | x | 1 | | | 0.5 | 64 b |
| Arithmetic instructions | | | | | | | | | | | |
| PADD/SUB(U,S)B/W/D/Q | mm/x, mm/x | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PADD/SUB(U,S)B/W/D/Q | mm/x,m | 1 | 1 | | x | x | 1 | | | 0.5 | |
| PHADD/SUB(S)W/D | mm/x, mm/x | 3 | 3 | | x | x | | | 2 | 1.5 | SSSE3 |
| PHADD/SUB(S)W/D | mm/x,m64 | 4 | 3 | | x | x | 1 | | | 1.5 | SSSE3 |
| PCMPEQ/GTB/W/D | mm/x,mm/x | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PCMPEQ/GTB/W/D | mm/x,m | 1 | 1 | | x | x | 1 | | | 0.5 | |
| PCMPEQQ | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PCMPEQQ | x,m128 | 1 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PCMPGTQ | x,x | 1 | 1 | 1 | | | | | 5 | 1 | SSE4.2 |
| PCMPGTQ | x,m128 | 1 | 1 | 1 | | | 1 | | | 1 | SSE4.2 |
| PSUBxx, PCMPGTx | x,same | 1 | 0 | | | | | | 0 | 0.25 | |
| PCMPEQx | x,same | 1 | 1 | | | | | | 0 | 0.5 | |
| PMULL/HW PMULHUW | mm/x,mm/x | 1 | 1 | 1 | | | | | 5 | 1 | |
| PMULL/HW PMULHUW | mm/x,m | 1 | 1 | 1 | | | 1 | | | 1 | |
| PMULHSW | mm/x,mm/x | 1 | 1 | 1 | | | | | 5 | 1 | SSSE3 |
| PMULHSW | mm/x,m | 1 | 1 | 1 | | | 1 | | | 1 | SSSE3 |
| PMULLD | x,x | 1 | 1 | 1 | | | | | 5 | 1 | SSE4.1 |
| PMULLD | x,m128 | 2 | 1 | 1 | | | 1 | | | 1 | SSE4.1 |
| PMULDQ | x,x | 1 | 1 | 1 | | | | | 5 | 1 | SSE4.1 |
| PMULDQ | x,m128 | 1 | 1 | 1 | | | 1 | | | 1 | SSE4.1 |
| PMULUDQ | mm/x,mm/x | 1 | 1 | 1 | | | | | 5 | 1 | |
| PMULUDQ | mm/x,m | 1 | 1 | 1 | | | 1 | | | 1 | |
| PMADDWD | mm/x,mm/x | 1 | 1 | 1 | | | | | 5 | 1 | |
| PMADDWD | mm/x,m | 1 | 1 | 1 | | | 1 | | | 1 | |
| PMADDUBSW | mm/x,mm/x | 1 | 1 | 1 | | | | | 5 | 1 | SSSE3 |
| PMADDUBSW | mm/x,m | 1 | 1 | 1 | | | 1 | | | 1 | SSSE3 |
| PAVGB/W | mm/x,mm/x | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PAVGB/W | mm/x,m | 1 | 1 | | x | x | 1 | | | 0.5 | |
| PMIN/MAXSB | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PMIN/MAXSB | x,m128 | 1 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |

Sandy Bridge

| | | | | | | | | | | | |
|---|-----------|----|----|---|---|---|---|--|-------|------|--------|
| PMIN/MAXUB | mm/x,mm/x | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PMIN/MAXUB | mm/x,m | 1 | 1 | | x | x | 1 | | | 0.5 | |
| PMIN/MAXSW | mm/x,mm/x | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PMIN/MAXSW | mm/x,m | 1 | 1 | | x | x | 1 | | | 0.5 | |
| PMIN/MAXUW | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PMIN/MAXUW | x,m | 1 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PMIN/MAXU/SD | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PMIN/MAXU/SD | x,m128 | 1 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PHMINPOSUW | x,x | 1 | 1 | 1 | | | | | 5 | 1 | SSE4.1 |
| PHMINPOSUW | x,m128 | 1 | 1 | 1 | | | 1 | | | 1 | SSE4.1 |
| PABSB/W/D | mm/x,mm/x | 1 | 1 | | x | x | | | 1 | 0.5 | SSSE3 |
| PABSB/W/D | mm/x,m | 1 | 1 | | x | x | 1 | | | 0.5 | SSSE3 |
| PSIGNB/W/D | mm/x,mm/x | 1 | 1 | | x | x | | | 1 | 0.5 | SSSE3 |
| PSIGNB/W/D | mm/x,m | 1 | 1 | | x | x | 1 | | | 0.5 | SSSE3 |
| PSADBW | mm/x,mm/x | 1 | 1 | 1 | | | | | 5 | 1 | |
| PSADBW | mm/x,m | 1 | 1 | 1 | | | 1 | | | 1 | |
| MPSADBW | x,x,i | 3 | 3 | 1 | 1 | 1 | | | 6 | 1 | SSE4.1 |
| MPSADBW | x,m,i | 4 | 3 | 1 | 1 | 1 | 1 | | | 1 | SSE4.1 |
| Logic instructions | | | | | | | | | | | |
| PAND(N) POR PXOR | mm/x,mm/x | 1 | 1 | x | x | x | | | 1 | | |
| PAND(N) POR PXOR | mm/x,m | 1 | 1 | x | x | x | 1 | | | 0.5 | |
| PXOR | x,same | 1 | 0 | | | | | | 0 | 0.25 | |
| PTEST | x,x | 1 | 2 | 1 | x | x | | | 1 | 1 | SSE4.1 |
| PTEST | x,m128 | 1 | 2 | 1 | x | x | 1 | | | 1 | SSE4.1 |
| PSLL/RL/RAW/D/Q | mm,mm/i | 1 | 1 | 1 | | | | | 1 | 1 | |
| PSLL/RL/RAW/D/Q | mm,m64 | 1 | 1 | 1 | | | 1 | | | 2 | |
| PSLL/RL/RAW/D/Q | x,i | 1 | 1 | 1 | | | | | 1 | 1 | |
| PSLL/RL/RAW/D/Q | x,x | 2 | 2 | 1 | x | x | | | 2 | 1 | |
| PSLL/RL/RAW/D/Q | x,m128 | 3 | 2 | 1 | x | x | 1 | | | 1 | |
| PSLL/RLDQ | x,i | 1 | 1 | | x | x | | | 1 | 1 | |
| String instructions | | | | | | | | | | | |
| PCMPESTRI | x,x,i | 8 | 8 | | | | | | 4 | 4 | SSE4.2 |
| PCMPESTRI | x,m128,i | 8 | 7 | | | | 1 | | | 4 | SSE4.2 |
| PCMPESTRM | x,x,i | 8 | 8 | | | | | | 11-12 | 4 | SSE4.2 |
| PCMPESTRM | x,m128,i | 8 | 7 | | | | 1 | | | 4 | SSE4.2 |
| PCMPISTRI | x,x,i | 3 | 3 | | | | | | 3 | 3 | SSE4.2 |
| PCMPISTRI | x,m128,i | 4 | 3 | | | | 1 | | | 3 | SSE4.2 |
| PCMPISTRM | x,x,i | 3 | 3 | | | | | | 11 | 3 | SSE4.2 |
| PCMPISTRM | x,m128,i | 4 | 3 | | | | 1 | | | 3 | SSE4.2 |
| Encryption instructions | | | | | | | | | | | |
| PCLMULQDQ | x,x,i | 18 | 18 | | | | | | 14 | 8 | CLMUL |
| AESDEC, AESDECLAST, AESENC, AESENCLAST | | | | | | | | | | | |
| | x,x | 2 | 2 | | | | | | 8 | 4 | AES |
| AESIMC | x,x | 2 | 2 | | | | | | | 2 | AES |
| AESKEYGENASSIST | x,x,i | 11 | 11 | | | | | | 8 | 8 | AES |
| Other | | | | | | | | | | | |
| EMMS | | 31 | 31 | | | | | | | 18 | |

Floating point XMM and YMM instructions

| Instruction | Operands | μops fused domain | μops unfused domain | | | | | | Latency | Reciprocal throughput | Comments |
|---------------------|-------------|-------------------|---------------------|----|----|----|-----|----|---------|-----------------------|----------|
| | | | p015 | p0 | p1 | p5 | p23 | p4 | | | |
| Move instructions | | | | | | | | | | | |
| MOVAPS/D | x,x | 1 | 1 | | | 1 | | | 1 | 1 | AVX |
| VMOVAPS/D | y,y | 1 | 1 | | | 1 | | | 1 | 1 | |
| MOVAPS/D MOVUPS/D | x,m128 | 1 | | | | | 1 | | 3 | 0.5 | |
| VMOVAPS/D VMOVUPS/D | y,m256 | 1 | | | | | 1+ | | 4 | 1 | AVX |
| MOVAPS/D MOVUPS/D | m128,x | 1 | | | | | 1 | 1 | 3 | 1 | |
| VMOVAPS/D VMOVUPS/D | m256,y | 1 | | | | | 1 | 1+ | 3 | 1 | AVX |
| MOVSS/D | x,x | 1 | 1 | | | 1 | | | 1 | 1 | |
| MOVSS/D | x,m32/64 | 1 | | | | | 1 | | 3 | 0.5 | |
| MOVSS/D | m32/64,x | 1 | | | | | 1 | 1 | 3 | 1 | AVX |
| MOVHPS/D MOVLPS/D | x,m64 | 1 | 1 | | | 1 | 1 | | 3 | 1 | |
| MOVH/LPS/D | m64,x | 1 | | | | | 1 | 1 | 3 | 1 | |
| MOVLHPS MOVHLPS | x,x | 1 | 1 | | | 1 | | | 1 | 1 | AVX |
| MOVMSKPS/D | r32,x | 1 | 1 | 1 | | | | | 2 | 1 | |
| VMOVMSKPS/D | r32,y | 1 | 1 | 1 | | | | | 2 | 1 | |
| MOVNTPS/D | m128,x | 1 | | | | | 1 | 1 | ~300 | 1 | AVX |
| VMOVNTPS/D | m256,y | 1 | | | | | 1 | 4 | ~300 | 25 | |
| SHUFPS/D | x,x,i | 1 | 1 | | | 1 | | | 1 | 1 | |
| SHUFPS/D | x,m128,i | 2 | 1 | | | 1 | 1 | | | 1 | AVX |
| VSHUFPS/D | y,y,y,i | 1 | 1 | | | 1 | | | 1 | 1 | |
| VSHUFPS/D | y, y,m256,i | 2 | 1 | | | 1 | 1+ | | | 1 | |
| VPERMILPS/PD | x,x,x/i | 1 | 1 | | | 1 | | | 1 | 1 | AVX |
| VPERMILPS/PD | y,y,y/i | 1 | 1 | | | 1 | | | 1 | 1 | |
| VPERMILPS/PD | x,x,m | 2 | 1 | | | 1 | 1 | | | 1 | |
| VPERMILPS/PD | y,y,m | 2 | | | | 1 | 1+ | | | 1 | AVX |
| VPERMILPS/PD | x,m,i | 2 | 1 | | | 1 | 1 | | | 1 | |
| VPERMILPS/PD | y,m,i | 2 | 1 | | | 1 | 1+ | | | 1 | |
| VPERM2F128 | y,y,y,i | 1 | 1 | | | 1 | | | 2 | 1 | AVX |
| VPERM2F128 | y,y,m,i | 2 | 1 | | | 1 | 1+ | | | 1 | |
| BLENDPS/PD | x,x,i | 1 | 1 | x | | x | | | 1 | 0.5 | SSE4.1 |
| BLENDPS/PD | x,m128,i | 2 | 1 | x | | x | 1 | | | 0.5 | |
| VBLENDPS/PD | y,y,i | 1 | 1 | x | | x | | | 1 | 1 | |
| VBLENDPS/PD | y,m256,i | 2 | 1 | x | | x | 1+ | | | 1 | AVX |
| BLENDVPS/PD | x,x,xmm0 | 2 | 2 | x | | x | | | 2 | 1 | |
| BLENDVPS/PD | x,m,xmm0 | 3 | 2 | x | | x | 1 | | | 1 | |
| VBLENDVPS/PD | y,y,y,y | 2 | 2 | x | | x | | | 2 | 1 | AVX |
| VBLENDVPS/PD | y,y,m,y | 3 | 2 | x | | x | 1+ | | | 1 | |
| MOVDDUP | x,x | 1 | 1 | | | 1 | | | 1 | 1 | SSE3 |
| MOVDDUP | x,m64 | 1 | | | | | 1 | | 3 | 0.5 | |
| VMOVDDUP | y,y | 1 | 1 | | | 1 | | | 1 | 1 | |
| VMOVDDUP | y,m256 | 1 | | | | | 1+ | | 3 | 1 | AVX |
| VBROADCASTSS | x,m32 | 1 | | | | | 1 | | | 1 | |
| VBROADCASTSS | y,m32 | 2 | 1 | | | 1 | 1 | | | 1 | |
| VBROADCASTSD | y,m64 | 2 | 1 | | | 1 | 1 | | | 1 | AVX |

Sandy Bridge

| | | | | | | | | | | | |
|-------------------|------------|---|---|---|--|---|----|----|---|-----|--------|
| VBROADCASTF128 | y,m128 | 2 | 1 | | | 1 | 1 | | | 1 | AVX |
| MOVSH/LDUP | x,x | 1 | 1 | | | 1 | | | 1 | 1 | SSE3 |
| MOVSH/LDUP | x,m128 | 1 | | | | | 1 | | 3 | 0.5 | SSE3 |
| VMOVSH/LDUP | y,y | 1 | 1 | | | 1 | | | 1 | 1 | AVX |
| VMOVSH/LDUP | y,m256 | 1 | | | | | 1+ | | 4 | 1 | AVX |
| UNPCKH/LPS/D | x,x | 1 | 1 | | | 1 | | | 1 | 1 | SSE3 |
| UNPCKH/LPS/D | x,m128 | 1 | 1 | | | 1 | 1 | | | 1 | SSE3 |
| VUNPCKH/LPS/D | y,y,y | 1 | 1 | | | 1 | | | 1 | 1 | AVX |
| VUNPCKH/LPS/D | y,y,m256 | 1 | 1 | | | 1 | 1+ | | | 1 | AVX |
| EXTRACTPS | r32,x,i | 2 | 2 | 1 | | 1 | | | 2 | 1 | SSE4.1 |
| EXTRACTPS | m32,x,i | 3 | 2 | | | 1 | 1 | 1 | | 1 | SSE4.1 |
| VEXTRACTF128 | x,y,i | 1 | 1 | | | 1 | | | 2 | 1 | AVX |
| VEXTRACTF128 | m128,y,i | 2 | 1 | | | | 1 | 1 | | 1 | AVX |
| INSERTPS | x,x,i | 1 | 1 | | | 1 | | | 1 | 1 | SSE4.1 |
| INSERTPS | x,m32,i | 2 | 1 | | | 1 | 1 | | | 1 | SSE4.1 |
| VINSERTF128 | y,y,x,i | 1 | 1 | | | 1 | | | 2 | 1 | AVX |
| VINSERTF128 | y,y,m128,i | 2 | 1 | | | 1 | 1 | | | 1 | AVX |
| VMASKMOVPS/D | x,x,m128 | 3 | 2 | | | | 1 | | | 1 | AVX |
| VMASKMOVPS/D | y,y,m256 | 3 | 2 | | | | 1+ | | | 1 | AVX |
| VMASKMOVPS/D | m128,x,x | 4 | 2 | | | | 1 | 1 | | 1 | AVX |
| VMASKMOVPS/D | m256,y,y | 4 | 2 | | | | 1 | 1+ | | 2 | AVX |
| Conversion | | | | | | | | | | | |
| CVTPD2PS | x,x | 2 | 2 | | | 1 | 1 | | 4 | 1 | |
| CVTPD2PS | x,m128 | 2 | 2 | | | 1 | 1 | 1 | | 1 | |
| VCVTPD2PS | x,y | 2 | 2 | | | 1 | 1 | | 4 | 1 | AVX |
| VCVTPD2PS | x,m256 | 2 | 2 | | | 1 | 1 | 1+ | | 1 | AVX |
| CVTSD2SS | x,x | 2 | 2 | | | 1 | 1 | | 3 | 1 | |
| CVTSD2SS | x,m64 | 2 | 2 | | | 1 | 1 | 1 | | 1 | |
| CVTPS2PD | x,x | 2 | 2 | 1 | | | 1 | | 3 | 1 | |
| CVTPS2PD | x,m64 | 2 | 1 | 1 | | | | 1 | | 1 | |
| VCVTPS2PD | y,x | 2 | 2 | 1 | | | 1 | | 4 | 1 | AVX |
| VCVTPS2PD | y,m128 | 3 | 2 | 1 | | | 1 | 1 | | 1 | AVX |
| CVTSS2SD | x,x | 2 | 2 | 1 | | | 1 | | 3 | 1 | |
| CVTSS2SD | x,m32 | 2 | 1 | 1 | | | | 1 | | 1 | |
| CVTDQ2PS | x,x | 1 | 1 | | | 1 | | | 3 | 1 | |
| CVTDQ2PS | x,m128 | 1 | 1 | | | 1 | | 1 | | 1 | |
| VCVTDQ2PS | y,y | 1 | 1 | | | 1 | | | 3 | 1 | AVX |
| VCVTDQ2PS | y,m256 | 1 | 1 | | | 1 | | 1+ | | 1 | AVX |
| CVT(T) PS2DQ | x,x | 1 | 1 | | | 1 | | | 3 | 1 | |
| CVT(T) PS2DQ | x,m128 | 1 | 1 | | | 1 | | 1 | | 1 | |
| VCVT(T) PS2DQ | y,y | 1 | 1 | | | 1 | | | 3 | 1 | AVX |
| VCVT(T) PS2DQ | y,m256 | 1 | 1 | | | 1 | | 1+ | | 1 | AVX |
| CVTDQ2PD | x,x | 2 | 2 | | | 1 | 1 | | 4 | 1 | |
| CVTDQ2PD | x,m64 | 2 | 2 | | | 1 | 1 | 1 | | 1 | |
| VCVTDQ2PD | y,x | 2 | 2 | | | 1 | 1 | | 5 | 1 | AVX |
| VCVTDQ2PD | y,m128 | 3 | 2 | | | 1 | 1 | 1 | | 1 | AVX |
| CVT(T)PD2DQ | x,x | 2 | 2 | | | 1 | 1 | | 4 | 1 | |
| CVT(T)PD2DQ | x,m128 | 2 | 2 | | | 1 | 1 | 1 | | 1 | |
| VCVT(T)PD2DQ | x,y | 2 | 2 | | | 1 | 1 | | 5 | 1 | AVX |
| VCVT(T)PD2DQ | x,m256 | 2 | 2 | | | 1 | 1 | 1+ | | 1 | AVX |
| CVTPI2PS | x,mm | 1 | 1 | | | 1 | | | 4 | 2 | |
| CVTPI2PS | x,m64 | 1 | 1 | | | 1 | | 1 | | 2 | |

Sandy Bridge

| | | | | | | | | | | |
|---------------------|----------|---|---|---|---|---|----|-------|-------|------|
| CVT(T)PS2PI | mm,x | 2 | 2 | | 1 | 1 | | 4 | 1 | |
| CVT(T)PS2PI | mm,m128 | 2 | 1 | | 1 | | 1 | | 1 | |
| CVTPI2PD | x,mm | 2 | 2 | | 1 | 1 | | 4 | 1 | |
| CVTPI2PD | x,m64 | 2 | 2 | | 1 | 1 | 1 | | 1 | |
| CVT(T) PD2PI | mm,x | 2 | 2 | | | | | 4 | 1 | |
| CVT(T) PD2PI | mm,m128 | 2 | 2 | | | | 1 | | 1 | |
| CVTSI2SS | x,r32 | 2 | 2 | | 1 | 1 | | 4 | 1.5 | |
| CVTSI2SS | x,m32 | 1 | 1 | | 1 | | 1 | | 1.5 | |
| CVT(T)SS2SI | r32,x | 2 | 2 | 1 | 1 | | | 4 | 1 | |
| CVT(T)SS2SI | r32,m32 | 2 | 2 | | 1 | | 1 | | 1 | |
| CVTSI2SD | x,r32 | 2 | 2 | 1 | 1 | | | 4 | 1.5 | |
| CVTSI2SD | x,m32 | 1 | 1 | | 1 | | 1 | | 1.5 | |
| CVT(T)SD2SI | r32,x | 2 | 2 | 1 | 1 | | | 4 | 1 | |
| CVT(T)SD2SI | r32,m64 | 2 | 2 | 1 | 1 | | 1 | | 1 | |
| Arithmetic | | | | | | | | | | |
| ADDSS/D SUBSS/D | x,x | 1 | 1 | | 1 | | | 3 | 1 | |
| ADDSS/D SUBSS/D | x,m32/64 | 1 | 1 | | 1 | | 1 | | 1 | |
| ADDPS/D SUBPS/D | x,x | 1 | 1 | | 1 | | | 3 | 1 | |
| ADDPS/D SUBPS/D | x,m128 | 1 | 1 | | 1 | | 1 | | 1 | |
| VADDPS/D VSUBPS/D | y,y,y | 1 | 1 | | 1 | | | 3 | 1 | AVX |
| VADDPS/D VSUBPS/D | y,y,m256 | 1 | 1 | | 1 | | 1+ | | 1 | AVX |
| ADDSUBPS/D | x,x | 1 | 1 | | 1 | | | 3 | 1 | SSE3 |
| ADDSUBPS/D | x,m128 | 1 | 1 | | 1 | | 1 | | 1 | SSE3 |
| VADDSUBPS/D | y,y,y | 1 | 1 | | 1 | | | 3 | 1 | AVX |
| VADDSUBPS/D | y,y,m256 | 1 | 1 | | 1 | | 1+ | | 1 | AVX |
| HADDPS/D HSUBPS/D | x,x | 3 | 3 | | 1 | 2 | | 5 | 2 | SSE3 |
| HADDPS/D HSUBPS/D | x,m128 | 4 | 3 | | 1 | 2 | 1 | | 2 | SSE3 |
| VHADDPS/D VHSUBPS/D | y,y,y | 3 | 3 | | 1 | 2 | | 5 | 2 | AVX |
| VHADDPS/D VHSUBPS/D | y,y,m256 | 4 | 3 | | 1 | 2 | 1+ | | 2 | AVX |
| MULSS MULPS | x,x | 1 | 1 | 1 | | | | 5 | 1 | |
| MULSS MULPS | x,m | 1 | 1 | 1 | | | 1 | | 1 | |
| VMULPS | y,y,y | 1 | 1 | 1 | | | | 5 | 1 | AVX |
| VMULPS | y,y,m256 | 1 | 1 | 1 | | | 1+ | | 1 | AVX |
| MULSD MULPD | x,x | 1 | 1 | 1 | | | | 5 | 1 | |
| MULSD MULPD | x,m | 1 | 1 | 1 | | | 1 | | 1 | |
| VMULPD | y,y,y | 1 | 1 | 1 | | | | 5 | 1 | AVX |
| VMULPD | y,y,m256 | 1 | 1 | 1 | | | 1+ | | 1 | AVX |
| DIVSS DIVPS | x,x | 1 | 1 | 1 | | | | 10-14 | 10-14 | |
| DIVSS DIVPS | x,m | 1 | 1 | 1 | | | 1 | | 10-14 | |
| VDIVPS | y,y,y | 3 | 3 | 2 | | 1 | | 21-29 | 20-28 | AVX |
| VDIVPS | y,y,m256 | 4 | 3 | 2 | | 1 | 1+ | | 20-28 | AVX |
| DIVSD DIVPD | x,x | 1 | 1 | 1 | | | | 10-22 | 10-22 | |
| DIVSD DIVPD | x,m | 1 | 1 | 1 | | | 1 | | 10-22 | |
| VDIVPD | y,y,y | 3 | 3 | 2 | | 1 | | 21-45 | 20-44 | AVX |
| VDIVPD | y,y,m256 | 4 | 3 | 2 | | 1 | 1+ | | 20-44 | AVX |
| RCPSS/PS | x,x | 1 | 1 | 1 | | | | 5 | 1 | |
| RCPSS/PS | x,m128 | 1 | 1 | 1 | | | 1 | | 1 | |
| VRCPPS | y,y | 3 | 3 | 2 | | 1 | | 7 | 2 | AVX |
| VRCPPS | y,m256 | 4 | 3 | | | | 1+ | | 2 | AVX |

Sandy Bridge

| | | | | | | | | | |
|-----------------------|--------------|----|---|---|----|----|-------|-------|-------------|
| CMPccSS/D CMPccPS/D | x,x | 1 | 1 | 1 | | | 3 | 1 | |
| CMPccSS/D CMPccPS/D | x,m128 | 2 | 1 | 1 | 1 | | | 1 | |
| VCMPccPS/D | y,y,y | 1 | 1 | 1 | | | 3 | 1 | AVX |
| VCMPccPS/D | y,y,m256 | 2 | 1 | 1 | 1+ | | | 1 | AVX |
| COMISS/D UCOMISS/D | x,x | 2 | 2 | 1 | 1 | | 2 | 1 | |
| COMISS/D UCOMISS/D | x,m32/64 | 2 | 2 | 1 | 1 | 1 | | 1 | |
| MAXSS/D MINSS/D | x,x | 1 | 1 | 1 | | | 3 | 1 | |
| MAXSS/D MINSS/D | x,m32/64 | 1 | 1 | 1 | 1 | | | 1 | |
| MAXPS/D MINPS/D | x,x | 1 | 1 | 1 | | | 3 | 1 | |
| MAXPS/D MINPS/D | x,m128 | 1 | 1 | 1 | 1 | | | 1 | |
| VMAXPS/D VMINPS/D | y,y,y | 1 | 1 | 1 | | | 3 | 1 | AVX |
| VMAXPS/D VMINPS/D | y,y,m256 | 1 | 1 | 1 | 1+ | | | 1 | AVX |
| ROUNDSS/SD/PS/PD | x,x,i | 1 | 1 | 1 | | | 3 | 1 | SSE4.1 |
| ROUNDSS/SD/PS/PD | x,m128,i | 2 | 1 | 1 | 1 | | | 1 | SSE4.1 |
| VROUNDSS/SD/PS/PD | y,y,i | 1 | 1 | 1 | | | 3 | 1 | AVX |
| VROUNDSS/SD/PS/PD | y,m256,i | 2 | 1 | 1 | 1+ | | | 1 | AVX |
| DPPS | x,x,i | 4 | 4 | 1 | 2 | 1 | 12 | 2 | SSE4.1 |
| DPPS | x,m128,i | 6 | 5 | | | 1 | | 4 | SSE4.1 |
| VDPPS | y,y,y,i | 4 | 4 | 1 | 2 | 1 | 12 | 2 | AVX |
| VDPPS | y,m256,i | 6 | 5 | | | 1+ | | 4 | AVX |
| DPPD | x,x,i | 3 | 3 | 1 | 1 | 1 | 9 | 2 | SSE4.1 |
| DPPD | x,m128,i | 4 | 3 | | | 1 | | 2 | SSE4.1 |
| Math | | | | | | | | | |
| SQRTSS/PS | x,x | 1 | 1 | 1 | | | 10-14 | 10-14 | |
| SQRTSS/PS | x,m128 | 1 | 1 | 1 | | 1 | | 10-14 | |
| VSQRTPS | y,y | 3 | 3 | | | | | 21-28 | AVX |
| VSQRTPS | y,m256 | 4 | 3 | | | 1+ | | 21-28 | AVX |
| SQRTSD/PD | x,x | 1 | 1 | 1 | | | 10-21 | 10-21 | |
| SQRTSD/PD | x,m128 | 2 | 1 | 1 | | 1 | | 10-21 | |
| VSQRTPD | y,y | 3 | 3 | | | | 21-43 | 21-43 | AVX |
| VSQRTPD | y,m256 | 4 | 3 | | | 1+ | | 21-43 | AVX |
| RSQRTSS/PS | x,x | 1 | 1 | 1 | | | 5 | 1 | |
| RSQRTSS/PS | x,m128 | 1 | 1 | 1 | | 1 | | 1 | |
| VRSQRTPS | y,y | 3 | 3 | | | | 7 | 2 | AVX |
| VRSQRTPS | y,m256 | 4 | 3 | | | 1+ | | 2 | AVX |
| Logic | | | | | | | | | |
| AND/ANDN/OR/XORPS/PD | x,x | 1 | 1 | | 1 | | 1 | 1 | |
| AND/ANDN/OR/XORPS/PD | x,m128 | 1 | 1 | | 1 | 1 | | 1 | |
| VAND/ANDN/OR/XORPS/PD | y,y,y | 1 | 1 | | 1 | | 1 | 1 | AVX |
| VAND/ANDN/OR/XORPS/PD | y,y,m256 | 1 | 1 | | 1 | 1+ | | 1 | AVX |
| (V)XORPS/PD | x/y,x/y,same | 1 | 0 | | | | 0 | 0.25 | |
| Other | | | | | | | | | |
| VZEROUPPER | | 4 | | | | | 2 | 1 | AVX |
| VZEROALL | | 12 | | | | | | 11 | AVX, 32 bit |
| VZEROALL | | 20 | | | | | | 9 | AVX, 64 bit |

Sandy Bridge

| | | | | | | | | | |
|----------|-------|---------|---|---|---|---|--------|----|-----|
| LDMXCSR | m32 | 3 | 3 | | | 1 | | 3 | |
| STMXCSR | m32 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | |
| VSTMXCSR | m32 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | AVX |
| FXSAVE | m4096 | 130 | | | | | | 68 | |
| FXRSTOR | m4096 | 116 | | | | | | 72 | |
| XSAVEOPT | m | 100-161 | | | | | 60-500 | | |

Intel Ivy Bridge

List of instruction timings and μ op breakdown

Explanation of column headings:

| | |
|--|---|
| Operands: | i = immediate data, r = register, mm = 64 bit mmx register, x = 128 bit xmm register, mm/x = mmx or xmm register, y = 256 bit ymm register, same = same register for both operands. m = memory operand, m32 = 32-bit memory operand, etc. |
| μops fused domain: | The number of μ ops at the decode, rename, allocate and retirement stages in the pipeline. Fused μ ops count as one. |
| μops unfused domain: | The number of μ ops for each execution port. Fused μ ops count as two. Fused macro-ops count as one. The instruction has μ op fusion if the sum of the numbers listed under p015 + p23 + p4 exceeds the number listed under μ ops fused domain. A number indicated as 1+ under a read or write port means a 256-bit read or write operation using two clock cycles for handling 128 bits each cycle. The port cannot receive another read or write μ op in the second clock cycle, but a read port can receive an address-calculation μ op in the second clock cycle. An x under p0, p1 or p5 means that at least one of the μ ops listed under p015 can optionally go to this port. For example, a 1 under p015 and an x under p0 and p5 means one μ op which can go to either port 0 or port 5, whichever is vacant first. A value listed under p015 but nothing under p0, p1 and p5 means that it is not known which of the three ports these μ ops go to. |
| p015: | The total number of μ ops going to port 0, 1 and 5. |
| p0: | The number of μ ops going to port 0 (execution units). |
| p1: | The number of μ ops going to port 1 (execution units). |
| p5: | The number of μ ops going to port 5 (execution units). |
| p23: | The number of μ ops going to port 2 or 3 (memory read or address calculation). |
| p4: | The number of μ ops going to port 4 (memory write data). |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Where hyperthreading is enabled, the use of the same execution units in the other thread leads to inferior performance. Denormal numbers, NAN's and infinity do not increase the latency. The time unit used is core clock cycles, not the reference clock cycles given by the time stamp counter. |
| Reciprocal throughput: | The average number of core clock cycles per instruction for a series of independent instructions of the same kind in the same thread. |

The latencies and throughputs listed below for addition and multiplication using full size YMM registers are obtained only after a warm-up period of a thousand instructions or more. The latencies may be one or two clock cycles longer and the reciprocal throughputs double the values for shorter sequences of code. There is no warm-up effect when vectors are 128 bits wide or less.

Integer instructions

| Instruction | Operands | μ ops fused domain | μ ops unfused domain | | | | | | Latency | Reciprocal throughput | Comments |
|-------------------|-------------|------------------------|--------------------------|----|----|----|-----|----|---------|-----------------------|----------|
| | | | p015 | p0 | p1 | p5 | p23 | p4 | | | |
| Move instructions | | | | | | | | | | | |
| MOV | r,i | 1 | 1 | x | x | x | | | 1 | 0.33 | |
| MOV | r8/16.r8/16 | 1 | 1 | x | x | x | | | 1 | 0.33 | |

Ivy Bridge

| | | | | | | | | | | | |
|--------------------------------|---------------|----|----|---|---|---|---|---|------|------|-----------------------------|
| MOV | r32/64,r32/64 | 1 | 1 | x | x | x | | | 0-1 | 0.25 | may be elimin. |
| MOV | r8/16,m8/16 | 1 | 1 | x | x | x | 1 | | 2 | 0.5 | |
| MOV | r32/64,m32/64 | 1 | | | | | 1 | | 2 | 0.5 | |
| MOV | r,m | 1 | | | | | 1 | | 2 | 1 | 64 b abs address |
| MOV | m,r | 1 | | | | | 1 | 1 | 3 | 1 | |
| MOV | m,i | 1 | | | | | 1 | 1 | | 1 | |
| MOVNTI | m,r | 2 | | | | | 1 | 1 | ~340 | 1 | |
| MOVSX MOVSD | r,r | 1 | 1 | x | x | x | | | 1 | 0.33 | |
| MOVZX | r16,r8 | 1 | 1 | x | x | x | | | 1 | 0.33 | |
| MOVZX | r32/64,r8 | 1 | 1 | x | x | x | | | 0-1 | 0.25 | may be elimin. |
| MOVZX | r32/64,r16 | 1 | 1 | x | x | x | | | 1 | 0.33 | |
| MOVSX MOVZX | r16,m8 | 2 | 1 | x | x | x | 1 | | 3 | 0.5 | |
| MOVSX MOVZX | r32/64,m | 1 | | | | | 1 | | 2 | 0.5 | |
| MOVSD | | | | | | | | | | | |
| CMOVcc | r,r | 2 | 2 | x | x | x | | | 2 | 0.67 | |
| CMOVcc | r,m | 2 | 2 | x | x | x | 1 | | | ~0.8 | |
| XCHG | r,r | 3 | 3 | x | x | x | | | 2 | 1 | |
| XCHG | r,m | 7 | x | | | | 2 | 3 | 25 | | implicit lock |
| XLAT | | 3 | 2 | | | | 1 | | 7 | 1 | |
| PUSH | r | 1 | | | | | 1 | 1 | 3 | 1 | |
| PUSH | i | 1 | | | | | 1 | 1 | | 1 | |
| PUSH | m | 2 | | | | | 2 | 1 | | 1 | |
| PUSH | (E/R)SP | 2 | 1 | x | x | x | 1 | 1 | 3 | 1 | |
| PUSHF(D/Q) | | 3 | 2 | x | x | x | 1 | 1 | | 1 | |
| PUSHA(D) | | 19 | 3 | x | x | x | 8 | 8 | | 8 | not 64 bit |
| POP | r | 1 | | | | | 1 | | 2 | 0.5 | |
| POP | (E/R)SP | 3 | 2 | x | x | x | 1 | | | 0.5 | |
| POP | m | 2 | | | | | 2 | 1 | | 1 | |
| POPF(D/Q) | | 9 | 8 | x | x | x | 1 | | | 18 | |
| POPA(D) | | 18 | 10 | x | x | x | 8 | | | 9 | not 64 bit |
| LAHF SAHF | | 1 | 1 | x | | x | | | 1 | 1 | |
| SALC | | 3 | 3 | x | x | x | | | 1 | 1 | not 64 bit |
| LEA | r16,m | 2 | 2 | x | 1 | x | | | 2-4 | 1 | |
| LEA | r32/64,m | 1 | 1 | x | x | | | | 1 | 0.5 | 1-2 com- ponents |
| LEA | r32/64,m | 1 | 1 | | 1 | | | | 3 | 1 | 3 com- ponents or RIP |
| BSWAP | r32 | 1 | 1 | | 1 | | | | 1 | 1 | |
| BSWAP | r64 | 2 | 2 | x | 1 | x | | | 2 | 1 | |
| PREFETCHNTA | m | 1 | | | | | 1 | | | 43 | |
| PREFETCHT0/1/2 | m | 1 | | | | | 1 | | | 43 | |
| LFENCE | | 2 | | | | | | | | 4 | |
| MFENCE | | 3 | | | | | 1 | 1 | | 36 | |
| SFENCE | | 2 | | | | | 1 | 1 | | 6 | |
| Arithmetic instructions | | | | | | | | | | | |
| ADD SUB | r,r/i | 1 | 1 | x | x | x | | | 1 | 0.33 | |
| ADD SUB | r,m | 1 | 1 | x | x | x | 1 | | | 0.5 | |
| ADD SUB | m,r/i | 2 | 1 | x | x | x | 2 | 1 | 6 | 1 | |

Ivy Bridge

| | | | | | | | | | | | |
|---------------------------|-----------|--------|----|---|---|---|---|---|--------|-------|------------|
| ADC SBB | r,r/i | 2 | 2 | x | x | x | | | 2 | 1 | |
| ADC SBB | r,m | 2 | 2 | x | x | x | 1 | | 2 | 1 | |
| ADC SBB | m,r/i | 4 | 3 | x | x | x | 2 | 1 | 7-8 | 2 | |
| CMP | r,r/i | 1 | 1 | x | x | x | | | 1 | 0.33 | |
| CMP | m,r/i | 1 | 1 | x | x | x | 1 | | 1 | 0.5 | |
| INC DEC NEG NOT | r | 1 | 1 | x | x | x | | | 1 | 0.33 | |
| INC DEC NEG NOT | m | 3 | 1 | x | x | x | 2 | 1 | 6 | 1 | |
| AAA AAS | | 2 | 2 | x | 1 | x | | | 4 | | not 64 bit |
| DAA DAS | | 3 | 3 | | | | | | 4 | | not 64 bit |
| AAD | | 3 | 3 | | | | | | 2 | | not 64 bit |
| AAM | | 8 | 8 | | | | | | 20 | 8 | not 64 bit |
| MUL IMUL | r8 | 1 | 1 | | 1 | | | | 3 | 1 | |
| MUL IMUL | r16 | 4 | 4 | | | | | | 4 | 2 | |
| MUL IMUL | r32 | 3 | 3 | | | | | | 4 | 2 | |
| MUL IMUL | r64 | 2 | 2 | | | | | | 3 | 1 | |
| IMUL | r,r | 1 | 1 | | 1 | | | | 3 | 1 | |
| IMUL | r16,r16,i | 2 | 2 | | | | | | 4 | 1 | |
| IMUL | r32,r32,i | 1 | 1 | | 1 | | | | 3 | 1 | |
| IMUL | r64,r64,i | 1 | 1 | | 1 | | | | 3 | 1 | |
| MUL IMUL | m8 | 1 | 1 | | 1 | | 1 | | 3 | 1 | |
| MUL IMUL | m16 | 4 | 3 | | | | 1 | | | 2 | |
| MUL IMUL | m32 | 3 | 2 | | | | 1 | | | 2 | |
| MUL IMUL | m64 | 2 | 1 | | | | 1 | | | 2 | |
| IMUL | r,m | 1 | 1 | | 1 | | 1 | | | 1 | |
| IMUL | r16,m16,i | 2 | 2 | | | | 1 | | | 1 | |
| IMUL | r32,m32,i | 1 | 1 | | 1 | | 1 | | | 1 | |
| IMUL | r64,m64,i | 1 | 1 | | 1 | | 1 | | | 1 | |
| DIV | r8 | 11 | 11 | | | | | | 19-22 | 9 | |
| DIV | r16 | 11 | 11 | | | | | | 20-24 | 10 | |
| DIV | r32 | 10 | 10 | | | | | | 19-27 | 11 | |
| DIV | r64 | 35-57 | x | | | | | | 29-94 | 22-76 | |
| IDIV | r8 | 11 | 11 | | | | | | 20-23 | 8 | |
| IDIV | r16 | 11 | 11 | | | | | | 20-24 | 8 | |
| IDIV | r32 | 9 | 9 | | | | | | 19-26 | 8-11 | |
| IDIV | r64 | 59-134 | x | | | | | | 28-103 | 26-88 | |
| CBW | | 1 | 1 | x | x | x | | | 1 | 0.33 | |
| CWDE | | 1 | 1 | x | x | x | | | 1 | | |
| CDQE | | 1 | 1 | x | x | x | | | 1 | | |
| CWD | | 2 | 2 | x | x | x | | | 1 | | |
| CDQ | | 1 | 1 | x | | x | | | 1 | | |
| CQO | | 1 | 1 | x | | x | | | 1 | 0.5 | |
| POPCNT | r,r | 1 | 1 | | 1 | | | | 3 | 1 | SSE4.2 |
| POPCNT | r,m | 1 | 1 | | 1 | | 1 | | | 1 | SSE4.2 |
| CRC32 | r,r | 1 | 1 | | 1 | | | | 3 | 1 | SSE4.2 |
| CRC32 | r,m | 1 | 1 | | 1 | | 1 | | | | SSE4.2 |
| Logic instructions | | | | | | | | | | | |
| AND OR XOR | r,r/i | 1 | 1 | x | x | x | | | 1 | 0.33 | |
| AND OR XOR | r,m | 1 | 1 | x | x | x | 1 | | | 0.5 | |
| AND OR XOR | m,r/i | 2 | 1 | x | x | x | 2 | 1 | 6 | 1 | |
| TEST | r,r/i | 1 | 1 | x | x | x | | | 1 | 0.33 | |
| TEST | m,r/i | 1 | 1 | x | x | x | 1 | | | 0.5 | |

Ivy Bridge

| | | | | | | | | | | | |
|--------------------------------------|------------|----|----|---|---|---|---|---|---|------|------------|
| SHR SHL SAR | r,i | 1 | 1 | x | | x | | | 1 | 0.5 | |
| SHR SHL SAR | m,i | 3 | 1 | | | | 2 | 1 | | 2 | |
| SHR SHL SAR | r,cl | 2 | 2 | 1 | | 1 | | | 1 | 1 | |
| SHR SHL SAR | m,cl | 5 | 3 | | | | 2 | 1 | | 4 | |
| ROR ROL | r,1 | 2 | 2 | x | | x | | | 1 | 1 | short form |
| ROR ROL | r,i | 1 | 1 | x | | x | | | 1 | 0.5 | |
| ROR ROL | m,i | 4 | 3 | | | | 2 | 1 | | 2 | |
| ROR ROL | r,cl | 2 | 2 | x | | x | | | 1 | 1 | |
| ROR ROL | m,cl | 5 | 3 | | | | 2 | 1 | | 4 | |
| RCL RCR | r,1 | 3 | 3 | x | x | x | | | 2 | 2 | |
| RCL RCR | r,i | 8 | 8 | x | x | x | | | 5 | 5 | |
| RCL RCR | m,i | 11 | 8 | x | x | x | 2 | 1 | | 6 | |
| RCL RCR | r,cl | 8 | 8 | x | x | x | | | 5 | 5 | |
| RCL RCR | m,cl | 11 | 8 | x | x | x | 2 | 1 | | 6 | |
| SHRD SHLD | r,r,i | 1 | 1 | x | | x | | | 1 | 0.5 | |
| SHRD SHLD | m,r,i | 3 | 3 | x | x | x | 2 | 1 | | 2 | |
| SHRD SHLD | r,r,cl | 4 | 4 | x | 1 | x | | | 2 | 2 | |
| SHRD SHLD | m,r,cl | 5 | 4 | x | 1 | x | 2 | 1 | | 4 | |
| BT | r,r/i | 1 | 1 | x | | x | | | 1 | 0.5 | |
| BT | m,r | 10 | 9 | x | x | x | 1 | | | 5 | |
| BT | m,i | 2 | 1 | x | | x | 1 | | | 0.5 | |
| BTR BTS BTC | r,r/i | 1 | 1 | x | | x | | | 1 | 0.5 | |
| BTR BTS BTC | m,r | 11 | 8 | x | x | x | 2 | 1 | | 5 | |
| BTR BTS BTC | m,i | 3 | 2 | x | | x | 1 | 1 | | 2 | |
| BSF BSR | r,r | 1 | 1 | | 1 | | | | 3 | 1 | |
| BSF BSR | r,m | 1 | 1 | | 1 | | 1 | | | 1 | |
| SETcc | r | 1 | 1 | x | | x | | | 1 | 0.5 | |
| SETcc | m | 2 | 1 | x | | x | 1 | 1 | | 1 | |
| CLC | | 1 | 0 | | | | | | | 0.25 | |
| STC CMC | | 1 | 1 | x | x | x | | | 1 | 0.33 | |
| CLD STD | | 3 | 3 | x | x | x | | | | 4 | |
| Control transfer instructions | | | | | | | | | | | |
| JMP | short/near | 1 | 1 | | | 1 | | | 0 | 2 | |
| JMP | r | 1 | 1 | | | 1 | | | 0 | 2 | |
| JMP | m | 1 | 1 | | | 1 | 1 | | 0 | 2 | |
| Conditional jump | short/near | 1 | 1 | | | 1 | | | 0 | 1-2 | |
| Fused arithmetic and branch | | 1 | 1 | | | 1 | | | 0 | 1-2 | |
| J(E/R)CXZ | short | 2 | 2 | x | x | 1 | | | | 1-2 | |
| LOOP | short | 7 | 7 | | | | | | | 4-5 | |
| LOOP(N)E | short | 11 | 11 | | | | | | | 6 | |
| CALL | near | 2 | 1 | | | 1 | 1 | 1 | | 2 | |
| CALL | r | 2 | 1 | | | 1 | 1 | 1 | | 2 | |
| CALL | m | 3 | 1 | | | 1 | 2 | 1 | | 2 | |
| RET | | 2 | 1 | | | 1 | 1 | | | 2 | |
| RET | i | 3 | 2 | x | x | 1 | 1 | | | 2 | |
| BOUND | r,m | 15 | 13 | | | | 2 | | | 7 | |
| INTO | | 4 | 4 | x | x | x | | | | 6 | |
| String instructions | | | | | | | | | | | |
| LODS | | 3 | 2 | x | x | x | 1 | | | 1 | |

Ivy Bridge

| | | | | | | | | | | | |
|-------------------------------------|-----|-------|----|---|---|---|---|---|---------|---------|-------------------------|
| REP LODS | | ~5n | 1 | x | x | x | 1 | 1 | ~2n | 1 | worst case best case |
| STOS | | 3 | | | | | | | n | | |
| REP STOS | | many | | | | | | | | | |
| REP STOS | | many | | | | | | | 1/16B | | worst case best case |
| MOVS | | 5 | 2 | x | x | x | 2 | 1 | n | 4 | |
| REP MOVS | | 2n | | | | | | | | | |
| REP MOVS | | 4/16B | | | | | | | 1/16B | | worst case best case |
| SCAS | | 3 | 2 | x | x | x | 1 | | ~2n | 1 | |
| REP SCAS | | ~6n | | | | | | | | | |
| CMPS | | 5 | 3 | x | x | x | 2 | | ~2n | 4 | worst case best case |
| REP CMPS | | ~8n | | | | | | | | | |
| | | | | | | | | | | | |
| Synchronization instructions | | | | | | | | | | | |
| XADD | m,r | 4 | 3 | x | x | x | 1 | 1 | 7 | | |
| LOCK XADD | m,r | 8 | 5 | x | x | x | 2 | 1 | 22 | | |
| LOCK ADD | m,r | 7 | 5 | x | x | x | 1 | 1 | 22 | | |
| CMPXCHG | m,r | 5 | 3 | x | x | x | 2 | 1 | 7 | | |
| LOCK CMPXCHG | m,r | 9 | 6 | x | x | x | 2 | 1 | 22 | | |
| CMPXCHG8B | m,r | 14 | 11 | x | x | x | 2 | 1 | 7 | | |
| LOCK CMPXCHG8B | m,r | 18 | 15 | x | x | x | 2 | 1 | 22 | | |
| CMPXCHG16B | m,r | 22 | 19 | x | x | x | 2 | 1 | 16 | | |
| LOCK CMPXCHG16B | m,r | 24 | 21 | x | x | x | 2 | 1 | 27 | | |
| Other | | | | | | | | | | | |
| NOP / Long NOP | | 1 | 0 | | | | | | | 0.25 | |
| PAUSE | | 7 | 7 | | | | | | | 10 | |
| ENTER | a,0 | 12 | 9 | x | x | x | 2 | 1 | | 8 | |
| ENTER | a,b | 45+7b | | | | | | | 84+3b | | |
| LEAVE | | 3 | 2 | x | x | x | 1 | | | 6 | |
| XGETBV | | 8 | | | | | | | | 9 | XGETBV |
| CPUID | | 37-82 | | | | | | | 100-340 | | |
| RDTSC | | 21 | | | | | | | | 27 | |
| RDPMSR | | 35 | | | | | | | | 39 | |
| RDRAND | r | 13 | 12 | x | x | x | 1 | | | 104-117 | RDRAND |

Floating point x87 instructions

| Instruction | Operands | μops fused domain | μops unfused domain | | | | | | Latency | Reciprocal throughput | Comments |
|-------------------|----------|-------------------|---------------------|----|----|----|-----|----|---------|-----------------------|----------|
| | | | p015 | p0 | p1 | p5 | p23 | p4 | | | |
| Move instructions | | | | | | | | | | | |
| FLD | r | 1 | 1 | | | 1 | | | 1 | 1 | |
| FLD | m32/64 | 1 | | | | | 1 | | 3 | 1 | |
| FLD | m80 | 4 | 2 | | 1 | 1 | 2 | | 5 | 2 | |
| FBLD | m80 | 43 | 40 | | | | 3 | | 45 | 21 | |
| FST(P) | r | 1 | 1 | | | 1 | | | 1 | 1 | |
| FST(P) | m32/m64 | 1 | | | | | 1 | 1 | 4 | 1 | |

Ivy Bridge

| | | | | | | | | | | | |
|--------------------------------|-----|--------|-----|---|---|---|---|---|--------|--------|------|
| FSTP | m80 | 7 | 3 | | | | 2 | 2 | 5 | 5 | |
| FBSTP | m80 | 243 | | | | | | | | 252 | |
| FXCH | r | 1 | 0 | | | | | | 0 | 0.5 | |
| FILD | m | 1 | 1 | | 1 | | 1 | | 6 | 1 | |
| FIST(P) | m | 3 | 1 | | 1 | | 1 | 1 | 7 | 1 | |
| FISTTP | m | 3 | 1 | | 1 | | 1 | 1 | 7 | 2 | SSE3 |
| FLDZ | | 1 | 1 | | | 1 | | | | | |
| FLD1 | | 2 | 2 | | 1 | 1 | | | | | |
| FLDPI FLDL2E etc. | | 2 | 2 | 1 | 1 | | | | | 2 | |
| FCMOVcc | r | 3 | 3 | 1 | | 2 | | | 2 | 2 | |
| FNSTSW | AX | 2 | 2 | 1 | x | x | | | 4 | 1 | |
| FNSTSW | m16 | 2 | 1 | | | | 1 | 1 | | 1 | |
| FLDCW | m16 | 3 | 2 | | | 2 | 1 | | | 3 | |
| FNSTCW | m16 | 2 | 1 | | | 1 | 1 | 1 | | 1 | |
| FINCSTP FDECSTP | | 1 | 1 | | | 1 | | | 1 | 1 | |
| FFREE(P) | r | 1 | 1 | | | 1 | | | | 1 | |
| FNSAVE | m | 143 | | | | | | | | 167 | |
| FRSTOR | m | 90 | | | | | | | | 162 | |
| Arithmetic instructions | | | | | | | | | | | |
| FADD(P) FSUB(R)(P) | r | 1 | 1 | | 1 | | | | 3 | 1 | |
| FADD(P) FSUB(R)(P) | m | 2 | 1 | | 1 | | 1 | | | 1 | |
| FMUL(P) | r | 1 | 1 | 1 | | | | | 5 | 1 | |
| FMUL(P) | m | 2 | 1 | 1 | | | 1 | | | 1 | |
| FDIV(R)(P) | r | 1 | 1 | 1 | | | | | 10-24 | 8-18 | |
| FDIV(R)(P) | m | 2 | 1 | 1 | | | 1 | | | 8-18 | |
| FABS | | 1 | 1 | | | 1 | | | 1 | 1 | |
| FCHS | | 1 | 1 | | | 1 | | | 1 | 1 | |
| FCOM(P) FUCOM | r | 1 | 1 | | 1 | | | | 3 | 1 | |
| FCOM(P) FUCOM | m | 1 | 1 | | 1 | | 1 | | | 1 | |
| FCOMPP FUCOMPP | | 2 | 2 | | 1 | 1 | | | 4 | 1 | |
| FCOMI(P) FUCOMI(P) | r | 3 | 3 | 1 | 1 | 1 | | | 5 | 1 | |
| FIADD FISUB(R) | m | 2 | 2 | | 2 | | 1 | | | 2 | |
| FIMUL | m | 2 | 2 | 1 | 1 | | 1 | | | 2 | |
| FIDIV(R) | m | 2 | 2 | 1 | 1 | | 1 | | | | |
| FICOM(P) | m | 2 | 2 | | 2 | | 1 | | | 2 | |
| FTST | | 1 | 1 | | 1 | | | | | 1 | |
| FXAM | | 2 | 2 | | 2 | | | | | 2 | |
| FPREM | | 28 | 28 | | | | | | 21-26 | 12 | |
| FPREM1 | | 41 | | | | | | | 27-50 | 19 | |
| FRNDINT | | 17 | 17 | | | | | | 22 | 11 | |
| Math | | | | | | | | | | | |
| FSCALE | | 25 | 25 | x | x | x | | | 49 | 49 | |
| FXTRACT | | 17 | 17 | x | x | x | | | 10 | 10 | |
| FSQRT | | 1 | 1 | 1 | | | | | 10-23 | 8-17 | |
| FSIN | | 21-78 | | x | x | x | | | 47-106 | 47-106 | |
| FCOS | | 23-100 | | x | x | x | | | 48-115 | 48-115 | |
| FSINCOS | | 20-110 | | x | x | x | | | 50-123 | 50-123 | |
| F2XM1 | | 16-23 | | x | x | x | | | ~68 | ~68 | |
| FYL2X | | 42 | 42 | x | x | x | | | 90-106 | | |
| FYL2XP1 | | 56 | 56 | x | x | x | | | 82 | | |
| FPTAN | | 102 | 102 | x | x | x | | | 130 | | |

Ivy Bridge

| | | | | | | | | | | | |
|--------------|--|-------|----|---|---|---|--|--|--------|----|--|
| FPATAN | | 28-72 | | x | x | x | | | 94-150 | | |
| Other | | | | | | | | | | | |
| FNOP | | 1 | 1 | | | 1 | | | | 1 | |
| WAIT | | 2 | 2 | x | x | 1 | | | | 1 | |
| FNCLEX | | 5 | 5 | x | x | x | | | | 22 | |
| FNINIT | | 26 | 26 | x | x | x | | | | 80 | |

Integer MMX and XMM instructions

| Instruction | Operands | μops fused do- main | μops unfused domain | | | | | | Latency | Reci- procal through- put | Com- ments |
|-------------------|-------------|------------------------|---------------------|----|----|----|-----|------|---------|------------------------------------|---------------|
| | | | p015 | p0 | p1 | p5 | p23 | p4 | | | |
| Move instructions | | | | | | | | | | | |
| MOVD | r32/64,mm/x | 1 | 1 | 1 | | | | | 1 | 1 | eliminat. |
| MOVD | m32/64,mm/x | 1 | | | | 1 | 1 | 3 | 1 | | |
| MOVD | mm/x,r32/64 | 1 | 1 | | 1 | | | 1 | 1 | | |
| MOVD | mm/x,m32/64 | 1 | | | | 1 | | 3 | 0.5 | | |
| MOVQ | mm/x,mm/x | 1 | 1 | x | x | x | | 1 | 0.33 | | |
| MOVQ | mm/x,m64 | 1 | | | | 1 | | 3 | 0.5 | | |
| MOVQ | m64, mm/x | 1 | | | | 1 | 1 | 3 | 1 | | |
| MOVDQA MOVDQU | x,x | 1 | 1 | x | x | x | | 0-1 | 0.25 | | |
| MOVDQA MOVDQU | x, m128 | 1 | | | | 1 | | 3 | 0.5 | | |
| MOVDQA MOVDQU | m128, x | 1 | | | | 1 | 1 | 3 | 1 | | |
| LDDQU | x, m128 | 1 | 1 | | | 1 | | 3 | 0.5 | SSE3 | |
| MOVDQ2Q | mm, x | 2 | 2 | x | x | 1 | | 1 | 1 | | |
| MOVQ2DQ | x,mm | 1 | 1 | | | | | 1 | 0.33 | | |
| MOVNTQ | m64,mm | 1 | | | | 1 | 1 | ~360 | 1 | | |
| MOVNTDQ | m128,x | 1 | | | | 1 | 1 | ~360 | 1 | SSE4.1 | |
| MOVNTDQA | x, m128 | 1 | | | | 1 | | 3 | 0.5 | | |
| PACKSSWB/DW | | | | | | | | | | | |
| PACKUSWB | mm,mm | 1 | 1 | 1 | | | | 1 | 1 | | |
| PACKSSWB/DW | | | | | | | | | | | |
| PACKUSWB | mm,m64 | 1 | 1 | 1 | | 1 | | | 1 | SSE4.1 SSE4.1 | |
| PACKSSWB/DW | | | | | | | | | | | |
| PACKUSWB | x,x | 1 | 1 | | x | x | | 1 | 0.5 | | |
| PACKSSWB/DW | | | | | | | | | | | |
| PACKUSWB | x,m128 | 1 | 1 | | x | x | 1 | 1 | 0.5 | | |
| PACKUSDW | x,x | 1 | 1 | | x | x | | 1 | 0.5 | | |
| PACKUSDW | x,m | 1 | 1 | | x | x | 1 | | 0.5 | | |
| PUNPCKH/LBW/WD/DQ | mm/x,mm/x | 1 | 1 | | x | x | | 1 | 0.5 | | |
| PUNPCKH/LBW/WD/DQ | mm/x,m | 1 | 1 | | x | x | 1 | | 0.5 | | |
| PUNPCKH/LQDQ | x,x | 1 | 1 | | x | x | | 1 | 0.5 | | |
| PUNPCKH/LQDQ | x, m128 | 2 | 1 | | x | x | 1 | | 0.5 | | |
| PMOVSX/ZXBW | x,x | 1 | 1 | | x | x | | 1 | 0.5 | SSE4.1 | |
| PMOVSX/ZXBW | x,m64 | 1 | 1 | | x | x | 1 | | 0.5 | SSE4.1 | |
| PMOVSX/ZXBD | x,x | 1 | 1 | | x | x | | 1 | 0.5 | SSE4.1 | |
| PMOVSX/ZXBD | x,m32 | 1 | 1 | | x | x | 1 | | 0.5 | SSE4.1 | |
| PMOVSX/ZXBQ | x,x | 1 | 1 | | x | x | | 1 | 0.5 | SSE4.1 | |
| PMOVSX/ZXBQ | x,m16 | 1 | 1 | | x | x | 1 | | 0.5 | SSE4.1 | |
| PMOVSX/ZXWD | x,x | 1 | 1 | | x | x | | 1 | 0.5 | SSE4.1 | |
| PMOVSX/ZXWD | x,m64 | 1 | 1 | | x | x | 1 | | 0.5 | SSE4.1 | |

Ivy Bridge

| | | | | | | | | | | | |
|--------------------------------|-------------|----|---|---|---|---|---|---|---|-----|--------|
| PMOVSX/ZXWQ | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PMOVSX/ZXWQ | x,m32 | 1 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PMOVSX/ZXDQ | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PMOVSX/ZXDQ | x,m64 | 1 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PSHUFB | mm/x,mm/x | 1 | 1 | | x | x | | | 1 | 0.5 | SSSE3 |
| PSHUFB | mm/x,m | 2 | 1 | | x | x | 1 | | | 0.5 | SSSE3 |
| PSHUFW | mm,mm,i | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PSHUFW | mm,m64,i | 2 | 1 | | x | x | 1 | | | 0.5 | |
| PSHUFD | xmm,x,i | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PSHUFD | x,m128,i | 2 | 1 | | x | x | 1 | | | 0.5 | |
| PSHUFL/HW | x,x,i | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PSHUFL/HW | x, m128,i | 2 | 1 | | x | x | 1 | | | 0.5 | |
| PALIGNR | mm/x,mm/x,i | 1 | 1 | | x | x | | | 1 | 0.5 | SSSE3 |
| PALIGNR | mm/x,m,i | 2 | 1 | | x | x | 1 | | | 0.5 | SSSE3 |
| PBLENDVB | x,x,xmm0 | 2 | 2 | | 1 | 1 | | | 2 | 1 | SSE4.1 |
| PBLENDVB | x,m,xmm0 | 3 | 2 | | 1 | 1 | 1 | | | 1 | SSE4.1 |
| PBLENDW | x,x,i | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PBLENDW | x,m,i | 2 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| MASKMOVQ | mm,mm | 4 | 1 | 1 | | | 2 | 1 | | 1 | |
| MASKMOVDQU | x,x | 10 | 4 | x | 1 | x | 4 | 2 | | 6 | |
| PMOVMASKB | r32,mm/x | 1 | 1 | 1 | | | | | 2 | 1 | |
| PEXTRB | r32,x,i | 2 | 2 | 1 | x | x | | | 2 | 1 | SSE4.1 |
| PEXTRB | m8,x,i | 2 | 1 | | x | x | 1 | 1 | | 1 | SSE4.1 |
| PEXTRW | r32,mm/x,i | 2 | 1 | 1 | x | x | | | 2 | 1 | |
| PEXTRW | m16,mm/x,i | 2 | 1 | | x | x | 1 | 1 | | 1 | SSE4.1 |
| PEXTRD | r32,x,i | 2 | 2 | 1 | x | x | | | 2 | 1 | SSE4.1 |
| PEXTRD | m32,x,i | 2 | 1 | | x | x | 1 | 1 | | 1 | SSE4.1 |
| PEXTRQ | r64,x,i | 2 | 2 | 1 | x | x | | | 2 | 1 | SSE4.1 |
| PEXTRQ | m64,x,i | 2 | 1 | | x | x | 1 | 1 | | 1 | |
| PINSRB | x,r32,i | 2 | 2 | | x | x | | | 2 | 1 | SSE4.1 |
| PINSRB | x,m8,i | 2 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PINSRW | mm/x,r32,i | 2 | 2 | | x | x | | | 2 | 1 | |
| PINSRW | mm/x,m16,i | 2 | 1 | | x | x | 1 | | | 0.5 | |
| PINSRD | x,r32,i | 2 | 1 | | x | x | | | 2 | 1 | SSE4.1 |
| PINSRD | x,m32,i | 2 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PINSRQ | x,r64,i | 2 | 1 | | x | x | | | 2 | 1 | SSE4.1 |
| PINSRQ | x,m64,i | 2 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| Arithmetic instructions | | | | | | | | | | | |
| PADD/SUB(U,S)B/W/D/Q | mm/x, mm/x | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PADD/SUB(U,S)B/W/D/Q | mm/x,m | 1 | 1 | | x | x | 1 | | | 0.5 | |
| PHADD/SUB(S)W/D | mm/x, mm/x | 3 | 3 | | x | x | | | 3 | 1.5 | SSSE3 |
| PHADD/SUB(S)W/D | mm/x,m64 | 4 | 3 | | x | x | 1 | | | 1.5 | SSSE3 |
| PCMPEQ/GTB/W/D | mm/x,mm/x | 1 | 1 | | x | x | | | 1 | 0.5 | |
| PCMPEQ/GTB/W/D | mm/x,m | 1 | 1 | | x | x | 1 | | | 0.5 | |
| PCMPEQQ | x,x | 1 | 1 | | x | x | | | 1 | 0.5 | SSE4.1 |
| PCMPEQQ | x,m128 | 1 | 1 | | x | x | 1 | | | 0.5 | SSE4.1 |
| PCMPGTQ | x,x | 1 | 1 | 1 | | | | | 5 | 1 | SSE4.2 |
| PCMPGTQ | x,m128 | 1 | 1 | 1 | | | 1 | | | 1 | SSE4.2 |
| PMULL/HW PMULHUW | mm/x,mm/x | 1 | 1 | 1 | | | | | 5 | 1 | |
| PMULL/HW PMULHUW | mm/x,m | 1 | 1 | 1 | | | 1 | | | 1 | |
| PMULHSW | mm/x,mm/x | 1 | 1 | 1 | | | | | 5 | 1 | SSSE3 |
| PMULHSW | mm/x,m | 1 | 1 | 1 | | | 1 | | | 1 | SSSE3 |

Ivy Bridge

| | | | | | | | | | | |
|----------------------------|-----------|---|---|---|---|---|---|----|------|--------|
| PMULLD | x,x | 1 | 1 | 1 | | | | 5 | 1 | SSE4.1 |
| PMULLD | x,m128 | 2 | 1 | 1 | | | 1 | | 1 | SSE4.1 |
| PMULDQ | x,x | 1 | 1 | 1 | | | | 5 | 1 | SSE4.1 |
| PMULDQ | x,m128 | 1 | 1 | 1 | | | 1 | | 1 | SSE4.1 |
| PMULUDQ | mm/x,mm/x | 1 | 1 | 1 | | | | 5 | 1 | |
| PMULUDQ | mm/x,m | 1 | 1 | 1 | | | 1 | | 1 | |
| PMADDWD | mm/x,mm/x | 1 | 1 | 1 | | | | 5 | 1 | |
| PMADDWD | mm/x,m | 1 | 1 | 1 | | | 1 | | 1 | |
| PMADDUBSW | mm/x,mm/x | 1 | 1 | 1 | | | | 5 | 1 | SSSE3 |
| PMADDUBSW | mm/x,m | 1 | 1 | 1 | | | 1 | | 1 | SSSE3 |
| PAVGB/W | mm/x,mm/x | 1 | 1 | | x | x | | 1 | 0.5 | |
| PAVGB/W | mm/x,m | 1 | 1 | | x | x | 1 | | 0.5 | |
| PMIN/MAXSB | x,x | 1 | 1 | | x | x | | 1 | 0.5 | SSE4.1 |
| PMIN/MAXSB | x,m128 | 1 | 1 | | x | x | 1 | | 0.5 | SSE4.1 |
| PMIN/MAXUB | mm/x,mm/x | 1 | 1 | | x | x | | 1 | 0.5 | |
| PMIN/MAXUB | mm/x,m | 1 | 1 | | x | x | 1 | | 0.5 | |
| PMIN/MAXSW | mm/x,mm/x | 1 | 1 | | x | x | | 1 | 0.5 | |
| PMIN/MAXSW | mm/x,m | 1 | 1 | | x | x | 1 | | 0.5 | |
| PMIN/MAXUW | x,x | 1 | 1 | | x | x | | 1 | 0.5 | SSE4.1 |
| PMIN/MAXUW | x,m | 1 | 1 | | x | x | 1 | | 0.5 | SSE4.1 |
| PMIN/MAXU/SD | x,x | 1 | 1 | | x | x | | 1 | 0.5 | SSE4.1 |
| PMIN/MAXU/SD | x,m128 | 1 | 1 | | x | x | 1 | | 0.5 | SSE4.1 |
| PHMINPOSUW | x,x | 1 | 1 | 1 | | | | 5 | 1 | SSE4.1 |
| PHMINPOSUW | x,m128 | 1 | 1 | 1 | | | 1 | | 1 | SSE4.1 |
| PABSB/W/D | mm/x,mm/x | 1 | 1 | | x | x | | 1 | 0.5 | SSSE3 |
| PABSB/W/D | mm/x,m | 1 | 1 | | x | x | 1 | | 0.5 | SSSE3 |
| PSIGNB/W/D | mm/x,mm/x | 1 | 1 | | x | x | | 1 | 0.5 | SSSE3 |
| PSIGNB/W/D | mm/x,m | 1 | 1 | | x | x | 1 | | 0.5 | SSSE3 |
| PSADBW | mm/x,mm/x | 1 | 1 | 1 | | | | 5 | 1 | |
| PSADBW | mm/x,m | 1 | 1 | 1 | | | 1 | | 1 | |
| MPSADBW | x,x,i | 3 | 3 | 1 | 1 | 1 | | 6 | 1 | SSE4.1 |
| MPSADBW | x,m,i | 4 | 3 | 1 | 1 | 1 | 1 | | 1 | SSE4.1 |
| Logic instructions | | | | | | | | | | |
| PAND(N) POR PXOR | mm/x,mm/x | 1 | 1 | x | x | x | | 1 | 0.33 | |
| PAND(N) POR PXOR | mm/x,m | 1 | 1 | x | x | x | 1 | | 0.5 | |
| PTEST | x,x | 2 | 2 | 1 | x | x | | 1 | 1 | SSE4.1 |
| PTEST | x,m128 | 3 | 2 | 1 | x | x | 1 | | 1 | SSE4.1 |
| PSLL/RL/RAW/D/Q | mm,mm/i | 1 | 1 | 1 | | | | 1 | 1 | |
| PSLL/RL/RAW/D/Q | mm,m64 | 1 | 1 | 1 | | | 1 | | 1 | |
| PSLL/RL/RAW/D/Q | xmm,i | 1 | 1 | 1 | | | | 1 | 1 | |
| PSLL/RL/RAW/D/Q | x,x | 2 | 2 | 1 | x | x | | 2 | 1 | |
| PSLL/RL/RAW/D/Q | x,m128 | 3 | 2 | 1 | x | x | 1 | | 1 | |
| PSLL/RDQ | x,i | 1 | 1 | | x | x | | 1 | 0.5 | |
| String instructions | | | | | | | | | | |
| PCMPESTRI | x,x,i | 8 | 8 | 3 | 1 | 4 | | 4 | 4 | SSE4.2 |
| PCMPESTRI | x,m128,i | 8 | 7 | 3 | 1 | 3 | 1 | | 4 | SSE4.2 |
| PCMPESTRM | x,x,i | 8 | 8 | 3 | 1 | 4 | | 12 | 4 | SSE4.2 |
| PCMPESTRM | x,m128,i | 8 | 7 | 3 | 1 | 3 | 1 | | 4 | SSE4.2 |
| PCMPISTRI | x,x,i | 3 | 3 | 3 | | | | 3 | | SSE4.2 |
| PCMPISTRI | x,m128,i | 4 | 3 | 3 | | | 1 | | 3 | SSE4.2 |
| PCMPISTRM | x,x,i | 3 | 3 | 3 | | | | 11 | | SSE4.2 |

Ivy Bridge

| | | | | | | | | | | | |
|---|----------|----|----|---|---|---|---|--|----|----|--------|
| PCMPISTRM | x,m128,i | 4 | 3 | 3 | | | 1 | | | 3 | SSE4.2 |
| Encryption instructions | | | | | | | | | | | |
| PCLMULQDQ | x,x,i | 18 | 18 | x | x | x | | | 14 | 8 | CLMUL |
| PCLMULQDQ | x,m,i | 18 | 17 | x | x | x | 1 | | | 8 | CLMUL |
| AESDEC, AESDECLAST, AESENC, AESENCLAST | x,x | 2 | 2 | x | x | 1 | | | 4 | 1 | AES |
| AESDEC, AESDECLAST, AESENC, AESENCLAST | x,m | 3 | 2 | x | x | 1 | 1 | | | 1 | AES |
| AESIMC | x,x | 2 | 2 | | | 2 | | | 14 | 2 | AES |
| AESIMC | x,m | 3 | 2 | | | 2 | 1 | | | 2 | AES |
| AESKEYGENASSIST | x,x,i | 11 | 11 | x | x | x | | | 10 | 8 | AES |
| AESKEYGENASSIST | x,m,i | 11 | 10 | x | x | x | 1 | | | 7 | AES |
| Other | | | | | | | | | | | |
| EMMS | | 31 | 31 | | | | | | | 18 | |

Floating point XMM and YMM instructions

| Instruction | Operands | μops fused do- main | μops unfused domain | | | | | | Latency | Reci- procal through- put | Com- ments |
|-------------------------|-------------|------------------------------|---------------------|----|----|----|-----|----|---------|------------------------------------|---------------|
| | | | p015 | p0 | p1 | p5 | p23 | p4 | | | |
| Move instructions | | | | | | | | | | | |
| MOVAPS/D | x,x | 1 | 1 | | | 1 | | | 0-1 | ≤1 | elimin. |
| VMOVAPS/D | y,y | 1 | 1 | | | 1 | | | 0-1 | ≤1 | elimin. |
| MOVAPS/D MOVUPS/D | x,m128 | 1 | | | | | 1 | | 3 | 0.5 | |
| VMOVAPS/D VMOVUPS/ D | y,m256 | 1 | | | | | 1+ | | 4 | 1 | AVX |
| MOVAPS/D MOVUPS/D | m128,x | 1 | | | | | 1 | 1 | 3 | 1 | |
| VMOVAPS/D VMOVUPS/ D | m256,y | 1 | | | | | 1 | 1+ | 4 | 2 | AVX |
| MOVSS/D | x,x | 1 | 1 | | | 1 | | | 1 | 1 | |
| MOVSS/D | x,m32/64 | 1 | | | | | 1 | | 3 | 0.5 | |
| MOVSS/D | m32/64,x | 1 | | | | | 1 | 1 | 3 | 1 | |
| MOVHPS/D MOVLPS/D | x,m64 | 2 | 1 | | | 1 | 1 | | 4 | 1 | |
| MOVH/LPS/D | m64,x | 2 | | | | | 1 | 1 | 3 | 1 | |
| MOVLHPS MOVHLPS | x,x | 1 | 1 | | | 1 | | | 1 | 1 | |
| MOVMSKPS/D | r32,x | 1 | 1 | 1 | | | | | 2 | 1 | |
| VMOVMSKPS/D | r32,y | 1 | 1 | 1 | | | | | 2 | 1 | |
| MOVNTPS/D | m128,x | 1 | | | | | 1 | 1 | ~380 | 1 | |
| VMOVNTPS/D | m256,y | 1 | | | | | 1 | 1+ | ~380 | 2 | AVX |
| SHUFPS/D | x,x,i | 1 | 1 | | | 1 | | | 1 | 1 | |
| SHUFPS/D | x,m128,i | 2 | 1 | | | 1 | 1 | | | 1 | |
| VSHUFPS/D | y,y,y,i | 1 | 1 | | | 1 | | | 1 | 1 | AVX |
| VSHUFPS/D | y, y,m256,i | 2 | 1 | | | 1 | 1+ | | | 1 | AVX |
| VPERMILPS/PD | x,x,x/i | 1 | 1 | | | 1 | | | 1 | 1 | AVX |
| VPERMILPS/PD | y,y,y/i | 1 | 1 | | | 1 | | | 1 | 1 | AVX |
| VPERMILPS/PD | x,x,m | 2 | 1 | | | 1 | 1 | | | 1 | AVX |
| VPERMILPS/PD | y,y,m | 2 | 1 | | | 1 | 1+ | | | 1 | AVX |
| VPERMILPS/PD | x,m,i | 2 | 1 | | | 1 | 1 | | | 1 | AVX |

Ivy Bridge

| | | | | | | | | | | | |
|----------------|------------|---|---|---|---|----|----|----|-----|---|--------|
| VPERMILPS/PD | y,m,i | 2 | 1 | | | 1 | 1+ | | | 1 | AVX |
| VPERM2F128 | y,y,y,i | 1 | 1 | | | 1 | | 2 | | 1 | AVX |
| VPERM2F128 | y,y,m,i | 2 | 1 | | | 1 | 1+ | | | 1 | AVX |
| BLENDPS/PD | x,x,i | 1 | 1 | x | x | | | 1 | 0.5 | | SSE4.1 |
| BLENDPS/PD | x,m128,i | 2 | 1 | x | x | 1 | | | 0.5 | | SSE4.1 |
| VBLENDPS/PD | y,y,i | 1 | 1 | x | x | | | 1 | 0.5 | | AVX |
| VBLENDPS/PD | y,m256,i | 2 | 1 | x | x | 1+ | | | 1 | | AVX |
| BLENDVPS/PD | x,x,xmm0 | 2 | 2 | x | x | | | 2 | 1 | | SSE4.1 |
| BLENDVPS/PD | x,m,xmm0 | 3 | 2 | x | x | 1 | | | 1 | | SSE4.1 |
| VBLENDVPS/PD | y,y,y,y | 2 | 2 | x | x | | | 2 | 1 | | AVX |
| VBLENDVPS/PD | y,y,m,y | 3 | 2 | x | x | 1+ | | | 1 | | AVX |
| MOVDDUP | x,x | 1 | 1 | | | 1 | | 1 | 1 | | SSE3 |
| MOVDDUP | x,m64 | 1 | | | | | 1 | 3 | 0.5 | | SSE3 |
| VMOVDDUP | y,y | 1 | 1 | | | 1 | | 1 | 1 | | AVX |
| VMOVDDUP | y,m256 | 1 | | | | | 1+ | 3 | 1 | | AVX |
| VBROADCASTSS | x,m32 | 1 | | | | | 1 | 4 | 0.5 | | AVX |
| VBROADCASTSS | y,m32 | 2 | 1 | | | 1 | 1 | 5 | 1 | | AVX |
| VBROADCASTSD | y,m64 | 2 | 1 | | | 1 | 1 | 5 | 1 | | AVX |
| VBROADCASTF128 | y,m128 | 2 | 1 | | | 1 | 1 | 5 | 1 | | AVX |
| MOVSH/LDUP | x,x | 1 | 1 | | | 1 | | 1 | 1 | | SSE3 |
| MOVSH/LDUP | x,m128 | 1 | | | | | 1 | 3 | 0.5 | | SSE3 |
| VMOVSH/LDUP | y,y | 1 | 1 | | | 1 | | 1 | 1 | | AVX |
| VMOVSH/LDUP | y,m256 | 1 | | | | | 1+ | | 1 | | AVX |
| UNPCKH/LPS/D | x,x | 1 | 1 | | | 1 | | 1 | 1 | | SSE3 |
| UNPCKH/LPS/D | x,m128 | 1 | 1 | | | 1 | 1 | | 1 | | SSE3 |
| VUNPCKH/LPS/D | y,y,y | 1 | 1 | | | 1 | | 1 | 1 | | AVX |
| VUNPCKH/LPS/D | y,y,m256 | 1 | 1 | | | 1 | 1+ | | 1 | | AVX |
| EXTRACTPS | r32,x,i | 2 | 2 | x | x | | | 2 | 1 | | SSE4.1 |
| EXTRACTPS | m32,x,i | 3 | 2 | x | x | 1 | 1 | | 1 | | SSE4.1 |
| VEXTRACTF128 | x,y,i | 1 | 1 | | | 1 | | 2 | 1 | | AVX |
| VEXTRACTF128 | m128,y,i | 2 | 0 | | | | 1 | 4 | 1 | | AVX |
| INSERTPS | x,x,i | 1 | 1 | | | 1 | | 1 | 1 | | SSE4.1 |
| INSERTPS | x,m32,i | 2 | 1 | | | 1 | 1 | | 1 | | SSE4.1 |
| VINSERTF128 | y,y,x,i | 1 | 1 | | | 1 | | 2 | 1 | | AVX |
| VINSERTF128 | y,y,m128,i | 2 | 1 | x | x | 1 | | 4 | 1 | | AVX |
| VMASKMOVPS/D | x,x,m128 | 3 | 2 | x | x | 1 | | 4 | 1 | | AVX |
| VMASKMOVPS/D | y,y,m256 | 3 | 2 | | | | 1+ | 5 | 1 | | AVX |
| VMASKMOVPS/D | m128,x,x | 4 | 2 | x | x | 1 | 1 | | 1 | | AVX |
| VMASKMOVPS/D | m256,y,y | 4 | 2 | x | x | 1 | 1+ | | 2 | | AVX |
| Conversion | | | | | | | | | | | |
| CVTPD2PS | x,x | 2 | 2 | | | 1 | 1 | 4 | 1 | | |
| CVTPD2PS | x,m128 | 2 | 2 | | | 1 | 1 | 1 | 1 | | |
| VCVTPD2PS | x,y | 2 | 2 | | | 1 | 1 | 4 | 1 | | AVX |
| VCVTPD2PS | x,m256 | 2 | 2 | | | 1 | 1 | 1+ | 1 | | AVX |
| CVTSD2SS | x,x | 2 | 2 | | | 1 | 1 | 4 | 1 | | |
| CVTSD2SS | x,m64 | 2 | 2 | | | 1 | 1 | 1 | 1 | | |
| CVTPS2PD | x,x | 2 | 2 | 1 | | 1 | | 1 | 1 | | |
| CVTPS2PD | x,m64 | 2 | 1 | 1 | | | 1 | | 1 | | |
| VCVTPS2PD | y,x | 2 | 2 | 1 | | 1 | | 4 | 1 | | AVX |
| VCVTPS2PD | y,m128 | 3 | 2 | 1 | | 1 | 1 | | 1 | | AVX |
| CVTSS2SD | x,x | 2 | 2 | 1 | | 1 | | 2 | 1 | | |
| CVTSS2SD | x,m32 | 2 | 1 | 1 | | | 1 | | 1 | | |

Ivy Bridge

| | | | | | | | | | | |
|---------------------|----------|---|---|---|---|----|---|----|---|------|
| CVTDQ2PS | x,x | 1 | 1 | 1 | | | | 3 | 1 | |
| CVTDQ2PS | x,m128 | 1 | 1 | 1 | | 1 | | | 1 | |
| VCVTDQ2PS | y,y | 1 | 1 | 1 | | | | 3 | 1 | AVX |
| VCVTDQ2PS | y,m256 | 1 | 1 | 1 | | 1+ | | | 1 | AVX |
| CVT(T) PS2DQ | x,x | 1 | 1 | 1 | | | | 3 | 1 | |
| CVT(T) PS2DQ | x,m128 | 1 | 1 | 1 | | 1 | | | 1 | |
| VCVT(T) PS2DQ | y,y | 1 | 1 | 1 | | | | 3 | 1 | AVX |
| VCVT(T) PS2DQ | y,m256 | 1 | 1 | 1 | | 1+ | | | 1 | AVX |
| CVTDQ2PD | x,x | 2 | 2 | 1 | 1 | | | 4 | 1 | |
| CVTDQ2PD | x,m64 | 2 | 2 | 1 | 1 | 1 | | | 1 | |
| VCVTDQ2PD | y,x | 2 | 2 | 1 | 1 | | | 5 | 1 | AVX |
| VCVTDQ2PD | y,m128 | 2 | 2 | 1 | 1 | 1 | | | 1 | AVX |
| CVT(T)PD2DQ | x,x | 2 | 2 | 1 | 1 | | | 4 | 1 | |
| CVT(T)PD2DQ | x,m128 | 2 | 2 | 1 | 1 | 1 | | | 1 | |
| VCVT(T)PD2DQ | x,y | 2 | 2 | 1 | 1 | | | 5 | 1 | AVX |
| VCVT(T)PD2DQ | x,m256 | 2 | 2 | 1 | 1 | 1+ | | | 1 | AVX |
| CVTPI2PS | x,mm | 1 | 1 | 1 | | | | 4 | | |
| CVTPI2PS | x,m64 | 1 | 1 | 1 | | 1 | | | 3 | |
| CVT(T)PS2PI | mm,x | 2 | 2 | 1 | 1 | | | 4 | 1 | |
| CVT(T)PS2PI | mm,m128 | 2 | 1 | 1 | | 1 | | | 1 | |
| CVTPI2PD | x,mm | 2 | 2 | 1 | 1 | | | 4 | 1 | |
| CVTPI2PD | x,m64 | 2 | 2 | 1 | 1 | 1 | | | 1 | |
| CVT(T) PD2PI | mm,x | 2 | 2 | 1 | 1 | | | 4 | 1 | |
| CVT(T) PD2PI | mm,m128 | 2 | 2 | 1 | 1 | 1 | | | 1 | |
| CVTSI2SS | x,r32 | 2 | 2 | 1 | 1 | | | 4 | 3 | |
| CVTSI2SS | x,m32 | 1 | 1 | 1 | | 1 | | | 3 | |
| CVT(T)SS2SI | r32,x | 2 | 2 | 1 | 1 | | | 4 | 1 | |
| CVT(T)SS2SI | r32,m32 | 2 | 2 | 1 | 1 | | 1 | | 1 | |
| CVTSI2SD | x,r32 | 2 | 2 | 1 | 1 | | | 4 | 3 | |
| CVTSI2SD | x,m32 | 2 | 1 | 1 | | 1 | | | 3 | |
| CVT(T)SD2SI | r32,x | 2 | 2 | 1 | 1 | | | 4 | 1 | |
| CVT(T)SD2SI | r32,m64 | 2 | 2 | 1 | 1 | | 1 | | 1 | |
| VCVTPS2PH | x,v,i | 3 | 3 | 1 | 1 | 1 | | 10 | 1 | F16C |
| VCVTPS2PH | m,v,i | 3 | 2 | 1 | 1 | | 1 | 1 | 1 | F16C |
| VCVTPH2PS | v,x | 2 | 2 | 1 | | 1 | | 6 | 1 | F16C |
| VCVTPH2PS | v,m | 2 | 1 | 1 | | 1 | | | 1 | F16C |
| Arithmetic | | | | | | | | | | |
| ADDSS/D SUBSS/D | x,x | 1 | 1 | 1 | | | | 3 | 1 | |
| ADDSS/D SUBSS/D | x,m32/64 | 1 | 1 | 1 | | 1 | | | 1 | |
| ADDPS/D SUBPS/D | x,x | 1 | 1 | 1 | | | | 3 | 1 | |
| ADDPS/D SUBPS/D | x,m128 | 1 | 1 | 1 | | 1 | | | 1 | |
| VADDPS/D VSUBPS/D | y,y,y | 1 | 1 | 1 | | | | 3 | 1 | AVX |
| VADDPS/D VSUBPS/D | y,y,m256 | 1 | 1 | 1 | | 1+ | | | 1 | AVX |
| ADDSUBPS/D | x,x | 1 | 1 | 1 | | | | 3 | 1 | SSE3 |
| ADDSUBPS/D | x,m128 | 1 | 1 | 1 | | 1 | | | 1 | SSE3 |
| VADDSUBPS/D | y,y,y | 1 | 1 | 1 | | | | 3 | 1 | AVX |
| VADDSUBPS/D | y,y,m256 | 1 | 1 | 1 | | 1+ | | | 1 | AVX |
| HADDPS/D HSUBPS/D | x,x | 3 | 3 | 1 | 2 | | | 5 | 2 | SSE3 |
| HADDPS/D HSUBPS/D | x,m128 | 4 | 3 | 1 | 2 | 1 | | | 2 | SSE3 |
| VHADDPS/D VHSUBPS/D | y,y,y | 3 | 3 | 1 | 2 | | | 5 | 2 | AVX |

Ivy Bridge

| | | | | | | | | | | |
|---------------------|----------|---|---|---|---|---|----|-------|-------|--------|
| VHADDPS/D VHSUBPS/D | y,y,m256 | 4 | 3 | | 1 | 2 | 1+ | | 2 | AVX |
| MULSS MULPS | x,x | 1 | 1 | 1 | | | | 5 | 1 | |
| MULSS MULPS | x,m | 1 | 1 | 1 | | | 1 | | 1 | |
| VMULPS | y,y,y | 1 | 1 | 1 | | | | 5 | 1 | AVX |
| VMULPS | y,y,m256 | 1 | 1 | 1 | | | 1+ | | 1 | AVX |
| MULSD MULPD | x,x | 1 | 1 | 1 | | | | 5 | 1 | |
| MULSD MULPD | x,m | 1 | 1 | 1 | | | 1 | | 1 | |
| VMULPD | y,y,y | 1 | 1 | 1 | | | | 5 | 1 | AVX |
| VMULPD | y,y,m256 | 1 | 1 | 1 | | | 1+ | | 1 | AVX |
| DIVSS DIVPS | x,x | 1 | 1 | 1 | | | | 10-13 | 7 | |
| DIVSS DIVPS | x,m | 1 | 1 | 1 | | | 1 | | 7 | |
| VDIVPS | y,y,y | 3 | 3 | 2 | | 1 | | 19-21 | 14 | AVX |
| VDIVPS | y,y,m256 | 4 | 3 | 2 | | 1 | 1+ | | 14 | AVX |
| DIVSD DIVPD | x,x | 1 | 1 | 1 | | | | 10-20 | 8-14 | |
| DIVSD DIVPD | x,m | 1 | 1 | 1 | | | 1 | | 8-14 | |
| VDIVPD | y,y,y | 3 | 3 | 2 | | 1 | | 20-35 | 16-28 | AVX |
| VDIVPD | y,y,m256 | 4 | 3 | 2 | | 1 | 1+ | | 16-28 | AVX |
| RCPSS/PS | x,x | 1 | 1 | 1 | | | | 5 | 1 | |
| RCPSS/PS | x,m128 | 1 | 1 | 1 | | | 1 | | 1 | |
| VRCPPS | y,y | 3 | 3 | 2 | | 1 | | 7 | 2 | AVX |
| VRCPPS | y,m256 | 4 | 3 | 2 | | 1 | 1+ | | 2 | AVX |
| CMPccSS/D CMPccPS/D | x,x | 1 | 1 | | 1 | | | 3 | 1 | |
| CMPccSS/D CMPccPS/D | x,m128 | 2 | 1 | | 1 | | 1 | | 1 | |
| VCMPccPS/D | y,y,y | 1 | 1 | | 1 | | | 3 | 1 | AVX |
| VCMPccPS/D | y,y,m256 | 2 | 1 | | 1 | | 1+ | | 1 | AVX |
| COMISS/D UCOMISS/D | x,x | 2 | 2 | 1 | 1 | | | | 1 | |
| COMISS/D UCOMISS/D | x,m32/64 | 2 | 2 | 1 | 1 | | 1 | | 1 | |
| MAXSS/D MINSS/D | x,x | 1 | 1 | | 1 | | | 3 | 1 | |
| MAXSS/D MINSS/D | x,m32/64 | 1 | 1 | | 1 | | 1 | | 1 | |
| MAXPS/D MINPS/D | x,x | 1 | 1 | | 1 | | | 3 | 1 | |
| MAXPS/D MINPS/D | x,m128 | 1 | 1 | | 1 | | 1 | | 1 | |
| VMAXPS/D VMINPS/D | y,y,y | 1 | 1 | | 1 | | | 3 | 1 | AVX |
| VMAXPS/D VMINPS/D | y,y,m256 | 1 | 1 | | 1 | | 1+ | | 1 | AVX |
| ROUNDSS/SD/PS/PD | x,x,i | 1 | 1 | | 1 | | | 3 | 1 | SSE4.1 |
| ROUNDSS/SD/PS/PD | x,m128,i | 2 | 1 | | 1 | | 1 | | 1 | SSE4.1 |
| VROUNDSS/SD/PS/PD | y,y,i | 1 | 1 | | 1 | | | 3 | 1 | AVX |
| VROUNDSS/SD/PS/PD | y,m256,i | 2 | 1 | | 1 | | 1+ | | 1 | AVX |
| DPPS | x,x,i | 4 | 4 | 1 | 2 | 1 | | 12 | 2 | SSE4.1 |
| DPPS | x,m128,i | 6 | 5 | 1 | 2 | 2 | 1 | | 4 | SSE4.1 |
| VDPPS | y,y,y,i | 4 | 4 | 1 | 2 | 1 | | 12 | 2 | AVX |
| VDPPS | y,m256,i | 6 | 5 | 1 | 2 | 2 | 1+ | | 4 | AVX |
| DPPD | x,x,i | 3 | 3 | 1 | 1 | 1 | | 9 | 1 | SSE4.1 |
| DPPD | x,m128,i | 4 | 3 | 1 | 1 | 1 | 1 | | 1 | SSE4.1 |
| Math | | | | | | | | | | |
| SQRTSS/PS | x,x | 1 | 1 | 1 | | | | 11 | 7 | |
| SQRTSS/PS | x,m128 | 1 | 1 | 1 | | | 1 | | 7 | |
| VSQRTPS | y,y | 3 | 3 | 2 | | 1 | | 19 | 14 | AVX |
| VSQRTPS | y,m256 | 4 | 3 | 2 | | 1 | 1+ | | 14 | AVX |
| SQRTSD/PD | x,x | 1 | 1 | 1 | | | | 16 | 8-14 | |

Ivy Bridge

| | | | | | | | | | | |
|-----------------------|----------|---------|---|---|---|----|---|--------|-------|--------|
| SQRTSD/PD | x,m128 | 1 | 1 | 1 | | 1 | | | 8-14 | |
| VSQRTPD | y,y | 3 | 3 | 2 | 1 | | | 28 | 16-28 | AVX |
| VSQRTPD | y,m256 | 4 | 3 | 2 | 1 | 1+ | | | 16-28 | AVX |
| RSQRTSS/PS | x,x | 1 | 1 | 1 | | | | 5 | 1 | |
| RSQRTSS/PS | x,m128 | 1 | 1 | 1 | | 1 | | | 1 | |
| VRSQRTPS | y,y | 3 | 3 | 2 | 1 | | | 7 | 2 | AVX |
| VRSQRTPS | y,m256 | 4 | 3 | 2 | 1 | 1+ | | | 2 | AVX |
| Logic | | | | | | | | | | |
| AND/ANDN/OR/XORPS/PD | x,x | 1 | 1 | | 1 | | | 1 | 1 | |
| AND/ANDN/OR/XORPS/PD | x,m128 | 1 | 1 | | 1 | 1 | | | 1 | |
| VAND/ANDN/OR/XORPS/PD | y,y,y | 1 | 1 | | 1 | | | 1 | 1 | AVX |
| VAND/ANDN/OR/XORPS/PD | y,y,m256 | 1 | 1 | | 1 | 1+ | | | 1 | AVX |
| Other | | | | | | | | | | |
| VZEROUPPER | | 4 | 0 | | | | | | 1 | AVX |
| VZEROALL | | 12 | 2 | | | | | | 11 | 32 bit |
| VZEROALL | | 20 | 2 | | | | | | 9 | 64 bit |
| LDMXCSR | m32 | 3 | 2 | 1 | 1 | 1 | | 6 | 3 | |
| STMXCSR | m32 | 3 | 2 | 1 | 1 | 1 | 1 | 7 | 1 | |
| FXSAVE | m4096 | 130 | | | | | | | 66 | |
| FXRSTOR | m4096 | 116 | | | | | | | 68 | |
| XSAVEOPT | m | 100-161 | | | | | | 60-500 | | |

Intel Haswell**List of instruction timings and μ op breakdown***Explanation of column headings:*

| | |
|--|---|
| Instruction: | Name of instruction. Multiple names mean that these instructions have the same data. Instructions with or without V name prefix behave the same unless otherwise noted. |
| Operands: | i = immediate data, r = register, mm = 64 bit mmx register, x = 128 bit xmm register, mm/x = mmx or xmm register, y = 256 bit ymm register, v = any vector register (mmx, xmm, ymm). same = same register for both operands. m = memory operand, m32 = 32-bit memory operand, etc. |
| μops fused domain: | The number of μ ops at the decode, rename and allocate stages in the pipeline. Fused μ ops count as one. |
| μops unfused domain: | The total number of μ ops for all execution port. Fused μ ops count as two. Fused macro-ops count as one. The instruction has μ op fusion if this number is higher than the number under fused domain. Some operations are not counted here if they do not go to any execution port or if the counters are inaccurate. |
| μops each port: | The number of μ ops for each execution port. p0 means a μ op to execution port 0. p01 means a μ op that can go to either port 0 or port 1. p0 p1 means two μ ops going to port 0 and 1, respectively. Port 0: Integer, f.p. and vector ALU, mul, div, branch Port 1: Integer, f.p. and vector ALU Port 2: Load Port 3: Load Port 4: Store Port 5: Integer and vector ALU Port 6: Integer ALU, branch Port 7: Store address |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Where hyperthreading is enabled, the use of the same execution units in the other thread leads to inferior performance. Denormal numbers, NAN's and infinity do not increase the latency. The time unit used is core clock cycles, not the reference clock cycles given by the time stamp counter. |
| Reciprocal throughput: | The average number of core clock cycles per instruction for a series of independent instructions of the same kind in the same thread. |

Integer instructions

| Instruction | Operands | μ ops fused domain | μ ops unfused domain | μ ops each port | Latency | Reciprocal throughput | Comments |
|--------------------------|---------------|------------------------|--------------------------|---------------------|---------|-----------------------|----------------------|
| Move instructions | | | | | | | |
| MOV | r,i | 1 | 1 | p0156 | | 0.25 | |
| MOV | r8/16,r8/16 | 1 | 1 | p0156 | 1 | 0.25 | |
| MOV | r32/64,r32/64 | 1 | 1 | p0156 | 0-1 | 0.25 | may be elim. |
| MOV | r8l,m | 1 | 2 | p23 p0156 | | 0.5 | |
| MOV | r8h,m | 1 | 1 | p23 | | 0.5 | |
| MOV | r16,m | 1 | 2 | p23 p0156 | | 0.5 | |
| MOV | r32/64,m | 1 | 1 | p23 | 2 | 0.5 | all addressing modes |
| MOV | m,r | 1 | 2 | p237 p4 | 3 | 1 | |
| MOV | m,i | 1 | 2 | p237 p4 | | 1 | |
| MOVNTI | m,r | 2 | 2 | p23 p4 | ~400 | 1 | |

Haswell

| | | | | | | | |
|--------------------------------|---------------|----|----|-----------------|----|------|------------------------------|
| MOVSX MOVZX | r,r | 1 | 1 | p0156 | 1 | 0.25 | |
| MOVSLD | | | | | | | |
| MOVSX MOVZX | r16,m8 | 1 | 2 | p23 p0156 | | 0.5 | |
| MOVSX MOVZX | r,m | 1 | 1 | p23 | | 0.5 | all other combinations |
| MOVSLD | | | | | | | |
| CMOVcc | r,r | 2 | 2 | 2p0156 | 2 | 0.5 | |
| CMOVcc | r,m | 3 | 3 | 2p0156 p23 | | 1 | |
| XCHG | r,r | 3 | 3 | 3p0156 | 2 | 1 | |
| XCHG | r,m | 8 | 8 | | 21 | | implicit lock |
| XLAT | | 3 | 3 | | 7 | 2 | |
| PUSH | r | 1 | 2 | p237 p4 | 3 | 1 | |
| PUSH | i | 1 | 2 | p237 p4 | | 1 | |
| PUSH | m | 2 | 3 | p4 2p237 | | 1 | |
| PUSH | stack pointer | 2 | 3 | p0156 p237 p4 | | 1 | |
| PUSHF(D/Q) | | 3 | 4 | p1 p4 p237 p06 | | 1 | |
| PUSHA(D) | | 11 | 19 | | | 8 | not 64 bit |
| POP | r | 1 | 1 | p23 | 2 | 0.5 | |
| POP | stack pointer | 3 | 3 | p23 2p0156 | | 4 | |
| POP | m | 2 | 3 | 2p237 p4 | | 1 | |
| POPF(D/Q) | | 9 | 9 | | | 18 | |
| POPA(D) | | 18 | 18 | | | 9 | not 64 bit |
| LAHF SAHF | | 1 | 1 | p06 | 1 | 1 | |
| SALC | | 3 | 3 | 3p0156 | 1 | 1 | not 64 bit |
| LEA | r16,m | 2 | 2 | p1 p0156 | 4 | 1 | 16 or 32 bit address size |
| LEA | r32/64,m | 1 | 1 | p15 | 1 | 0.5 | 1 or 2 components in address |
| LEA | r32/64,m | 1 | 1 | p1 | 3 | 1 | 3 components in address |
| LEA | r32/64,m | 1 | 1 | p1 | | 1 | rip relative address |
| BSWAP | r32 | 1 | 1 | p15 | 1 | 0.5 | |
| BSWAP | r64 | 2 | 2 | p06 p15 | 2 | 1 | |
| MOVBE | r16,m16 | 3 | 3 | 2p0156 p23 | | 0.5 | MOVBE |
| MOVBE | r32,m32 | 2 | 2 | p15 p23 | | 0.5 | MOVBE |
| MOVBE | r64,m64 | 3 | 3 | 2p0156 p23 | | 0.5 | MOVBE |
| MOVBE | m16,r16 | 2 | 3 | p06 p237 p4 | | 1 | MOVBE |
| MOVBE | m32,r32 | 2 | 3 | p15 p237 p4 | | 1 | MOVBE |
| MOVBE | m64,r64 | 3 | 4 | p06 p15 p237 p4 | | 1 | MOVBE |
| PREFETCHNTA/0/1/2 | m | 1 | 1 | p23 | | 0.5 | |
| LFENCE | | 2 | | none counted | | 4 | |
| MFENCE | | 3 | 2 | p23 p4 | | 33 | |
| SFENCE | | 2 | 2 | p23 p4 | | 5 | |
| Arithmetic instructions | | | | | | | |
| ADD SUB | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | |
| ADD SUB | r,m | 1 | 2 | p0156 p23 | | 0.5 | |
| ADD SUB | m,r/i | 2 | 4 | 2p0156 2p237 p4 | 6 | 1 | |
| ADC SBB | r,r/i | 2 | 2 | 2p0156 | 2 | 1 | |
| ADC SBB | r,m | 2 | 3 | 2p0156 p23 | | 1 | |

Haswell

| | | | | | | | |
|---------------------------|-------------|----|----|-----------------|--------|-------|------------|
| ADC SBB | m,r/i | 4 | 6 | 3p0156 2p237 p4 | 7 | 2 | |
| CMP | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | |
| CMP | m,r/i | 1 | 2 | p0156 p23 | 1 | 0.5 | |
| INC DEC NEG | r | 1 | 1 | p0156 | 1 | 0.25 | |
| NOT | | | | | | | |
| INC DEC NOT | m | 3 | 4 | p0156 2p237 p4 | 6 | 1 | |
| NEG | m | 2 | 4 | p0156 2p237 p4 | 6 | 1 | |
| AAA | | 2 | 2 | p1 p0156 | 4 | | not 64 bit |
| AAS | | 2 | 2 | p1 p56 | 6 | | not 64 bit |
| DAA DAS | | 3 | 3 | p1 2p0156 | 4 | | not 64 bit |
| AAD | | 3 | 3 | p1 2p0156 | 4 | | not 64 bit |
| AAM | | 8 | 8 | p0 p1 p5 p6 | 21 | 8 | not 64 bit |
| MUL IMUL | r8 | 1 | 1 | p1 | 3 | 1 | |
| MUL IMUL | r16 | 4 | 4 | p1 p0156 | 4 | 2 | |
| MUL IMUL | r32 | 3 | 3 | p1 p0156 | 4 | 2 | |
| MUL IMUL | r64 | 2 | 2 | p1 p6 | 3 | 1 | |
| MUL IMUL | m8 | 1 | 2 | p1 p23 | | 1 | |
| MUL IMUL | m16 | 4 | 5 | p1 3p0156 p23 | | 2 | |
| MUL IMUL | m32 | 3 | 4 | p1 2p0156 p23 | | 2 | |
| MUL IMUL | m64 | 2 | 3 | p1 p6 p23 | | 1 | |
| IMUL | r,r | 1 | 1 | p1 | 3 | 1 | |
| IMUL | r,m | 1 | 2 | p1 p23 | | 1 | |
| IMUL | r16,r16,i | 2 | 2 | p1 p0156 | 4 | 1 | |
| IMUL | r32,r32,i | 1 | 1 | p1 | 3 | 1 | |
| IMUL | r64,r64,i | 1 | 1 | p1 | 3 | 1 | |
| IMUL | r16,m16,i | 2 | 3 | p1 p0156 p23 | | 1 | |
| IMUL | r32,m32,i | 1 | 2 | p1 p23 | | 1 | |
| IMUL | r64,m64,i | 1 | 2 | p1 p23 | | 1 | |
| MULX | r32,r32,r32 | 3 | 3 | p1 2p056 | 4 | 1 | BMI2 |
| MULX | r32,r32,m32 | 3 | 4 | p1 2p056 p23 | | 1 | BMI2 |
| MULX | r64,r64,r64 | 2 | 2 | p1 p6 | 4 | 1 | BMI2 |
| MULX | r64,r64,m64 | 2 | 3 | p1 p6 p23 | | 1 | BMI2 |
| DIV | r8 | 9 | 9 | p0 p1 p5 p6 | 22-25 | 9 | |
| DIV | r16 | 11 | 11 | p0 p1 p5 p6 | 23-26 | 9 | |
| DIV | r32 | 10 | 10 | p0 p1 p5 p6 | 22-29 | 9-11 | |
| DIV | r64 | 36 | 36 | p0 p1 p5 p6 | 32-96 | 21-74 | |
| IDIV | r8 | 9 | 9 | p0 p1 p5 p6 | 23-26 | 8 | |
| IDIV | r16 | 10 | 10 | p0 p1 p5 p6 | 23-26 | 8 | |
| IDIV | r32 | 9 | 9 | p0 p1 p5 p6 | 22-29 | 8-11 | |
| IDIV | r64 | 59 | 59 | p0 p1 p5 p6 | 39-103 | 24-81 | |
| CBW | | 1 | 1 | p0156 | 1 | | |
| CWDE | | 1 | 1 | p0156 | 1 | | |
| CDQE | | 1 | 1 | p0156 | 1 | | |
| CWD | | 2 | 2 | p0156 | 1 | | |
| CDQ | | 1 | 1 | p06 | 1 | | |
| CQO | | 1 | 1 | p06 | 1 | | |
| POPCNT | r,r | 1 | 1 | p1 | 3 | 1 | SSE4.2 |
| POPCNT | r,m | 1 | 2 | p1 p23 | | 1 | SSE4.2 |
| CRC32 | r,r | 1 | 1 | p1 | 3 | 1 | SSE4.2 |
| CRC32 | r,m | 1 | 2 | p1 p23 | | 1 | SSE4.2 |
| Logic instructions | | | | | | | |

Haswell

| | | | | | | | |
|----------------|--------|----|----|-----------------|---|------|------------|
| AND OR XOR | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | |
| AND OR XOR | r,m | 1 | 2 | p0156 p23 | | 0.5 | |
| AND OR XOR | m,r/i | 2 | 4 | 2p0156 2p237 p4 | 6 | 1 | |
| TEST | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | |
| TEST | m,r/i | 1 | 2 | p0156 p23 | | 0.5 | |
| SHR SHL SAR | r,i | 1 | 1 | p06 | 1 | 0.5 | |
| SHR SHL SAR | m,i | 3 | 4 | 2p06 p237 p4 | | 2 | |
| SHR SHL SAR | r,cl | 3 | 3 | 3p06 | 2 | 2 | |
| SHR SHL SAR | m,cl | 5 | 6 | 3p06 2p23 p4 | | 4 | |
| ROR ROL | r,1 | 2 | 2 | 2p06 | 1 | 1 | short form |
| ROR ROL | r,i | 1 | 1 | p06 | 1 | 0.5 | |
| ROR ROL | m,i | 4 | 5 | 2p06 2p237 p4 | | 2 | |
| ROR ROL | r,cl | 3 | 3 | 3p06 | 2 | 2 | |
| ROR ROL | m,cl | 5 | 6 | | | 4 | |
| RCR RCL | r,1 | 3 | 3 | 2p06 p0156 | 2 | 2 | |
| RCR RCL | m,1 | 4 | 6 | | | 3 | |
| RCR RCL | r,i | 8 | 8 | p0156 | 6 | 6 | |
| RCR RCL | m,i | 11 | 11 | | | 6 | |
| RCR RCL | r,cl | 8 | 8 | p0156 | 6 | 6 | |
| RCR RCL | m,cl | 11 | 11 | | | 6 | |
| SHRD SHLD | r,r,i | 1 | 1 | p1 | 3 | 1 | |
| SHRD SHLD | m,r,i | 3 | 5 | | | 2 | |
| SHLD | r,r,cl | 4 | 4 | p0156 | 3 | 2 | |
| SHRD | r,r,cl | 4 | 4 | p0156 | 4 | 2 | |
| SHRD SHLD | m,r,cl | 5 | 7 | | | 4 | |
| SHLX SHRX SARX | r,r,r | 1 | 1 | p06 | 1 | 0.5 | BMI2 |
| SHLX SHRX SARX | r,m,r | 2 | 2 | p06 p23 | | 0.5 | BMI2 |
| RORX | r,r,i | 1 | 1 | p06 | 1 | 0.5 | BMI2 |
| RORX | r,m,i | 2 | 2 | p06 p23 | | 0.5 | BMI2 |
| BT | r,r/i | 1 | 1 | p06 | 1 | 0.5 | |
| BT | m,r | 10 | 10 | | | 5 | |
| BT | m,i | 2 | 2 | p06 p23 | | 0.5 | |
| BTR BTS BTC | r,r/i | 1 | 1 | p06 | 1 | 0.5 | |
| BTR BTS BTC | m,r | 10 | 11 | | | 5 | |
| BTR BTS BTC | m,i | 3 | 4 | 2p06 p23 p4 | | 2 | |
| BSF BSR | r,r | 1 | 1 | p1 | 3 | 1 | |
| BSF BSR | r,m | 1 | 2 | p1 p23 | | 1 | |
| SETcc | r | 1 | 1 | p06 | 1 | 0.5 | |
| SETcc | m | 2 | 3 | p06 p237 p4 | | 1 | |
| CLC | | 1 | 0 | none | | 0.25 | |
| STC | | 1 | 1 | p0156 | | 0.25 | |
| CMC | | 1 | 1 | p0156 | 1 | | |
| CLD STD | | 3 | 3 | p15 p6 | | 4 | |
| LZCNT | r,r | 1 | 1 | p1 | 3 | 1 | LZCNT |
| LZCNT | r,m | 1 | 2 | p1 p23 | | 1 | LZCNT |
| TZCNT | r,r | 1 | 1 | p1 | 3 | 1 | BMI1 |
| TZCNT | r,m | 1 | 2 | p1 p23 | | 1 | BMI1 |
| ANDN | r,r,r | 1 | 1 | p15 | 1 | 0.5 | BMI1 |
| ANDN | r,r,m | 1 | 2 | p15 p23 | 1 | 0.5 | BMI1 |
| BLSI BLSMSK | r,r | 1 | 1 | p15 | 1 | 0.5 | BMI1 |
| BLSR | | | | | | | |
| BLSI BLSMSK | r,m | 1 | 2 | p15 p23 | | 0.5 | BMI1 |
| BLSR | | | | | | | |

Haswell

| | | | | | | | |
|--------------------------------------|------------|---------|----|----------------|---|--------|-------------------------|
| BEXTR | r,r,r | 2 | 2 | 2p0156 | 2 | 0.5 | BMI1 |
| BEXTR | r,m,r | 3 | 3 | 2p0156 p23 | | 1 | BMI1 |
| BZHI | r,r,r | 1 | 1 | p15 | 1 | 0.5 | BMI2 |
| BZHI | r,m,r | 1 | 2 | p15 p23 | | 0.5 | BMI2 |
| PDEP | r,r,r | 1 | 1 | p1 | 3 | 1 | BMI2 |
| PDEP | r,r,m | 1 | 2 | p1 p23 | | 1 | BMI2 |
| PEXT | r,r,r | 1 | 1 | p1 | 3 | 1 | BMI2 |
| PEXT | r,r,m | 1 | 2 | p1 p23 | | 1 | BMI2 |
| Control transfer instructions | | | | | | | |
| JMP | short/near | 1 | 1 | p6 | | 1-2 | |
| JMP | r | 1 | 1 | p6 | | 2 | |
| JMP | m | 1 | 2 | p23 p6 | | 2 | |
| Conditional jump | short/near | 1 | 1 | p6 | | 1-2 | predicted taken |
| Conditional jump | short/near | 1 | 1 | p06 | | 0.5-1 | predicted not taken |
| Fused arithmetic and branch | | 1 | 1 | p6 | | 1-2 | predicted taken |
| Fused arithmetic and branch | | 1 | 1 | p06 | | 0.5-1 | predicted not taken |
| J(E/R)CXZ | short | 2 | 2 | p0156 p6 | | 0.5-2 | |
| LOOP | short | 7 | 7 | | | 5 | |
| LOOP(N)E | short | 11 | 11 | | | 6 | |
| CALL | near | 2 | 3 | p237 p4 p6 | | 2 | |
| CALL | r | 2 | 3 | p237 p4 p6 | | 2 | |
| CALL | m | 3 | 4 | 2p237 p4 p6 | | 3 | |
| RET | | 1 | 2 | p237 p6 | | 1 | |
| RET | i | 3 | 4 | p23 2p6 p015 | | 2 | |
| BOUND | r,m | 15 | 15 | | | 8 | not 64 bit |
| INTO | | 4 | 4 | | | 5 | not 64 bit |
| String instructions | | | | | | | |
| LODSB/W | | 3 | 3 | 2p0156 p23 | | 1 | |
| LODSD/Q | | 2 | 2 | p0156 p23 | | 1 | |
| REP LODS | | 5n+12 | | | | ~2n | |
| STOS | | 3 | 3 | p23 p0156 p4 | | 1 | |
| REP STOS | | <2n | | | | ~0.5n | worst case |
| REP STOS | | 2.6/32B | | | | 1/32B | best case aligned by 32 |
| MOVS | | 5 | 5 | 2p23 p4 2p0156 | | 4 | |
| REP MOVS | | ~2n | | | | ~1.5 n | worst case |
| REP MOVS | | 4/32B | | | | 1/32B | best case aligned by 32 |
| SCAS | | 3 | 3 | p23 2p0156 | | 1 | |
| REP SCAS | | ≥6n | | | | ≥2n | |
| CMPS | | 5 | 5 | 2p23 3p0156 | | 4 | |
| REP CMPS | | ≥8n | | | | ≥2n | |
| Synchronization instructions | | | | | | | |
| XADD | m,r | 4 | 5 | | | 7 | |
| LOCK XADD | m,r | 9 | 9 | | | 19 | |
| LOCK ADD | m,r | 8 | 8 | | | 19 | |

Haswell

| | | | | | | | |
|------------------|-----|--------|--------|-------------|--------|------|--------|
| CMPXCHG | m,r | 5 | 6 | | | 8 | |
| LOCK CMPXCHG | m,r | 10 | 10 | | | 19 | |
| CMPXCHG8B | m,r | 15 | 15 | | | 9 | |
| LOCK CMPXCHG8B | m,r | 19 | 19 | | | 19 | |
| CMPXCHG16B | m,r | 22 | 22 | | | 15 | |
| LOCK CMPXCHG16B | m,r | 24 | 24 | | | 25 | |
| Other | | | | | | | |
| NOP (90) | | 1 | 0 | none | | 0.25 | |
| Long NOP (0F 1F) | | 1 | 0 | none | | 0.25 | |
| PAUSE | | 5 | 5 | p05 3p6 | | 9 | |
| ENTER | a,0 | 12 | 12 | | | 8 | |
| ENTER | a,b | ~14+7b | ~45+7b | | ~87+2b | | |
| LEAVE | | 3 | 3 | 2p0156 p23 | | 6 | |
| XGETBV | | 8 | 8 | | | 9 | XGETBV |
| RDTSC | | 15 | 15 | | | 24 | |
| RDPMSR | | 34 | 34 | | | 37 | |
| RDRAND | r | 17 | 17 | p23 16p0156 | | ~320 | RDRAND |

Floating point x87 instructions

| Instruction | Operands | μops fused domain | μops unfused domain | μops each port | Latency | Reciprocal throughput | Comments |
|--------------------------|----------|-------------------|---------------------|-----------------|---------|-----------------------|----------|
| Move instructions | | | | | | | |
| FLD | r | 1 | 1 | p01 | 1 | 0.5 | |
| FLD | m32/64 | 1 | 1 | p23 | 3 | 0.5 | |
| FLD | m80 | 4 | 4 | 2p01 2p23 | 4 | 2 | |
| FBLD | m80 | 43 | 43 | | 47 | 22 | |
| FST(P) | r | 1 | 1 | p01 | 1 | 0.5 | |
| FST(P) | m32/m64 | 1 | 2 | p4 p237 | 4 | 1 | |
| FSTP | m80 | 7 | 7 | 3p0156 2p23 2p4 | 1 | 5 | |
| FBSTP | m80 | 238 | 226 | | | 265 | |
| FXCH | r | 2 | 0 | none | 0 | 0.5 | |
| FILD | m | 1 | 2 | p01 p23 | 6 | 1 | |
| FIST(P) | m | 3 | 3 | p1 p23 p4 | 7 | 1 | |
| FISTTP | m | 3 | 3 | p1 p23 p4 | 7 | 2 | SSE3 |
| FLDZ | | 1 | 1 | p01 | | 1 | |
| FLD1 | | 2 | 2 | 2p01 | | 2 | |
| FLDPI FLDL2E etc. | | 2 | 2 | 2p01 | | 2 | |
| FCMOVcc | r | 3 | 3 | 2p0 p5 | 2 | 2 | |
| FNSTSW | AX | 2 | 2 | p0 p0156 | | 1 | |
| FNSTSW | m16 | 2 | 3 | p0 p4 p237 | 6 | 1 | |
| FLDCW | m16 | 3 | 3 | p01 p23 p6 | 7 | 2 | |
| FNSTCW | m16 | 2 | 3 | p237 p4 p6 | | 1 | |
| FINCSTP FDECSTP | | 1 | 1 | p01 | 0 | 0.5 | |
| FFREE(P) | r | 1 | 1 | p01 | | 0.5 | |
| FNSAVE | m | 147 | 147 | | | 150 | |
| FRSTOR | m | 90 | 90 | | | 164 | |

Haswell

| Arithmetic in-structions | | | | | | |
|--------------------------|---|---------|----|-----------|--------|------|
| FADD(P) | | | | | | |
| FSUB(R)(P) | r | 1 | 1 | p1 | 3 | 1 |
| FADD(P) | | | | | | |
| FSUB(R)(P) | m | 1 | 2 | p1 p23 | | 1 |
| FMUL(P) | r | 1 | 1 | p0 | 5 | 1 |
| FMUL(P) | m | 1 | 2 | p0 p23 | | 1 |
| FDIV(R)(P) | r | 1 | 1 | p0 | 10-24 | 8-18 |
| FDIV(R)(P) | m | 1 | 2 | p0 p23 | | 8-18 |
| FABS | | 1 | 1 | p0 | 1 | 1 |
| FCHS | | 1 | 1 | p0 | 1 | 1 |
| FCOM(P) FUCOM | r | 1 | 1 | p1 | | 1 |
| FCOM(P) FUCOM | m | 1 | 2 | p1 p23 | | 1 |
| FCOMPP FUCOMPP | | 2 | 2 | 2p01 | | 1 |
| FCOMI(P) FUCOM | r | 3 | 3 | 3p01 | | 1.5 |
| FIADD FISUB(R) | m | 2 | 3 | 2p1 p23 | | 2 |
| FIMUL | m | 2 | 3 | p0 p1 p23 | | 2 |
| FIDIV(R) | m | 2 | 3 | p0 p1 p23 | | |
| FICOM(P) | m | 2 | 3 | 2p1 p23 | | 2 |
| FTST | | 1 | 1 | p1 | | 1 |
| FXAM | | 2 | 2 | 2p1 | | 2 |
| FPREM | | 28 | 28 | | 19 | 13 |
| FPREM1 | | 41 | 41 | | 27 | 17 |
| FRNDINT | | 17 | 17 | | 11 | 23 |
| Math | | | | | | |
| FSCALE | | 25-75 | | | 49-125 | |
| FEXTRACT | | 17 | 17 | | 15 | 11 |
| FSQRT | | 1 | 1 | p0 | 10-23 | 8-17 |
| FSIN | | 71-100 | | | 47-106 | |
| FCOS | | 110 | | | 112 | |
| FSINCOS | | 70-120 | | | 52-123 | |
| F2XM1 | | 58-89 | | | 63-68 | |
| FYL2X | | 55-417 | | | 58-680 | |
| FYL2XP1 | | 55-228 | | | 58-360 | |
| FPTAN | | 110-121 | | | 130 | |
| FPATAN | | 78-160 | | | 96-156 | |
| Other | | | | | | |
| FNOP | | 1 | 1 | p01 | | 0.5 |
| WAIT | | 2 | 2 | p01 | | 1 |
| FNCLEX | | 5 | 5 | p0156 | | 22 |
| FNINIT | | 26 | 26 | | | 83 |

Integer vector instructions

| Instruction | Operands | μops fused domain | μops unfused domain | μops each port | Latency | Reciprocal through put | Comments |
|-------------------|-------------|-------------------|---------------------|----------------|---------|------------------------|----------|
| Move instructions | | | | | | | |
| MOVD | r32/64,mm/x | 1 | 1 | p0 | 1 | 1 | |

Haswell

| | | | | | | | |
|---------------|-------------|---|---|---------|------|------|---------------------|
| MOVD | m32/64,mm/x | 1 | 2 | p237 p4 | 3 | 1 | |
| MOVD | mm/x,r32/64 | 1 | 1 | p5 | 1 | 1 | |
| MOVD | mm/x,m32/64 | 1 | 1 | p23 | 3 | 0.5 | |
| MOVQ | r64,mm/x | 1 | 1 | p0 | 1 | 1 | |
| MOVQ | mm/x,r64 | 1 | 1 | p5 | 1 | 1 | |
| MOVQ | mm/x,mm/x | 1 | | p015 | 1 | 0.33 | |
| MOVQ | mm/x,m64 | 1 | 1 | p23 | 3 | 0.5 | |
| MOVQ | m64, mm/x | 1 | 2 | p237 p4 | 3 | 1 | |
| MOVDQA/U | x,x | 1 | 1 | p015 | 0-1 | 0.33 | may be elim. |
| MOVDQA/U | x, m128 | 1 | 1 | p23 | 3 | 0.5 | |
| MOVDQA/U | m128, x | 1 | 2 | p237 p4 | 3 | 1 | |
| VMOVDQA/U | y,y | 1 | 1 | p015 | 0-1 | 0.33 | AVX may be elim. |
| VMOVDQA/U | y,m256 | 1 | 1 | p23 | 3 | 0.5 | AVX |
| VMOVDQA/U | m256,y | 1 | 2 | p237 p4 | 4 | 1 | AVX |
| LDDQU | x, m128 | 1 | 1 | p23 | 3 | 0.5 | SSE3 |
| MOVDQ2Q | mm, x | 2 | 2 | p01 p5 | 1 | 1 | |
| MOVQ2DQ | x,mm | 1 | 1 | p015 | 1 | 0.33 | |
| MOVNTQ | m64,mm | 1 | 2 | p237 p4 | ~400 | 1 | |
| MOVNTDQ | m128,x | 1 | 2 | p237 p4 | ~400 | 1 | |
| VMOVNTDQ | m256,y | 1 | 2 | p237 p4 | ~400 | 1 | AVX2 |
| MOVNTDQA | x, m128 | 1 | 1 | p23 | 3 | 0.5 | SSE4.1 |
| VMOVNTDQA | y,m256 | 1 | 1 | p23 | 3 | 0.5 | AVX2 |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | mm,mm | 3 | 3 | p5 | 2 | 2 | |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | mm,m64 | 3 | 3 | p23 2p5 | | 2 | |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | x,x / y,y,y | 1 | 1 | p5 | 1 | 1 | |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | x,m / y,y,m | 1 | 2 | p23 p5 | | 1 | |
| PACKUSDW | x,x / y,y,y | 1 | 1 | p5 | 1 | 1 | SSE4.1 |
| PACKUSDW | x,m / y,y,m | 1 | 2 | p23 p5 | | 1 | SSE4.1 |
| PUNPCKH/L | | | | | | | |
| BW/WD/DQ | v,v / v,v,v | 1 | 1 | p5 | 1 | 1 | |
| PUNPCKH/L | | | | | | | |
| BW/WD/DQ | v,m / v,v,m | 1 | 2 | p23 p5 | | 1 | |
| PUNPCKH/L | | | | | | | |
| QDQ | x,x / y,y,y | 1 | 1 | p5 | 1 | 1 | |
| PUNPCKH/L | | | | | | | |
| QDQ | x,m / y,y,m | 2 | 2 | p23 p5 | | 1 | |
| PMOVSX/ZX BW | | | | | | | |
| BD BQ DW DQ | x,x | 1 | 1 | p5 | 1 | 1 | SSE4.1 |
| PMOVSX/ZX BW | | | | | | | |
| BD BQ DW DQ | x,m | 1 | 2 | p23 p5 | | 1 | SSE4.1 |
| VPMOVSX/ZX BW | | | | | | | |
| BD BQ DW DQ | y,x | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPMOVSX/ZX BW | | | | | | | |
| BD BQ DW DQ | y,m | 2 | 2 | p5 p23 | | 1 | AVX2 |
| PSHUFB | v,v / v,v,v | 1 | 1 | p5 | 1 | 1 | SSSE3 |
| PSHUFB | v,m / v,v,m | 2 | 2 | p23 p5 | | 1 | SSSE3 |
| PSHUFW | mm,mm,i | 1 | 1 | p5 | 1 | 1 | |
| PSHUFW | mm,m64,i | 2 | 2 | p23 p5 | | 1 | |
| PSHUFD | v,v,i | 1 | 1 | p5 | 1 | 1 | |
| PSHUFD | v,m,i | 2 | 2 | p23 p5 | | 1 | |

Haswell

| | | | | | | | |
|----------------|-----------------|----|----|----------------|--------|------|--------|
| PSHUFL/HW | v,v,i | 1 | 1 | p5 | 1 | 1 | |
| PSHUFL/HW | v,m,i | 2 | 2 | p23 p5 | | 1 | |
| PALIGNR | v,v,i / v,v,v,i | 1 | 1 | p5 | 1 | 1 | SSSE3 |
| PALIGNR | v,m,i / v,v,m,i | 2 | 2 | p23 p5 | | 1 | SSSE3 |
| PBLENDVB | x,x,xmm0 | 2 | 2 | 2p5 | 2 | 2 | SSE4.1 |
| PBLENDVB | x,m,xmm0 | 3 | 3 | 2p5 p23 | | 2 | SSE4.1 |
| VPBLENDVB | v,v,v,v | 2 | 2 | 2p5 | 2 | 2 | AVX2 |
| VPBLENDVB | v,v,m,v | 3 | 3 | 2p5 p23 | | 2 | AVX2 |
| PBLENDW | x,x,i / v,v,v,i | 1 | 1 | p5 | 1 | 1 | SSE4.1 |
| PBLENDW | x,m,i / v,v,m,i | 2 | 2 | p23 p5 | | 1 | SSE4.1 |
| VPBLEND | v,v,v,i | 1 | 1 | p015 | 1 | 0.33 | AVX2 |
| VPBLEND | v,v,m,i | 2 | 2 | p015 p23 | | 0.5 | AVX2 |
| VPERMD | y,y,y | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPERMD | y,y,m | 1 | 2 | p5 p23 | | 1 | AVX2 |
| VPERMQ | y,y,i | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPERMQ | y,m,i | 2 | 2 | p5 p23 | | 1 | AVX2 |
| VPERM2I128 | y,y,y,i | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPERM2I128 | y,y,m,i | 2 | 2 | p5 p23 | | 1 | AVX2 |
| MASKMOVQ | mm,mm | 4 | 4 | p0 p4 2p23 | 13-413 | 1 | |
| MASKMOVQ | x,x | 10 | 10 | 4p04 2p56 4p23 | 14-438 | 6 | |
| VPMASKMOVQ | v,v,m | 3 | 3 | p23 2p5 | 4 | 2 | AVX2 |
| VPMASKMOVQ | m,v,v | 4 | 4 | p0 p1 p4 p23 | 13-14 | 1 | AVX2 |
| PMOVMSKB | r,v | 1 | 1 | p0 | 3 | 1 | |
| PEXTRB/W/D/Q | r32,x,i | 2 | 2 | p0 p5 | 2 | 1 | SSE4.1 |
| PEXTRB/W/D/Q | m8,x,i | 2 | 3 | p23 p4 p5 | | 1 | SSE4.1 |
| VEXTRACTI128 | x,y,i | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VEXTRACTI128 | m,y,i | 2 | 2 | p23 p4 | 4 | 1 | AVX2 |
| PINSRB | x,r32,i | 2 | 2 | p5 | 2 | 2 | SSE4.1 |
| PINSRB | x,m8,i | 2 | 2 | p23 p5 | | 1 | SSE4.1 |
| PINSRW | mm/x,r32,i | 2 | 2 | p5 | 2 | 2 | |
| PINSRW | mm/x,m16,i | 2 | 2 | p23 p5 | | 1 | |
| PINSRD/Q | x,r32,i | 2 | 2 | p5 | 2 | 2 | SSE4.1 |
| PINSRD/Q | x,m32,i | 2 | 2 | p23 p5 | | 1 | SSE4.1 |
| VINSERTI128 | y,y,x,i | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VINSERTI128 | y,y,m,i | 2 | 2 | p015 p23 | 4 | 0.5 | AVX2 |
| VPBROADCAST | | | | | | | |
| B/W/D/Q | x,x | 1 | 1 | p5 | 1 | 1 | AVX2 |
| VPBROADCAST | | | | | | | |
| B/W | x,m8/16 | 3 | 3 | p01 p23 p5 | 5 | 1 | AVX2 |
| VPBROADCAST | | | | | | | |
| D/Q | x,m32/64 | 1 | 1 | p23 | 4 | 0.5 | AVX2 |
| VPBROADCAST | | | | | | | |
| B/W/D/Q | y,x | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPBROADCAST | | | | | | | |
| B/W | y,m8/16 | 3 | 3 | p01 p23 p5 | 7 | 1 | AVX2 |
| VPBROADCAST | | | | | | | |
| D/Q | y,m32/64 | 1 | 1 | p23 | 5 | 0.5 | AVX2 |
| VBROADCASTI128 | y,m128 | 1 | 1 | p23 | 3 | 0.5 | AVX2 |
| VPGATHERDD | x,[r+s*x],x | 20 | 20 | | | 9 | AVX2 |
| VPGATHERDD | y,[r+s*y],y | 34 | 34 | | | 12 | AVX2 |
| VPGATHERQD | x,[r+s*x],x | 15 | 15 | | | 8 | AVX2 |
| VPGATHERQD | x,[r+s*y],x | 22 | 22 | | | 7 | AVX2 |
| VPGATHERDQ | x,[r+s*x],x | 12 | 12 | | | 7 | AVX2 |
| VPGATHERDQ | y,[r+s*x],y | 20 | 20 | | | 9 | AVX2 |
| VPGATHERQQ | x,[r+s*x],x | 14 | 14 | | | 7 | AVX2 |

Haswell

| | | | | | | | |
|--------------------------------|-----------------|----|----|------------|----|-----|--------|
| VPGATHERQQ | y,[r+s*y],y | 22 | 22 | | | 9 | AVX2 |
| Arithmetic instructions | | | | | | | |
| PADD/SUB(S,US) B/W/D/Q | v,v / v,v,v | 1 | 1 | p15 | 1 | 0.5 | |
| PADD/SUB(S,US) B/W/D/Q | v,m / v,v,m | 1 | 2 | p15 p23 | | 0.5 | |
| PHADD(S)W/D | | | | | | | |
| PHSUB(S)W/D | v,v / v,v,v | 3 | 3 | p1 2p5 | 3 | 2 | SSSE3 |
| PHADD(S)W/D | | | | | | | |
| PHSUB(S)W/D | v,m / v,v,m | 4 | 4 | p1 2p5 p23 | | 2 | SSSE3 |
| PCMPEQB/W/D | | | | | | | |
| PCMPGTB/W/D | v,v / v,v,v | 1 | 1 | p15 | 1 | 0.5 | |
| PCMPEQB/W/D | | | | | | | |
| PCMPGTB/W/D | v,m / v,v,m | 1 | 2 | p15 p23 | | 0.5 | |
| PCMPEQQ | v,v / v,v,v | 1 | 1 | p15 | 1 | 0.5 | SSE4.1 |
| PCMPEQQ | v,m / v,v,m | 1 | 2 | p15 p23 | | 0.5 | SSE4.1 |
| PCMPGTQ | v,v / v,v,v | 1 | 1 | p0 | 5 | 1 | SSE4.2 |
| PCMPGTQ | v,m / v,v,m | 1 | 2 | p0 p23 | | 1 | SSE4.2 |
| PMULL/HW | | | | | | | |
| PMULHUW | v,v / v,v,v | 1 | 1 | p0 | 5 | 1 | |
| PMULL/HW | | | | | | | |
| PMULHUW | v,m / v,v,m | 1 | 2 | p0 p23 | | 1 | |
| PMULHRSW | v,v / v,v,v | 1 | 1 | p0 | 5 | 1 | SSSE3 |
| PMULHRSW | v,m / v,v,m | 1 | 2 | p0 p23 | | 1 | SSSE3 |
| PMULLD | x,x / y,y,y | 2 | 2 | 2p0 | 10 | 2 | SSE4.1 |
| PMULLD | x,m / y,y,m | 3 | 3 | 2p0 p23 | | 2 | SSE4.1 |
| PMULDQ | x,x / y,y,y | 1 | 1 | p0 | 5 | 1 | SSE4.1 |
| PMULDQ | x,m / y,y,m | 1 | 2 | p0 p23 | | 1 | SSE4.1 |
| PMULUDQ | v,v / v,v,v | 1 | 1 | p0 | 5 | 1 | |
| PMULUDQ | v,m / v,v,m | 1 | 2 | p0 p23 | | 1 | |
| PMADDWD | v,v / v,v,v | 1 | 1 | p0 | 5 | 1 | |
| PMADDWD | v,m / v,v,m | 1 | 2 | p0 p23 | | 1 | |
| PMADDUBSW | v,v / v,v,v | 1 | 1 | p0 | 5 | 1 | SSSE3 |
| PMADDUBSW | v,m / v,v,m | 1 | 2 | p0 p23 | | 1 | SSSE3 |
| PAVGB/W | v,v / v,v,v | 1 | 1 | p15 | 1 | 0.5 | |
| PAVGB/W | v,m / v,v,m | 1 | 2 | p15 p23 | | 0.5 | |
| PMIN/PMAX SB/SW/SD | | | | | | | |
| UB/UW/UD | x,x / y,y,y | 1 | 1 | p15 | 1 | 0.5 | SSE4.1 |
| PMIN/PMAX SB/SW/SD | | | | | | | |
| UB/UW/UD | x,m / y,y,m | 1 | 2 | p15 p23 | | 0.5 | SSE4.1 |
| PHMINPOSUW | x,x | 1 | 1 | p0 | 5 | 1 | SSE4.1 |
| PHMINPOSUW | x,m128 | 1 | 2 | p0 p23 | | 1 | SSE4.1 |
| PABSB/W/D | v,v | 1 | 1 | p15 | 1 | 0.5 | SSSE3 |
| PABSB/W/D | v,m | 1 | 2 | p15 p23 | | 0.5 | SSSE3 |
| PSIGNB/W/D | v,v / v,v,v | 1 | 1 | p15 | 1 | 0.5 | SSSE3 |
| PSIGNB/W/D | v,m / v,v,m | 1 | 2 | p15 p23 | | 0.5 | SSSE3 |
| PSADBW | v,v / v,v,v | 1 | 1 | p0 | 5 | 1 | |
| PSADBW | v,m / v,v,m | 1 | 2 | p0 p23 | | 1 | |
| MPSADBW | x,x,i / v,v,v,i | 3 | 3 | p0 2p5 | 6 | 2 | SSE4.1 |
| MPSADBW | x,m,i / v,v,m,i | 4 | 4 | p0 2p5 p23 | | 2 | SSE4.1 |

Haswell

| Logic instructions | | | | | | | |
|---|-------------|---|---|------------------|----|------|--------|
| PAND PANDN | v,v / v,v,v | 1 | 1 | p015 | 1 | 0.33 | |
| POR PXOR | | | | | | | |
| PAND PANDN | | | | | | | |
| POR PXOR | | 1 | 2 | p015 p23 | | 0.5 | |
| PTEST | v,v | 2 | 2 | p0 p5 | 2 | 1 | SSE4.1 |
| PTEST | v,m | 2 | 3 | p0 p5 p23 | | 1 | SSE4.1 |
| PSLLW/D/Q | | | | | | | |
| PSRLW/D/Q | | | | | | | |
| PSRAW/D/Q | mm,mm | 1 | 1 | p0 | 1 | 1 | |
| PSLLW/D/Q | | | | | | | |
| PSRLW/D/Q | | | | | | | |
| PSRAW/D/Q | mm,m64 | 1 | 2 | p0 p23 | | 1 | |
| PSLLW/D/Q | | | | | | | |
| PSRLW/D/Q | | | | | | | |
| PSRAW/D/Q | x,x / v,v,x | 2 | 2 | p0 p5 | 2 | 1 | |
| PSLLW/D/Q | | | | | | | |
| PSRLW/D/Q | | | | | | | |
| PSRAW/D/Q | x,m / v,v,m | 2 | 2 | p0 p23 | | 1 | |
| PSLLW/D/Q | | | | | | | |
| PSRLW/D/Q | | | | | | | |
| PSRAW/D/Q | v,i / v,v,i | 1 | 1 | p0 | 1 | 1 | |
| VPSLLVD/Q | | | | | | | |
| VPSRAVD | | | | | | | |
| VPSRLVD/Q | v,v,v | 3 | 3 | 2p0 p5 | 2 | 2 | AVX2 |
| VPSLLVD/Q | | | | | | | |
| VPSRAVD | | | | | | | |
| VPSRLVD/Q | v,v,m | 4 | 4 | 2p0 p5 p23 | | 2 | AVX2 |
| PSLLDQ | | | | | | | |
| PSRLDQ | x,i / v,v,i | 1 | 1 | p5 | 1 | 1 | |
| String instructions | | | | | | | |
| PCMPESTRI | x,x,i | 8 | 8 | 6p05 2p16 | 11 | 4 | SSE4.2 |
| PCMPESTRI | x,m128,i | 8 | 8 | 3p0 2p16 2p5 p23 | | 4 | SSE4.2 |
| PCMPESTRM | x,x,i | 9 | 9 | 3p0 2p16 4p5 | 10 | 5 | SSE4.2 |
| PCMPESTRM | x,m128,i | 9 | 9 | 6p05 2p16 p23 | | 5 | SSE4.2 |
| PCMPISTRI | x,x,i | 3 | 3 | 3p0 | 11 | 3 | SSE4.2 |
| PCMPISTRI | x,m128,i | 4 | 4 | 3p0 p23 | | 3 | SSE4.2 |
| PCMPISTRM | x,x,i | 3 | 3 | 3p0 | 10 | 3 | SSE4.2 |
| PCMPISTRM | x,m128,i | 4 | 4 | 3p0 p23 | | 3 | SSE4.2 |
| Encryption instructions | | | | | | | |
| PCLMULQDQ | x,x,i | 3 | 3 | 2p0 p5 | 7 | 2 | CLMUL |
| PCLMULQDQ | x,m,i | 4 | 4 | 2p0 p5 p23 | | 2 | CLMUL |
| AESDEC, AESDECLAST, AESENC, AESENCLAST | x,x | 1 | 1 | p5 | 7 | 1 | AES |
| AESDEC, AESDECLAST, AESENC, AESENCLAST | x,m | 2 | 2 | p5 p23 | | 1.5 | AES |
| AESIMC | x,x | 2 | 2 | 2p5 | 14 | 2 | AES |
| AESIMC | x,m | 3 | 3 | 2p5 p23 | | 2 | AES |

Haswell

| | | | | | | | |
|---------------------|-------|----|----|-------------|----|----|-----|
| AESKEYGENAS SIST | x,x,i | 10 | 10 | 2p0 8p5 | 10 | 9 | AES |
| AESKEYGENAS SIST | x,m,i | 10 | 10 | 2p0 p23 7p5 | | 8 | AES |
| Other | | | | | | | |
| EMMS | | 31 | 31 | | | 13 | |

Floating point XMM and YMM instructions

| Instruction | Operands | μops fused domain | μops unfused domain | μops each port | Latency | Recipro- cal through put | Comments |
|--------------------------------|-----------------|-------------------------|---------------------------|----------------|---------|-----------------------------------|--------------|
| Move instruc- tions | | | | | | | |
| MOVAPS/D | x,x | 1 | 1 | p5 | 0-1 | 1 | may be elim. |
| VMOVAPS/D | y,y | 1 | 1 | p5 | 0-1 | 1 | may be elim. |
| MOVAPS/D | | | | | | | |
| MOVUPS/D | x,m128 | 1 | 1 | p23 | 3 | 0.5 | |
| VMOVAPS/D | | | | | | | |
| VMOVUPS/D | y,m256 | 1 | 1 | p23 | 3 | 0.5 | AVX |
| MOVAPS/D | | | | | | | |
| MOVUPS/D | m128,x | 1 | 2 | p237 p4 | 3 | 1 | |
| VMOVAPS/D | | | | | | | |
| VMOVUPS/D | m256,y | 1 | 2 | p237 p4 | 4 | 1 | AVX |
| MOVSS/D | x,x | 1 | 1 | p5 | 1 | 1 | |
| MOVSS/D | x,m32/64 | 1 | 1 | p23 | 3 | 0.5 | |
| MOVSS/D | m32/64,x | 1 | 2 | p237 p4 | 3 | 1 | |
| MOVHPS/D | x,m64 | 1 | 2 | p23 p5 | 4 | 1 | |
| MOVHPS/D | m64,x | 1 | 2 | p4 p237 | 3 | 1 | |
| MOVLPS/D | x,m64 | 1 | 2 | p23 p5 | 4 | 1 | |
| MOVLPS/D | m64,x | 1 | 2 | p4 p237 | 3 | 1 | |
| MOVHLPs | x,x | 1 | 1 | p5 | 1 | 1 | |
| MOVLHPs | x,x | 1 | 1 | p5 | 1 | 1 | |
| MOVMSKPS/D | r32,x | 1 | 1 | p0 | 3 | 1 | |
| VMOVMSKPS/D | r32,y | 1 | 1 | p0 | 2 | 1 | |
| MOVNTPS/D | m128,x | 1 | 2 | p4 p237 | ~400 | 1 | |
| VMOVNTPS/D | m256,y | 1 | 2 | p4 p237 | ~400 | 1 | AVX |
| SHUFPS/D | x,x,i / v,v,v,i | 1 | 1 | p5 | 1 | 1 | |
| SHUFPS/D | x,m,i / v,v,m,i | 2 | 2 | p5 p23 | | 1 | |
| VPERMILPS/PD | v,v,i | 1 | 1 | p5 | 1 | 1 | AVX |
| VPERMILPS/PD | v,m,i | 2 | 2 | p5 p23 | | 1 | AVX |
| VPERMILPS/PD | v,v,v | 1 | 1 | p5 | 1 | 1 | AVX |
| VPERMILPS/PD | v,v,m | 2 | 2 | p5 p23 | | 1 | AVX |
| VPERM2F128 | y,y,y,i | 1 | 1 | p5 | 3 | 1 | AVX |
| VPERM2F128 | y,y,m,i | 2 | 2 | p5 p23 | | 1 | AVX |
| VPERMPS | y,y,y | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPERMPS | y,y,m | 1 | 2 | p5 p23 | | 1 | AVX2 |
| VPERMPD | y,y,i | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPERMPD | y,m,i | 2 | 2 | p5 p23 | | 1 | AVX2 |
| BLENDPS/PD | x,x,i / v,v,v,i | 1 | 1 | p015 | 1 | 0.33 | SSE4.1 |
| BLENDPS/PD | x,m,i / v,v,m,i | 2 | 2 | p015 p23 | | 0.5 | SSE4.1 |
| BLENDVPS/PD | x,x,xmm0 | 2 | 2 | 2p5 | 2 | 2 | SSE4.1 |
| BLENDVPS/PD | x,m,xmm0 | 3 | 3 | 2p5 p23 | | 2 | SSE4.1 |

Haswell

| | | | | | | | |
|-------------------|-------------|----|----|--------------|----|-----|--------|
| VBLENDVPS/PD | v,v,v,v | 2 | 2 | 2p5 | 2 | 2 | AVX |
| VBLENDVPS/PD | v,v,m,v | 3 | 3 | 2p5 p23 | | 2 | AVX |
| MOVDDUP | v,v | 1 | 1 | p5 | 1 | 1 | SSE3 |
| MOVDDUP | v,m | 1 | 1 | p23 | 3 | 0.5 | SSE3 |
| VBROADCASTSS | x,m32 | 1 | 1 | p23 | 4 | 0.5 | AVX |
| VBROADCASTSS | y,m32 | 1 | 1 | p23 | 5 | 0.5 | AVX |
| VBROADCASTSS | x,x | 1 | 1 | p5 | 1 | 1 | AVX2 |
| VBROADCASTSS | y,x | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VBROADCASTSD | y,m64 | 1 | 1 | p23 | 5 | 0.5 | AVX |
| VBROADCASTSD | y,x | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VBROADCASTF128 | y,m128 | 1 | 1 | p23 | 3 | 0.5 | AVX |
| MOVSH/LDUP | v,v | 1 | 1 | p5 | 1 | 1 | SSE3 |
| MOVSH/LDUP | v,m | 1 | 1 | p23 | 3 | 0.5 | SSE3 |
| UNPCKH/LPS/D | x,x / v,v,v | 1 | 1 | p5 | 1 | 1 | SSE3 |
| UNPCKH/LPS/D | x,m / v,v,m | 1 | 2 | p5 p23 | | 1 | SSE3 |
| EXTRACTPS | r32,x,i | 2 | 2 | p0 p5 | | 1 | SSE4.1 |
| EXTRACTPS | m32,x,i | 3 | 3 | p0 p5 p23 | 4 | 1 | SSE4.1 |
| VEEXTRACTF128 | x,y,i | 1 | 1 | p5 | 3 | 1 | AVX |
| VEEXTRACTF128 | m128,y,i | 2 | 2 | p23 p4 | 4 | 1 | AVX |
| INSERTPS | x,x,i | 1 | 1 | p5 | 1 | 1 | SSE4.1 |
| INSERTPS | x,m32,i | 2 | 2 | p23 p5 | 4 | 1 | SSE4.1 |
| VINSERTF128 | y,y,x,i | 1 | 1 | p5 | 3 | 1 | AVX |
| VINSERTF128 | y,y,m128,i | 2 | 2 | p015 p23 | 4 | 2 | AVX |
| VMASKMOVPS/D | v,v,m | 3 | 3 | 2p5 p23 | 4 | 2 | AVX |
| VMASKMOVPS/D | m128,x,x | 4 | 4 | p0 p1 p4 p23 | 13 | 1 | AVX |
| VMASKMOVPS/D | m256,y,y | 4 | 4 | p0 p1 p4 p23 | 14 | 2 | AVX |
| VGATHERDPS | x,[r+s*x],x | 20 | 20 | | | 9 | AVX2 |
| VGATHERDPS | y,[r+s*y],y | 34 | 34 | | | 12 | AVX2 |
| VGATHERQPS | x,[r+s*x],x | 15 | 15 | | | 8 | AVX2 |
| VGATHERQPS | x,[r+s*y],x | 22 | 22 | | | 7 | AVX2 |
| VGATHERDPD | x,[r+s*x],x | 12 | 12 | | | 7 | AVX2 |
| VGATHERDPD | y,[r+s*x],y | 20 | 20 | | | 9 | AVX2 |
| VGATHERQPD | x,[r+s*x],x | 14 | 14 | | | 7 | AVX2 |
| VGATHERQPD | y,[r+s*y],y | 22 | 22 | | | 9 | AVX2 |
| Conversion | | | | | | | |
| CVTPD2PS | x,x | 2 | 2 | p1 p5 | 4 | 1 | |
| CVTPD2PS | x,m128 | 2 | 3 | p1 p5 p23 | | 1 | |
| VCVTPD2PS | x,y | 2 | 2 | p1 p5 | 5 | 1 | AVX |
| VCVTPD2PS | x,m256 | 2 | 3 | p1 p5 p23 | | 1 | AVX |
| CVTSD2SS | x,x | 2 | 2 | p1 p5 | 4 | 1 | |
| CVTSD2SS | x,m64 | 2 | 3 | p1 p5 p23 | | 1 | |
| CVTPS2PD | x,x | 2 | 2 | p0 p5 | 2 | 1 | |
| CVTPS2PD | x,m64 | 2 | 2 | p0 p23 | | 1 | |
| VCVTPS2PD | y,x | 2 | 2 | p0 p5 | 5 | 1 | AVX |
| VCVTPS2PD | y,m128 | 2 | 2 | p0 p23 | | 1 | AVX |
| CVTSS2SD | x,x | 2 | 2 | p0 p5 | 2 | 1 | |
| CVTSS2SD | x,m32 | 2 | 2 | p0 p23 | | 1 | |
| CVTDQ2PS | x,x | 1 | 1 | p1 | 3 | 1 | |
| CVTDQ2PS | x,m128 | 1 | 2 | p1 p23 | | 1 | |
| VCVTDQ2PS | y,y | 1 | 1 | p1 | 3 | 1 | AVX |
| VCVTDQ2PS | y,m256 | 1 | 2 | p1 p23 | | 1 | AVX |
| CVT(T) PS2DQ | x,x | 1 | 1 | p1 | 3 | 1 | |

Haswell

| | | | | | | | |
|-------------------|-------------|---|---|--------------|-------|-------|------|
| CVT(T) PS2DQ | x,m128 | 1 | 2 | p1 p23 | | 1 | |
| VCVT(T) PS2DQ | y,y | 1 | 1 | p1 | 3 | 1 | AVX |
| VCVT(T) PS2DQ | y,m256 | 1 | 2 | p1 p23 | | 1 | AVX |
| CVTDQ2PD | x,x | 2 | 2 | p1 p5 | 4 | 1 | |
| CVTDQ2PD | x,m64 | 2 | 2 | p1 p23 | | 1 | |
| VCVTDQ2PD | y,x | 2 | 2 | p1 p5 | 6 | 1 | AVX |
| VCVTDQ2PD | y,m128 | 2 | 2 | p1 p23 | | 1 | AVX |
| CVT(T)PD2DQ | x,x | 2 | 2 | p1 p5 | 4 | 1 | |
| CVT(T)PD2DQ | x,m128 | 2 | 3 | p1 p5 p23 | | 1 | |
| VCVT(T)PD2DQ | x,y | 2 | 2 | p1 p5 | 6 | 1 | AVX |
| VCVT(T)PD2DQ | x,m256 | 2 | 3 | p1 p5 p23 | | 1 | AVX |
| CVTPI2PS | x,mm | 1 | 1 | p1 | 4 | 4 | |
| CVTPI2PS | x,m64 | 1 | 2 | p1 p23 | | 3 | |
| CVT(T)PS2PI | mm,x | 2 | 2 | p1 p5 | 4 | 1 | |
| CVT(T)PS2PI | mm,m128 | 2 | 2 | p1 p23 | | 1 | |
| CVTPI2PD | x,mm | 2 | 2 | p1 p5 | 4 | 1 | |
| CVTPI2PD | x,m64 | 2 | 2 | p1 p23 | | 1 | |
| CVT(T) PD2PI | mm,x | 2 | 2 | p1 p5 | 4 | 1 | |
| CVT(T) PD2PI | mm,m128 | 2 | 3 | p1 p5 p23 | | 1 | |
| CVTSI2SS | x,r32 | 2 | 2 | p1 p5 | 4 | 3 | |
| CVTSI2SS | x,m32 | 1 | 2 | p1 p23 | | 3 | |
| CVT(T)SS2SI | r32,x | 2 | 2 | p0 p1 | 4 | 1 | |
| CVT(T)SS2SI | r32,m32 | 2 | 3 | p0 p1 p23 | | 1 | |
| CVTSI2SD | x,r32/64 | 2 | 2 | p1 p5 | 4 | 3 | |
| CVTSI2SD | x,m32 | 2 | 2 | p1 p23 | | 3 | |
| CVT(T)SD2SI | r32/64,x | 2 | 2 | p0 p1 | 4 | 1 | |
| CVT(T)SD2SI | r32,m64 | 2 | 3 | p0 p1 p23 | | 1 | |
| VCVTPS2PH | x,v,i | 2 | 2 | p1 p5 | 4 | 1 | F16C |
| VCVTPS2PH | m,v,i | 4 | 4 | p1 p4 p5 p23 | | 1 | F16C |
| VCVTPH2PS | v,x | 2 | 2 | p1 p5 | 4 | 1 | F16C |
| VCVTPH2PS | v,m | 2 | 2 | p1 p23 | | 1 | F16C |
| Arithmetic | | | | | | | |
| ADDSS/D PS/D | | | | | | | |
| SUBSS/D PS/D | x,x / v,v,v | 1 | 1 | p1 | 3 | 1 | |
| ADDSS/D PS/D | | | | | | | |
| SUBSS/D PS/D | x,m / v,v,m | 1 | 2 | p1 p23 | | 1 | |
| ADDSS/D PS/D | x,x / v,v,v | 1 | 1 | p1 | 3 | 1 | SSE3 |
| ADDSS/D PS/D | x,m / v,v,m | 1 | 2 | p1 p23 | | 1 | SSE3 |
| HADDPS/D | | | | | | | |
| HSUBPS/D | x,x / v,v,v | 3 | 3 | p1 2p5 | 5 | 2 | SSE3 |
| HADDPS/D | | | | | | | |
| HSUBPS/D | x,m / v,v,m | 4 | 4 | p1 2p5 p23 | | 2 | SSE3 |
| MULSS/D PS/D | x,x / v,v,v | 1 | 1 | p01 | 5 | 0.5 | |
| MULSS/D PS/D | x,m / v,v,m | 1 | 2 | p01 p23 | | 0.5 | |
| DIVSS DIVPS | x,x | 1 | 1 | p0 | 10-13 | 7 | |
| DIVSS DIVPS | x,m | 1 | 2 | p0 p23 | | 7 | |
| DIVSD DIVPD | x,x | 1 | 1 | p0 | 10-20 | 8-14 | |
| DIVSD DIVPD | x,m | 1 | 2 | p0 p23 | | 8-14 | |
| VDIVPS | y,y,y | 3 | 3 | 2p0 p15 | 18-21 | 14 | AVX |
| VDIVPS | y,y,m256 | 4 | 4 | 2p0 p15 p23 | | 14 | AVX |
| VDIVPD | y,y,y | 3 | 3 | 2p0 p15 | 19-35 | 16-28 | AVX |
| VDIVPD | y,y,m256 | 4 | 4 | 2p0 p15 p23 | | 16-28 | AVX |
| RCPSS/PS | x,x | 1 | 1 | p0 | 5 | 1 | |

Haswell

| | | | | | | | |
|--------------------------|-----------------|----|----|------------------|-------|-------|----------------|
| RCPSS/PS | x,m128 | 1 | 2 | p0 p23 | | 1 | |
| VRCPPS | y,y | 3 | 3 | 2p0 p15 | 7 | 2 | AVX |
| VRCPPS | y,m256 | 4 | 4 | 2p0 p15 p23 | | 2 | AVX |
| CMPccSS/D | | | | | | | |
| CMPccPS/D | x,x / v,v,v | 1 | 1 | p1 | 3 | 1 | |
| CMPccSS/D | | | | | | | |
| CMPccPS/D | x,m / v,v,m | 2 | 2 | p1 p23 | | 1 | |
| (U)COMISS/D | x,x | 1 | 1 | p1 | | 1 | |
| (U)COMISS/D | x,m32/64 | 2 | 2 | p1 p23 | | 1 | |
| MAXSS/D PS/D | | | | | | | |
| MINSS/D PS/D | x,x / v,v,v | 1 | 1 | p1 | 3 | 1 | |
| MAXSS/D PS/D | | | | | | | |
| MINSS/D PS/D | x,m / v,v,m | 1 | 2 | p1 p23 | | 1 | |
| ROUNDSS/D PS/D | v,v,i | 2 | 2 | 2p1 | 6 | 2 | SSE4.1 |
| ROUNDSS/D PS/D | v,m,i | 3 | 3 | 2p1 p23 | | 2 | SSE4.1 |
| DPPS | x,x,i / v,v,v,i | 4 | 4 | 2p0 p1 p5 | 14 | 2 | SSE4.1 |
| DPPS | x,m,i / v,v,m,i | 6 | 6 | 2p0 p1 p5 p23 p6 | | 4 | SSE4.1 |
| DPPD | x,x,i | 3 | 3 | p0 p1 p5 | 9 | 1 | SSE4.1 |
| DPPD | x,m128,i | 4 | 4 | p0 p1 p5 p23 | | 1 | SSE4.1 |
| VFMADD... | | | | | | | |
| (all FMA instr.) | v,v,v | 1 | 1 | p01 | 5 | 0.5 | FMA |
| VFMADD... | | | | | | | |
| (all FMA instr.) | v,v,m | 1 | 2 | p01 p23 | | 0.5 | FMA |
| Math | | | | | | | |
| SQRTSS/PS | x,x | 1 | 1 | p0 | 11 | 7 | |
| SQRTSS/PS | x,m128 | 1 | 2 | p0 p23 | | 7 | |
| VSQRTPS | y,y | 3 | 3 | 2p0 p15 | 19 | 14 | AVX |
| VSQRTPS | y,m256 | 4 | 4 | 2p0 p15 p23 | | 14 | AVX |
| SQRTSD/PD | x,x | 1 | 1 | p0 | 16 | 8-14 | |
| SQRTSD/PD | x,m128 | 1 | 2 | p0 p23 | | 8-14 | |
| VSQRTPD | y,y | 3 | 3 | 2p0 p15 | 28-29 | 16-28 | AVX |
| VSQRTPD | y,m256 | 4 | 4 | 2p0 p15 p23 | | 16-28 | AVX |
| RSQRTSS/PS | x,x | 1 | 1 | p0 | 5 | 1 | |
| RSQRTSS/PS | x,m128 | 1 | 2 | p0 p23 | | 1 | |
| VRSQRTPS | y,y | 3 | 3 | 2p0 p15 | 7 | 2 | AVX |
| VRSQRTPS | y,m256 | 4 | 4 | 2p0 p15 p23 | | 2 | AVX |
| Logic | | | | | | | |
| AND/ANDN/OR/ XORPS/PD | x,x / v,v,v | 1 | 1 | p5 | 1 | 1 | |
| AND/ANDN/OR/ XORPS/PD | x,m / v,v,m | 1 | 2 | p5 p23 | | 1 | |
| Other | | | | | | | |
| VZEROUPPER | | 4 | 4 | none | | 1 | AVX |
| VZEROALL | | 12 | 12 | none | | 10 | AVX, 32 bit |
| VZEROALL | | 20 | 20 | none | | 8 | AVX, 64 bit |
| LDMXCSR | m32 | 3 | 3 | p0 p6 p23 | 6 | 3 | |
| STMXCSR | m32 | 3 | 4 | p0 p4 p6 p237 | 7 | 1 | |
| VSTMXCSR | m32 | 3 | | | | 1 | AVX |

Haswell

| | | | | | | | |
|----------|-------|-----|--|--|--|-----|--|
| FXSAVE | m4096 | 130 | | | | 68 | |
| FXRSTOR | m4096 | 116 | | | | 72 | |
| XSAVE | | 224 | | | | 84 | |
| XRSTOR | | 173 | | | | 111 | |
| XSAVEOPT | m | | | | | | |

Intel Broadwell

List of instruction timings and μ op breakdown

Explanation of column headings:

| | |
|--|---|
| Instruction: | Name of instruction. Multiple names mean that these instructions have the same data. Instructions with or without V name prefix behave the same unless otherwise noted. |
| Operands: | i = immediate data, r = register, mm = 64 bit mmx register, x = 128 bit xmm register, mm/x = mmx or xmm register, y = 256 bit ymm register, v = any vector register (mmx, xmm, ymm). same = same register for both operands. m = memory operand, m32 = 32-bit memory operand, etc. |
| μops fused domain: | The number of μ ops at the decode, rename and allocate stages in the pipeline. Fused μ ops count as one. |
| μops unfused domain: | The total number of μ ops for all execution port. Fused μ ops count as two. Fused macro-ops count as one. The instruction has μ op fusion if this number is higher than the number under fused domain. Some operations are not counted here if they do not go to any execution port or if the counters are inaccurate. |
| μops each port: | The number of μ ops for each execution port. p0 means a μ op to execution port 0. p01 means a μ op that can go to either port 0 or port 1. p0 p1 means two μ ops going to port 0 and 1, respectively. Port 0: Integer, f.p. and vector ALU, mul, div, branch Port 1: Integer, f.p. and vector ALU Port 2: Load Port 3: Load Port 4: Store Port 5: Integer and vector ALU Port 6: Integer ALU, branch Port 7: Store address |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Where hyperthreading is enabled, the use of the same execution units in the other thread leads to inferior performance. Denormal numbers, NAN's and infinity do not increase the latency. The time unit used is core clock cycles, not the reference clock cycles given by the time stamp counter. |
| Reciprocal throughput: | The average number of core clock cycles per instruction for a series of independent instructions of the same kind in the same thread. |

Integer instructions

| Instruction | Operands | μ ops fused domain | μ ops unfused domain | μ ops each port | Latency | Reciprocal throughput | Comments |
|--------------------------|---------------|------------------------|--------------------------|---------------------|---------|-----------------------|----------------------|
| Move instructions | | | | | | | |
| MOV | r,i | 1 | 1 | p0156 | | 0.25 | |
| MOV | r8/16,r8/16 | 1 | 1 | p0156 | 1 | 0.25 | |
| MOV | r32/64,r32/64 | 1 | 1 | p0156 | 0-1 | 0.25 | may be elim. |
| MOV | r8l,m | 1 | 2 | p23 p0156 | | 0.5 | |
| MOV | r8h,m | 1 | 1 | p23 | | 0.5 | |
| MOV | r16,m | 1 | 2 | p23 p0156 | | 0.5 | |
| MOV | r32/64,m | 1 | 1 | p23 | 2 | 0.5 | all addressing modes |
| MOV | m,r | 1 | 2 | p237 p4 | 3 | 1 | |
| MOV | m,i | 1 | 2 | p237 p4 | | 1 | |
| MOVNTI | m,r | 2 | 2 | p23 p4 | ~400 | 1 | |

Broadwell

| | | | | | | | |
|--------------------------------|---------------|----|----|-----------------|-----|-------|------------------------------|
| MOVSBX MOVZXB | r,r | 1 | 1 | p0156 | 1 | 0.25 | |
| MOVSBXD | | | | | | | |
| MOVSBX MOVZXB | r16,m8 | 1 | 2 | p23 p0156 | | 0.5 | |
| MOVSBX MOVZXB | r,m | 1 | 1 | p23 | | 0.5 | all other combinations |
| MOVSBXD | | | | | | | |
| CMOVBcc | r,r | 1 | 1 | p06 | 1 | 0.5 | |
| CMOVBcc | r,m | 2 | 2 | p06 p23 | | 0.5 | |
| XCHG | r,r | 3 | 3 | 3p0156 | 2 | 1 | |
| XCHG | r,m | 8 | 8 | | 21 | | implicit lock |
| XLAT | | 3 | 3 | p23 2p0156 | 7 | 2 | |
| PUSH | r | 1 | 2 | p237 p4 | 3 | 1 | |
| PUSH | i | 1 | 2 | p237 p4 | | 1 | |
| PUSH | m | 2 | 3 | p4 2p237 | | 1 | |
| PUSH | stack pointer | 2 | 3 | p0156 p237 p4 | | 1 | |
| PUSHF(D/Q) | | 3 | 4 | p1 p4 p237 p06 | | 1 | |
| PUSHA(D) | | 11 | 19 | | | 8 | not 64 bit |
| POP | r | 1 | 1 | p23 | 2 | 0.5 | |
| POP | stack pointer | 3 | 3 | p23 2p0156 | | 4 | |
| POP | m | 2 | 3 | 2p237 p4 | | 1 | |
| POPF(D/Q) | | 9 | 9 | | | 18 | |
| POPA(D) | | 18 | 18 | | | 8 | not 64 bit |
| LAHF SAHF | | 1 | 1 | p06 | 1 | 1 | |
| SALC | | 3 | 3 | 3p0156 | 1 | 1 | not 64 bit |
| LEA | r16,m | 2 | 2 | p1 p05 | 2-4 | 1 | 16 or 32 bit address size |
| LEA | r32/64,m | 1 | 1 | p15 | 1 | 0.5 | 1 or 2 components in address |
| LEA | r32/64,m | 1 | 1 | p1 | 3 | 1 | 3 components in address |
| LEA | r32/64,m | 1 | 1 | p1 | | 1 | rip relative address |
| BSWAP | r32 | 1 | 1 | p15 | 1 | 0.5 | |
| BSWAP | r64 | 2 | 2 | p06 p15 | 2 | 1 | |
| MOVBE | r16,m16 | 3 | 3 | 2p0156 p23 | | 0.5-1 | MOVBE |
| MOVBE | r32,m32 | 2 | 2 | p15 p23 | | 0.5 | MOVBE |
| MOVBE | r64,m64 | 3 | 3 | 2p0156 p23 | | 0.5 | MOVBE |
| MOVBE | m16,r16 | 2 | 3 | p06 p237 p4 | | 1 | MOVBE |
| MOVBE | m32,r32 | 2 | 3 | p15 p237 p4 | | 1 | MOVBE |
| MOVBE | m64,r64 | 3 | 4 | p06 p15 p237 p4 | | 1 | MOVBE |
| PREFETCHNTA/0/1/2 | m | 1 | 1 | p23 | | 0.5 | |
| PREFETCHW | m | 1 | 1 | p23 | | 1 | PREFETCHW |
| LFENCE | | 2 | | none counted | | 4 | |
| MFENCE | | 3 | 3 | p23 p4 | | 33 | |
| SFENCE | | 2 | 2 | p23 p4 | | 6 | |
| Arithmetic instructions | | | | | | | |
| ADD SUB | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | |
| ADD SUB | r,m | 1 | 2 | p0156 p23 | | 0.5 | |
| ADD SUB | m,r/i | 2 | 4 | 2p0156 2p237 p4 | 6 | 1 | |
| ADC SBB | r,r/i | 1 | 1 | p06 | 1 | 1 | |

Broadwell

| | | | | | | | |
|-------------|-------------|----|----|-----------------|--------|-------|------------|
| ADC SBB | r,m | 2 | 2 | p06 p23 | | 1 | |
| ADC SBB | m,r/i | 4 | 6 | 3p0156 2p237 p4 | 7 | 2 | |
| CMP | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | |
| CMP | m,r/i | 1 | 2 | p0156 p23 | 1 | 0.5 | |
| INC DEC NEG | r | 1 | 1 | p0156 | 1 | 0.25 | |
| NOT | | | | | | | |
| INC DEC NOT | m | 3 | 4 | p0156 2p237 p4 | 6 | 1 | |
| NEG | m | 2 | 4 | p0156 2p237 p4 | 6 | 1 | |
| AAA | | 2 | 2 | p1 p56 | 4 | | not 64 bit |
| AAS | | 2 | 2 | p1 p056 | 6 | | not 64 bit |
| DAA DAS | | 3 | 3 | p1 2p056 | 4 | | not 64 bit |
| AAD | | 3 | 3 | p1 2p056 | 6 | | not 64 bit |
| AAM | | 8 | 8 | p0 p1 p5 p6 | 21 | 7 | not 64 bit |
| MUL IMUL | r8 | 1 | 1 | p1 | 3 | 1 | |
| MUL IMUL | r16 | 4 | 4 | p1 p0156 | 4 | 2 | |
| MUL IMUL | r32 | 3 | 3 | p1 p0156 | 4 | 2 | |
| MUL IMUL | r64 | 2 | 2 | p1 p6 | 3 | 1 | |
| MUL IMUL | m8 | 1 | 2 | p1 p23 | | 1 | |
| MUL IMUL | m16 | 4 | 5 | p1 3p0156 p23 | | 2 | |
| MUL IMUL | m32 | 3 | 4 | p1 2p0156 p23 | | 2 | |
| MUL IMUL | m64 | 2 | 3 | p1 p6 p23 | | 1 | |
| IMUL | r,r | 1 | 1 | p1 | 3 | 1 | |
| IMUL | r,m | 1 | 2 | p1 p23 | | 1 | |
| IMUL | r16,r16,i | 2 | 2 | p1 p0156 | 4 | 1 | |
| IMUL | r32,r32,i | 1 | 1 | p1 | 3 | 1 | |
| IMUL | r64,r64,i | 1 | 1 | p1 | 3 | 1 | |
| IMUL | r16,m16,i | 2 | 3 | p1 p0156 p23 | | 1 | |
| IMUL | r32,m32,i | 1 | 2 | p1 p23 | | 1 | |
| IMUL | r64,m64,i | 1 | 2 | p1 p23 | | 1 | |
| MULX | r32,r32,r32 | 3 | 3 | p1 2p056 | 4 | 1 | BMI2 |
| MULX | r32,r32,m32 | 3 | 4 | p1 2p056 p23 | | 1 | BMI2 |
| MULX | r64,r64,r64 | 2 | 2 | p1 p5 | 4 | 1 | BMI2 |
| MULX | r64,r64,m64 | 2 | 3 | p1 p6 p23 | | 1 | BMI2 |
| DIV | r8 | 9 | 9 | p0 p1 p5 p6 | 22-25 | 9 | |
| DIV | r16 | 11 | 11 | p0 p1 p5 p6 | 23-26 | 9 | |
| DIV | r32 | 10 | 10 | p0 p1 p5 p6 | 22-29 | 9 | |
| DIV | r64 | 36 | 36 | p0 p1 p5 p6 | 32-95 | 21-73 | |
| IDIV | r8 | 9 | 9 | p0 p1 p5 p6 | 23-26 | 6 | |
| IDIV | r16 | 10 | 10 | p0 p1 p5 p6 | 23-26 | 6 | |
| IDIV | r32 | 9 | 9 | p0 p1 p5 p6 | 22-29 | 6 | |
| IDIV | r64 | 59 | 59 | p0 p1 p5 p6 | 39-103 | 24-81 | |
| CBW | | 1 | 1 | p0156 | 1 | | |
| CWDE | | 1 | 1 | p0156 | 1 | | |
| CDQE | | 1 | 1 | p0156 | 1 | | |
| CWD | | 2 | 2 | p0156 | 1 | | |
| CDQ | | 1 | 1 | p06 | 1 | | |
| CQO | | 1 | 1 | p06 | 1 | | |
| POPCNT | r,r | 1 | 1 | p1 | 3 | 1 | SSE4.2 |
| POPCNT | r,m | 1 | 2 | p1 p23 | | 1 | SSE4.2 |
| CRC32 | r,r | 1 | 1 | p1 | 3 | 1 | SSE4.2 |
| CRC32 | r,m | 1 | 2 | p1 p23 | | 1 | SSE4.2 |

Broadwell

| Logic instructions | | | | | | | |
|--------------------|--------|----|----|-----------------|---|------|------------|
| AND OR XOR | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | |
| AND OR XOR | r,m | 1 | 2 | p0156 p23 | | 0.5 | |
| AND OR XOR | m,r/i | 2 | 4 | 2p0156 2p237 p4 | 6 | 1 | |
| TEST | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | |
| TEST | m,r/i | 1 | 2 | p0156 p23 | 1 | 0.5 | |
| SHR SHL SAR | r,i | 1 | 1 | p06 | 1 | 0.5 | |
| SHR SHL SAR | m,i | 3 | 4 | 2p06 p237 p4 | | 2 | |
| SHR SHL SAR | r,cl | 3 | 3 | 3p06 | 2 | 2 | |
| SHR SHL SAR | m,cl | 5 | 6 | 3p06 2p23 p4 | | 4 | |
| ROR ROL | r,1 | 2 | 2 | 2p06 | 1 | 1 | short form |
| ROR ROL | r,i | 1 | 1 | p06 | 1 | 0.5 | |
| ROR ROL | m,i | 4 | 5 | 2p06 2p237 p4 | | 2 | |
| ROR ROL | r,cl | 3 | 3 | 3p06 | 2 | 2 | |
| ROR ROL | m,cl | 5 | 6 | 3p06 p23 p4 | | 4 | |
| RCR RCL | r,1 | 3 | 3 | 2p06 p0156 | 2 | 2 | |
| RCR RCL | m,1 | 4 | 6 | | | 3 | |
| RCR RCL | r,i | 8 | 8 | p0156 | 6 | 6 | |
| RCR RCL | m,i | 11 | 11 | | | 6 | |
| RCR RCL | r,cl | 8 | 8 | p0156 | 6 | 6 | |
| RCR RCL | m,cl | 11 | 11 | | | 6 | |
| SHRD SHLD | r,r,i | 1 | 1 | p1 | 3 | 1 | |
| SHRD SHLD | m,r,i | 3 | 5 | | | 2 | |
| SHLD | r,r,cl | 4 | 4 | p0156 | 3 | 2 | |
| SHRD | r,r,cl | 4 | 4 | p0156 | 4 | 2 | |
| SHRD SHLD | m,r,cl | 5 | 7 | | | 4 | |
| SHLX SHRX SARX | r,r,r | 1 | 1 | p06 | 1 | 0.5 | BMI2 |
| SHLX SHRX SARX | r,m,r | 2 | 2 | p06 p23 | | 0.5 | BMI2 |
| RORX | r,r,i | 1 | 1 | p06 | 1 | 0.5 | BMI2 |
| RORX | r,m,i | 2 | 2 | p06 p23 | | 0.5 | BMI2 |
| BT | r,r/i | 1 | 1 | p06 | 1 | 0.5 | |
| BT | m,r | 10 | 10 | | | 5 | |
| BT | m,i | 2 | 2 | p06 p23 | | 0.5 | |
| BTR BTS BTC | r,r/i | 1 | 1 | p06 | 1 | 0.5 | |
| BTR BTS BTC | m,r | 10 | 10 | | | 5 | |
| BTR BTS BTC | m,i | 2 | 2 | p06 p23 | | 0.5 | |
| BSF BSR | r,r | 1 | 1 | p1 | 3 | 1 | |
| BSF BSR | r,m | 1 | 2 | p1 p23 | | 1 | |
| SETcc | r | 1 | 1 | p06 | 1 | 0.5 | |
| SETcc | m | 2 | 3 | p06 p237 p4 | | 1 | |
| CLC | | 1 | 0 | none | | 0.25 | |
| STC | | 1 | 1 | p0156 | | 0.25 | |
| CMC | | 1 | 1 | p0156 | 1 | 1 | |
| CLD STD | | 3 | 3 | p15 p6 | | 4 | |
| LZCNT | r,r | 1 | 1 | p1 | 3 | 1 | LZCNT |
| LZCNT | r,m | 1 | 2 | p1 p23 | | 1 | LZCNT |
| TZCNT | r,r | 1 | 1 | p1 | 3 | 1 | BMI1 |
| TZCNT | r,m | 1 | 2 | p1 p23 | | 1 | BMI1 |
| ANDN | r,r,r | 1 | 1 | p15 | 1 | 0.5 | BMI1 |
| ANDN | r,r,m | 1 | 2 | p15 p23 | 1 | 0.5 | BMI1 |
| BLSI BLSMSK | r,r | 1 | 1 | p15 | 1 | 0.5 | BMI1 |
| BLSR | | | | | | | |

Broadwell

| | | | | | | | |
|--------------------------------------|------------|---------|----|----------------|---|-------|-------------------------|
| BLSI BLSMSK | r,m | 1 | 2 | p15 p23 | | 0.5 | BMI1 |
| BLSR | | | | | | | |
| BEXTR | r,r,r | 2 | 2 | 2p0156 | 2 | 0.5 | BMI1 |
| BEXTR | r,m,r | 3 | 3 | 2p0156 p23 | | 1 | BMI1 |
| BZHI | r,r,r | 1 | 1 | p15 | 1 | 0.5 | BMI2 |
| BZHI | r,m,r | 1 | 2 | p15 p23 | | 0.5 | BMI2 |
| PDEP | r,r,r | 1 | 1 | p1 | 3 | 1 | BMI2 |
| PDEP | r,r,m | 1 | 2 | p1 p23 | | 1 | BMI2 |
| PEXT | r,r,r | 1 | 1 | p1 | 3 | 1 | BMI2 |
| PEXT | r,r,m | 1 | 2 | p1 p23 | | 1 | BMI2 |
| Control transfer instructions | | | | | | | |
| JMP | short/near | 1 | 1 | p6 | | 1-2 | |
| JMP | r | 1 | 1 | p6 | | 2 | |
| JMP | m | 1 | 2 | p23 p6 | | 2 | |
| Conditional jump | short/near | 1 | 1 | p6 | | 1-2 | predicted taken |
| Conditional jump | short/near | 1 | 1 | p06 | | 0.5-1 | predicted not taken |
| Fused arithmetic and branch | | 1 | 1 | p6 | | 1-2 | predicted taken |
| Fused arithmetic and branch | | 1 | 1 | p06 | | 0.5-1 | predicted not taken |
| J(E/R)CXZ | short | 2 | 2 | p0156 p6 | | 0.5-2 | |
| LOOP | short | 7 | 7 | | | 5 | |
| LOOP(N)E | short | 11 | 11 | | | 6 | |
| CALL | near | 2 | 3 | p237 p4 p6 | | 2 | |
| CALL | r | 2 | 3 | p237 p4 p6 | | 2 | |
| CALL | m | 3 | 4 | 2p237 p4 p6 | | 3 | |
| RET | | 1 | 2 | p237 p6 | | 1 | |
| RET | i | 3 | 4 | p23 2p6 p015 | | 2 | |
| BOUND | r,m | 15 | 15 | | | 8 | not 64 bit |
| INTO | | 4 | 4 | | | 5 | not 64 bit |
| String instructions | | | | | | | |
| LODSB/W | | 3 | 3 | 2p0156 p23 | | 1 | |
| LODSD/Q | | 2 | 2 | p0156 p23 | | 1 | |
| REP LODS | | 5n+12 | | | | ~2n | |
| STOS | | 3 | 3 | p23 p0156 p4 | | 1 | |
| REP STOS | | <2n | | | | ~0.5n | worst case |
| REP STOS | | 2.6/32B | | | | 1/32B | best case aligned by 32 |
| MOVS | | 5 | 5 | 2p23 p4 2p0156 | | 4 | |
| REP MOVS | | ~2n | | | | < 1n | worst case |
| REP MOVS | | 4/32B | | | | 1/32B | best case aligned by 32 |
| SCAS | | 3 | 3 | p23 2p0156 | | 1 | |
| REP SCAS | | ≥6n | | | | ≥2n | |
| CMPS | | 5 | 5 | 2p23 3p0156 | | 4 | |
| REP CMPS | | ≥8n | | | | ≥2n | |
| Synchronization instructions | | | | | | | |
| XADD | m,r | 4 | 5 | | | 6 | |

Broadwell

| | | | | | | | |
|------------------|-----|--------|--------|-------------|--------|------|--------|
| LOCK XADD | m,r | 9 | 9 | | | 21 | |
| LOCK ADD | m,r | 8 | 8 | | | 21 | |
| CMPXCHG | m,r | 5 | 6 | | | 7 | |
| LOCK CMPXCHG | m,r | 10 | 10 | | | 21 | |
| CMPXCHG8B | m,r | 15 | 15 | | | 8 | |
| LOCK CMPXCHG8B | m,r | 19 | 19 | | | 21 | |
| CMPXCHG16B | m,r | 22 | 22 | | | 15 | |
| LOCK CMPXCHG16B | m,r | 24 | 24 | | | 27 | |
| Other | | | | | | | |
| NOP (90) | | 1 | 0 | none | | 0.25 | |
| Long NOP (0F 1F) | | 1 | 0 | none | | 0.25 | |
| PAUSE | | 5 | 5 | p05 3p6 | | 9 | |
| ENTER | a,0 | 12 | 12 | | | 8 | |
| ENTER | a,b | ~14+7b | ~45+7b | | ~87+2b | | |
| LEAVE | | 3 | 3 | 2p0156 p23 | | 5 | |
| XGETBV | | 8 | 8 | | | 5 | XGETBV |
| RDTSC | | 15 | 15 | | | 24 | |
| RDTSCP | | 21 | 21 | | | 30 | RDTSCP |
| RDPMSR | | 34 | 34 | | | 37 | |
| RDRAND | r | 16 | 16 | p23 15p0156 | | ~230 | RDRAND |
| RDSEED | r | 16 | 16 | p23 15p0156 | | ~230 | RDSEED |

Floating point x87 instructions

| Instruction | Operands | μops fused domain | μops unfused domain | μops each port | Latency | Reciprocal throughput | Comments |
|--------------------------|----------|-------------------|---------------------|-----------------|---------|-----------------------|----------|
| Move instructions | | | | | | | |
| FLD | r | 1 | 1 | p01 | 1 | 0.5 | |
| FLD | m32/64 | 1 | 1 | p23 | 3 | 0.5 | |
| FLD | m80 | 4 | 4 | 2p01 2p23 | 4 | 2 | |
| FBLD | m80 | 43 | 43 | | 47 | 22 | |
| FST(P) | r | 1 | 1 | p01 | 1 | 0.5 | |
| FST(P) | m32/m64 | 1 | 2 | p4 p237 | 4 | 1 | |
| FSTP | m80 | 7 | 7 | 3p0156 2p23 2p4 | 5 | 5 | |
| FBSTP | m80 | 238 | 226 | | 269 | 267 | |
| FXCH | r | 2 | 0 | none | 0 | 0.5 | |
| FILD | m | 1 | 2 | p01 p23 | 6 | 1 | |
| FIST(P) | m | 3 | 3 | p1 p23 p4 | 7 | 1 | |
| FISTTP | m | 3 | 3 | p1 p23 p4 | 7 | 2 | SSE3 |
| FLDZ | | 1 | 1 | p01 | | 1 | |
| FLD1 | | 2 | 2 | 2p01 | | 2 | |
| FLDPI FLDL2E etc. | | 2 | 2 | 2p01 | | 2 | |
| FCMOVcc | r | 3 | 3 | 2p0 p5 | 2 | 2 | |
| FNSTSW | AX | 2 | 2 | p0 p0156 | 6 | 1 | |
| FNSTSW | m16 | 2 | 3 | p0 p4 p237 | 6 | 1 | |
| FLDCW | m16 | 3 | 3 | p01 p23 p6 | 7 | 2 | |
| FNSTCW | m16 | 2 | 3 | p237 p4 p6 | 6 | 1 | |
| FINCSTP FDECSTP | | 1 | 1 | p01 | 0 | 0.5 | |
| FFREE(P) | r | 1 | 1 | p01 | | 0.5 | |

Broadwell

| | | | | | | |
|--------------------------------|---|--------|-----|-----------|--------|-----|
| FNSAVE | m | 152 | 152 | | 173 | 173 |
| FRSTOR | m | 95 | 95 | | 175 | 175 |
| Arithmetic instructions | | | | | | |
| FADD(P) | | | | | | |
| FSUB(R)(P) | r | 1 | 1 | p1 | 3 | 1 |
| FADD(P) | | | | | | |
| FSUB(R)(P) | m | 1 | 2 | p1 p23 | | 1 |
| FMUL(P) | r | 1 | 1 | p0 | 5 | 1 |
| FMUL(P) | m | 1 | 2 | p0 p23 | | 1 |
| FDIV(R)(P) | r | 1 | 1 | p0 | 10-15 | 4-5 |
| FDIV(R)(P) | m | 1 | 2 | p0 p23 | | 4-5 |
| FABS | | 1 | 1 | p0 | 1 | 1 |
| FCHS | | 1 | 1 | p0 | 1 | 1 |
| FCOM(P) FUCOM | r | 1 | 1 | p1 | 3 | 1 |
| FCOM(P) FUCOM | m | 1 | 2 | p1 p23 | | 1 |
| FCOMPP FUCOMPP | | 2 | 2 | 2p01 | | 1 |
| FCOMI(P) | | | | | | |
| FUCOMI(P) | r | 3 | 3 | 3p01 | 7 | 1.5 |
| FIADD FISUB(R) | m | 2 | 3 | 2p1 p23 | | 2 |
| FIMUL | m | 2 | 3 | p0 p1 p23 | | 2 |
| FIDIV(R) | m | 2 | 3 | p0 p1 p23 | | |
| FICOM(P) | m | 2 | 3 | 2p1 p23 | | 2 |
| FTST | | 1 | 1 | p1 | 3 | 1 |
| FXAM | | 2 | 2 | 2p1 | 6 | 2 |
| FPREM | | 28 | 28 | | 20-24 | 13 |
| FPREM1 | | 28 | 28 | | 23-48 | 13 |
| FRNDINT | | 17 | 17 | | 11 | 23 |
| Math | | | | | | |
| FSCALE | | 27 | 27 | | 125 | 130 |
| FXTRACT | | 17 | 17 | | 12 | 11 |
| FSQRT | | 1 | 1 | p0 | 10-23 | 4-9 |
| FSIN | | 75-100 | | | 48-106 | |
| FCOS | | 70-100 | | | 49-112 | |
| FSINCOS | | 70-110 | | | 52-124 | |
| F2XM1 | | 16-86 | | | 63-68 | |
| FYL2X | | 55-96 | | | 92 | |
| FYL2XP1 | | 56 | | | 74 | |
| FPTAN | | 71-102 | | | 132 | |
| FPATAN | | 27-71 | | | 97-147 | |
| Other | | | | | | |
| FNOP | | 1 | 1 | p01 | | 0.5 |
| WAIT | | 2 | 2 | p01 | | 1 |
| FNCLEX | | 5 | 5 | p0156 | | 22 |
| FNINIT | | 26 | 26 | | | 84 |

Integer vector instructions

Broadwell

| Instruction | Operands | μops fused domain | μops unfused domain | μops each port | Latency | Reciprocal throughput | Comments |
|--------------------------|-------------|-------------------|---------------------|----------------|---------|-----------------------|---------------------|
| Move instructions | | | | | | | |
| MOVD | r32/64,mm/x | 1 | 1 | p0 | 1 | 1 | |
| MOVD | m32/64,mm/x | 1 | 2 | p237 p4 | 3 | 1 | |
| MOVD | mm/x,r32/64 | 1 | 1 | p5 | 1 | 1 | |
| MOVD | mm/x,m32/64 | 1 | 1 | p23 | 3 | 0.5 | |
| MOVQ | r64,mm/x | 1 | 1 | p0 | 1 | 1 | |
| MOVQ | mm/x,r64 | 1 | 1 | p5 | 1 | 1 | |
| MOVQ | mm/x,mm/x | 1 | | p015 | 1 | 0.33 | |
| MOVQ | mm/x,m64 | 1 | 1 | p23 | 3 | 0.5 | |
| MOVQ | m64, mm/x | 1 | 2 | p237 p4 | 3 | 1 | |
| MOVDQA/U | x,x | 1 | 1 | p015 | 0-1 | 0.25 | may be elim. |
| MOVDQA/U | x, m128 | 1 | 1 | p23 | 3 | 0.5 | |
| MOVDQA/U | m128, x | 1 | 2 | p237 p4 | 3 | 1 | |
| VMOVDQA/U | y,y | 1 | 1 | p015 | 0-1 | 0.25 | AVX may be elim. |
| VMOVDQA/U | y,m256 | 1 | 1 | p23 | 3 | 0.5 | AVX |
| VMOVDQA/U | m256,y | 1 | 2 | p237 p4 | 4 | 1 | AVX |
| LDDQU | x, m128 | 1 | 1 | p23 | 3 | 0.5 | SSE3 |
| MOVDQ2Q | mm, x | 2 | 2 | p01 p5 | 1 | 1 | |
| MOVQ2DQ | x,mm | 1 | 1 | p015 | 1 | 0.33 | |
| MOVNTQ | m64,mm | 1 | 2 | p237 p4 | ~400 | 1 | |
| MOVNTDQ | m128,x | 1 | 2 | p237 p4 | ~400 | 1 | |
| VMOVNTDQ | m256,y | 1 | 2 | p237 p4 | ~400 | 1 | AVX2 |
| MOVNTDQA | x, m128 | 1 | 1 | p23 | 3 | 0.5 | SSE4.1 |
| VMOVNTDQA | y,m256 | 1 | 1 | p23 | 3 | 0.5 | AVX2 |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | mm,mm | 3 | 3 | p5 | 2 | 2 | |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | mm,m64 | 3 | 3 | p23 2p5 | | 2 | |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | x,x / y,y,y | 1 | 1 | p5 | 1 | 1 | |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | x,m / y,y,m | 1 | 2 | p23 p5 | | 1 | |
| PACKUSDW | x,x / y,y,y | 1 | 1 | p5 | 1 | 1 | SSE4.1 |
| PACKUSDW | x,m / y,y,m | 1 | 2 | p23 p5 | | 1 | SSE4.1 |
| PUNPCKH/L | | | | | | | |
| BW/WD/DQ | v,v / v,v,v | 1 | 1 | p5 | 1 | 1 | |
| PUNPCKH/L | | | | | | | |
| BW/WD/DQ | v,m / v,v,m | 1 | 2 | p23 p5 | | 1 | |
| PUNPCKH/L | | | | | | | |
| QDQ | x,x / y,y,y | 1 | 1 | p5 | 1 | 1 | |
| PUNPCKH/L | | | | | | | |
| QDQ | x,m / y,y,m | 2 | 2 | p23 p5 | | 1 | |
| PMOVSX/ZX BW | | | | | | | |
| BD BQ DW DQ | x,x | 1 | 1 | p5 | 1 | 1 | SSE4.1 |
| PMOVSX/ZX BW | | | | | | | |
| BD BQ DW DQ | x,m | 1 | 2 | p23 p5 | | 1 | SSE4.1 |
| VPMOVSX/ZX BW | | | | | | | |
| BD BQ DW DQ | y,x | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPMOVSX/ZX BW | | | | | | | |
| BD BQ DW DQ | y,m | 2 | 2 | p5 p23 | | 1 | AVX2 |

Broadwell

| | | | | | | | |
|----------------|-----------------|----|----|----------------|--------|------|--------|
| PSHUFB | v,v / v,v,v | 1 | 1 | p5 | 1 | 1 | SSSE3 |
| PSHUFB | v,m / v,v,m | 2 | 2 | p23 p5 | | 1 | SSSE3 |
| PSHUFW | mm,mm,i | 1 | 1 | p5 | 1 | 1 | |
| PSHUFW | mm,m64,i | 2 | 2 | p23 p5 | | 1 | |
| PSHUFD | v,v,i | 1 | 1 | p5 | 1 | 1 | |
| PSHUFD | v,m,i | 2 | 2 | p23 p5 | | 1 | |
| PSHUFL/HW | v,v,i | 1 | 1 | p5 | 1 | 1 | |
| PSHUFL/HW | v,m,i | 2 | 2 | p23 p5 | | 1 | |
| PALIGNR | v,v,i / v,v,v,i | 1 | 1 | p5 | 1 | 1 | SSSE3 |
| PALIGNR | v,m,i / v,v,m,i | 2 | 2 | p23 p5 | | 1 | SSSE3 |
| PBLENDB | x,x,xmm0 | 2 | 2 | 2p5 | 2 | 2 | SSE4.1 |
| PBLENDB | x,m,xmm0 | 3 | 3 | 2p5 p23 | | 2 | SSE4.1 |
| VPBLENDB | v,v,v,v | 2 | 2 | 2p5 | 2 | 2 | AVX2 |
| VPBLENDB | v,v,m,v | 3 | 3 | 2p5 p23 | | 2 | AVX2 |
| PBLENDD | x,x,i / v,v,v,i | 1 | 1 | p5 | 1 | 1 | SSE4.1 |
| PBLENDD | x,m,i / v,v,m,i | 2 | 2 | p23 p5 | | 1 | SSE4.1 |
| VPBLENDD | v,v,v,i | 1 | 1 | p015 | 1 | 0.33 | AVX2 |
| VPBLENDD | v,v,m,i | 2 | 2 | p015 p23 | | 0.5 | AVX2 |
| VPERMD | y,y,y | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPERMD | y,y,m | 1 | 2 | p5 p23 | | 1 | AVX2 |
| VPERMQ | y,y,i | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPERMQ | y,m,i | 2 | 2 | p5 p23 | | 1 | AVX2 |
| VPERM2I128 | y,y,y,i | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPERM2I128 | y,y,m,i | 2 | 2 | p5 p23 | | 1 | AVX2 |
| MASKMOVQ | mm,mm | 4 | 4 | p0 p4 2p23 | 18-500 | 1 | |
| MASKMOVQ | x,x | 10 | 10 | 4p04 2p56 4p23 | 18-500 | 6 | |
| VPMASKMOVQ | v,v,m | 3 | 3 | p23 2p5 | 4 | 2 | AVX2 |
| VPMASKMOVQ | m,v,v | 4 | 4 | p0 p1 p4 p23 | 15 | 1 | AVX2 |
| PMOVB | r,v | 1 | 1 | p0 | 3 | 1 | |
| PEXTRB/W/D/Q | r32,x,i | 2 | 2 | p0 p5 | 2 | 1 | SSE4.1 |
| PEXTRB/W/D/Q | m8,x,i | 2 | 3 | p23 p4 p5 | | 1 | SSE4.1 |
| VEXTRACTI128 | x,y,i | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VEXTRACTI128 | m,y,i | 2 | 2 | p23 p4 | 4 | 1 | AVX2 |
| PINSRB | x,r32,i | 2 | 2 | p5 | 2 | 2 | SSE4.1 |
| PINSRB | x,m8,i | 2 | 2 | p23 p5 | | 1 | SSE4.1 |
| PINSRW | mm/x,r32,i | 2 | 2 | p5 | 2 | 2 | |
| PINSRW | mm/x,m16,i | 2 | 2 | p23 p5 | | 1 | |
| PINSRD/Q | x,r32,i | 2 | 2 | p5 | 2 | 2 | SSE4.1 |
| PINSRD/Q | x,m32,i | 2 | 2 | p23 p5 | | 1 | SSE4.1 |
| VINSERTI128 | y,y,x,i | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VINSERTI128 | y,y,m,i | 2 | 2 | p015 p23 | 4 | 0.5 | AVX2 |
| VPBROADCAST | | | | | | | |
| B/W/D/Q | x,x | 1 | 1 | p5 | 1 | 1 | AVX2 |
| VPBROADCAST | | | | | | | |
| B/W | x,m8/16 | 3 | 3 | p01 p23 p5 | 5 | 1 | AVX2 |
| VPBROADCAST | | | | | | | |
| D/Q | x,m32/64 | 1 | 1 | p23 | 4 | 0.5 | AVX2 |
| VPBROADCAST | | | | | | | |
| B/W/D/Q | y,x | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPBROADCAST | | | | | | | |
| B/W | y,m8/16 | 3 | 3 | p01 p23 p5 | 7 | 1 | AVX2 |
| VPBROADCAST | | | | | | | |
| D/Q | y,m32/64 | 1 | 1 | p23 | 5 | 0.5 | AVX2 |
| VBROADCASTI128 | y,m128 | 1 | 1 | p23 | 3 | 0.5 | AVX2 |
| VPGATHERDD | x,[r+s*x],x | 10 | 10 | | | 6 | AVX2 |

Broadwell

| | | | | | | | |
|--------------------------------|-------------|----|----|------------|----|-----|--------|
| VPGATHERDD | y,[r+s*y],y | 14 | 14 | | | 7 | AVX2 |
| VPGATHERQD | x,[r+s*x],x | 9 | 9 | | | 6 | AVX2 |
| VPGATHERQD | x,[r+s*y],x | 10 | 10 | | | 6 | AVX2 |
| VPGATHERDQ | x,[r+s*x],x | 7 | 7 | | | 5 | AVX2 |
| VPGATHERDQ | y,[r+s*x],y | 9 | 9 | | | 6 | AVX2 |
| VPGATHERQQ | x,[r+s*x],x | 7 | 7 | | | 5 | AVX2 |
| VPGATHERQQ | y,[r+s*y],y | 9 | 9 | | | 6 | AVX2 |
| Arithmetic instructions | | | | | | | |
| PADD/SUB(S,US) B/W/D/Q | v,v / v,v,v | 1 | 1 | p15 | 1 | 0.5 | |
| PADD/SUB(S,US) B/W/D/Q | v,m / v,v,m | 1 | 2 | p15 p23 | | 0.5 | |
| PHADD(S)W/D | | | | | | | |
| PHSUB(S)W/D | v,v / v,v,v | 3 | 3 | p1 2p5 | 3 | 2 | SSSE3 |
| PHADD(S)W/D | | | | | | | |
| PHSUB(S)W/D | v,m / v,v,m | 4 | 4 | p1 2p5 p23 | | 2 | SSSE3 |
| PCMPEQB/W/D | | | | | | | |
| PCMPGTB/W/D | v,v / v,v,v | 1 | 1 | p15 | 1 | 0.5 | |
| PCMPEQB/W/D | | | | | | | |
| PCMPGTB/W/D | v,m / v,v,m | 1 | 2 | p15 p23 | | 0.5 | |
| PCMPEQQ | v,v / v,v,v | 1 | 1 | p15 | 1 | 0.5 | SSE4.1 |
| PCMPEQQ | v,m / v,v,m | 1 | 2 | p15 p23 | | 0.5 | SSE4.1 |
| PCMPGTQ | v,v / v,v,v | 1 | 1 | p0 | 5 | 1 | SSE4.2 |
| PCMPGTQ | v,m / v,v,m | 1 | 2 | p0 p23 | | 1 | SSE4.2 |
| PMULL/HW | | | | | | | |
| PMULHUW | v,v / v,v,v | 1 | 1 | p0 | 5 | 1 | |
| PMULL/HW | | | | | | | |
| PMULHUW | v,m / v,v,m | 1 | 2 | p0 p23 | | 1 | |
| PMULHRSW | v,v / v,v,v | 1 | 1 | p0 | 5 | 1 | SSSE3 |
| PMULHRSW | v,m / v,v,m | 1 | 2 | p0 p23 | | 1 | SSSE3 |
| PMULLD | x,x / y,y,y | 2 | 2 | 2p0 | 10 | 2 | SSE4.1 |
| PMULLD | x,m / y,y,m | 3 | 3 | 2p0 p23 | | 2 | SSE4.1 |
| PMULDQ | x,x / y,y,y | 1 | 1 | p0 | 5 | 1 | SSE4.1 |
| PMULDQ | x,m / y,y,m | 1 | 2 | p0 p23 | | 1 | SSE4.1 |
| PMULUDQ | v,v / v,v,v | 1 | 1 | p0 | 5 | 1 | |
| PMULUDQ | v,m / v,v,m | 1 | 2 | p0 p23 | | 1 | |
| PMADDWD | v,v / v,v,v | 1 | 1 | p0 | 5 | 1 | |
| PMADDWD | v,m / v,v,m | 1 | 2 | p0 p23 | | 1 | |
| PMADDUBSW | v,v / v,v,v | 1 | 1 | p0 | 5 | 1 | SSSE3 |
| PMADDUBSW | v,m / v,v,m | 1 | 2 | p0 p23 | | 1 | SSSE3 |
| PAVGB/W | v,v / v,v,v | 1 | 1 | p15 | 1 | 0.5 | |
| PAVGB/W | v,m / v,v,m | 1 | 2 | p15 p23 | | 0.5 | |
| PMIN/PMAX SB/SW/SD | | | | | | | |
| UB/UW/UD | x,x / y,y,y | 1 | 1 | p15 | 1 | 0.5 | SSE4.1 |
| PMIN/PMAX SB/SW/SD | | | | | | | |
| UB/UW/UD | x,m / y,y,m | 1 | 2 | p15 p23 | | 0.5 | SSE4.1 |
| PHMINPOSUW | x,x | 1 | 1 | p0 | 5 | 1 | SSE4.1 |
| PHMINPOSUW | x,m128 | 1 | 2 | p0 p23 | | 1 | SSE4.1 |
| PABSB/W/D | v,v | 1 | 1 | p15 | 1 | 0.5 | SSSE3 |
| PABSB/W/D | v,m | 1 | 2 | p15 p23 | | 0.5 | SSSE3 |
| PSIGNB/W/D | v,v / v,v,v | 1 | 1 | p15 | 1 | 0.5 | SSSE3 |

Broadwell

| | | | | | | | |
|---|-----------------|---|---|------------------|----|------|--------|
| PSIGNB/W/D | v,m / v,v,m | 1 | 2 | p15 p23 | | 0.5 | SSSE3 |
| PSADBW | v,v / v,v,v | 1 | 1 | p0 | 5 | 1 | |
| PSADBW | v,m / v,v,m | 1 | 2 | p0 p23 | | 1 | |
| MPSADBW | x,x,i / v,v,v,i | 3 | 3 | p0 2p5 | 6 | 2 | SSE4.1 |
| MPSADBW | x,m,i / v,v,m,i | 4 | 4 | p0 2p5 p23 | | 2 | SSE4.1 |
| Logic instructions | | | | | | | |
| PAND PANDN | v,v / v,v,v | 1 | 1 | p015 | 1 | 0.33 | SSE4.1 |
| POR PXOR | | | | | | | |
| PAND PANDN | v,m / v,v,m | 1 | 2 | p015 p23 | 2 | 0.5 | |
| POR PXOR | | | | | | | |
| PTEST | v,v | 2 | 2 | p0 p5 | 1 | | |
| PTEST | v,m | 2 | 3 | p0 p5 p23 | 1 | | |
| PSLLW/D/Q | mm,mm | 1 | 1 | p0 | 1 | 1 | |
| PSRLW/D/Q | | | | | | | |
| PSRAW/D/Q | mm,m64 | 1 | 2 | p0 p23 | | 1 | |
| PSLLW/D/Q | | | | | | | |
| PSRLW/D/Q | x,x / v,v,x | 2 | 2 | p0 p5 | 2 | 1 | |
| PSRAW/D/Q | | | | | | | |
| PSLLW/D/Q | x,m / v,v,m | 2 | 2 | p0 p23 | | 1 | |
| PSRLW/D/Q | | | | | | | |
| PSRAW/D/Q | v,i / v,v,i | 1 | 1 | p0 | 1 | 1 | |
| PSLLW/D/Q | | | | | | | |
| PSRLW/D/Q | v,v,v | 3 | 3 | 2p0 p5 | 2 | 2 | |
| VPSLLVD/Q | | | | | | | |
| VPSRAVD | v,v,m | 4 | 4 | 2p0 p5 p23 | | 2 | |
| VPSRLVD/Q | | | | | | | |
| VPSLLVD/Q | x,i / v,v,i | 1 | 1 | p5 | 1 | 1 | |
| VPSRAVD | | | | | | | |
| VPSRLVD/Q | x,x,i | 8 | 8 | 6p05 2p16 | 4 | 4 | |
| PCMPESTRM | | | | | | | |
| PCMPESTRM | x,m128,i | 8 | 8 | 3p0 2p16 2p5 p23 | | 4 | |
| PCMPESTRM | x,x,i | 9 | 9 | 3p0 2p16 4p5 | 11 | 11 | |
| PCMPESTRM | x,m128,i | 9 | 9 | 6p05 2p16 p23 | | 5 | |
| PCMPISTRM | x,x,i | 3 | 3 | 3p0 | 3 | 3 | |
| PCMPISTRM | x,m128,i | 4 | 4 | 3p0 p23 | | 3 | |
| PCMPISTRM | x,x,i | 3 | 3 | 3p0 | 11 | 11 | |
| PCMPISTRM | x,m128,i | 4 | 4 | 3p0 p23 | | 3 | |
| Encryption instructions | | | | | | | |
| PCLMULQDQ | x,x,i | 1 | 1 | p0 | 5 | 1 | CLMUL |
| PCLMULQDQ | x,m,i | 2 | 2 | p0 p23 | | 1 | CLMUL |
| AESDEC, AESDECLAST, AESENC, AESENCLAST | x,x | 1 | 1 | p5 | 7 | 1 | AES |

Broadwell

| | | | | | | | |
|--|-------|----|----|-------------|----|-----|-----|
| AESDEC, AESDECLAST, AESENC, AESENCCLAST | x,m | 2 | 2 | p5 p23 | 14 | 1.5 | AES |
| AESIMC | x,x | 2 | 2 | 2p5 | | 2 | AES |
| AESIMC | x,m | 3 | 3 | 2p5 p23 | | 2 | AES |
| AESKEYGENAS SIST | x,x,i | 10 | 10 | 2p0 8p5 | 10 | 9 | AES |
| AESKEYGENAS SIST | x,m,i | 10 | 10 | 2p0 p23 7p5 | | 8 | AES |
| Other | | | | | | | |
| EMMS | | 31 | 31 | | | 12 | |

Floating point XMM and YMM instructions

| Instruction | Operands | μops fused domain | μops unfused domain | μops each port | Latency | Recipro- cal through put | Comments |
|--------------------------------|-----------------|-------------------------|---------------------------|----------------|---------|-----------------------------------|--------------|
| Move instruc- tions | | | | | | | |
| MOVAPS/D | x,x | 1 | 1 | p5 | 0-1 | 1 | may be elim. |
| VMOVAPS/D | y,y | 1 | 1 | p5 | 0-1 | 1 | may be elim. |
| MOVAPS/D | x,m128 | 1 | 1 | p23 | 3 | 0.5 | |
| VMOVAPS/D | y,m256 | 1 | 1 | p23 | 3 | 0.5 | AVX |
| MOVUPS/D | m128,x | 1 | 2 | p237 p4 | 3 | 1 | |
| VMOVUPS/D | m256,y | 1 | 2 | p237 p4 | 4 | 1 | AVX |
| MOVSS/D | x,x | 1 | 1 | p5 | 1 | 1 | |
| MOVSS/D | x,m32/64 | 1 | 1 | p23 | 3 | 0.5 | |
| MOVSS/D | m32/64,x | 1 | 2 | p237 p4 | 3 | 1 | |
| MOVHPS/D | x,m64 | 1 | 2 | p23 p5 | 4 | 1 | |
| MOVHPS/D | m64,x | 1 | 2 | p4 p237 | 3 | 1 | |
| MOVLPS/D | x,m64 | 1 | 2 | p23 p5 | 4 | 1 | |
| MOVLPS/D | m64,x | 1 | 2 | p4 p237 | 3 | 1 | |
| MOVHLPs | x,x | 1 | 1 | p5 | 1 | 1 | |
| MOVLHPs | x,x | 1 | 1 | p5 | 1 | 1 | |
| MOVMSKPS/D | r32,x | 1 | 1 | p0 | 3 | 1 | |
| VMOVMSKPS/D | r32,y | 1 | 1 | p0 | 3 | 1 | |
| MOVNTPS/D | m128,x | 1 | 2 | p4 p237 | ~400 | 1 | |
| VMOVNTPS/D | m256,y | 1 | 2 | p4 p237 | ~400 | 1 | AVX |
| SHUFPS/D | x,x,i / v,v,v,i | 1 | 1 | p5 | 1 | 1 | |
| SHUFPS/D | x,m,i / v,v,m,i | 2 | 2 | p5 p23 | | 1 | |
| VPERMILPS/PD | v,v,i | 1 | 1 | p5 | 1 | 1 | AVX |
| VPERMILPS/PD | v,m,i | 2 | 2 | p5 p23 | | 1 | AVX |
| VPERMILPS/PD | v,v,v | 1 | 1 | p5 | 1 | 1 | AVX |
| VPERMILPS/PD | v,v,m | 2 | 2 | p5 p23 | | 1 | AVX |
| VPERM2F128 | y,y,y,i | 1 | 1 | p5 | 3 | 1 | AVX |
| VPERM2F128 | y,y,m,i | 2 | 2 | p5 p23 | | 1 | AVX |
| VPERMPS | y,y,y | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPERMPS | y,y,m | 1 | 2 | p5 p23 | | 1 | AVX2 |

Broadwell

| | | | | | | | |
|-------------------|-----------------|----|----|--------------|----|------|--------|
| VPERMPD | y,y,i | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPERMPD | y,m,i | 2 | 2 | p5 p23 | | 1 | AVX2 |
| BLENDPS/PD | x,x,i / v,v,v,i | 1 | 1 | p015 | 1 | 0.33 | SSE4.1 |
| BLENDPS/PD | x,m,i / v,v,m,i | 2 | 2 | p015 p23 | | 0.5 | SSE4.1 |
| BLENDVPS/PD | x,x,xmm0 | 2 | 2 | 2p5 | 2 | 2 | SSE4.1 |
| BLENDVPS/PD | x,m,xmm0 | 3 | 3 | 2p5 p23 | | 2 | SSE4.1 |
| VBLENDVPS/PD | v,v,v,v | 2 | 2 | 2p5 | 2 | 2 | AVX |
| VBLENDVPS/PD | v,v,m,v | 3 | 3 | 2p5 p23 | | 2 | AVX |
| MOVDDUP | v,v | 1 | 1 | p5 | 1 | 1 | SSE3 |
| MOVDDUP | v,m | 1 | 1 | p23 | 3 | 0.5 | SSE3 |
| VBROADCASTSS | x,m32 | 1 | 1 | p23 | 4 | 0.5 | AVX |
| VBROADCASTSS | y,m32 | 1 | 1 | p23 | 5 | 0.5 | AVX |
| VBROADCASTSS | x,x | 1 | 1 | p5 | 1 | 1 | AVX2 |
| VBROADCASTSS | y,x | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VBROADCASTSD | y,m64 | 1 | 1 | p23 | 5 | 0.5 | AVX |
| VBROADCASTSD | y,x | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VBROADCASTF128 | y,m128 | 1 | 1 | p23 | 4 | 0.5 | AVX |
| MOVSH/LDUP | v,v | 1 | 1 | p5 | 1 | 1 | SSE3 |
| MOVSH/LDUP | v,m | 1 | 1 | p23 | 3 | 0.5 | SSE3 |
| UNPCKH/LPS/D | x,x / v,v,v | 1 | 1 | p5 | 1 | 1 | SSE3 |
| UNPCKH/LPS/D | x,m / v,v,m | 1 | 2 | p5 p23 | | 1 | SSE3 |
| EXTRACTPS | r32,x,i | 2 | 2 | p0 p5 | | 1 | SSE4.1 |
| EXTRACTPS | m32,x,i | 2 | 3 | p0 p5 p23 | 4 | 1 | SSE4.1 |
| VEEXTRACTF128 | x,y,i | 1 | 1 | p5 | 3 | 1 | AVX |
| VEEXTRACTF128 | m128,y,i | 2 | 2 | p23 p4 | 4 | 1 | AVX |
| INSERTPS | x,x,i | 1 | 1 | p5 | 1 | 1 | SSE4.1 |
| INSERTPS | x,m32,i | 2 | 2 | p23 p5 | 4 | 1 | SSE4.1 |
| VINSERTF128 | y,y,x,i | 1 | 1 | p5 | 3 | 1 | AVX |
| VINSERTF128 | y,y,m128,i | 2 | 2 | p015 p23 | 4 | 2 | AVX |
| VMASKMOVPS/D | v,v,m | 3 | 3 | 2p5 p23 | 4 | 2 | AVX |
| VMASKMOVPS/D | m128,x,x | 4 | 4 | p0 p1 p4 p23 | 15 | 1 | AVX |
| VMASKMOVPS/D | m256,y,y | 4 | 4 | p0 p1 p4 p23 | 16 | 1 | AVX |
| VGATHERDPS | x,[r+s*x],x | 10 | 10 | | | 6 | AVX2 |
| VGATHERDPS | y,[r+s*y],y | 14 | 14 | | | 7 | AVX2 |
| VGATHERQPS | x,[r+s*x],x | 9 | 9 | | | 6 | AVX2 |
| VGATHERQPS | x,[r+s*y],x | 10 | 10 | | | 6 | AVX2 |
| VGATHERDPD | x,[r+s*x],x | 7 | 7 | | | 5 | AVX2 |
| VGATHERDPD | y,[r+s*x],y | 9 | 9 | | | 6 | AVX2 |
| VGATHERQPD | x,[r+s*x],x | 7 | 7 | | | 5 | AVX2 |
| VGATHERQPD | y,[r+s*y],y | 9 | 9 | | | 6 | AVX2 |
| Conversion | | | | | | | |
| CVTPD2PS | x,x | 2 | 2 | p1 p5 | 4 | 1 | |
| CVTPD2PS | x,m128 | 2 | 3 | p1 p5 p23 | | 1 | |
| VCVTPD2PS | x,y | 2 | 2 | p1 p5 | 5 | 1 | AVX |
| VCVTPD2PS | x,m256 | 2 | 3 | p1 p5 p23 | | 1 | AVX |
| CVTSD2SS | x,x | 2 | 2 | p1 p5 | 4 | 1 | |
| CVTSD2SS | x,m64 | 2 | 3 | p1 p5 p23 | | 1 | |
| CVTPS2PD | x,x | 2 | 2 | p0 p5 | 2 | 1 | |
| CVTPS2PD | x,m64 | 2 | 2 | p0 p23 | | 1 | |
| VCVTPS2PD | y,x | 2 | 2 | p0 p5 | 5 | 1 | AVX |
| VCVTPS2PD | y,m128 | 2 | 2 | p0 p23 | | 1 | AVX |

Broadwell

| | | | | | | | |
|-------------------|-------------|---|---|------------|-----|-----|------|
| CVTSS2SD | x,x | 2 | 2 | p0 p5 | 2 | 1 | |
| CVTSS2SD | x,m32 | 2 | 2 | p0 p23 | | 1 | |
| CVTDQ2PS | x,x | 1 | 1 | p1 | 3 | 1 | |
| CVTDQ2PS | x,m128 | 1 | 2 | p1 p23 | | 1 | |
| VCVTDQ2PS | y,y | 1 | 1 | p1 | 3 | 1 | AVX |
| VCVTDQ2PS | y,m256 | 1 | 2 | p1 p23 | | 1 | AVX |
| CVT(T) PS2DQ | x,x | 1 | 1 | p1 | 3 | 1 | |
| CVT(T) PS2DQ | x,m128 | 1 | 2 | p1 p23 | | 1 | |
| VCVT(T) PS2DQ | y,y | 1 | 1 | p1 | 3 | 1 | AVX |
| VCVT(T) PS2DQ | y,m256 | 1 | 2 | p1 p23 | | 1 | AVX |
| CVTDQ2PD | x,x | 2 | 2 | p1 p5 | 4 | 1 | |
| CVTDQ2PD | x,m64 | 2 | 2 | p1 p23 | | 1 | |
| VCVTDQ2PD | y,x | 2 | 2 | p1 p5 | 6 | 1 | AVX |
| VCVTDQ2PD | y,m128 | 2 | 2 | p1 p23 | | 1 | AVX |
| CVT(T)PD2DQ | x,x | 2 | 2 | p1 p5 | 4 | 1 | |
| CVT(T)PD2DQ | x,m128 | 2 | 3 | p1 p5 p23 | | 1 | |
| VCVT(T)PD2DQ | x,y | 2 | 2 | p1 p5 | 6 | 1 | AVX |
| VCVT(T)PD2DQ | x,m256 | 2 | 3 | p1 p5 p23 | | 1 | AVX |
| CVTPI2PS | x,mm | 1 | 1 | p1 | 4 | 4 | |
| CVTPI2PS | x,m64 | 1 | 2 | p1 p23 | | 3 | |
| CVT(T)PS2PI | mm,x | 2 | 2 | p1 p5 | 4 | 1 | |
| CVT(T)PS2PI | mm,m128 | 2 | 2 | p1 p23 | | 1 | |
| CVTPI2PD | x,mm | 2 | 2 | p1 p5 | 4 | 1 | |
| CVTPI2PD | x,m64 | 2 | 2 | p1 p23 | | 1 | |
| CVT(T) PD2PI | mm,x | 2 | 2 | p1 p5 | 4 | 1 | |
| CVT(T) PD2PI | mm,m128 | 2 | 3 | p1 p5 p23 | | 1 | |
| CVTSI2SS | x,r32 | 2 | 2 | p1 p5 | 4 | 3 | |
| CVTSI2SS | x,r64 | 3 | 3 | p1 2p5 | 5 | 4 | |
| CVTSI2SS | x,m32 | 1 | 2 | p1 p23 | | 3 | |
| CVT(T)SS2SI | r32,x | 2 | 2 | p0 p1 | 4 | 1 | |
| CVT(T)SS2SI | r32,m32 | 2 | 3 | p0 p1 p23 | | 1 | |
| CVTSI2SD | x,r32/64 | 2 | 2 | p1 p5 | 4 | 3 | |
| CVTSI2SD | x,m32 | 2 | 2 | p1 p23 | | 3 | |
| CVT(T)SD2SI | r32/64,x | 2 | 2 | p0 p1 | 4 | 1 | |
| CVT(T)SD2SI | r32,m64 | 2 | 3 | p0 p1 p23 | | 1 | |
| VCVTPS2PH | x,v,i | 2 | 2 | p1 p5 | 4-6 | 1 | F16C |
| VCVTPS2PH | m,v,i | 3 | 3 | p1 p4 p23 | | 1 | F16C |
| VCVTPH2PS | v,x | 2 | 2 | p1 p5 | 4-6 | 1 | F16C |
| VCVTPH2PS | v,m | 2 | 2 | p1 p23 | | 1 | F16C |
| Arithmetic | | | | | | | |
| ADDSS/D PS/D | | | | | | | |
| SUBSS/D PS/D | x,x / v,v,v | 1 | 1 | p1 | 3 | 1 | |
| ADDSS/D PS/D | | | | | | | |
| SUBSS/D PS/D | x,m / v,v,m | 1 | 2 | p1 p23 | | 1 | |
| ADDSUBPS/D | x,x / v,v,v | 1 | 1 | p1 | 3 | 1 | SSE3 |
| ADDSUBPS/D | x,m / v,v,m | 1 | 2 | p1 p23 | | 1 | SSE3 |
| HADDPS/D | | | | | | | |
| HSUBPS/D | x,x / v,v,v | 3 | 3 | p1 2p5 | 5 | 2 | SSE3 |
| HADDPS/D | | | | | | | |
| HSUBPS/D | x,m / v,v,m | 4 | 4 | p1 2p5 p23 | | 2 | SSE3 |
| MULSS/D PS/D | x,x / v,v,v | 1 | 1 | p01 | 3 | 0.5 | |
| MULSS/D PS/D | x,m / v,v,m | 1 | 2 | p01 p23 | | 0.5 | |
| DIVSS | x,x | 1 | 1 | p0 | 11 | 2.5 | |

Broadwell

| | | | | | | | |
|--------------------------|-----------------|---|---|------------------|-------|-------|--------|
| DIVPS | x,x | 1 | 1 | p0 | 11 | 5 | |
| DIVSS DIVPS | x,m | 1 | 2 | p0 p23 | | 3-5 | |
| DIVSD | x,x | 1 | 1 | p0 | 10-14 | 4-5 | |
| DIVPD | x,x | 1 | 1 | p0 | 10-14 | 8 | |
| DIVSD DIVPD | x,m | 1 | 2 | p0 p23 | | 4-5 | |
| VDIVPS | y,y,y | 3 | 3 | 2p0 p15 | 17 | 10 | AVX |
| VDIVPS | y,y,m256 | 4 | 4 | 2p0 p15 p23 | | 10 | AVX |
| VDIVPD | y,y,y | 3 | 3 | 2p0 p15 | 19-23 | 16 | AVX |
| VDIVPD | y,y,m256 | 4 | 4 | 2p0 p15 p23 | | 16 | AVX |
| RCPSS/PS | x,x | 1 | 1 | p0 | 5 | 1 | |
| RCPSS/PS | x,m128 | 1 | 2 | p0 p23 | | 1 | |
| VRCPSS | y,y | 3 | 3 | 2p0 p15 | 7 | 2 | AVX |
| VRCPSS | y,m256 | 4 | 4 | 2p0 p15 p23 | | 2 | AVX |
| CMPccSS/D | | | | | | | |
| CMPccPS/D | x,x / v,v,v | 1 | 1 | p1 | 3 | 1 | |
| CMPccSS/D | | | | | | | |
| CMPccPS/D | x,m / v,v,m | 2 | 2 | p1 p23 | | 1 | |
| (U)COMISS/D | x,x | 1 | 1 | p1 | | 1 | |
| (U)COMISS/D | x,m32/64 | 2 | 2 | p1 p23 | | 1 | |
| MAXSS/D PS/D | | | | | | | |
| MINSS/D PS/D | x,x / v,v,v | 1 | 1 | p1 | 3 | 1 | |
| MAXSS/D PS/D | | | | | | | |
| MINSS/D PS/D | x,m / v,v,m | 1 | 2 | p1 p23 | | 1 | |
| ROUNDSS/D PS/D | v,v,i | 2 | 2 | 2p1 | 6 | 2 | SSE4.1 |
| ROUNDSS/D PS/D | v,m,i | 3 | 3 | 2p1 p23 | | 2 | SSE4.1 |
| DPPS | x,x,i / v,v,v,i | 4 | 4 | 2p0 p1 p5 | 12 | 2 | SSE4.1 |
| DPPS | x,m,i / v,v,m,i | 6 | 6 | 2p0 p1 p5 p23 p6 | | 4 | SSE4.1 |
| DPPD | x,x,i | 3 | 3 | p0 p1 p5 | 7 | 1 | SSE4.1 |
| DPPD | x,m128,i | 4 | 4 | p0 p1 p5 p23 | | 1 | SSE4.1 |
| VFMADD... | | | | | | | |
| (all FMA instr.) | v,v,v | 1 | 1 | p01 | 5 | 0.5 | FMA |
| VFMADD... | | | | | | | |
| (all FMA instr.) | v,v,m | 1 | 2 | p01 p23 | | 0.5 | FMA |
| Math | | | | | | | |
| SQRTSS | x,x | 1 | 1 | p0 | 11 | 4 | |
| SQRTPS | x,x | 1 | 1 | p0 | 11 | 7 | |
| SQRTSS/PS | x,m128 | 1 | 2 | p0 p23 | | 4-7 | |
| VSQRTPS | y,y | 3 | 3 | 2p0 p15 | 19 | 14 | AVX |
| VSQRTPS | y,m256 | 4 | 4 | 2p0 p15 p23 | | 14 | AVX |
| SQRTSD | x,x | 1 | 1 | p0 | 15-16 | 4-8 | |
| SQRTPD | x,x | 1 | 1 | p0 | 15-16 | 8-14 | |
| SQRTSD/PD | x,m128 | 1 | 2 | p0 p23 | | 4-14 | |
| VSQRTPD | y,y | 3 | 3 | 2p0 p15 | 27-29 | 16-28 | AVX |
| VSQRTPD | y,m256 | 4 | 4 | 2p0 p15 p23 | | 16-28 | AVX |
| RSQRTSS/PS | x,x | 1 | 1 | p0 | 5 | 1 | |
| RSQRTSS/PS | x,m128 | 1 | 2 | p0 p23 | | 1 | |
| VRSQRTPS | y,y | 3 | 3 | 2p0 p15 | 7 | 2 | AVX |
| VRSQRTPS | y,m256 | 4 | 4 | 2p0 p15 p23 | | 2 | AVX |
| Logic | | | | | | | |
| AND/ANDN/OR/ XORPS/PD | x,x / v,v,v | 1 | 1 | p5 | 1 | 1 | |

Broadwell

| | | | | | | | |
|--------------------------|-------------|-----|----|---------------|-----|-----|----------------|
| AND/ANDN/OR/ XORPS/PD | x,m / v,v,m | 1 | 2 | p5 p23 | | 1 | |
| Other | | | | | | | |
| VZEROUPPER | | 4 | 4 | none | | 1 | AVX |
| VZEROALL | | 12 | 12 | none | | 10 | AVX, 32 bit |
| VZEROALL | | 20 | 20 | none | | 8 | AVX, 64 bit |
| LDMXCSR | m32 | 3 | 3 | p0 p6 p23 | 6 | 3 | |
| STMXCSR | m32 | 3 | 4 | p0 p4 p6 p237 | 7 | 1 | |
| FXSAVE | m4096 | 111 | | | 66 | 66 | 32 bit mode |
| FXSAVE | m4096 | 141 | | | 66 | 66 | 64 bit mode |
| FXRSTOR | m4096 | 107 | | | 80 | 80 | 32 bit mode |
| FXRSTOR | m4096 | 115 | | | 80 | 80 | 64 bit mode |
| XSAVE | | 174 | | | 70 | 70 | 32 bit mode |
| XSAVE | | 224 | | | 84 | 84 | 64 bit mode |
| XRSTOR | | 172 | | | 111 | 111 | 32 bit mode |
| XRSTOR | | 173 | | | 112 | 112 | 64 bit mode |
| XSAVEOPT | m | 114 | | | 51 | 51 | |

Intel Skylake

List of instruction timings and μ op breakdown

Explanation of column headings:

| | |
|--|---|
| Instruction: | Name of instruction. Multiple names mean that these instructions have the same data. Instructions with or without V name prefix behave the same unless otherwise noted. |
| Operands: | i = immediate data, r = register, mm = 64 bit mmx register, x = 128 bit xmm register, mm/x = mmx or xmm register, y = 256 bit ymm register, v = any vector register (mmx, xmm, ymm). m = memory operand, m32 = 32-bit memory operand, etc. |
| μops fused domain: | The number of μ ops at the decode, rename and allocate stages in the pipeline. Fused μ ops count as one. |
| μops unfused domain: | The total number of μ ops for all execution port. Fused μ ops count as two. Fused macro-ops count as one. The instruction has μ op fusion if this number is higher than the number under fused domain. Some operations are not counted here if they do not go to any execution port or if the counters are inaccurate. |
| μops each port: | The number of μ ops for each execution port. p0 means a μ op to execution port 0. p01 means a μ op that can go to either port 0 or port 1. p0 p1 means two μ ops going to port 0 and 1, respectively. Port 0: Integer, f.p. and vector ALU, mul, div, branch Port 1: Integer, f.p. and vector ALU Port 2: Load Port 3: Load Port 4: Store Port 5: Integer and vector ALU Port 6: Integer ALU, branch Port 7: Store address |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Where hyperthreading is enabled, the use of the same execution units in the other thread leads to inferior performance. Denormal numbers, NAN's and infinity do not increase the latency. The time unit used is core clock cycles, not the reference clock cycles given by the time stamp counter. |
| Reciprocal throughput: | The average number of core clock cycles per instruction for a series of independent instructions of the same kind in the same thread. |

Integer instructions

| Instruction | Operands | μ ops fused domain | μ ops unfused domain | μ ops each port | Latency | Reciprocal throughput | Comments |
|--------------------------|---------------|------------------------|--------------------------|---------------------|---------|-----------------------|----------------------|
| Move instructions | | | | | | | |
| MOV | r,i | 1 | 1 | p0156 | | 0.25 | |
| MOV | r8/16,r8/16 | 1 | 1 | p0156 | 1 | 0.25 | |
| MOV | r32/64,r32/64 | 1 | 1 | p0156 | 0-1 | 0.25 | may be elim. |
| MOV | r8l,m | 1 | 2 | p23 p0156 | | 0.5 | |
| MOV | r8h,m | 1 | 1 | p23 | | 0.5 | |
| MOV | r16,m | 1 | 2 | p23 p0156 | | 0.5 | |
| MOV | r32/64,m | 1 | 1 | p23 | 2 | 0.5 | all addressing modes |
| MOV | m,r | 1 | 2 | p237 p4 | 2 | 1 | |
| MOV | m,i | 1 | 2 | p237 p4 | | 1 | |
| MOVNTI | m,r | 2 | 2 | p23 p4 | ~400 | 1 | |

Skylake

| | | | | | | | |
|--------------------------------|---------------|----|----|-----------------|-----|-------|------------------------------|
| MOVSBX MOVZXB | r,r | 1 | 1 | p0156 | 1 | 0.25 | |
| MOVSBXD | | | | | | | |
| MOVSBX MOVZXB | r16,m8 | 1 | 2 | p23 p0156 | | 0.5 | |
| MOVSBX MOVZXB | r,m | 1 | 1 | p23 | | 0.5 | all other combinations |
| MOVSBXD | | | | | | | |
| CMOVBcc | r,r | 1 | 1 | p06 | 1 | 0.5 | |
| CMOVBcc | r,m | 2 | 2 | p06 p23 | | 0.5 | |
| XCHGB | r,r | 3 | 3 | 3p0156 | 2 | 1 | |
| XCHGB | r,m | 8 | 8 | | 23 | | implicit lock |
| XLATB | | 3 | 3 | p23 2p0156 | 7 | 2 | |
| PUSHB | r | 1 | 2 | p237 p4 | 3 | 1 | |
| PUSHB | i | 1 | 2 | p237 p4 | | 1 | |
| PUSHB | m | 2 | 3 | p4 2p237 | | 1 | |
| PUSHB | stack pointer | 2 | 3 | p0156 p237 p4 | | 1 | |
| PUSHFB(D/Q) | | 3 | 4 | p1 p4 p237 p06 | | 1 | |
| PUSHAB(D) | | 11 | 19 | | | 8 | not 64 bit |
| POPB | r | 1 | 1 | p23 | 2 | 0.5 | |
| POPB | stack pointer | 3 | 3 | p23 2p0156 | | 3 | |
| POPB | m | 2 | 3 | 2p237 p4 | | 1 | |
| POPF(D/Q) | | 9 | 9 | | | 20 | |
| POPAB(D) | | 18 | 18 | | | 8 | not 64 bit |
| LAHFB SAHFB | | 1 | 1 | p06 | 1 | 1 | |
| SALCB | | 3 | 3 | 3p0156 | 1 | 1 | not 64 bit |
| LEAB | r16,m | 2 | 2 | p1 p05 | 2-4 | 1 | 16 or 32 bit address size |
| LEAB | r32/64,m | 1 | 1 | p15 | 1 | 0.5 | 1 or 2 components in address |
| LEAB | r32/64,m | 1 | 1 | p1 | 3 | 1 | 3 components in address |
| LEAB | r32/64,m | 1 | 1 | p1 | | 1 | rip relative address |
| BSWAPB | r32 | 1 | 1 | p15 | 1 | 0.5 | |
| BSWAPB | r64 | 2 | 2 | p06 p15 | 2 | 1 | |
| MOVBBE | r16,m16 | 3 | 3 | 2p0156 p23 | | 0.5-1 | MOVBBE |
| MOVBBE | r32,m32 | 2 | 2 | p15 p23 | | 0.5 | MOVBBE |
| MOVBBE | r64,m64 | 3 | 3 | 2p0156 p23 | | 0.75 | MOVBBE |
| MOVBBE | m16,r16 | 2 | 3 | p06 p237 p4 | | 1 | MOVBBE |
| MOVBBE | m32,r32 | 2 | 3 | p15 p237 p4 | | 1 | MOVBBE |
| MOVBBE | m64,r64 | 3 | 4 | p06 p15 p237 p4 | | 1 | MOVBBE |
| PREFETCHNTA/0/1/2 | m | 1 | 1 | p23 | | 0.5 | |
| PREFETCHWB | m | 1 | 1 | p23 | | 1 | PREFETCHWB |
| LFENCE | | 2 | | none counted | | 4 | |
| MFENCE | | 4 | 4 | p23 p4 | | 33 | |
| SFENCE | | 2 | 2 | p23 p4 | | 6 | |
| Arithmetic instructions | | | | | | | |
| ADD SUB | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | |
| ADD SUB | r,m | 1 | 2 | p0156 p23 | | 0.5 | |
| ADD SUB | m,r/i | 2 | 4 | 2p0156 2p237 p4 | 5 | 1 | |
| ADC SBB | r,r/i | 1 | 1 | p06 | 1 | 1 | |

Skylake

| | | | | | | | |
|-------------|-------------|----|----|-----------------|-------|-------|------------|
| ADC SBB | r,m | 2 | 2 | p06 p23 | | 1 | |
| ADC SBB | m,r/i | 4 | 6 | 3p0156 2p237 p4 | 5 | 2 | |
| CMP | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | |
| CMP | m,r/i | 1 | 2 | p0156 p23 | 1 | 0.5 | |
| INC DEC NEG | r | 1 | 1 | p0156 | 1 | 0.25 | |
| NOT | | | | | | | |
| INC DEC NOT | m | 3 | 4 | p0156 2p237 p4 | 5-6 | 1 | |
| NEG | m | 2 | 4 | p0156 2p237 p4 | 5-6 | 1 | |
| AAA | | 2 | 2 | p1 p56 | 4 | | not 64 bit |
| AAS | | 2 | 2 | p1 p056 | 4 | | not 64 bit |
| DAA DAS | | 3 | 3 | p1 2p056 | 4 | | not 64 bit |
| AAD | | 3 | 3 | p1 2p056 | 4 | | not 64 bit |
| AAM | | 11 | 11 | p0 p1 p5 p6 | 23 | 7 | not 64 bit |
| MUL IMUL | r8 | 1 | 1 | p1 | 3 | 1 | |
| MUL IMUL | r16 | 4 | 4 | p1 p0156 | 4 | 2 | |
| MUL IMUL | r32 | 3 | 3 | p1 p0156 | 4 | 1 | |
| MUL IMUL | r64 | 2 | 2 | p1 p6 | 3 | 1 | |
| MUL IMUL | m8 | 1 | 2 | p1 p23 | | 1 | |
| MUL IMUL | m16 | 4 | 5 | p1 3p0156 p23 | | 2 | |
| MUL IMUL | m32 | 3 | 4 | p1 2p0156 p23 | | 2 | |
| MUL IMUL | m64 | 2 | 3 | p1 p6 p23 | | 1 | |
| IMUL | r,r | 1 | 1 | p1 | 3 | 1 | |
| IMUL | r,m | 1 | 2 | p1 p23 | | 1 | |
| IMUL | r16,r16,i | 2 | 2 | p1 p0156 | 4 | 1 | |
| IMUL | r32,r32,i | 1 | 1 | p1 | 3 | 1 | |
| IMUL | r64,r64,i | 1 | 1 | p1 | 3 | 1 | |
| IMUL | r16,m16,i | 2 | 3 | p1 p0156 p23 | | 1 | |
| IMUL | r32,m32,i | 1 | 2 | p1 p23 | | 1 | |
| IMUL | r64,m64,i | 1 | 2 | p1 p23 | | 1 | |
| MULX | r32,r32,r32 | 3 | 3 | p1 2p056 | 4 | 1 | BMI2 |
| MULX | r32,r32,m32 | 3 | 4 | p1 2p056 p23 | | 1 | BMI2 |
| MULX | r64,r64,r64 | 2 | 2 | p1 p5 | 4 | 1 | BMI2 |
| MULX | r64,r64,m64 | 2 | 3 | p1 p6 p23 | | 1 | BMI2 |
| DIV | r8 | 10 | 10 | p0 p1 p5 p6 | 23 | 6 | |
| DIV | r16 | 10 | 10 | p0 p1 p5 p6 | 23 | 6 | |
| DIV | r32 | 10 | 10 | p0 p1 p5 p6 | 26 | 6 | |
| DIV | r64 | 36 | 36 | p0 p1 p5 p6 | 35-88 | 21-83 | |
| IDIV | r8 | 11 | 11 | p0 p1 p5 p6 | 24 | 6 | |
| IDIV | r16 | 10 | 10 | p0 p1 p5 p6 | 23 | 6 | |
| IDIV | r32 | 10 | 10 | p0 p1 p5 p6 | 26 | 6 | |
| IDIV | r64 | 57 | 57 | p0 p1 p5 p6 | 42-95 | 24-90 | |
| CBW | | 1 | 1 | p0156 | 1 | | |
| CWDE | | 1 | 1 | p0156 | 1 | | |
| CDQE | | 1 | 1 | p0156 | 1 | | |
| CWD | | 2 | 2 | p0156 | 1 | | |
| CDQ | | 1 | 1 | p06 | 1 | | |
| CQO | | 1 | 1 | p06 | 1 | | |
| POPCNT | r,r | 1 | 1 | p1 | 3 | 1 | SSE4.2 |
| POPCNT | r,m | 1 | 2 | p1 p23 | | 1 | SSE4.2 |
| CRC32 | r,r | 1 | 1 | p1 | 3 | 1 | SSE4.2 |
| CRC32 | r,m | 1 | 2 | p1 p23 | | 1 | SSE4.2 |

Skylake

| Logic instructions | | | | | | | |
|--------------------|--------|----|----|-----------------|---|------|------------|
| AND OR XOR | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | |
| AND OR XOR | r,m | 1 | 2 | p0156 p23 | | 0.5 | |
| AND OR XOR | m,r/i | 2 | 4 | 2p0156 2p237 p4 | 5 | 1 | |
| TEST | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | |
| TEST | m,r/i | 1 | 2 | p0156 p23 | 1 | 0.5 | |
| SHR SHL SAR | r,i | 1 | 1 | p06 | 1 | 0.5 | |
| SHR SHL SAR | m,i | 3 | 4 | 2p06 p237 p4 | | 2 | |
| SHR SHL SAR | r,cl | 3 | 3 | 3p06 | 2 | 2 | |
| SHR SHL SAR | m,cl | 5 | 6 | 3p06 2p23 p4 | | 4 | |
| ROR ROL | r,1 | 2 | 2 | 2p06 | 1 | 1 | short form |
| ROR ROL | r,i | 1 | 1 | p06 | 1 | 0.5 | |
| ROR ROL | m,i | 4 | 5 | 2p06 2p237 p4 | | 2 | |
| ROR ROL | r,cl | 3 | 3 | 3p06 | 2 | 2 | |
| ROR ROL | m,cl | 5 | 6 | 3p06 p23 p4 | | 4 | |
| RCR RCL | r,1 | 3 | 3 | 2p06 p0156 | 2 | 2 | |
| RCR RCL | m,1 | 4 | 6 | | | 3 | |
| RCR RCL | r,i | 8 | 8 | p0156 | 6 | 6 | |
| RCR RCL | m,i | 11 | 11 | | | 6 | |
| RCR RCL | r,cl | 8 | 8 | p0156 | 6 | 6 | |
| RCR RCL | m,cl | 11 | 11 | | | 6 | |
| SHRD SHLD | r,r,i | 1 | 1 | p1 | 3 | 1 | |
| SHRD SHLD | m,r,i | 3 | 5 | | | 2 | |
| SHLD | r,r,cl | 4 | 4 | p0156 | 3 | 2 | |
| SHRD | r,r,cl | 4 | 4 | p0156 | 4 | 2 | |
| SHRD SHLD | m,r,cl | 5 | 7 | | | 4 | |
| SHLX SHRX SARX | r,r,r | 1 | 1 | p06 | 1 | 0.5 | BMI2 |
| SHLX SHRX SARX | r,m,r | 2 | 2 | p06 p23 | | 0.5 | BMI2 |
| RORX | r,r,i | 1 | 1 | p06 | 1 | 0.5 | BMI2 |
| RORX | r,m,i | 2 | 2 | p06 p23 | | 0.5 | BMI2 |
| BT | r,r/i | 1 | 1 | p06 | 1 | 0.5 | |
| BT | m,r | 10 | 10 | | | 5 | |
| BT | m,i | 2 | 2 | p06 p23 | | 0.5 | |
| BTR BTS BTC | r,r/i | 1 | 1 | p06 | 1 | 0.5 | |
| BTR BTS BTC | m,r | 10 | 11 | | | 5 | |
| BTR BTS BTC | m,i | 3 | 4 | p06 p4 p23 | | 1 | |
| BSF BSR | r,r | 1 | 1 | p1 | 3 | 1 | |
| BSF BSR | r,m | 1 | 2 | p1 p23 | | 1 | |
| SETcc | r | 1 | 1 | p06 | 1 | 0.5 | |
| SETcc | m | 2 | 3 | p06 p237 p4 | | 1 | |
| CLC | | 1 | 0 | none | | 0.25 | |
| STC | | 1 | 1 | p0156 | | 0.25 | |
| CMC | | 1 | 1 | p0156 | 1 | 1 | |
| CLD STD | | 3 | 3 | p15 p6 | | 4 | |
| LZCNT | r,r | 1 | 1 | p1 | 3 | 1 | LZCNT |
| LZCNT | r,m | 1 | 2 | p1 p23 | | 1 | LZCNT |
| TZCNT | r,r | 1 | 1 | p1 | 3 | 1 | BMI1 |
| TZCNT | r,m | 1 | 2 | p1 p23 | | 1 | BMI1 |
| ANDN | r,r,r | 1 | 1 | p15 | 1 | 0.5 | BMI1 |
| ANDN | r,r,m | 1 | 2 | p15 p23 | 1 | 0.5 | BMI1 |
| BLSI BLSMSK | r,r | 1 | 1 | p15 | 1 | 0.5 | BMI1 |
| BLSR | | | | | | | |

Skylake

| | | | | | | | |
|--------------------------------------|------------|---------|----|----------------|---|-------|-------------------------|
| BLSI BLSMSK | r,m | 1 | 2 | p15 p23 | | 0.5 | BMI1 |
| BLSR | | | | | | | |
| BEXTR | r,r,r | 2 | 2 | 2p0156 | 2 | 0.5 | BMI1 |
| BEXTR | r,m,r | 3 | 3 | 2p0156 p23 | | 1 | BMI1 |
| BZHI | r,r,r | 1 | 1 | p15 | 1 | 0.5 | BMI2 |
| BZHI | r,m,r | 1 | 2 | p15 p23 | | 0.5 | BMI2 |
| PDEP | r,r,r | 1 | 1 | p1 | 3 | 1 | BMI2 |
| PDEP | r,r,m | 1 | 2 | p1 p23 | | 1 | BMI2 |
| PEXT | r,r,r | 1 | 1 | p1 | 3 | 1 | BMI2 |
| PEXT | r,r,m | 1 | 2 | p1 p23 | | 1 | BMI2 |
| Control transfer instructions | | | | | | | |
| JMP | short/near | 1 | 1 | p6 | | 1-2 | |
| JMP | r | 1 | 1 | p6 | | 2 | |
| JMP | m | 1 | 2 | p23 p6 | | 2 | |
| Conditional jump | short/near | 1 | 1 | p6 | | 1-2 | predicted taken |
| Conditional jump | short/near | 1 | 1 | p06 | | 0.5-1 | predicted not taken |
| Fused arithmetic and branch | | 1 | 1 | p6 | | 1-2 | predicted taken |
| Fused arithmetic and branch | | 1 | 1 | p06 | | 0.5-1 | predicted not taken |
| J(E/R)CXZ | short | 2 | 2 | p0156 p6 | | 0.5-2 | |
| LOOP | short | 7 | 7 | | | 5 | |
| LOOP(N)E | short | 11 | 11 | | | 6 | |
| CALL | near | 2 | 3 | p237 p4 p6 | | 3 | |
| CALL | r | 2 | 3 | p237 p4 p6 | | 2 | |
| CALL | m | 3 | 4 | 2p237 p4 p6 | | 3 | |
| RET | | 1 | 2 | p237 p6 | | 1 | |
| RET | i | | 2 | | | 2 | |
| BOUND | r,m | 15 | 15 | | | 8 | not 64 bit |
| INTO | | 5 | 5 | | | 6 | not 64 bit |
| String instructions | | | | | | | |
| LODSB/W | | 3 | 3 | 2p0156 p23 | | 1 | |
| LODSD/Q | | 2 | 2 | p0156 p23 | | 1 | |
| REP LODS | | 5n+12 | | | | ~2n | |
| STOS | | 3 | 3 | p23 p0156 p4 | | 1 | |
| REP STOS | | <2n | | | | ~0.5n | worst case |
| REP STOS | | 2.6/32B | | | | 1/32B | best case aligned by 32 |
| MOVS | | 5 | 5 | 2p23 p4 2p0156 | | 4 | |
| REP MOVS | | ~2n | | | | < 1n | worst case |
| REP MOVS | | 4/32B | | | | 1/32B | best case aligned by 32 |
| SCAS | | 3 | 3 | p23 2p0156 | | 1 | |
| REP SCAS | | ≥6n | | | | ≥2n | |
| CMPS | | 5 | 5 | 2p23 3p0156 | | 4 | |
| REP CMPS | | ≥8n | | | | ≥2n | |
| Synchronization instructions | | | | | | | |
| XADD | m,r | 4 | 5 | | | 5 | |

Skylake

| | | | | | | | |
|------------------|-----|--------|--------|-------------|--------|---------|--------|
| LOCK XADD | m,r | 9 | 9 | | | 18 | |
| LOCK ADD | m,r | 8 | 8 | | | 18 | |
| CMPXCHG | m,r | 5 | 6 | | | 6 | |
| LOCK CMPXCHG | m,r | 10 | 10 | | | 18 | |
| CMPXCHG8B | m,r | 16 | 16 | | | 11 | |
| LOCK CMPXCHG8B | m,r | 20 | 20 | | | 19 | |
| CMPXCHG16B | m,r | 23 | 23 | | | 16 | |
| LOCK CMPXCHG16B | m,r | 25 | 25 | | | 26 | |
| Other | | | | | | | |
| NOP (90) | | 1 | 0 | none | | 0.25 | |
| Long NOP (0F 1F) | | 1 | 0 | none | | 0.25 | |
| PAUSE | | 4 | 4 | p6 | | | |
| ENTER | a,0 | 12 | 12 | | | 8 | |
| ENTER | a,b | ~14+7b | ~45+7b | | ~87+2b | | |
| LEAVE | | 3 | 3 | 2p0156 p23 | | 5 | |
| XGETBV | | 15 | 15 | | | 9 | XGETBV |
| CPUID | | 27-118 | | | | 100-250 | |
| RDTSC | | 20 | 20 | | | 25 | |
| RDTSCP | | 22 | 22 | | | 32 | RDTSCP |
| RDPMC | | 35 | 35 | | | 40 | |
| RDRAND | r | 16 | 16 | p23 15p0156 | | ~460 | RDRAND |
| RDSEED | r | 16 | 16 | p23 15p0156 | | ~460 | RDSEED |

Floating point x87 instructions

| Instruction | Operands | μops fused domain | μops unfused domain | μops each port | Latency | Reciprocal throughput | Comments |
|--------------------------|----------|-------------------|---------------------|-----------------|---------|-----------------------|----------|
| Move instructions | | | | | | | |
| FLD | r | 1 | 1 | p05 | 1 | 0.5 | |
| FLD | m32/64 | 1 | 1 | p23 | 3 | 0.5 | |
| FLD | m80 | 4 | 4 | 2p01 2p23 | 4 | 2 | |
| FBLD | m80 | 43 | 43 | | 46 | 22 | |
| FST(P) | r | 1 | 1 | p05 | 1 | 0.5 | |
| FST(P) | m32/m64 | 1 | 2 | p4 p237 | 3 | 1 | |
| FSTP | m80 | 7 | 7 | 3p0156 2p23 2p4 | 4 | 5 | |
| FBSTP | m80 | 244 | 226 | | 264 | 266 | |
| FXCH | r | 2 | 0 | none | 0 | 0.5 | |
| FILD | m | 1 | 2 | p05 p23 | 5 | 1 | |
| FIST(P) | m | 3 | 3 | p5 p23 p4 | 7 | 1 | |
| FISTTP | m | 3 | 3 | p1 p23 p4 | 7 | 2 | SSE3 |
| FLDZ | | 1 | 1 | p05 | | 1 | |
| FLD1 | | 2 | 2 | 2p05 | | 2 | |
| FLDPI FLDL2E etc. | | 2 | 2 | 2p05 | | 2 | |
| FCMOVcc | r | 4 | 4 | p0 p1 p56 | 3 | 2 | |
| FNSTSW | AX | 2 | 2 | p0 p0156 | 6 | 2 | |
| FNSTSW | m16 | 2 | 3 | p0 p4 p237 | 6 | 1 | |
| FLDCW | m16 | 3 | 3 | p01 p23 p6 | 7 | 2 | |
| FNSTCW | m16 | 2 | 3 | p237 p4 p6 | 6 | 1 | |
| FINCSTP FDECSTP | | 1 | 1 | p05 | 0 | 0.5 | |

Skylake

| | | | | | | |
|--------------------------------|---|--------|-----|-----------|---------|-----|
| FFREE(P) | r | 1 | 1 | p05 | | 0.5 |
| FNSAVE | m | 133 | 133 | | 176 | 176 |
| FRSTOR | m | 89 | 89 | | 175 | 175 |
| Arithmetic instructions | | | | | | |
| FADD(P) | | | | | | |
| FSUB(R)(P) | r | 1 | 1 | p5 | 3 | 1 |
| FADD(P) | | | | | | |
| FSUB(R)(P) | m | 2 | 3 | p5 p23 | | 1 |
| FMUL(P) | r | 1 | 1 | p0 | 5 | 1 |
| FMUL(P) | m | 2 | 3 | p0 p23 | | 1 |
| FDIV(R)(P) | r | 1 | 1 | p0 | 14-16 | 4-5 |
| FDIV(R)(P) | m | 1 | 2 | p0 p23 | | 4-5 |
| FABS | | 1 | 1 | p0 | 1 | 1 |
| FCHS | | 1 | 1 | p0 | 1 | 1 |
| FCOM(P) FUCOM | r | 1 | 1 | p5 | 3 | 1 |
| FCOM(P) FUCOM | m | 1 | 2 | p5 p23 | | 1 |
| FCOMPP FUCOMPP | | 2 | 2 | p0 p5 | | 1 |
| FCOMI(P) | | | | | | |
| FUCOMI(P) | r | 3 | 3 | p5 | | 1 |
| FIADD FISUB(R) | m | 3 | 4 | 2p5 p23 | | 2 |
| FIMUL | m | 2 | 3 | p0 p5 p23 | | 1 |
| FIDIV(R) | m | 2 | 3 | p0 p5 p23 | | |
| FICOM(P) | m | 2 | 3 | 2p5 p23 | | 2 |
| FTST | | 1 | 1 | p5 | 3 | 1 |
| FXAM | | 2 | 2 | 2p5 | 6 | 2 |
| FPREM | | 31 | 31 | | 26-30 | 17 |
| FPREM1 | | 31 | 31 | | 30-57 | 17 |
| FRNDINT | | 17 | 17 | | 21 | 11 |
| Math | | | | | | |
| FSCALE | | 27 | 27 | | 130 | 130 |
| FEXTRACT | | 17 | 17 | | 11 | 11 |
| FSQRT | | 1 | 1 | p0 | 14-21 | 4-7 |
| FSIN | | 53-105 | | | 50-120 | |
| FCOS | | 53-105 | | | 50-130 | |
| FSINCOS | | 55-120 | | | 55-150 | |
| F2XM1 | | 16-90 | | | 65-80 | |
| FYL2X | | 40-100 | | | 103 | |
| FYL2XP1 | | 56 | | | 77 | |
| FPTAN | | 40-112 | | | 140-160 | |
| FPATAN | | 30-160 | | | 100-160 | |
| Other | | | | | | |
| FNOP | | 1 | 1 | p05 | | 0.5 |
| WAIT | | 2 | 2 | p05 | | 2 |
| FNCLEX | | 5 | 5 | p156 | | 22 |
| FNINIT | | 18 | 18 | | | 78 |

Integer vector instructions

Skylake

| Instruction | Operands | μops fused domain | μops unfused domain | μops each port | Latency | Reciprocal throughput | Comments |
|--------------------------|-------------|-------------------|---------------------|----------------|---------|-----------------------|---------------|
| Move instructions | | | | | | | |
| MOVD | r32/64,mm/x | 1 | 1 | p0 | 2 | 1 | |
| MOVD | m32/64,mm/x | 1 | 2 | p237 p4 | 3 | 1 | |
| MOVD | mm/x,r32/64 | 1 | 1 | p5 | 2 | 1 | |
| MOVD | mm/x,m32/64 | 1 | 1 | p23 | 2 | 0.5 | |
| MOVQ | r64,mm/x | 1 | 1 | p0 | 2 | 1 | |
| MOVQ | mm/x,r64 | 1 | 1 | p5 | 1 | 1 | |
| MOVQ | mm,mm | 1 | | p05 | 1 | 0.5 | |
| MOVQ | x,x | 1 | | p015 | 1 | 0.33 | |
| MOVQ | mm/x,m64 | 1 | 1 | p23 | 2 | 0.5 | |
| MOVQ | m64, mm/x | 1 | 2 | p237 p4 | 3 | 1 | |
| MOVDQA/U | x,x | 1 | 1 | p015 | 0-1 | 0.25 | may eliminate |
| MOVDQA/U | x, m128 | 1 | 1 | p23 | 2 | 0.5 | |
| MOVDQA/U | m128, x | 1 | 2 | p237 p4 | 3 | 1 | |
| VMOVDQA/U | y,y | 1 | 1 | p015 | 0-1 | 0.25 | may eliminate |
| VMOVDQA/U | y,m256 | 1 | 1 | p23 | 3 | 0.5 | AVX |
| VMOVDQA/U | m256,y | 1 | 2 | p237 p4 | 3 | 1 | AVX |
| LDDQU | x, m128 | 1 | 1 | p23 | 3 | 0.5 | SSE3 |
| MOVDQ2Q | mm, x | 2 | 2 | p0 p5 | 2 | 1 | |
| MOVQ2DQ | x,mm | 2 | 2 | p0 p15 | 2 | 1 | |
| MOVNTQ | m64,mm | 1 | 2 | p237 p4 | ~418 | 1 | |
| MOVNTDQ | m128,x | 1 | 2 | p237 p4 | ~450 | 1 | |
| VMOVNTDQ | m256,y | 1 | 2 | p237 p4 | ~400 | 1 | AVX2 |
| MOVNTDQA | x, m128 | 2 | 2 | p23 p015 | 3 | 0.5 | SSE4.1 |
| VMOVNTDQA | y,m256 | 2 | 2 | p23 p015 | 3 | 0.5 | AVX2 |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | mm,mm | 3 | 3 | p5 | 2 | 2 | |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | mm,m64 | 3 | 3 | p23 2p5 | | 2 | |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | x,x / y,y,y | 1 | 1 | p5 | 1 | 1 | |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | x,m / y,y,m | 1 | 2 | p23 p5 | | 1 | |
| PACKUSDW | x,x / y,y,y | 1 | 1 | p5 | 1 | 1 | SSE4.1 |
| PACKUSDW | x,m / y,y,m | 1 | 2 | p23 p5 | | 1 | SSE4.1 |
| PUNPCKH/L | | | | | | | |
| BW/WD/DQ | v,v / v,v,v | 1 | 1 | p5 | 1 | 1 | |
| PUNPCKH/L | | | | | | | |
| BW/WD/DQ | v,m / v,v,m | 1 | 2 | p23 p5 | | 1 | |
| PUNPCKH/L | | | | | | | |
| QDQ | x,x / y,y,y | 1 | 1 | p5 | 1 | 1 | |
| PUNPCKH/L | | | | | | | |
| QDQ | x,m / y,y,m | 1 | 2 | p23 p5 | | 1 | |
| PMOVSX/ZX BW | | | | | | | |
| BD BQ DW DQ | x,x | 1 | 1 | p5 | 1 | 1 | SSE4.1 |
| PMOVSX/ZX BW | | | | | | | |
| BD BQ DW DQ | x,m | 1 | 2 | p23 p5 | | 1 | SSE4.1 |
| VPMOVSX/ZX BW | | | | | | | |
| BD BQ DW DQ | y,x | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPMOVSX/ZX BW | | | | | | | |
| BD BQ DW DQ | y,m | 2 | 2 | p5 p23 | | 1 | AVX2 |

Skylake

| | | | | | | | |
|----------------|-----------------|-----|----|----------------|--------|------|--------|
| PSHUFB | v,v / v,v,v | 1 | 1 | p5 | 1 | 1 | SSSE3 |
| PSHUFB | v,m / v,v,m | 2 | 2 | p23 p5 | | 1 | SSSE3 |
| PSHUFW | mm,mm,i | 1 | 1 | p5 | 1 | 1 | |
| PSHUFW | mm,m64,i | 2 | 2 | p23 p5 | | 1 | |
| PSHUFD | v,v,i | 1 | 1 | p5 | 1 | 1 | |
| PSHUFD | v,m,i | 1-2 | 2 | p23 p5 | | 1 | |
| PSHUFL/HW | v,v,i | 1 | 1 | p5 | 1 | 1 | |
| PSHUFL/HW | v,m,i | 2 | 2 | p23 p5 | | 1 | |
| PALIGNR | v,v,i / v,v,v,i | 1 | 1 | p5 | 1 | 1 | SSSE3 |
| PALIGNR | v,m,i / v,v,m,i | 2 | 2 | p23 p5 | | 1 | SSSE3 |
| PBLENDVB | x,x,xmm0 | 1 | 1 | p015 | 1 | 1 | SSE4.1 |
| PBLENDVB | x,m,xmm0 | 2 | 2 | p015 p23 | | 2 | SSE4.1 |
| VPBLENDVB | v,v,v,v | 2 | 2 | 2p015 | 2 | 1 | AVX2 |
| VPBLENDVB | v,v,m,v | 3 | 3 | 2p015 p23 | | 2 | AVX2 |
| PBLENDW | x,x,i / v,v,v,i | 1 | 1 | p5 | 1 | 1 | SSE4.1 |
| PBLENDW | x,m,i / v,v,m,i | 2 | 2 | p23 p5 | | 1 | SSE4.1 |
| VPBLEND | v,v,v,i | 1 | 1 | p015 | 1 | 0.33 | AVX2 |
| VPBLEND | v,v,m,i | 2 | 2 | p015 p23 | | 0.5 | AVX2 |
| VPERMD | y,y,y | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPERMD | y,y,m | 1 | 2 | p5 p23 | | 1 | AVX2 |
| VPERMQ | y,y,i | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPERMQ | y,m,i | 2 | 2 | p5 p23 | | 1 | AVX2 |
| VPERM2I128 | y,y,y,i | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPERM2I128 | y,y,m,i | 2 | 2 | p5 p23 | | 1 | AVX2 |
| MASKMOVQ | mm,mm | 4 | 4 | p0 p4 2p23 | ~450 | 2 | |
| MASKMOVQ | x,x | 10 | 10 | 4p04 2p56 4p23 | 18-500 | 6 | |
| VPMASKMOVQ | v,v,m | 2 | 2 | p23 p015 | 4 | 0.5 | AVX2 |
| VPMASKMOVQ | m,v,v | 3 | 3 | p0 p4 p23 | 14 | 1 | AVX2 |
| PMOVBMSKB | r,v | 1 | 1 | p0 | 2-3 | 1 | |
| PEXTRB/W/D/Q | r,x,i | 2 | 2 | p0 p5 | 3 | 1 | SSE4.1 |
| PEXTRB/W/D/Q | m,x,i | 2 | 3 | p23 p4 p5 | | 1 | SSE4.1 |
| VEXTRACTI128 | x,y,i | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VEXTRACTI128 | m,y,i | 2 | 2 | p23 p4 | 4 | 1 | AVX2 |
| PINSRB | x,r32,i | 2 | 2 | 2p5 | 3 | 2 | SSE4.1 |
| PINSRB | x,m8,i | 2 | 2 | p23 p5 | | 1 | SSE4.1 |
| PINSRW | mm/x,r32,i | 2 | 2 | p5 | 3 | 2 | |
| PINSRW | mm/x,m16,i | 2 | 2 | p23 p5 | | 1 | |
| PINSRD/Q | x,r32,i | 2 | 2 | 2p5 | 3 | 2 | SSE4.1 |
| PINSRD/Q | x,m32,i | 2 | 2 | p23 p5 | | 1 | SSE4.1 |
| VINSERTI128 | y,y,x,i | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VINSERTI128 | y,y,m,i | 2 | 2 | p015 p23 | 3 | 0.5 | AVX2 |
| VPBROADCAST | | | | | | | |
| B/W/D/Q | x,x | 1 | 1 | p5 | 1 | 1 | AVX2 |
| VPBROADCAST | | | | | | | |
| B/W | x,m8/16 | 2 | 2 | p23 p5 | 7 | 1 | AVX2 |
| VPBROADCAST | | | | | | | |
| D/Q | x,m32/64 | 1 | 1 | p23 | 4 | 0.5 | AVX2 |
| VPBROADCAST | | | | | | | |
| B/W/D/Q | y,x | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPBROADCAST | | | | | | | |
| B/W | y,m8/16 | 2 | 2 | p23 p5 | 7 | 1 | AVX2 |
| VPBROADCAST | | | | | | | |
| D/Q | y,m32/64 | 1 | 1 | p23 | 3 | 0.5 | AVX2 |
| VBROADCASTI128 | y,m128 | 1 | 1 | p23 | 3 | 0.5 | AVX2 |
| VPGATHERDD | x,[r+s*x],x | 4 | 4 | p0 p1 p23 p5 | | 4 | AVX2 |

Skylake

| | | | | | | | |
|--------------------------------|-------------|---|---|--------------|----|------|--------|
| VPGATHERDD | y,[r+s*y],y | 4 | 4 | p0 p1 p23 p5 | | 5 | AVX2 |
| VPGATHERQD | x,[r+s*x],x | 5 | 5 | p0 p1 p23 p5 | | 2 | AVX2 |
| VPGATHERQD | x,[r+s*y],x | 4 | 4 | p0 p1 p23 p5 | | 4 | AVX2 |
| VPGATHERDQ | x,[r+s*x],x | 5 | 5 | p0 p1 p23 p5 | | 2 | AVX2 |
| VPGATHERDQ | y,[r+s*x],y | 4 | 4 | p0 p1 p23 p5 | | 4 | AVX2 |
| VPGATHERQQ | x,[r+s*x],x | 5 | 5 | p0 p1 p23 p5 | | 2 | AVX2 |
| VPGATHERQQ | y,[r+s*y],y | 4 | 4 | p0 p1 p23 p5 | | 4 | AVX2 |
| Arithmetic instructions | | | | | | | |
| PADD/SUB(S,US) B/W/D/Q | v,v / v,v,v | 1 | 1 | p015 | 1 | 0.33 | |
| PADD/SUB(S,US) B/W/D/Q | v,m / v,v,m | 1 | 2 | p015 p23 | | 0.5 | |
| PHADD(S)W/D | | | | | | | |
| PHSUB(S)W/D | v,v / v,v,v | 3 | 3 | p01 2p5 | 3 | 2 | SSSE3 |
| PHADD(S)W/D | | | | | | | |
| PHSUB(S)W/D | v,m / v,v,m | 4 | 4 | p01 2p5 p23 | | 2 | SSSE3 |
| PCMPEQB/W/D | | | | | | | |
| PCMPGTB/W/D | mm,mm | 1 | 1 | p0 | 1 | 1 | |
| PCMPEQB/W/D | | | | | | | |
| PCMPGTB/W/D | x,x / y,y,y | 1 | 1 | p01 | 1 | 0.5 | |
| PCMPEQB/W/D | | | | | | | |
| PCMPGTB/W/D | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 | |
| PCMPEQQ | v,v / v,v,v | 1 | 1 | p01 | 1 | 0.5 | SSE4.1 |
| PCMPEQQ | v,m / v,v,m | 1 | 2 | p01 p23 | | 0.5 | SSE4.1 |
| PCMPGTQ | v,v / v,v,v | 1 | 1 | p5 | 3 | 1 | SSE4.2 |
| PCMPGTQ | v,m / v,v,m | 1 | 2 | p5 p23 | | 1 | SSE4.2 |
| PMULL/HW | | | | | | | |
| PMULHUW | mm,mm | 1 | 1 | p0 | 5 | 1 | |
| PMULL/HW | | | | | | | |
| PMULHUW | x,x / y,y,y | 1 | 1 | p01 | 5 | 0.5 | |
| PMULL/HW | | | | | | | |
| PMULHUW | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 | |
| PMULHSW | mm,mm | 1 | 1 | p0 | 5 | 1 | SSSE3 |
| PMULHSW | x,x / y,y,y | 1 | 1 | p01 | 5 | 0.5 | SSSE3 |
| PMULHSW | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 | SSSE3 |
| PMULLD | x,x / y,y,y | 2 | 2 | 2p01 | 10 | 1 | SSE4.1 |
| PMULLD | x,m / y,y,m | 3 | 3 | 2p01 p23 | | 1 | SSE4.1 |
| PMULDQ | x,x / y,y,y | 1 | 1 | p01 | 5 | 0.5 | SSE4.1 |
| PMULDQ | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 | SSE4.1 |
| PMULUDQ | mm,mm | 1 | 1 | p0 | 5 | 1 | |
| PMULUDQ | x,x / y,y,y | 1 | 1 | p01 | 5 | 0.5 | |
| PMULUDQ | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 | |
| PMADDWD | mm,mm | 1 | 1 | p0 | 5 | 1 | |
| PMADDWD | x,x / y,y,y | 1 | 1 | p01 | 5 | 0.5 | |
| PMADDWD | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 | |
| PMADDUBSW | mm,mm | 1 | 1 | p0 | 5 | 1 | SSSE3 |
| PMADDUBSW | x,x / y,y,y | 1 | 1 | p01 | 5 | 0.5 | SSSE3 |
| PMADDUBSW | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 | SSSE3 |
| PAVGB/W | mm,mm | 1 | 1 | p0 | 1 | 1 | |
| PAVGB/W | x,x / y,y,y | 1 | 1 | p01 | 1 | 0.5 | |
| PAVGB/W | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 | |

Skylake

| | | | | | | | |
|-------------------------------------|-----------------|---|---|-----------|---|------|--------|
| PMIN/PMAX SB/SW/SD UB/UW/UD | mm,mm | 1 | 1 | p0 | 1 | 1 | SSE4.1 |
| PMIN/PMAX SB/SW/SD UB/UW/UD | x,x / y,y,y | 1 | 1 | p01 | 1 | 0.5 | SSE4.1 |
| PMIN/PMAX SB/SW/SD UB/UW/UD | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 | SSE4.1 |
| PHMINPOSUW | x,x | 1 | 1 | p0 | 4 | 1 | SSE4.1 |
| PHMINPOSUW | x,m128 | 1 | 2 | p0 p23 | | 1 | SSE4.1 |
| PABSB/W/D | mm,mm | 1 | 1 | p0 | 1 | 1 | SSSE3 |
| PABSB/W/D | x,x / y,y | 1 | 1 | p01 | 1 | 0.5 | SSSE3 |
| PABSB/W/D | x,m / y,m | 1 | 2 | p01 p23 | | 0.5 | SSSE3 |
| PSIGNB/W/D | mm,mm | 1 | 1 | p0 | 1 | 1 | SSSE3 |
| PSIGNB/W/D | x,x / y,y,y | 1 | 1 | p01 | 1 | 0.5 | SSSE3 |
| PSIGNB/W/D | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 | SSSE3 |
| PSADBW | v,v / v,v,v | 1 | 1 | p5 | 3 | 1 | |
| PSADBW | v,m / v,v,m | 1 | 2 | p5 p23 | | 1 | |
| MPSADBW | x,x,i / v,v,v,i | 2 | 2 | 2p5 | 4 | 2 | SSE4.1 |
| MPSADBW | x,m,i / v,v,m,i | 3 | 3 | 2p5 p23 | | 2 | SSE4.1 |
| Logic instructions | | | | | | | |
| PAND PANDN POR PXOR | mm,mm | 1 | 1 | p05 | 1 | 0.5 | |
| PAND PANDN POR PXOR | x,x / y,y,y | 1 | 1 | p015 | 1 | 0.33 | |
| PAND PANDN POR PXOR | v,m / v,v,m | 1 | 2 | p015 p23 | | 0.5 | |
| PTEST | v,v | 2 | 2 | p0 p5 | 3 | 1 | SSE4.1 |
| PTEST | v,m | 2 | 3 | p0 p5 p23 | | 1 | SSE4.1 |
| PSLLW/D/Q PSRLW/D/Q PSRAW/D/Q | mm,mm | 1 | 1 | p0 | 1 | 1 | |
| PSLLW/D/Q PSRLW/D/Q PSRAW/D/Q | mm,m64 | 2 | 2 | p0 p23 | | 1 | |
| PSLLW/D/Q PSRLW/D/Q PSRAW/D/Q | x,x / v,v,x | 2 | 2 | p01 p5 | 1 | 1 | |
| PSLLW/D/Q PSRLW/D/Q PSRAW/D/Q | x,m / v,v,m | 2 | 2 | p01 p23 | | 0.5 | |
| PSLLW/D/Q PSRLW/D/Q PSRAW/D/Q | mm,i | 1 | 1 | p0 | 1 | 1 | |
| PSLLW/D/Q PSRLW/D/Q PSRAW/D/Q | x,i / y,y,i | 1 | 1 | p01 | 1 | 0.5 | |
| VPSLLVD/Q VPSRAVD | | | | | | | |
| VPSRLVD/Q | v,v,v | 1 | 1 | p01 | 1 | 0.5 | AVX2 |
| VPSLLVD/Q VPSRAVD | | | | | | | |
| VPSRLVD/Q | v,v,m | 1 | 2 | p01 p23 | | 0.5 | AVX2 |

Skylake

| | | | | | | | | |
|--|-------------|----------|----|---------|------------------|-----|--------|--------|
| PSLLDQ | x,i / v,v,i | 1 | 1 | p5 | 1 | 1 | | |
| PSRLDQ | | | | | | | | |
| String instructions | | | | | | | | |
| PCMPESTRI | | x,x,i | 8 | 8 | 6p05 2p16 | 12 | 4 | SSE4.2 |
| PCMPESTRI | | x,m128,i | 8 | 8 | 3p0 2p16 2p5 p23 | | 4 | SSE4.2 |
| PCMPESTRM | | x,x,i | 9 | 9 | 3p0 2p16 4p5 | 9 | 5 | SSE4.2 |
| PCMPESTRM | | x,m128,i | 9 | 9 | 6p05 2p16 p23 | | 5 | SSE4.2 |
| PCMPISTRI | | x,x,i | 3 | 3 | 3p0 | 10 | 3 | SSE4.2 |
| PCMPISTRI | | x,m128,i | 4 | 4 | 3p0 p23 | | 3 | SSE4.2 |
| PCMPISTRM | | x,x,i | 3 | 3 | 3p0 | 9 | 3 | SSE4.2 |
| PCMPISTRM | x,m128,i | 4 | 4 | 3p0 p23 | | 3 | SSE4.2 | |
| Encryption instructions | | | | | | | | |
| PCLMULQDQ | x,x,i | 1 | 1 | p5 | 7 | 1 | CLMUL | |
| PCLMULQDQ | x,m,i | 2 | 2 | p5 p23 | | 1 | CLMUL | |
| AESDEC, AESDECLAST, AESENC, AESENCCLAST | x,x | 1 | 1 | p0 | 4 | 1 | AES | |
| AESDEC, AESDECLAST, AESENC, AESENCCLAST | x,m | 2 | 2 | p0 p23 | | 1.5 | AES | |
| AESIMC | x,x | 2 | 2 | 2p0 | 8 | 2 | AES | |
| AESIMC | x,m | 3 | 3 | 2p0 p23 | | 2 | AES | |
| AESKEYGENAS SIST | x,x,i | 13 | 13 | p0 p5 | 12 | 12 | AES | |
| AESKEYGENAS SIST | x,m,i | 13 | 13 | | | 12 | AES | |
| Other | | | | | | | | |
| EMMS | | 10 | 10 | p05 | | 6 | | |

Floating point XMM and YMM instructions

| Instruction | Operands | μops fused domain | μops unfused domain | μops each port | Latency | Reciprocal throughput | Comments |
|--------------------------|----------|-------------------|---------------------|----------------|---------|-----------------------|--------------------------------|
| Move instructions | | | | | | | |
| MOVAPS/D | x,x | 1 | 1 | p015 | 0-1 | 0.25 | may eliminate may eliminate |
| VMOVAPS/D | y,y | 1 | 1 | p015 | 0-1 | 0.25 | |
| MOVAPS/D MOVUPS/D | x,m128 | 1 | 1 | p23 | 2 | 0.5 | AVX |
| VMOVAPS/D VMOVUPS/D | y,m256 | 1 | 1 | p23 | 3 | 0.5 | |
| MOVAPS/D MOVUPS/D | m128,x | 1 | 2 | p237 p4 | 3 | 1 | AVX |
| VMOVAPS/D VMOVUPS/D | m256,y | 1 | 2 | p237 p4 | 3 | 1 | |
| MOVSS/D | x,x | 1 | 1 | p5 | 1 | 1 | |
| MOVSS/D | x,m32/64 | 1 | 1 | p23 | 3 | 0.5 | |
| MOVSS/D | m32/64,x | 1 | 2 | p237 p4 | 3 | 1 | |

Skylake

| | | | | | | | |
|----------------|-----------------|---|---|--------------|------|------|--------|
| MOVHPS/D | x,m64 | 1 | 2 | p23 p5 | 4 | 1 | |
| MOVHPS/D | m64,x | 1 | 2 | p4 p237 | 3 | 1 | |
| MOVLPS/D | x,m64 | 1 | 2 | p23 p5 | 4 | 1 | |
| MOVLPS/D | m64,x | 1 | 2 | p4 p237 | 3 | 1 | |
| MOVHLPS | x,x | 1 | 1 | p5 | 1 | 1 | |
| MOVLHPS | x,x | 1 | 1 | p5 | 1 | 1 | |
| MOVMSKPS/D | r32,x | 1 | 1 | p0 | 2 | 1 | |
| VMOVMSKPS/D | r32,y | 1 | 1 | p0 | 3 | 1 | |
| MOVNTPS/D | m128,x | 1 | 2 | p4 p237 | ~400 | 1 | |
| VMOVNTPS/D | m256,y | 1 | 2 | p4 p237 | ~400 | 1 | AVX |
| SHUFPS/D | x,x,i / v,v,v,i | 1 | 1 | p5 | 1 | 1 | |
| SHUFPS/D | x,m,i / v,v,m,i | 2 | 2 | p5 p23 | | 1 | |
| VPERMILPS/PD | v,v,i | 1 | 1 | p5 | 1 | 1 | AVX |
| VPERMILPS/PD | v,m,i | 2 | 2 | p5 p23 | | 1 | AVX |
| VPERMILPS/PD | v,v,v | 1 | 1 | p5 | 1 | 1 | AVX |
| VPERMILPS/PD | v,v,m | 2 | 2 | p5 p23 | | 1 | AVX |
| VPERM2F128 | y,y,y,i | 1 | 1 | p5 | 3 | 1 | AVX |
| VPERM2F128 | y,y,m,i | 2 | 2 | p5 p23 | | 1 | AVX |
| VPERMPS | y,y,y | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPERMPS | y,y,m | 1 | 2 | p5 p23 | | 1 | AVX2 |
| VPERMPD | y,y,i | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VPERMPD | y,m,i | 2 | 2 | p5 p23 | | 1 | AVX2 |
| BLENDPS/PD | x,x,i / v,v,v,i | 1 | 1 | p015 | 1 | 0.33 | SSE4.1 |
| BLENDPS/PD | x,m,i / v,v,m,i | 2 | 2 | p015 p23 | | 0.5 | SSE4.1 |
| BLENDVPS/PD | x,x,xmm0 | 1 | 1 | p015 | 1 | 1 | SSE4.1 |
| BLENDVPS/PD | x,m,xmm0 | 2 | 2 | p015 p23 | | 1 | SSE4.1 |
| VBLENDVPS/PD | v,v,v,v | 2 | 2 | 2p015 | 2 | 1 | AVX |
| VBLENDVPS/PD | v,v,m,v | 3 | 3 | 2p015 p23 | | 1 | AVX |
| MOVDDUP | v,v | 1 | 1 | p5 | 1 | 1 | SSE3 |
| MOVDDUP | v,m | 1 | 1 | p23 | 3 | 0.5 | SSE3 |
| VBROADCASTSS | x,m32 | 1 | 1 | p23 | 2 | 0.5 | AVX |
| VBROADCASTSS | y,m32 | 1 | 1 | p23 | 3 | 0.5 | AVX |
| VBROADCASTSS | x,x | 1 | 1 | p5 | 1 | 1 | AVX2 |
| VBROADCASTSS | y,x | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VBROADCASTSD | y,m64 | 1 | 1 | p23 | 3 | 0.5 | AVX |
| VBROADCASTSD | y,x | 1 | 1 | p5 | 3 | 1 | AVX2 |
| VBROADCASTF128 | y,m128 | 1 | 1 | p23 | 3 | 0.5 | AVX |
| MOVSH/LDUP | v,v | 1 | 1 | p5 | 1 | 1 | SSE3 |
| MOVSH/LDUP | v,m | 1 | 1 | p23 | 3 | 0.5 | SSE3 |
| UNPCKH/LPS/D | x,x / v,v,v | 1 | 1 | p5 | 1 | 1 | SSE3 |
| UNPCKH/LPS/D | x,m / v,v,m | 1 | 2 | p5 p23 | | 1 | SSE3 |
| EXTRACTPS | r32,x,i | 2 | 2 | p0 p5 | | 1 | SSE4.1 |
| EXTRACTPS | m32,x,i | 2 | 3 | p4 p5 p23 | 5 | 1 | SSE4.1 |
| VEEXTRACTF128 | x,y,i | 1 | 1 | p5 | 3 | 1 | AVX |
| VEEXTRACTF128 | m128,y,i | 2 | 2 | p23 p4 | 6 | 1 | AVX |
| INSERTPS | x,x,i | 1 | 1 | p5 | 1 | 1 | SSE4.1 |
| INSERTPS | x,m32,i | 2 | 2 | p23 p5 | 4 | 1 | SSE4.1 |
| VINSERTF128 | y,y,x,i | 1 | 1 | p5 | 3 | 1 | AVX |
| VINSERTF128 | y,y,m128,i | 2 | 2 | p015 p23 | 5 | 0.5 | AVX |
| VMASKMOVPS/D | v,v,m | 2 | 2 | p015 p23 | 3 | 0.5 | AVX |
| VMASKMOVPS/D | m128,x,x | 4 | 4 | p0 p4 p23 | 13 | 1 | AVX |
| VMASKMOVPS/D | m256,y,y | 4 | 4 | p0 p4 p23 | 13 | 1 | AVX |
| VGATHERDPS | x,[r+s*x],x | 4 | 4 | p0 p1 p23 p5 | 12 | 4 | AVX2 |

Skylake

| | | | | | | | |
|-------------------|-------------|---|---|--------------|----|-----|------|
| VGATHERDPS | y,[r+s*y],y | 4 | 4 | p0 p1 p23 p5 | 13 | 5 | AVX2 |
| VGATHERQPS | x,[r+s*x],x | 5 | 5 | p0 p1 p23 p5 | | 2 | AVX2 |
| VGATHERQPS | x,[r+s*y],x | 4 | 4 | p0 p1 p23 p5 | | 4 | AVX2 |
| VGATHERDPD | x,[r+s*x],x | 5 | 5 | p0 p1 p23 p5 | | 2 | AVX2 |
| VGATHERDPD | y,[r+s*x],y | 4 | 4 | p0 p1 p23 p5 | | 4 | AVX2 |
| VGATHERQPD | x,[r+s*x],x | 5 | 5 | p0 p1 p23 p5 | | 2 | AVX2 |
| VGATHERQPD | y,[r+s*y],y | 4 | 4 | p0 p1 p23 p5 | | 4 | AVX2 |
| Conversion | | | | | | | |
| CVTPD2PS | x,x | 2 | 2 | p01 p5 | 5 | 1 | |
| CVTPD2PS | x,m128 | 2 | 3 | p01 p5 p23 | | 1 | |
| VCVTPD2PS | x,y | 2 | 2 | p01 p5 | 7 | 1 | AVX |
| VCVTPD2PS | x,m256 | 2 | 3 | p01 p5 p23 | | 1 | AVX |
| CVTSD2SS | x,x | 2 | 2 | p01 p5 | 5 | 1 | |
| CVTSD2SS | x,m64 | 2 | 3 | p01 p5 p23 | | 1 | |
| CVTPS2PD | x,x | 2 | 2 | p01 p5 | 5 | 1 | |
| CVTPS2PD | x,m64 | 1 | 2 | p01 p5 p23 | | 0.5 | |
| VCVTPS2PD | y,x | 2 | 2 | p01 p5 | 7 | 1 | AVX |
| VCVTPS2PD | y,m128 | 1 | 2 | p01 p5 p23 | | 0.5 | AVX |
| CVTSS2SD | x,x | 2 | 2 | p01 p5 | 5 | 2 | |
| CVTSS2SD | x,m32 | 1 | 2 | p01 p5 p23 | | 2 | |
| CVTDQ2PS | x,x | 1 | 1 | p01 | 4 | 0.5 | |
| CVTDQ2PS | x,m128 | 1 | 2 | p01 p23 | | 0.5 | |
| VCVTDQ2PS | y,y | 1 | 1 | p01 | 4 | 0.5 | AVX |
| VCVTDQ2PS | y,m256 | 1 | 2 | p01 p23 | | 0.5 | AVX |
| CVT(T) PS2DQ | x,x | 1 | 1 | p01 | 4 | 0.5 | |
| CVT(T) PS2DQ | x,m128 | 1 | 2 | p01 p23 | | 0.5 | |
| VCVT(T) PS2DQ | y,y | 1 | 1 | p01 | 4 | 0.5 | AVX |
| VCVT(T) PS2DQ | y,m256 | 1 | 2 | p01 p23 | | 0.5 | AVX |
| CVTDQ2PD | x,x | 2 | 2 | p01 p5 | 5 | 1 | |
| CVTDQ2PD | x,m64 | 2 | 2 | p01 p23 | | 0.5 | |
| VCVTDQ2PD | y,x | 2 | 2 | p01 p5 | 7 | 1 | AVX |
| VCVTDQ2PD | y,m128 | 1 | 2 | p01 p23 | | 0.5 | AVX |
| CVT(T)PD2DQ | x,x | 2 | 2 | p01 p5 | 5 | 1 | |
| CVT(T)PD2DQ | x,m128 | 3 | 3 | p01 p23 p5 | | 1 | |
| VCVT(T)PD2DQ | x,y | 2 | 2 | p01 p5 | 7 | 1 | AVX |
| VCVT(T)PD2DQ | x,m256 | 2 | 3 | p01 p23 p5 | | 1 | AVX |
| CVTPI2PS | x,mm | 2 | 2 | p0 p1 | 6 | 2 | |
| CVTPI2PS | x,m64 | 1 | 2 | p01 p23 | | 3 | |
| CVT(T)PS2PI | mm,x | 2 | 2 | p0 p5 | 7 | 1 | |
| CVT(T)PS2PI | mm,m128 | 2 | 2 | p0 p23 | | 1 | |
| CVTPI2PD | x,mm | 2 | 2 | p01 p5 | 5 | 1 | |
| CVTPI2PD | x,m64 | 1 | 2 | p01 p23 | | 0.5 | |
| CVT(T) PD2PI | mm,x | 2 | 2 | p01 p5 | 5 | 1 | |
| CVT(T) PD2PI | mm,m128 | 2 | 3 | p01 p23 p5 | | 1 | |
| CVTSI2SS | x,r32 | 2 | 2 | p01 p5 | 6 | 2 | |
| CVTSI2SS | x,r64 | 3 | 3 | p01 2p5 | 7 | 2 | |
| CVTSI2SS | x,m32 | 1 | 2 | p1 p23 | | 3 | |
| CVT(T)SS2SI | r32,x | 2 | 2 | 2p01 | 6 | 1 | |
| CVT(T)SS2SI | r64,x | 3 | 3 | 2p01 p5 | 7 | 1 | |
| CVT(T)SS2SI | r32,m32 | 3 | 3 | 2p01 p23 | | 1 | |
| CVTSI2SD | x,r32/64 | 2 | 2 | p01 p5 | 6 | 2 | |
| CVTSI2SD | x,m32 | 1 | 2 | p01 p23 | | 2 | |

Skylake

| | | | | | | | |
|-------------------|-----------------|---|---|----------------|-------|-----|--------|
| CVT(T)SD2SI | r32/64,x | 2 | 2 | p0 p1 | 6 | 1 | |
| CVT(T)SD2SI | r32,m64 | 3 | 3 | 2p01 p23 | | 1 | |
| VCVTPS2PH | x,v,i | 2 | 2 | p01 p5 | 5-7 | 1 | F16C |
| VCVTPS2PH | m,v,i | 3 | 3 | p01 p4 p23 | | 1 | F16C |
| VCVTPH2PS | v,x | 2 | 2 | p01 p5 | 5-7 | 1 | F16C |
| VCVTPH2PS | v,m | 1 | 2 | p01 p23 | | 1 | F16C |
| Arithmetic | | | | | | | |
| ADDSS/D PS/D | | | | | | | |
| SUBSS/D PS/D | x,x / v,v,v | 1 | 1 | p01 | 4 | 0.5 | |
| ADDSS/D PS/D | | | | | | | |
| SUBSS/D PS/D | x,m / v,v,m | 1 | 2 | p01 p23 | | 0.5 | |
| ADDSUBPS/D | x,x / v,v,v | 1 | 1 | p01 | 4 | 0.5 | SSE3 |
| ADDSUBPS/D | x,m / v,v,m | 1 | 2 | p01 p23 | | 0.5 | SSE3 |
| HADDPS/D | | | | | | | |
| HSUBPS/D | x,x / v,v,v | 3 | 3 | p01 2p5 | 6 | 2 | SSE3 |
| HADDPS/D | | | | | | | |
| HSUBPS/D | x,m / v,v,m | 4 | 4 | p1 2p5 p23 | | 2 | SSE3 |
| MULSS/D PS/D | x,x / v,v,v | 1 | 1 | p01 | 4 | 0.5 | |
| MULSS/D PS/D | x,m / v,v,m | 1 | 2 | p01 p23 | | 0.5 | |
| DIVSS | x,x | 1 | 1 | p0 | 11 | 3 | |
| DIVPS | x,x | 1 | 1 | p0 | 11 | 3 | |
| DIVSS DIVPS | x,m | 1 | 2 | p0 p23 | | 3-5 | |
| DIVSD | x,x | 1 | 1 | p0 | 13-14 | 4 | |
| DIVPD | x,x | 1 | 1 | p0 | 13-14 | 4 | |
| DIVSD DIVPD | x,m | 1 | 2 | p0 p23 | | 4 | |
| VDIVPS | y,y,y | 1 | 1 | p0 | 11 | 5 | AVX |
| VDIVPS | y,y,m256 | 1 | 2 | p0 p23 | | 5 | AVX |
| VDIVPD | y,y,y | 1 | 1 | p0 | 13-14 | 8 | AVX |
| VDIVPD | y,y,m256 | 4 | 4 | p0 p23 | | 8 | AVX |
| RCPSS/PS | v,v | 1 | 1 | p0 | 4 | 1 | |
| RCPSS/PS | v,m | 1 | 2 | p0 p23 | | 1 | |
| CMPccSS/D | | | | | | | |
| CMPccPS/D | x,x / v,v,v | 1 | 1 | p01 | 4 | 0.5 | |
| CMPccSS/D | | | | | | | |
| CMPccPS/D | x,m / v,v,m | 2 | 2 | p01 p23 | | 0.5 | |
| (U)COMISS/D | x,x | 1 | 1 | p0 | | 1 | |
| (U)COMISS/D | x,m32/64 | 2 | 2 | p0 p23 | | 1 | |
| MAXSS/D PS/D | | | | | | | |
| MINSS/D PS/D | x,x / v,v,v | 1 | 1 | p01 | 4 | 0.5 | |
| MAXSS/D PS/D | | | | | | | |
| MINSS/D PS/D | x,m / v,v,m | 1 | 2 | p01 p23 | | 0.5 | |
| ROUNDSS/D PS/D | v,v,i | 2 | 2 | 2p01 | 8 | 1 | SSE4.1 |
| ROUNDSS/D PS/D | v,m,i | 3 | 3 | 2p01 p23 | | 1 | SSE4.1 |
| DPPS | x,x,i / v,v,v,i | 4 | 4 | 3p01 p5 | 13 | 1.5 | SSE4.1 |
| DPPS | x,m,i / v,v,m,i | 6 | 6 | 3p01 p23 p5 p6 | | 1.5 | SSE4.1 |
| DPPD | x,x,i | 3 | 3 | 2p01 p5 | 9 | 1 | SSE4.1 |
| DPPD | x,m128,i | 4 | 4 | 2p01 p23 p5 | | 1 | SSE4.1 |
| VFMADD... | | | | | | | |
| (all FMA instr.) | v,v,v | 1 | 1 | p01 | 4 | 0.5 | FMA |
| VFMADD... | | | | | | | |
| (all FMA instr.) | v,v,m | 1 | 2 | p01 p23 | | 0.5 | FMA |

Skylake

| Math | | | | | | | |
|--------------------------|-------------|-----|----|---------------|-------|------|----------------|
| SQRTSS/PS | x,x | 1 | 1 | p0 | 12 | 3 | |
| SQRTSS/PS | x,m128 | 1 | 2 | p0 p23 | | 3 | |
| VSQRTPS | y,y | 1 | 1 | p0 | 12 | 6 | AVX |
| VSQRTPS | y,m256 | 4 | 4 | p0 p23 | | 6 | AVX |
| SQRTSD | x,x | 1 | 1 | p0 | 15-16 | 4-6 | |
| SQRTPD | x,x | 1 | 1 | p0 | 15-16 | 4-6 | |
| SQRTSD/PD | x,m128 | 1 | 2 | p0 p23 | | 4-6 | |
| VSQRTPD | y,y | 1 | 1 | p0 | 15-16 | 9-12 | AVX |
| VSQRTPD | y,m256 | 4 | 4 | p0 p23 | | 9-12 | AVX |
| RSQRTSS/PS | v,v | 1 | 1 | p0 | 4 | 1 | |
| RSQRTSS/PS | v,m | 1 | 2 | p0 p23 | | 1 | |
| Logic | | | | | | | |
| AND/ANDN/OR/ XORPS/PD | x,x / v,v,v | 1 | 1 | p015 | 1 | 0.33 | |
| AND/ANDN/OR/ XORPS/PD | x,m / v,v,m | 1 | 2 | p015 p23 | | 0.5 | |
| Other | | | | | | | |
| VZEROUPPER | | 4 | 4 | none | | 1 | AVX |
| VZEROALL | | 25 | 25 | p0 p1 p5 p6 | | 12 | AVX, 32 bit |
| VZEROALL | | 34 | 34 | p0 p1 p5 p6 | | 12 | AVX, 64 bit |
| LDMXCSR | m32 | 4 | 4 | p0 p5 p6 p23 | 5 | 3 | |
| STMXCSR | m32 | 3 | 4 | p0 p4 p6 p237 | 5 | 2 | |
| FXSAVE | m4096 | 106 | | | 78 | 78 | 32 bit mode |
| FXSAVE | m4096 | 136 | | | 64 | 64 | 64 bit mode |
| FXRSTOR | m4096 | 105 | | | 76 | 76 | 32 bit mode |
| FXRSTOR | m4096 | 121 | | | 77 | 77 | 64 bit mode |
| XSAVE | | 247 | | | 107 | 107 | 32 bit mode |
| XSAVE | | 304 | | | 107 | 107 | 64 bit mode |
| XRSTOR | | 257 | | | 122 | 122 | 32 bit mode |
| XRSTOR | | 257 | | | 122 | 122 | 64 bit mode |
| XSAVEOPT | m | 168 | | | 74 | 74 | |

Intel Skylake-X

List of instruction timings and μ op breakdown

Explanation of column headings:

| | |
|--|--|
| Instruction: | Name of instruction. Multiple names mean that these instructions have the same data. Instructions with or without V name prefix behave the same unless otherwise noted. |
| Operands: | i = immediate data, r = register, mm = 64 bit mmx register, x = 128 bit xmm register, mm/x = mmx or xmm register, y = 256 bit ymm register, z = 512 bit zmm register, xy = xmm or ymm register, v = any vector register (mmx, xmm, ymm, zmm). m = memory operand, m32 = 32-bit memory operand, etc. |
| μops fused domain: | The number of μ ops at the decode, rename, allocate and retire stages in the pipeline. Fused μ ops count as one. |
| μops unfused domain: | The total number of μ ops for all execution ports. Fused μ ops count as two. Fused macro-ops count as one. The instruction has μ op fusion if this number is higher than the number under fused domain. Some operations are not counted here if they do not go to any execution port or if the counters are inaccurate. |
| μops each port: | The number of μ ops for each execution port. p0 means a μ op to execution port 0. p01 means a μ op that can go to either port 0 or port 1. p0 p1 means two μ ops going to port 0 and 1, respectively. Port 0: Integer, f.p. and 256 bit vector ALU, mul, div, branch Port 1: Integer, f.p. and 256 bit vector ALU (re-routed to port 0 for 512-bit vectors) Port 2: Load Port 3: Load Port 4: Store Port 5: Integer and 512 bit vector ALU Port 6: Integer ALU, branch Port 7: Store address |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Where hyperthreading is enabled, the use of the same execution units in the other thread leads to inferior performance. Denormal numbers, NAN's and infinity do not increase the latency. The time unit used is core clock cycles, not the reference clock cycles given by the time stamp counter. |
| Reciprocal throughput: | The average number of core clock cycles per instruction for a series of independent instructions of the same kind in the same thread. |

Integer instructions

| Instruction | Operands | μ ops fused domain | μ ops unfused domain | μ ops each port | Latency | Reciprocal throughput | Comments |
|--------------------------|---------------|------------------------|--------------------------|---------------------|---------|-----------------------|----------------------|
| Move instructions | | | | | | | |
| MOV | r,i | 1 | 1 | p0156 | | 0.25 | may eliminate |
| MOV | r8/16,r8/16 | 1 | 1 | p0156 | 1 | 0.25 | |
| MOV | r32/64,r32/64 | 1 | 1 | p0156 | 0-1 | 0.25 | |
| MOV | r8l,m | 1 | 2 | p23 p0156 | | 0.5 | all addressing modes |
| MOV | r8h,m | 1 | 1 | p23 | | 0.5 | |
| MOV | r16,m | 1 | 2 | p23 p0156 | | 0.5 | |
| MOV | r32/64,m | 1 | 1 | p23 | 2 | 0.5 | |
| MOV | m,r | 1 | 2 | p237 p4 | 2 | 1 | |
| MOV | m,i | 1 | 2 | p237 p4 | | 1 | |
| MOVNTI | m,r | 2 | 2 | p23 p4 | ~400 | 1 | |

SkyLakeX

| | | | | | | | | |
|-------------------------|---------------|----|----|-----------------|-----|-------|------------------------------|--|
| MOVSX MOVZX MOVSD | r,r | 1 | 1 | p0156 | 1 | 0.25 | all other combinations | |
| MOVSX MOVZX | r16,m8 | 1 | 2 | p23 p0156 | 2 | 0.5 | | |
| MOVSX MOVZX MOVSD | r,m | 1 | 1 | p23 | 2 | 0.5 | | |
| CMOVcc | r,r | 1 | 1 | p06 | 1 | 0.5 | | |
| CMOVcc | r,m | 1 | 2 | p06 p23 | | 0.5 | implicit lock | |
| XCHG | r,r | 3 | 3 | 3p0156 | 2 | 1 | | |
| XCHG | r,m | 8 | 8 | | 23 | | | |
| XLAT | | 3 | 4 | p23 2p0156 | 7 | 1 | | |
| PUSH | r | 1 | 2 | p237 p4 | 3 | 1 | not 64 bit | |
| PUSH | i | 1 | 2 | p237 p4 | | 1 | | |
| PUSH | m | 2 | 3 | p4 2p237 | | 1 | | |
| PUSH | stack pointer | 2 | 3 | p0156 p237 p4 | | 1 | | |
| PUSHF(D/Q) | | 3 | 4 | p1 p4 p237 p06 | | 1 | | |
| PUSHA(D) | | 11 | 19 | | | 8 | | |
| POP | r | 1 | 1 | p23 | 2 | 0.5 | | |
| POP | stack pointer | 3 | 3 | p23 2p0156 | | 3 | | |
| POP | m | 2 | 3 | 2p237 p4 | | 1 | | |
| POPF(D/Q) | | 9 | 9 | | | 20 | | |
| POPA(D) | | 18 | 18 | | | 8 | not 64 bit | |
| LAHF SAHF | | 1 | 1 | p06 | 1 | 1 | not 64 bit | |
| SALC | | 3 | 3 | 3p0156 | 1 | 1 | | |
| LEA | r16,m | 2 | 2 | p1 p05 | 2-4 | 1 | | |
| LEA | r32/64,m | 1 | 1 | p15 | 1 | 0.5 | 1 or 2 components in address | |
| LEA | r32/64,m | 1 | 1 | p1 | 3 | 1 | 3 components in address | |
| LEA | r32/64,m | 1 | 1 | p1 | | 1 | rip relative address | |
| BSWAP | r32 | 1 | 1 | p15 | 1 | 0.5 | | |
| BSWAP | r64 | 2 | 2 | p06 p15 | 2 | 1 | | |
| MOVBE | r16,m16 | 3 | 3 | 2p0156 p23 | | 0.5-1 | | |
| MOVBE | r32,m32 | 2 | 2 | p15 p23 | | 0.5 | | |
| MOVBE | r64,m64 | 3 | 3 | 2p0156 p23 | | 0.75 | | |
| MOVBE | m16,r16 | 2 | 3 | p06 p237 p4 | | 1 | | |
| MOVBE | m32,r32 | 2 | 3 | p15 p237 p4 | | 1 | | |
| MOVBE | m64,r64 | 3 | 4 | p06 p15 p237 p4 | | 1 | | |
| PREFETCHNTA/0/1/2 | m | 1 | 1 | p23 | | 0.5 | | |
| PREFETCHW | m | 1 | 1 | p23 | | 1 | | |
| LFENCE | | 2 | | none counted | | 4 | | |
| MFENCE | | 4 | 4 | p23 p4 | | 33 | | |
| SFENCE | | 2 | 2 | p23 p4 | | 6 | | |
| Arithmetic instructions | | | | | | | | |
| ADD SUB | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | | |
| ADD SUB | r,m | 1 | 2 | p0156 p23 | | 0.5 | | |
| ADD SUB | m,r/i | 2 | 4 | 2p0156 2p237 p4 | 5 | 1 | | |
| ADC SBB | r,r/i | 1 | 1 | p06 | 1 | 1 | | |

SkylakeX

| | | | | | | | |
|--------------------|-------------|----|----|-----------------|-------|-------|------------|
| ADC SBB | r,m | 1 | 2 | p06 p23 | | 1 | |
| ADC SBB | m,r/i | 4 | 6 | 3p0156 2p237 p4 | 5 | 2 | |
| ADCX ADOX | r,r | 1 | 1 | p06 | 1 | 1 | |
| CMP | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | |
| CMP | m,r/i | 1 | 2 | p0156 p23 | 1 | 0.5 | |
| INC DEC NEG NOT | r | 1 | 1 | p0156 | 1 | 0.25 | |
| INC DEC NOT | m | 3 | 4 | p0156 2p237 p4 | 5 | 1 | |
| NEG | m | 2 | 4 | p0156 2p237 p4 | 5 | 1 | |
| AAA | | 2 | 2 | p1 p56 | 4 | | not 64 bit |
| AAS | | 2 | 2 | p1 p056 | 4 | | not 64 bit |
| DAA DAS | | 3 | 3 | p1 2p056 | 4 | | not 64 bit |
| AAD | | 3 | 3 | p1 2p056 | 4 | | not 64 bit |
| AAM | | 11 | 11 | p0 p1 p5 p6 | 23 | 7 | not 64 bit |
| MUL IMUL | r8 | 1 | 1 | p1 | 3 | 1 | |
| MUL IMUL | r16 | 4 | 4 | p1 p0156 | 4 | 2 | |
| MUL IMUL | r32 | 3 | 3 | p1 p0156 | 4 | 1 | |
| MUL IMUL | r64 | 2 | 2 | p1 p6 | 3 | 1 | |
| MUL IMUL | m8 | 1 | 2 | p1 p23 | | 1 | |
| MUL IMUL | m16 | 4 | 5 | p1 3p0156 p23 | | 2 | |
| MUL IMUL | m32 | 3 | 4 | p1 2p0156 p23 | | 2 | |
| MUL IMUL | m64 | 2 | 3 | p1 p6 p23 | | 1 | |
| IMUL | r,r | 1 | 1 | p1 | 3 | 1 | |
| IMUL | r,m | 1 | 2 | p1 p23 | | 1 | |
| IMUL | r16,r16,i | 2 | 2 | p1 p0156 | 4 | 1 | |
| IMUL | r32,r32,i | 1 | 1 | p1 | 3 | 1 | |
| IMUL | r64,r64,i | 1 | 1 | p1 | 3 | 1 | |
| IMUL | r16,m16,i | 2 | 3 | p1 p0156 p23 | | 1 | |
| IMUL | r32,m32,i | 1 | 2 | p1 p23 | | 1 | |
| IMUL | r64,m64,i | 1 | 2 | p1 p23 | | 1 | |
| MULX | r32,r32,r32 | 3 | 3 | p1 2p056 | 4 | 1 | |
| MULX | r32,r32,m32 | 3 | 4 | p1 2p056 p23 | | 1 | |
| MULX | r64,r64,r64 | 2 | 2 | p1 p5 | 4 | 1 | |
| MULX | r64,r64,m64 | 2 | 3 | p1 p6 p23 | | 1 | |
| DIV | r8 | 10 | 10 | p0 p1 p5 p6 | 23 | 6 | |
| DIV | r16 | 10 | 10 | p0 p1 p5 p6 | 23 | 6 | |
| DIV | r32 | 10 | 10 | p0 p1 p5 p6 | 26 | 6 | |
| DIV | r64 | 36 | 36 | p0 p1 p5 p6 | 35-88 | 21-83 | |
| IDIV | r8 | 11 | 11 | p0 p1 p5 p6 | 24 | 6 | |
| IDIV | r16 | 10 | 10 | p0 p1 p5 p6 | 23 | 6 | |
| IDIV | r32 | 10 | 10 | p0 p1 p5 p6 | 26 | 6 | |
| IDIV | r64 | 57 | 57 | p0 p1 p5 p6 | 42-95 | 24-90 | |
| CBW | | 1 | 1 | p0156 | 1 | 0.5 | |
| CWDE | | 1 | 1 | p0156 | 1 | 1 | |
| CDQE | | 1 | 1 | p0156 | 1 | 0.5 | |
| CWD | | 2 | 2 | p0156 | 1 | 1 | |
| CDQ | | 1 | 1 | p06 | 1 | 1 | |
| CQO | | 1 | 1 | p06 | 1 | 0.5 | |
| POPCNT | r,r | 1 | 1 | p1 | 3 | 1 | |
| POPCNT | r,m | 1 | 2 | p1 p23 | | 1 | |
| CRC32 | r,r | 1 | 1 | p1 | 3 | 1 | |
| CRC32 | r,m | 1 | 2 | p1 p23 | | 1 | |
| Logic instructions | | | | | | | |

SkylakeX

| | | | | | | | |
|-----------------|--------|----|----|-----------------|---|------|------------|
| AND OR XOR | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | short form |
| AND OR XOR | r,m | 1 | 2 | p0156 p23 | | 0.5 | |
| AND OR XOR | m,r/i | 2 | 4 | 2p0156 2p237 p4 | 5 | 1 | |
| TEST | r,r/i | 1 | 1 | p0156 | 1 | 0.25 | |
| TEST | m,r/i | 1 | 2 | p0156 p23 | 1 | 0.5 | |
| SHR SHL SAR | r,i | 1 | 1 | p06 | 1 | 0.5 | |
| SHR SHL SAR | m,i | 3 | 4 | 2p06 p237 p4 | | 2 | |
| SHR SHL SAR | r,cl | 3 | 3 | 3p06 | 2 | 2 | |
| SHR SHL SAR | m,cl | 5 | 6 | 3p06 2p23 p4 | | 4 | |
| ROR ROL | r,1 | 2 | 2 | 2p06 | 1 | 1 | |
| ROR ROL | r,i | 1 | 1 | p06 | 1 | 0.5 | |
| ROR ROL | m,i | 4 | 5 | 2p06 2p237 p4 | | 2 | |
| ROR ROL | r,cl | 3 | 3 | 3p06 | 2 | 2 | |
| ROR ROL | m,cl | 5 | 6 | 3p06 p23 p4 | | 4 | |
| RCR RCL | r,1 | 3 | 3 | 2p06 p0156 | 2 | 2 | |
| RCR RCL | m,1 | 4 | 6 | | | 3 | |
| RCR RCL | r,i | 8 | 8 | p0156 | 6 | 6 | |
| RCR RCL | m,i | 11 | 11 | | | 6 | |
| RCR RCL | r,cl | 8 | 8 | p0156 | 6 | 6 | |
| RCR RCL | m,cl | 11 | 11 | | | 6 | |
| SHRD SHLD | r,r,i | 1 | 1 | p1 | 3 | 1 | |
| SHRD SHLD | m,r,i | 3 | 5 | | | 2 | |
| SHLD | r,r,cl | 4 | 4 | p0156 | 3 | 2 | |
| SHRD | r,r,cl | 4 | 4 | p0156 | 4 | 2 | |
| SHRD SHLD | m,r,cl | 5 | 7 | | | 4 | |
| SHLX SHR X SARX | r,r,r | 1 | 1 | p06 | 1 | 0.5 | |
| SHLX SHR X SARX | r,m,r | 2 | 2 | p06 p23 | | 0.5 | |
| RORX | r,r,i | 1 | 1 | p06 | 1 | 0.5 | |
| RORX | r,m,i | 2 | 2 | p06 p23 | | 0.5 | |
| BT | r,r/i | 1 | 1 | p06 | 1 | 0.5 | |
| BT | m,r | 10 | 10 | | | 5 | |
| BT | m,i | 2 | 2 | p06 p23 | | 0.5 | |
| BTR BTS BTC | r,r/i | 1 | 1 | p06 | 1 | 0.5 | |
| BTR BTS BTC | m,r | 10 | 11 | | | 5 | |
| BTR BTS BTC | m,i | 3 | 4 | p06 p4 p23 | | 1 | |
| BSF BSR | r,r | 1 | 1 | p1 | 3 | 1 | |
| BSF BSR | r,m | 1 | 2 | p1 p23 | | 1 | |
| SETcc | r | 1 | 1 | p06 | 1 | 0.5 | |
| SETcc | m | 2 | 3 | p06 p237 p4 | | 1 | |
| CLC | | 1 | 0 | none | | 0.25 | |
| STC | | 1 | 1 | p0156 | | 0.25 | |
| CMC | | 1 | 1 | p0156 | 1 | | |
| CLD STD | | 3 | 3 | p15 p6 | | 4 | |
| LZCNT | r,r | 1 | 1 | p1 | 3 | 1 | |
| LZCNT | r,m | 1 | 2 | p1 p23 | | 1 | |
| TZCNT | r,r | 1 | 1 | p1 | 3 | 1 | |
| TZCNT | r,m | 1 | 2 | p1 p23 | | 1 | |
| ANDN | r,r,r | 1 | 1 | p15 | 1 | 0.5 | |
| ANDN | r,r,m | 1 | 2 | p15 p23 | 1 | 0.5 | |
| BLSI BLSMSK | r,r | 1 | 1 | p15 | 1 | 0.5 | |
| BLSR | | | | | | | |
| BLSI BLSMSK | r,m | 1 | 2 | p15 p23 | | 0.5 | |
| BLSR | | | | | | | |

SkylakeX

| | | | | | | | |
|--------------------------------------|------------|---------|----|----------------|---|-------|-------------------------|
| BEXTR | r,r,r | 2 | 2 | 2p0156 | 2 | 0.5 | |
| BEXTR | r,m,r | 3 | 3 | 2p0156 p23 | | 1 | |
| BZHI | r,r,r | 1 | 1 | p15 | 1 | 0.5 | |
| BZHI | r,m,r | 1 | 2 | p15 p23 | | 0.5 | |
| PDEP | r,r,r | 1 | 1 | p1 | 3 | 1 | |
| PDEP | r,r,m | 1 | 2 | p1 p23 | | 1 | |
| PEXT | r,r,r | 1 | 1 | p1 | 3 | 1 | |
| PEXT | r,r,m | 1 | 2 | p1 p23 | | 1 | |
| Control transfer instructions | | | | | | | |
| JMP | short/near | 1 | 1 | p6 | | 1-2 | |
| JMP | r | 1 | 1 | p6 | | 2 | |
| JMP | m | 1 | 2 | p23 p6 | | 2 | |
| Conditional jump | short/near | 1 | 1 | p6 | | 1-2 | predicted taken |
| Conditional jump | short/near | 1 | 1 | p06 | | 0.5-1 | predicted not taken |
| Fused arithmetic and branch | | 1 | 1 | p6 | | 1-2 | predicted taken |
| Fused arithmetic and branch | | 1 | 1 | p06 | | 0.5-1 | predicted not taken |
| J(E/R)CXZ | short | 2 | 2 | p0156 p6 | | 0.5-2 | |
| LOOP | short | 7 | 7 | | | 4 | |
| LOOP(N)E | short | 10 | 10 | | | 6 | |
| CALL | near | 2 | 3 | p237 p4 p6 | | 3 | |
| CALL | r | 2 | 3 | p237 p4 p6 | | 3 | |
| CALL | m | 3 | 4 | 2p237 p4 p6 | | 4 | |
| RET | | 1 | 2 | p237 p6 | | 1 | |
| RET | i | | 2 | | | 1 | |
| BOUND | r,m | 15 | 15 | | | 8 | not 64 bit |
| INTO | | 5 | 5 | | | 6 | not 64 bit |
| String instructions | | | | | | | |
| LODSB/W | | 3 | 3 | 2p0156 p23 | | 1 | |
| LODSD/Q | | 2 | 2 | p0156 p23 | | 1 | |
| REP LODS | | 5n+12 | | | | ~2n | |
| STOS | | 3 | 3 | p23 p0156 p4 | | 1 | |
| REP STOS | | <2n | | | | ~0.5n | worst case |
| REP STOS | | 2.6/32B | | | | 1/32B | best case aligned by 32 |
| MOVS | | 5 | 5 | 2p23 p4 2p0156 | | 4 | |
| REP MOVS | | ~2n | | | | < 1n | worst case |
| REP MOVS | | 4/32B | | | | 1/32B | best case aligned by 32 |
| SCAS | | 3 | 3 | p23 2p0156 | | 1 | |
| REP SCAS | | ≥8n | | | | ≥n | |
| CMPS | | 5 | 5 | 2p23 3p0156 | | 4 | |
| REP CMPS | | ≥8n | | | | ≥2n | |
| Synchronization instructions | | | | | | | |
| XADD | m,r | 4 | 5 | | | 5 | |
| LOCK XADD | m,r | 9 | 9 | | | 18 | |
| LOCK ADD | m,r | 8 | 8 | | | 18 | |
| CMPXCHG | m,r | 5 | 6 | | | 5 | |
| LOCK CMPXCHG | m,r | 10 | 10 | | | 18 | |

SkyLakeX

| | | | | | | | |
|-----------------|-----|--------|--------|-------------|--------|---------|--|
| CMPXCHG8B | m,r | 16 | 16 | | | 11 | |
| LOCK CMPXCHG8B | m,r | 20 | 20 | | | 19 | |
| CMPXCHG16B | m,r | 23 | 23 | | | 16 | |
| LOCK CMPXCHG16B | m,r | 25 | 25 | | | 26 | |
| Other | | | | | | | |
| NOP | | 1 | 0 | none | | 0.25 | |
| Long NOP | | 1 | 0 | none | | 0.25 | |
| PAUSE | | 4 | 4 | p6 | | 141 | |
| ENTER | a,0 | 12 | 12 | | | 8 | |
| ENTER | a,b | ~14+7b | ~45+7b | | ~87+2b | | |
| LEAVE | | 3 | 3 | 2p0156 p23 | | 4 | |
| XGETBV | | 15 | 15 | | | 7 | |
| CPUID | | 27-118 | | | | 100-250 | |
| RDTSC | | 20 | 20 | | | 25 | |
| RDTSCP | | 22 | 22 | | | 32 | |
| RDPMC | | 33 | 33 | | | 16 | |
| RDRAND | r | 16 | 16 | p23 15p0156 | | ~200 | |
| RDSEED | r | 16 | 16 | p23 15p0156 | | ~200 | |

Floating point x87 instructions

| Instruction | Operands | μops fused domain | μops unfused domain | μops each port | Latency | Reciprocal through put | Comments |
|--------------------------------|----------|-------------------|---------------------|-----------------|---------|------------------------|----------|
| Move instructions | | | | | | | |
| FLD | r | 1 | 1 | p05 | 1 | 0.5 | |
| FLD | m32/64 | 1 | 1 | p23 | 3 | 0.5 | |
| FLD | m80 | 4 | 4 | 2p01 2p23 | 4 | 2 | |
| FBLD | m80 | 43 | 43 | | 46 | 22 | |
| FST(P) | r | 1 | 1 | p05 | 1 | 0.5 | |
| FST(P) | m32/m64 | 1 | 2 | p4 p237 | 3 | 1 | |
| FSTP | m80 | 7 | 7 | 3p0156 2p23 2p4 | 4 | 5 | |
| FBSTP | m80 | 244 | 226 | | 264 | 266 | |
| FXCH | r | 2 | 0 | none | 0 | 0.5 | |
| FILD | m | 1 | 2 | p05 p23 | 5 | 1 | |
| FIST(P) | m | 3 | 3 | p5 p23 p4 | 7 | 1 | |
| FISTTP | m | 3 | 3 | p1 p23 p4 | 7 | 2 | |
| FLDZ | | 1 | 1 | p05 | | 1 | |
| FLD1 | | 2 | 2 | 2p05 | | 2 | |
| FLDPI FLDL2E etc. | | 2 | 2 | 2p05 | | 2 | |
| FCMOVcc | r | 4 | 4 | p0 p1 p56 | 3 | 2 | |
| FNSTSW | AX | 2 | 2 | p0 p0156 | 6 | 1 | |
| FNSTSW | m16 | 2 | 3 | p0 p4 p237 | 6 | 1 | |
| FLDCW | m16 | 3 | 3 | p01 p23 p6 | 7 | 2 | |
| FNSTCW | m16 | 2 | 3 | p237 p4 p6 | 6 | 1 | |
| FINCSTP FDECSTP | | 1 | 1 | p05 | 0 | 0.5 | |
| FFREE(P) | r | 1 | 1 | p05 | | 0.5 | |
| FNSAVE | m | 133 | 133 | | 176 | 176 | |
| FRSTOR | m | 89 | 89 | | 175 | 175 | |
| Arithmetic instructions | | | | | | | |

SkylakeX

| | | | | | | | |
|----------------|---|--------|----|-----------|---------|-----|--|
| FADD(P) | | | | | | | |
| FSUB(R)(P) | r | 1 | 1 | p5 | 3 | 1 | |
| FADD(P) | | | | | | | |
| FSUB(R)(P) | m | 2 | 3 | p5 p23 | | 1 | |
| FMUL(P) | r | 1 | 1 | p0 | 5 | 1 | |
| FMUL(P) | m | 2 | 3 | p0 p23 | | 1 | |
| FDIV(R)(P) | r | 1 | 1 | p0 | 14-16 | 4-5 | |
| FDIV(R)(P) | m | 1 | 2 | p0 p23 | | 4-5 | |
| FABS | | 1 | 1 | p0 | 1 | 1 | |
| FCHS | | 1 | 1 | p0 | 1 | 1 | |
| FCOM(P) FUCOM | r | 1 | 1 | p5 | 3 | 1 | |
| FCOM(P) FUCOM | m | 1 | 2 | p5 p23 | | 1 | |
| FCOMPP FUCOMPP | | 2 | 2 | p0 p5 | | 1 | |
| FCOMI(P) | | | | | | | |
| FUCOMI(P) | r | 3 | 3 | p5 | 4 | 1 | |
| FIADD FISUB(R) | m | 3 | 4 | 2p5 p23 | | 2 | |
| FIMUL | m | 3 | 4 | p0 p5 p23 | | 1 | |
| FIDIV(R) | m | 2 | 3 | p0 p5 p23 | | | |
| FICOM(P) | m | 2 | 3 | 2p5 p23 | | 2 | |
| FTST | | 1 | 1 | p5 | 3 | 1 | |
| FXAM | | 2 | 2 | 2p5 | 6 | 2 | |
| FPREM | | 31 | 31 | | 26-30 | 17 | |
| FPREM1 | | 31 | 31 | | 30-57 | 17 | |
| FRNDINT | | 17 | 17 | | 21 | 11 | |
| Math | | | | | | | |
| FSCALE | | 27 | 27 | | 11 | 11 | |
| FEXTRACT | | 17 | 17 | | 11 | 11 | |
| FSQRT | | 1 | 1 | p0 | 14-21 | 4-7 | |
| FSIN | | 53-105 | | | 50-120 | | |
| FCOS | | 53-105 | | | 50-130 | | |
| FSINCOS | | 55-120 | | | 55-150 | | |
| F2XM1 | | 16-90 | | | 65-80 | | |
| FYL2X | | 40-100 | | | 103 | | |
| FYL2XP1 | | 56 | | | 77 | | |
| FPTAN | | 40-112 | | | 140-160 | | |
| FPATAN | | 30-160 | | | 100-160 | | |
| Other | | | | | | | |
| FNOP | | 1 | 1 | p05 | | 0.5 | |
| WAIT | | 2 | 2 | p05 | | 1 | |
| FNCLEX | | 5 | 5 | p156 | | 22 | |
| FNINIT | | 18 | 18 | | | 78 | |

Integer vector instructions

| Instruction | Operands | μops fused domain | μops unfused domain | μops each port | Latency | Reciprocal throughput | Comments |
|--------------------------|-------------|-------------------|---------------------|----------------|---------|-----------------------|----------|
| Move instructions | | | | | | | |
| MOVD | r32/64,mm/x | 1 | 1 | p0 | 2 | 1 | |
| MOVD | m32/64,mm/x | 1 | 2 | p237 p4 | 3 | 1 | |
| MOVD | mm/x,r32/64 | 1 | 1 | p5 | 2 | 1 | |

SkylakeX

| | | | | | | | |
|------------------|-------------|---|---|----------|------|------|---------------|
| MOVD | mm/x,m32/64 | 1 | 1 | p23 | 2 | 0.5 | |
| MOVQ | r64,mm/x | 1 | 1 | p0 | 2 | 1 | |
| MOVQ | mm/x,r64 | 1 | 1 | p5 | 1 | 1 | |
| MOVQ | mm,mm | 1 | | p05 | 1 | 0.5 | |
| MOVQ | x,x | 1 | | p015 | 1 | 0.33 | |
| MOVQ | mm/x,m64 | 1 | 1 | p23 | 2 | 0.5 | |
| MOVQ | m64, mm/x | 1 | 2 | p237 p4 | 3 | 1 | |
| MOVDQA/U | x,x | 1 | 1 | p015 | 0-1 | 0.25 | may eliminate |
| MOVDQA/U | x, m128 | 1 | 1 | p23 | 2 | 0.5 | |
| MOVDQA/U | m128, x | 1 | 2 | p237 p4 | 3 | 1 | |
| VMOVDQA/U | y,y | 1 | 1 | p015 | 0-1 | 0.25 | may eliminate |
| VMOVDQA/U | z,z | 1 | 1 | p05 | 0-1 | 0.33 | may eliminate |
| VMOVDQA32/64 | xy{k},xy | 1 | 1 | p015 | 1 | 0.33 | |
| VMOVDQA32/64 | z{k},z | 1 | 1 | p05 | 1 | 0.5 | |
| VMOVDQA/U/32/64 | yz,m | 1 | 1 | p23 | 3 | 0.5 | |
| VMOVDQA/U | /8/16/32/64 | 1 | 2 | p05 p23 | 4 | 0.5 | |
| VMOVDQA/U/32/64 | m,v | 1 | 2 | p237 p4 | 3 | 1 | |
| VMOVDQA/U | /8/16/32/64 | 1 | 2 | | 14 | 1 | |
| LDDQU | x, m128 | 1 | 1 | p23 | 3 | 0.5 | |
| MOVDQ2Q | mm, x | 2 | 2 | p0 p5 | 2 | 1 | |
| MOVQ2DQ | x,mm | 2 | 2 | p0 p15 | 2 | 1 | |
| MOVNTQ | m64,mm | 1 | 2 | p237 p4 | ~600 | 1 | |
| MOVNTDQ | m128,x | 1 | 2 | p237 p4 | ~700 | 1 | |
| VMOVNTDQ | m256,y | 1 | 2 | p237 p4 | ~700 | 1 | |
| MOVNTDQA | v,m | 2 | 2 | p23 p015 | 3 | 0.5 | |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | mm,mm | 3 | 3 | p5 | 2 | 2 | |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | mm,m64 | 3 | 3 | p23 2p5 | | 2 | |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | x,x / v,v,v | 1 | 1 | p5 | 1 | 1 | |
| PACKSSWB/DW | | | | | | | |
| PACKUSWB | x,m / y,y,m | 1 | 2 | p23 p5 | | 1 | |
| PACKUSDW | x,x / v,v,v | 1 | 1 | p5 | 1 | 1 | |
| PACKUSDW | x,m / v,v,m | 1 | 2 | p23 p5 | | 1 | |
| PUNPCKH/L | | | | | | | |
| BW/WD/DQ | v,v / v,v,v | 1 | 1 | p5 | 1 | 1 | |
| PUNPCKH/L | | | | | | | |
| BW/WD/DQ | v,m / v,v,m | 1 | 2 | p23 p5 | | 1 | |
| PUNPCKH/L | | | | | | | |
| QDQ | x,x / y,y,y | 1 | 1 | p5 | 1 | 1 | |
| PUNPCKH/L | | | | | | | |
| QDQ | x,m / y,y,m | 1 | 2 | p23 p5 | | 1 | |
| PMOVSX/ZX BW BD | | | | | | | |
| BQ DW DQ | x,x | 1 | 1 | p5 | 1 | 1 | |
| PMOVSX/ZX BW BD | | | | | | | |
| BQ DW DQ | x,m | 1 | 2 | p23 p5 | | 1 | |
| VPMOVSX/ZX BW BD | | | | | | | |
| BQ DW DQ | y,x | 1 | 1 | p5 | 3 | 1 | |
| VPMOVSX/ZX BW BD | | | | | | | |
| BQ DW DQ | z,xy | 1 | 1 | p5 | 3 | 1 | |

SkylakeX

| | | | | | | |
|-------------------------------|-----------------|-----|----|----------------|---------|------|
| VPMOVSX/ZX BW BD BQ DW DQ | y,m | 2 | 2 | p5 p23 | | 1 |
| VPMOVWB/DB/QB/ DW/QW/QD | | | | | | |
| WPMOVSWB/DB/QB/ DW/QW/QD | | | | | | |
| WPMOVUSWB/DB/QB/ /DW/QW/QD | v,v | 2 | 2 | 2p5 | 4 | 2 |
| PSHUFB | mm,mm | 3 | 3 | p0 p5 | 3 | 1 |
| PSHUFB | x,x / v,v,v | 1 | 1 | p5 | 1 | 1 |
| PSHUFB | v,m / v,v,m | 2 | 2 | p23 p5 | | 1 |
| PSHUFW | mm,mm,i | 1 | 1 | p5 | 1 | 1 |
| PSHUFW | mm,m64,i | 2 | 2 | p23 p5 | | 1 |
| PSHUFD | v,v,i | 1 | 1 | p5 | 1 | 1 |
| PSHUFD | v,m,i | 1-2 | 2 | p23 p5 | | 1 |
| PSHUFL/HW | v,v,i | 1 | 1 | p5 | 1 | 1 |
| PSHUFL/HW | v,m,i | 2 | 2 | p23 p5 | | 1 |
| VSHUFI32X4 | v,v,v,i | 1 | 1 | p5 | 3 | 1 |
| VSHUFI64X2 | v,v,v,i | 1 | 1 | p5 | 3 | 1 |
| PALIGNR | v,v,i / v,v,v,i | 1 | 1 | p5 | 1 | 1 |
| PALIGNR | v,m,i / v,v,m,i | 2 | 2 | p23 p5 | | 1 |
| VALIGND/Q | v,v,v,i | 1 | 1 | p5 | 3 | 1 |
| VALIGND/Q | v,v,m,i | 2 | 2 | p23 p5 | | 1 |
| PBLENDVB | x,x,xmm0 | 1 | 1 | p015 | 1 | 0.5 |
| PBLENDVB | x,m,xmm0 | 2 | 2 | p015 p23 | | 1 |
| VPBLENDVB | v,v,v,v | 2 | 2 | 2p015 | 2 | 1 |
| VPBLENDVB | v,v,m,v | 3 | 3 | 2p015 p23 | | 2 |
| PBLENDW | x,x,i / v,v,v,i | 1 | 1 | p5 | 1 | 1 |
| PBLENDW | x,m,i / v,v,m,i | 2 | 2 | p23 p5 | | 1 |
| VPBLEND | v,v,v,i | 1 | 1 | p015 | 1 | 0.33 |
| VPBLEND/Q | v,v,m,i | 2 | 2 | p015 p23 | | 0.5 |
| VPBLENDMB/W | v{k},v,v | 1 | 1 | p05 | 3 | 0.5 |
| VPBLENDMD/Q | v{k},v,v | 1 | 1 | p05 | 1-2 | 0.5 |
| VPERMW | v,v,v | 2 | 2 | p5 | 4 | 2 |
| VPERMW | v,v,m | 3 | 3 | | | 2 |
| VPERMD | y,y,y | 1 | 1 | p5 | 3 | 1 |
| VPERMD | y,y,m | 1 | 2 | p5 p23 | | 1 |
| VPERMQ | v,v,i | 1 | 1 | p5 | 3 | 1 |
| VPERMQ | v,m,i | 2 | 2 | p5 p23 | | 1 |
| VPERMQ | v,v,v | 1 | 1 | p5 | 3 | 1 |
| VPERMQ | v,v,m | 1 | 2 | p5 p23 | | 1 |
| VPERM2128 | y,y,y,i | 1 | 1 | p5 | 3 | 1 |
| VPERM2128 | y,y,m,i | 2 | 2 | p5 p23 | | 1 |
| VPERMI2W | v,v,v | 3 | 3 | p01 2p5 | 7 | 2 |
| VPERMI2D | v,v,v | 1 | 1 | p5 | 3 | 1 |
| VPERMI2Q | v,v,v | 1 | 1 | p5 | 3 | 1 |
| VPERMT2W | v,v,v | 3 | 3 | p01 2p5 | 6 | 2 |
| VPERMT2D | v,v,v | 1 | 1 | p5 | 3 | 1 |
| VPERMT2Q | v,v,v | 1 | 1 | p5 | 3 | 1 |
| MASKMOVQ | mm,mm | 4 | 4 | p0 p4 2p23 | ~300 | 2 |
| MASKMOVDQU | x,x | 10 | 10 | 4p04 2p56 4p23 | 300-800 | 6 |
| VPMASKMOVD/Q | v,v,m | 2 | 2 | p23 p015 | 4 | 0.5 |
| VPMASKMOVD/Q | m,v,v | 3 | 3 | p0 p4 p23 | 4 | 1 |
| PMOVMSKB | r,v | 1 | 1 | p0 | 2-3 | 1 |

SkylakeX

| | | | | | | |
|------------------------|--------------|----|----|--------------|-------|-----|
| PEXTRB/W/D/Q | r,x,i | 2 | 2 | p0 p5 | 3 | 1 |
| PEXTRB/W/D/Q | m,x,i | 2 | 3 | p23 p4 p5 | | 1 |
| VEXTRACTI128 | x,y,i | 1 | 1 | p5 | 3 | 1 |
| VEXTRACTI128 | m,y,i | 2 | 2 | p23 p4 | 4 | 1 |
| PINSRB | x,r32,i | 2 | 2 | 2p5 | 3 | 2 |
| PINSRB | x,m8,i | 2 | 2 | p23 p5 | | 1 |
| PINSRW | mm/x,r32,i | 2 | 2 | p5 | 3 | 2 |
| PINSRW | mm/x,m16,i | 2 | 2 | p23 p5 | | 1 |
| PINSRD/Q | x,r,i | 2 | 2 | 2p5 | 3 | 2 |
| PINSRD/Q | x,m,i | 2 | 2 | p23 p5 | | 1 |
| VINSERTI128 | y,y,x,i | 1 | 1 | p5 | 3 | 1 |
| VINSERTI128 | y,y,m,i | 2 | 2 | p015 p23 | 4-5 | 0.5 |
| VINSERTI32x4 | z,z,x,i | 1 | 1 | p5 | 3 | 1 |
| VINSERTI32x4 | z,z,m,i | 1 | 2 | p05 p23 | 4-5 | 0.5 |
| VINSERTI64x4 | z,z,y,i | 1 | 1 | p5 | 3 | 1 |
| VINSERTI64x4 | z,z,m,i | 1 | 2 | p05 p23 | 4-5 | 0.5 |
| VPBROADCAST B/W/D/Q | x,x | 1 | 1 | p5 | 1 | 1 |
| VPBROADCAST B/W | v,m | 2 | 2 | p23 p5 | | 1 |
| VPBROADCAST D/Q | v,m | 1 | 1 | p23 | | 0.5 |
| VPBROADCAST B/W/D/Q | v,x | 1 | 1 | p5 | 3 | 1 |
| VBROADCASTI128 | y,m128 | 1 | 1 | p23 | 3 | 0.5 |
| VBROADCASTI32X2 | z,x,i | 1 | 1 | p5 | 3 | 1 |
| VBROADCASTI32X4/I64x2 | z,m128,i | 1 | 1 | p23 | 4 | 0.5 |
| VBROADCASTI32X8/I64x4 | z,m256,i | 1 | 1 | p23 | 4 | 0.5 |
| VPCOMPRESSD/Q | v{k},v | 2 | 2 | p5 | 3 | 2 |
| VPEXPANDD/Q | v{k},v | 2 | 2 | p5 | 3 | 2 |
| VPGATHERDD | x,[r+s*x],x | 4 | 4 | p0 p1 p23 p5 | | 4 |
| VPGATHERDD | y,[r+s*y],y | 4 | 4 | p0 p1 p23 p5 | | 5 |
| VPGATHERDQ | x,[r+s*x],x | 4 | 5 | p0 p1 p23 p5 | | 2 |
| VPGATHERDQ | y,[r+s*x],y | 4 | 4 | p0 p1 p23 p5 | | 4 |
| VPGATHERQD | x,[r+s*x],x | 4 | 5 | p0 p1 p23 p5 | | 2 |
| VPGATHERQD | x,[r+s*y],x | 4 | 4 | p0 p1 p23 p5 | | 4 |
| VPGATHERQQ | x,[r+s*x],x | 4 | 5 | p0 p1 p23 p5 | | 2 |
| VPGATHERQQ | y,[r+s*y],y | 4 | 4 | p0 p1 p23 p5 | | 4 |
| VPGATHERDD | x{k},[r+s*x] | 4 | 4 | p0 p1 p23 p5 | | 4 |
| VPGATHERDD | y{k},[r+s*y] | 4 | 4 | p0 p1 p23 p5 | | 5 |
| VPGATHERDD | z{k},[r+s*z] | 4 | 4 | p0 p1 p23 p5 | | 9 |
| VPGATHERDQ | x{k},[r+s*x] | 4 | 5 | p0 p1 p23 p5 | | 3 |
| VPGATHERDQ | y{k},[r+s*x] | 4 | 4 | p0 p1 p23 p5 | | 4 |
| VPGATHERDQ | z{k},[r+s*y] | 4 | 4 | p0 p1 p23 p5 | | 5 |
| VPGATHERQD | x{k},[r+s*x] | 4 | 5 | p0 p1 p23 p5 | | 3 |
| VPGATHERQD | x{k},[r+s*y] | 4 | 4 | p0 p1 p23 p5 | | 4 |
| VPGATHERQD | y{k},[r+s*z] | 4 | 4 | p0 p1 p23 p5 | | 5 |
| VPGATHERQQ | x{k},[r+s*x] | 4 | 5 | p0 p1 p23 p5 | | 3 |
| VPGATHERQQ | y{k},[r+s*y] | 4 | 4 | p0 p1 p23 p5 | | 4 |
| VPGATHERQQ | z{k},[r+s*z] | 4 | 4 | p0 p1 p23 p5 | | 5 |
| VPSCATTERDD | [r+s*x]{k},x | 19 | 19 | | | 10 |
| VPSCATTERDD | [r+s*y]{k},y | 27 | 27 | | | 12 |
| VPSCATTERDD | [r+s*z]{k},z | 43 | 43 | | 17-20 | 17 |
| VPSCATTERDQ | [r+s*x]{k},x | 15 | 15 | | | 9 |

SkylakeX

| | | | | | | |
|--------------------------------|----------------|----|----|-------------|----|-------|
| VPSCATTERDQ | [r+s*x]{k},y | 19 | 19 | | | 10 |
| VPSCATTERDQ | [r+s*y]{k},z | 27 | 27 | | | 12 |
| VPSCATTERQD | [r+s*x]{k},x | 15 | 15 | | | 9 |
| VPSCATTERQD | [r+s*y]{k},y | 19 | 19 | | | 10 |
| VPSCATTERQD | [r+s*z]{k},z | 27 | 27 | | | 12 |
| VPSCATTERQQ | [r+s*x]{k},x | 15 | 15 | | | 9 |
| VPSCATTERQQ | [r+s*y]{k},y | 19 | 9 | | | 10 |
| VPSCATTERQQ | [r+s*z]{k},z | 27 | 27 | | | 12 |
| Arithmetic instructions | | | | | | |
| PADD/SUB(S,US) B/ W/D/Q | mm,mm | 1 | 1 | p05 | 1 | 0.5 |
| PADD/SUB(S,US) B/ W/D/Q | x,x / xy,xy,xy | 1 | 1 | p015 | 1 | 0.33 |
| PADD/SUB(S,US) B/ W/D/Q | z,z,z | 1 | 1 | p05 | 1 | 0.5 |
| PADD/SUB(S,US) B/ W/D/Q | v,m / v,v,m | 1 | 2 | p015 p23 | | 0.5 |
| PHADD(S)W/D | v,v / v,v,v | 3 | 3 | p01 2p5 | 3 | 2 |
| PHSUB(S)W/D | v,m / v,v,m | 4 | 4 | p01 2p5 p23 | | 2 |
| PCMPEQB/W/D | mm,mm | 1 | 1 | p0 | 1 | 1 |
| PCMPGTB/W/D | x,x / y,y,y | 1 | 1 | p01 | 1 | 0.5 |
| PCMPEQB/W/D | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 |
| PCMPGTB/W/D | v,v / v,v,v | 1 | 1 | p01 | 1 | 0.5 |
| PCMPEQQ | v,m / v,v,m | 1 | 2 | p01 p23 | | 0.5 |
| PCMPGTQ | v,v / v,v,v | 1 | 1 | p5 | 3 | 1 |
| PCMPGTQ | v,m / v,v,m | 1 | 2 | p5 p23 | | 1 |
| VPCMP..B/W/D/Q | k{k},v,v | 1 | 1 | p5 | 3 | 1 |
| PMULL/HW | mm,mm | 1 | 1 | p0 | 5 | 1 |
| PMULHUW | x,x / y,y,y | 1 | 1 | p01 | 5 | 0.5 |
| VPMULL/HW | z,z,z | 1 | 1 | p0 | 5 | 1 |
| PMULL/HW | x,m / xy,xy,m | 1 | 2 | p01 p23 | | 0.5 |
| PMULLQ | v,v,v | 3 | 3 | p01 | 15 | 1.5-3 |
| PMULHRSW | mm,mm | 1 | 1 | p0 | 5 | 1 |
| PMULHRSW | x,x / y,y,y | 1 | 1 | p01 | 5 | 0.5 |
| PMULHRSW | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 |
| PMULLD | x,x / y,y,y | 2 | 2 | 2p01 | 10 | 1 |
| PMULLD | z,z,z | 2 | 2 | 2p0 | 10 | 2 |
| PMULLD | x,m / y,y,m | 3 | 3 | 2p01 p23 | | 1 |
| PMULDQ | x,x / y,y,y | 1 | 1 | p01 | 5 | 0.5 |
| PMULDQ | z,z,z | 2 | 2 | 2p0 | 10 | 2 |
| PMULDQ | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 |
| PMULUDQ | mm,mm | 1 | 1 | p0 | 5 | 1 |
| PMULUDQ | x,x / y,y,y | 1 | 1 | p01 | 5 | 0.5 |
| PMULUDQ | z,z,z | 1 | 1 | p0 | 5 | 1 |

SkylakeX

| | | | | | | |
|-------------------------------------|-----------------|---|---|-----------|---|------|
| PMULUDQ | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 |
| PMADDWD | mm,mm | 1 | 1 | p0 | 5 | 1 |
| PMADDWD | x,x / y,y,y | 1 | 1 | p01 | 5 | 0.5 |
| PMADDWD | z,z,z | 1 | 1 | p0 | 5 | 1 |
| PMADDWD | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 |
| PMADDUBSW | mm,mm | 1 | 1 | p0 | 5 | 1 |
| PMADDUBSW | x,x / y,y,y | 1 | 1 | p01 | 5 | 0.5 |
| PMADDUBSW | z,z,z | 1 | 1 | p0 | 5 | 1 |
| PMADDUBSW | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 |
| PAVGB/W | mm,mm | 1 | 1 | p0 | 1 | 1 |
| PAVGB/W | x,x / y,y,y | 1 | 1 | p01 | 1 | 0.5 |
| PAVGB/W | z,z,z | 1 | 1 | p0 | 1 | 1 |
| PAVGB/W | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 |
| PMIN/PMAX SB/SW/SD UB/UW/UD | mm,mm | 1 | 1 | p0 | 1 | 1 |
| PMIN/PMAX SB/SW/SD UB/UW/UD | x,x / y,y,y | 1 | 1 | p01 | 1 | 0.5 |
| PMIN/PMAX SB/SW/SD UB/UW/UD | z,z,z | 1 | 1 | p0 | 1 | 1 |
| PMIN/PMAX SB/SW/SD UB/UW/UD | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 |
| PHMINPOSUW | x,x | 1 | 1 | p0 | 4 | 1 |
| PHMINPOSUW | x,m128 | 1 | 2 | p0 p23 | | 1 |
| PABSB/W/D | mm,mm | 1 | 1 | p0 | 1 | 1 |
| PABSB/W/D | x,x / y,y | 1 | 1 | p01 | 1 | 0.5 |
| PABSB/W/D | z,z,z | 1 | 1 | p0 | 1 | 1 |
| PABSB/W/D | x,m / y,m | 1 | 2 | p01 p23 | | 0.5 |
| PSIGNB/W/D | mm,mm | 1 | 1 | p0 | 1 | 1 |
| PSIGNB/W/D | x,x / y,y,y | 1 | 1 | p01 | 1 | 0.5 |
| PSIGNB/W/D | x,m / y,y,m | 1 | 2 | p01 p23 | | 0.5 |
| PSADBW | v,v / v,v,v | 1 | 1 | p5 | 3 | 1 |
| PSADBW | v,m / v,v,m | 1 | 2 | p5 p23 | | 1 |
| MPSADBW | x,x,i / v,v,v,i | 2 | 2 | 2p5 | 4 | 2 |
| MPSADBW | x,m,i / v,v,m,i | 3 | 3 | 2p5 p23 | | 2 |
| Logic instructions | | | | | | |
| PAND PANDN POR PXOR | mm,mm | 1 | 1 | p05 | 1 | 0.5 |
| PAND PANDN POR PXOR | x,x / y,y,y | 1 | 1 | p015 | 1 | 0.33 |
| PAND PANDN POR PXOR | z,z,z | 1 | 1 | p05 | 1 | 0.5 |
| PAND PANDN POR PXOR | v,m / v,v,m | 1 | 2 | p015 p23 | | 0.5 |
| PTEST | v,v | 2 | 2 | p0 p5 | 3 | 1 |
| PTEST | v,m | 2 | 3 | p0 p5 p23 | | 1 |
| VPTESTMB/W/D/Q VPTESTNMB/W/D/Q | k,v,v | 1 | 1 | | 3 | 1 |
| PSLLW/D/Q PSRLW/D/Q PSRAW/D/Q | mm,mm | 1 | 1 | p0 | 1 | 1 |

SkylakeX

| | | | | | | |
|--------------------------------|-------------|----|----|--------------|----|-----|
| PSLLW/D/Q | | | | | | |
| PSRLW/D/Q | | | | | | |
| PSRAW/D/Q | x,x / v,v,x | 2 | 2 | p01 p5 | 1 | 1 |
| PSLLW/D/Q | | | | | | |
| PSRLW/D/Q | | | | | | |
| PSRAW/D/Q | x,m / v,v,m | 2 | 2 | p01 p23 | | 0.5 |
| PSLLW/D/Q | | | | | | |
| PSRLW/D/Q | | | | | | |
| PSRAW/D/Q | mm,i | 1 | 1 | p0 | 1 | 1 |
| PSLLW/D/Q | | | | | | |
| PSRLW/D/Q | | | | | | |
| PSRAW/D/Q | x,i / y,y,i | 1 | 1 | p01 | 1 | 0.5 |
| VPSLLW/D/Q | | | | | | |
| VPSRLW/D/Q | | | | | | |
| VPSRAW/D/Q | z,z,i | 1 | 1 | p0 | 1 | |
| VPSLLW/D/Q | | | | | | |
| VPSRLW/D/Q | | | | | | |
| VPSRAW/D/Q | v,v,x | 2 | 2 | p01/05 | 2 | 1 |
| VPSLLVD/Q | | | | | | |
| VPSRAVD | | | | | | |
| VPSRLVD/Q | xy,xy,xy | 1 | 1 | p01 | 1 | 0.5 |
| VPSLLVW/D/Q | | | | | | |
| VPSRAVW/D/Q | | | | | | |
| VPSRLVW/D/Q | z,z,z | 1 | 1 | p0 | 1 | 1 |
| PSLLDQ | | | | | | |
| PSRLDQ | x,i / v,v,i | 1 | 1 | p5 | 1 | 1 |
| VPROL/RD/Q | xy,xy,i | 1 | 1 | p01 | 1 | 0.5 |
| VPROL/RD/Q | z,z,i | 1 | 1 | p0 | 1 | 1 |
| VPROL/RVD/Q | xy,xy,xy | 1 | 1 | p01 | 1 | 0.5 |
| VPROL/RVD/Q | z,z,z | 1 | 1 | p0 | 1 | 1 |
| VPLZCNTD/Q | xy,xy | 1 | 1 | p01 | 4 | 0.5 |
| VPLZCNTD/Q | v,v | 1 | 1 | p0 | 4 | 1 |
| VPTERNLOGD/Q | v,v,v | 1 | 1 | p05 | 1 | 0.5 |
| VPCONFLICTD | x,x | 15 | 15 | p01 p5 | 12 | 7 |
| VPCONFLICTD | y,y | 22 | 22 | p01 p5 | 17 | 10 |
| VPCONFLICTD | z,z | 37 | 37 | p0 p5 | 26 | 20 |
| VPCONFLICTQ | x,x | 3 | 3 | p01 p5 | 4 | 2 |
| VPCONFLICTQ | y,y | 15 | 15 | p01 p5 | 13 | 7 |
| VPCONFLICTQ | z,z | 22 | 22 | p0 p5 | 17 | 12 |
| String instructions | | | | | | |
| PCMPESTRI | x,x,i | 8 | 8 | 6p05 2p16 | 12 | 4 |
| PCMPESTRI | x,m128,i | 8 | 8 | | | 4 |
| PCMPESTRM | x,x,i | 9 | 9 | 3p0 2p16 4p5 | 9 | 5 |
| PCMPESTRM | x,m128,i | 9 | 9 | | | 5 |
| PCMPISTRI | x,x,i | 3 | 3 | 3p0 | 10 | 3 |
| PCMPISTRI | x,m128,i | 4 | 4 | | | 3 |
| PCMPISTRM | x,x,i | 3 | 3 | 3p0 | 9 | 3 |
| PCMPISTRM | x,m128,i | 4 | 4 | | | 3 |
| Encryption instructions | | | | | | |
| PCLMULQDQ | x,x,i | 1 | 1 | p5 | 7 | 1 |
| PCLMULQDQ | x,m,i | 2 | 2 | p5 p23 | | 1 |

SkylakeX

| | | | | | | | |
|---|-------|----|----|---------|----|----|--|
| AESDEC, AESDECLAST, AESENC, AESENCLAST | x,x | 1 | 1 | p0 | 4 | 1 | |
| AESDEC, AESDECLAST, AESENC, AESENCLAST | x,m | 2 | 2 | p0 p23 | | 1 | |
| AESIMC | x,x | 2 | 2 | 2p0 | 8 | 2 | |
| AESIMC | x,m | 3 | 3 | 2p0 p23 | | 2 | |
| AESKEYGENASSIST | x,x,i | 13 | 13 | p0 p5 | 12 | 12 | |
| AESKEYGENASSIST | x,m,i | 13 | 13 | | | 12 | |
| Other | | | | | | | |
| EMMS | | 10 | 10 | p05 | | 6 | |

Floating point vector instructions

| Instruction | Operands | μops fused domain | μops unfused domain | μops each port | Latency | Reciprocal through put | Comments |
|--------------------------|-----------------|-------------------|---------------------|----------------|---------|------------------------|---------------|
| Move instructions | | | | | | | |
| MOVAPS/D | x,x | 1 | 1 | p015 | 0-1 | 0.25 | may eliminate |
| VMOVAPS/D | y,y | 1 | 1 | p015 | 0-1 | 0.25 | may eliminate |
| VMOVAPS/D | z,z | 1 | 1 | p05 | 0-1 | 0.33 | may eliminate |
| MOVUPS/D | x,m128 | 1 | 1 | p23 | 2 | 0.5 | |
| VMOVUPS/D | v,m | 1 | 1 | p23 | 3 | 0.5 | |
| MOVUPS/D | m128,x | 1 | 2 | p237 p4 | 3 | 1 | |
| VMOVUPS/D | m256,y | 1 | 2 | p237 p4 | 3 | 1 | |
| MOVSS/D | x,x | 1 | 1 | p5 | 1 | 1 | |
| MOVSS/D | x,m32/64 | 1 | 1 | p23 | 2 | 0.5 | |
| MOVSS/D | m32/64,x | 1 | 2 | p237 p4 | 3 | 1 | |
| MOVHPS/D | x,m64 | 1 | 2 | p23 p5 | 4 | 1 | |
| MOVHPS/D | m64,x | 1 | 2 | p4 p237 | 3 | 1 | |
| MOVLPS/D | x,m64 | 1 | 2 | p23 p5 | 4 | 1 | |
| MOVLPS/D | m64,x | 1 | 2 | p4 p237 | 3 | 1 | |
| MOVHLPs | x,x | 1 | 1 | p5 | 1 | 1 | |
| MOVLHPs | x,x | 1 | 1 | p5 | 1 | 1 | |
| MOVMSKPS/D | r32,x | 1 | 1 | p0 | 2 | 1 | |
| VMOVMSKPS/D | r32,y | 1 | 1 | p0 | 3 | 1 | |
| MOVNTPS/D | m128,x | 1 | 2 | p4 p237 | ~400 | 1 | |
| VMOVNTPS/D | m256,y | 1 | 2 | p4 p237 | ~400 | 1 | |
| SHUFFPS/D | x,x,i / v,v,v,i | 1 | 1 | p5 | 1 | 1 | |
| SHUFFPS/D | x,m,i / v,v,m,i | 2 | 2 | p5 p23 | | 1 | |
| VSHUFF32X4 | v,v,v,i | 1 | 1 | p5 | 3 | 1 | |
| VSHUFF64X2 | v,v,v,i | 1 | 1 | p5 | 3 | 1 | |
| VPERMILPS/PD | v,v,i | 1 | 1 | p5 | 1 | 1 | |
| VPERMILPS/PD | v,m,i | 2 | 2 | p5 p23 | | 1 | |

SkylakeX

| | | | | | | |
|----------------|-----------------|---|---|--------------|-----|----------|
| VPERMILPS/PD | v,v,v | 1 | 1 | p5 | 1 | 1 |
| VPERMILPS/PD | v,v,m | 2 | 2 | p5 p23 | | 1 |
| VPERM2F128 | y,y,y,i | 1 | 1 | p5 | 3 | 1 |
| VPERM2F128 | y,y,m,i | 2 | 2 | p5 p23 | | 1 |
| VPERMPS | v,v,v | 1 | 1 | p5 | 3 | 1 |
| VPERMPS | y,y,m | 1 | 2 | p5 p23 | | 1 |
| VPERMPD | y,y,i | 1 | 1 | p5 | 3 | 1 |
| VPERMPD | y,m,i | 2 | 2 | p5 p23 | | 1 |
| VPERMI2PS/PD | v,v,v | 1 | 1 | p5 | 3 | 1 |
| VPERMT2PS/PD | v,v,v | 1 | 1 | p5 | 3 | 1 |
| BLENDPS/PD | x,x,i / v,v,v,i | 1 | 1 | p015 | 1 | 0.33-0.5 |
| BLENDPS/PD | x,m,i / v,v,m,i | 2 | 2 | p015 p23 | | 0.5 |
| BLENDVPS/PD | x,x,xmm0 | 1 | 1 | p015 | 1 | 1 |
| BLENDVPS/PD | x,m,xmm0 | 2 | 2 | p015 p23 | | 1 |
| VBLENDVPS/PD | v,v,v,v | 2 | 2 | 2p015 | 2 | 1 |
| VBLENDVPS/PD | v,v,m,v | 3 | 3 | 2p015 p23 | | 1 |
| VBLENDMPS/PD | v{k},v,v | 1 | 1 | p05 | 1 | 0.5 |
| MOVDDUP | v,v | 1 | 1 | p5 | 1 | 1 |
| MOVDDUP | v,m | 1 | 1 | p23 | 3 | 0.5 |
| VBROADCASTSS | x,m32 | 1 | 1 | p23 | 2 | 0.5 |
| VBROADCASTSS | y,m32 | 1 | 1 | p23 | 3 | 0.5 |
| VBROADCASTSS | z,m32 | 1 | 1 | p23 | 4 | 0.5 |
| VBROADCASTSS | x,x | 1 | 1 | p5 | 1 | 1 |
| VBROADCASTSS | v,x | 1 | 1 | p5 | 3 | 1 |
| VBROADCASTSD | y,m64 | 1 | 1 | p23 | 3 | 0.5 |
| VBROADCASTSD | z,m64 | 1 | 1 | p23 | 4 | 0.5 |
| VBROADCASTSD | v,x | 1 | 1 | p5 | 3 | 1 |
| VBROADCASTF128 | y,m128 | 1 | 1 | p23 | 3 | 0.5 |
| MOVSH/LDUP | v,v | 1 | 1 | p5 | 1 | 1 |
| MOVSH/LDUP | v,m | 1 | 1 | p23 | 3-4 | 0.5 |
| UNPCKH/LPS/D | x,x / v,v,v | 1 | 1 | p5 | 1 | 1 |
| UNPCKH/LPS/D | x,m / v,v,m | 1 | 2 | p5 p23 | | 1 |
| EXTRACTPS | r32,x,i | 2 | 2 | p0 p5 | 3 | 1 |
| EXTRACTPS | m32,x,i | 2 | 3 | p4 p5 p23 | 5 | 1 |
| VEEXTRACTF128 | x,y,i | 1 | 1 | p5 | 3 | 1 |
| VEEXTRACTF128 | m128,y,i | 2 | 2 | p23 p4 | 5 | 1 |
| VEEXTRACTF32X4 | | | | | | |
| VEEXTRACTF64X2 | x,z,i | 1 | 1 | p5 | 3 | 1 |
| VEEXTRACTF32X4 | | | | | | |
| VEEXTRACTF64X2 | m128,z,i | 2 | 2 | p23 p4 | 3 | 1 |
| INSERTPS | x,x,i | 1 | 1 | p5 | 1 | 1 |
| INSERTPS | x,m32,i | 2 | 2 | p23 p5 | 4 | 1 |
| VINSERTF128 | y,y,x,i | 1 | 1 | p5 | 3 | 1 |
| VINSERTF128 | y,y,m128,i | 2 | 2 | p015 p23 | 5 | 0.5 |
| VEEXTRACTF32X4 | | | | | | |
| VEEXTRACTF64X2 | z,z,x,i | 1 | 1 | p5 | 3 | 1 |
| VEEXTRACTF32X4 | | | | | | |
| VEEXTRACTF64X2 | z,z,m128,i | 2 | 2 | p05 p23 | 5 | 0.5 |
| VMASKMOVPS/D | v,v,m | 2 | 2 | p015 p23 | 3 | 0.5 |
| VMASKMOVPS/D | m,xy,xy | 3 | 3 | p0 p4 p23 | 13 | 1 |
| VCOMPRESPPS/PD | v{k},v | 2 | 2 | p5 | 3 | 2 |
| VEXPANDPS/PD | v{k},v | 2 | 2 | p5 | 3 | 2 |
| VGATHERDPS | x,[r+s*x],x | 4 | 4 | p0 p1 p23 p5 | 12 | 4 |
| VGATHERDPS | y,[r+s*y],y | 4 | 4 | p0 p1 p23 p5 | 13 | 5 |

SkylakeX

| | | | | | | |
|---------------|--------------|----|----|--------------|-------|-------|
| VGATHERQPS | x,[r+s*x],x | 4 | 5 | p0 p1 p23 p5 | | 2 |
| VGATHERQPS | x,[r+s*y],x | 4 | 4 | p0 p1 p23 p5 | | 4 |
| VGATHERDPD | x,[r+s*x],x | 4 | 5 | p0 p1 p23 p5 | | 2 |
| VGATHERDPD | y,[r+s*x],y | 4 | 4 | p0 p1 p23 p5 | | 4 |
| VGATHERQPD | x,[r+s*x],x | 4 | 4 | p0 p1 p23 p5 | | 2 |
| VGATHERQPD | y,[r+s*y],y | 4 | 4 | p0 p1 p23 p5 | | 4 |
| VGATHERDPS | x{k},[r+s*x] | 4 | 4 | p0 p1 p23 p5 | | 4 |
| VGATHERDPS | y{k},[r+s*y] | 4 | 4 | p0 p1 p23 p5 | | 5 |
| VGATHERDPS | z{k},[r+s*z] | 4 | 4 | p0 p1 p23 p5 | | 9 |
| VGATHERDPD | x{k},[r+s*x] | 4 | 5 | p0 p1 p23 p5 | | 3 |
| VGATHERDPD | y{k},[r+s*x] | 4 | 4 | p0 p1 p23 p5 | | 4 |
| VGATHERDPD | z{k},[r+s*y] | 4 | 4 | p0 p1 p23 p5 | | 5 |
| VGATHERQPS | x{k},[r+s*x] | 4 | 5 | p0 p1 p23 p5 | | 3 |
| VGATHERQPS | x{k},[r+s*y] | 4 | 4 | p0 p1 p23 p5 | | 4 |
| VGATHERQPS | y{k},[r+s*z] | 4 | 4 | p0 p1 p23 p5 | | 5 |
| VGATHERQPD | x{k},[r+s*x] | 4 | 5 | p0 p1 p23 p5 | | 3 |
| VGATHERQPD | y{k},[r+s*y] | 4 | 4 | p0 p1 p23 p5 | | 4 |
| VGATHERQPD | z{k},[r+s*z] | 4 | 4 | p0 p1 p23 p5 | | 5 |
| VPSCATTERDPS | [r+s*x]{k},x | 19 | 19 | | 17-20 | 10 |
| VPSCATTERDPS | [r+s*y]{k},y | 27 | 27 | | | 12 |
| VPSCATTERDPS | [r+s*z]{k},z | 43 | 43 | | | 17 |
| VPSCATTERDPD | [r+s*x]{k},x | 15 | 15 | | | 9 |
| VPSCATTERDPD | [r+s*x]{k},y | 19 | 19 | | | 10 |
| VPSCATTERDPD | [r+s*y]{k},z | 27 | 27 | | | 12 |
| VPSCATTERQPS | [r+s*x]{k},x | 15 | 15 | | | 9 |
| VPSCATTERQPS | [r+s*y]{k},y | 19 | 19 | | | 10 |
| VPSCATTERQPS | [r+s*z]{k},z | 27 | 27 | | | 12 |
| VPSCATTERQPD | [r+s*x]{k},x | 15 | 15 | | | 9 |
| VPSCATTERQPD | [r+s*y]{k},y | 19 | 9 | | | 10 |
| VPSCATTERQPD | [r+s*z]{k},z | 27 | 27 | | | 12 |
| Conversion | | | | | | |
| CVTPD2PS | x,x | 2 | 2 | p01 p5 | 5 | 1 |
| VCVTPD2PS | x,y / y,z | 2 | 2 | p01 p5 | 7 | 1 |
| CVTPS2PD | x,x | 2 | 2 | p01 p5 | 5 | 1 |
| VCVTPS2PD | y,x / z,y | 2 | 2 | p01 p5 | 7 | 1 |
| CVTSS2SD | x,x | 2 | 2 | p01 p5 | 5 | 2 |
| CVTSD2SS | x,x | 2 | 2 | p01 p5 | 5 | 1 |
| CVT(T) PS2DQ | x,x | 1 | 1 | p01 | 4 | 0.5 |
| VCVT(T) PS2DQ | y,y | 1 | 1 | p01 | 4 | 0.5 |
| VCVT(T) PS2DQ | z,z | 1 | 1 | p05 | 4 | 0.5 |
| CVTDQ2PS | x,x | 1 | 1 | p01 | 4 | 0.5 |
| VCVTDQ2PS | y,y | 1 | 1 | p01 | 4 | 0.5 |
| VCVTDQ2PS | z,z | 1 | 1 | p05 | 4 | 0.5 |
| VCVT(T)PS2QQ | v,v | 2 | 2 | p01 p5 | 7 | 1 |
| VCVTQQ2PS | v,v | 2 | 2 | p01 p5 | 7 | 1 |
| VCVT(T)PS2UDQ | v,v | 1 | 1 | p01 | 4 | 0.5-1 |
| VCVTUDQ2PS | v,v | 1 | 1 | p01 | 4 | 0.5-1 |
| VCVT(T)PS2UQQ | v,v | 2 | 2 | p01 p5 | 7 | 1 |
| VCVTUQQ2PS | v,v | 2 | 2 | p01 p5 | 7 | 1 |
| CVT(T)PS2PI | mm,x | 2 | 2 | p0 p5 | 7 | 1 |
| CVTPI2PS | x,mm | 2 | 2 | p0 p1 | 6 | 2 |
| CVT(T)SS2SI | r32,x | 2 | 2 | 2p01 | 6 | 1 |

SkylakeX

| | | | | | | |
|-------------------|-------------|---|---|------------|-------|-------|
| CVT(T)SS2SI | r64,x | 3 | 3 | 2p01 p5 | 7 | 1 |
| CVT(T)SS2SI | r32,m32 | 3 | 3 | 2p01 p23 | | 1 |
| CVTSI2SS | x,r32 | 2 | 2 | p01 p5 | 6 | 2 |
| CVTSI2SS | x,r64 | 3 | 3 | p01 2p5 | 7 | 2 |
| CVTSI2SS | x,m32 | 1 | 2 | p1 p23 | | 3 |
| CVT(T)SS2USI | r32,x | 2 | 2 | 2p01 | 6 | 1 |
| VCVTPS2PH | x,v,i | 2 | 2 | p01 p5 | 5-7 | 1 |
| VCVTPS2PH | y,z,i | 2 | 2 | p0 p5 | 5-7 | 1 |
| VCVTPH2PS | v,x | 2 | 2 | p01 p5 | 5-7 | 1 |
| VCVTPH2PS | z,y | 2 | 2 | p0 p5 | 5-7 | 1 |
| CVT(T)PD2DQ | x,x | 2 | 2 | p01 p5 | 5 | 1 |
| VCVT(T)PD2DQ | x,y | 2 | 2 | p01 p5 | 7 | 1 |
| VCVT(T)PD2DQ | y,z | 2 | 2 | p0 p5 | 7 | 1 |
| CVTDQ2PD | x,x | 2 | 2 | p01 p5 | 5 | 1 |
| VCVTDQ2PD | y,x | 2 | 2 | p01 p5 | 7 | 1 |
| VCVTDQ2PD | z,y | 2 | 2 | p0 p5 | 7 | 1 |
| VCVT(T)PD2QQ | v,v | 1 | 1 | p01 | 4 | 0.5-1 |
| VCVTQQ2PD | v,v | 1 | 1 | p01 | 4 | 0.5-1 |
| VCVT(T)PD2UDQ | x,x | 2 | 2 | p01 p5 | 5 | 1 |
| VCVT(T)PD2UDQ | x,y | 2 | 2 | p01 p5 | 7 | 1 |
| VCVT(T)PD2UDQ | y,z | 2 | 2 | p0 p5 | 7 | 1 |
| VCVTUDQ2PD | x,x | 2 | 2 | p01 p5 | 5 | 1 |
| VCVTUDQ2PD | y,x | 2 | 2 | p01 p5 | 7 | 1 |
| VCVTUDQ2PD | z,y | 2 | 2 | p0 p5 | 7 | 1 |
| VCVT(T)PD2UQQ | v,v | 1 | 1 | p01 | 4 | 0.5-1 |
| VCVTUQQ2PD | v,v | 1 | 1 | p01 | 4 | 0.5-1 |
| CVT(T) PD2PI | mm,x | 2 | 2 | p01 p5 | 5 | 1 |
| CVTPI2PD | x,mm | 2 | 2 | p01 p5 | 5 | 1 |
| CVT(T)SD2SI | r32/64,x | 2 | 2 | p0 p1 | 6 | 1 |
| CVTSI2SD | x,r32/64 | 2 | 2 | p01 p5 | 6 | 2 |
| CVT(T)SD2USI | r32/64,x | 2 | 2 | p0 p1 | 6 | 1 |
| CVTUSI2SD | x,r32/64 | 2 | 2 | p01 p5 | 6 | 1 |
| Arithmetic | | | | | | |
| ADDSS/D PS/D | | | | | | |
| SUBSS/D PS/D | x,x / y,y,y | 1 | 1 | p01 | 4 | 0.5 |
| VADDPS/D | | | | | | |
| VSUBPS/D | z,z,z | 1 | 1 | p05 | 4 | 0.5-1 |
| ADDSS/D PS/D | | | | | | |
| SUBSS/D PS/D | x,m / v,v,m | 1 | 2 | p01/05 p23 | | 0.5-1 |
| ADDPS/D | x,x / v,v,v | 1 | 1 | p01/05 | 4 | 0.5-1 |
| ADDPS/D | x,m / v,v,m | 1 | 2 | p01/05 p23 | | 0.5-1 |
| HADDPS/D | | | | | | |
| HSUBPS/D | x,x / v,v,v | 3 | 3 | p01 2p5 | 6 | 2 |
| HADDPS/D | | | | | | |
| HSUBPS/D | x,m / v,v,m | 4 | 4 | p1 2p5 p23 | | 2 |
| MULSS/D PS/D | x,x / v,v,v | 1 | 1 | p01 | 4 | 0.5-1 |
| VMULPS/D | z,z,z | 1 | 1 | p05 | 4 | 0.5-1 |
| MULSS/D PS/D | x,m / v,v,m | 1 | 2 | p01/05 p23 | | 0.5-1 |
| DIVSS | x,x | 1 | 1 | p0 | 11 | 3 |
| DIVPS | x,x | 1 | 1 | p0 | 11 | 3 |
| DIVSS DIVPS | x,m | 1 | 2 | p0 p23 | | 3-5 |
| DIVSD | x,x | 1 | 1 | p0 | 13-14 | 4 |
| DIVPD | x,x | 1 | 1 | p0 | 13-14 | 4 |

SkylakeX

| | | | | | | |
|-------------------|-----------------|---|---|----------------|-------|-------|
| DIVSD DIVPD | x,m | 1 | 2 | p0 p23 | | 4 |
| VDIVPS | y,y,y | 1 | 1 | p0 | 11 | 5 |
| VDIVPS | z,z,z | 3 | 3 | 2p0 p5 | 18 | 10 |
| VDIVPS | y,y,m256 | 1 | 2 | p0 p23 | | 5 |
| VDIVPD | y,y,y | 1 | 1 | p0 | 13-14 | 8 |
| VDIVPD | z,z,z | 3 | 3 | 2p0 p5 | 24 | 16 |
| VDIVPD | y,y,m256 | 4 | 4 | p0 p23 | | 8 |
| RCPSS/PS | v,v | 1 | 1 | p0 | 4 | 1 |
| RCPSS/PS | v,m | 1 | 2 | p0 p23 | | 1 |
| VRCP14SS/PS | xy,xy | 1 | 1 | p0 | 4 | 1 |
| VRCP14SS/PS | z,z | 3 | 3 | 2p0 p5 | 7 | 2 |
| CMPccSS/D | | | | | | |
| CMPccPS/D | x,x / v,v,v | 1 | 1 | p01 | 4 | 0.5 |
| CMPccSS/D | | | | | | |
| CMPccPS/D | x,m / v,v,m | 2 | 2 | p01 p23 | | 0.5 |
| VCMPPS/PD | k,v,v | 1 | 1 | | 3 | 1 |
| (U)COMISS/D | x,x | 1 | 1 | p0 | 2 | 1 |
| (U)COMISS/D | x,m32/64 | 2 | 2 | p0 p23 | | 1 |
| MAXSS/SD/PS/PD | | | | | | |
| MINSS/SD/PS/PD | x,x / v,v,v | 1 | 1 | p01/05 | 4 | 0.5-1 |
| MAXSS/SD/PS/PD | | | | | | |
| MINSS/SD/PS/PD | x,m / v,v,m | 1 | 2 | p01/05 p23 | | 0.5-1 |
| VRANGESS/SD/PS/PD | v,v,v | 1 | 1 | p01 | 4 | 0.5-1 |
| ROUNDSS/SD/PS/PD | v,v,i | 2 | 2 | 2p01 | 8 | 1 |
| ROUNDSS/SD/PS/PD | v,m,i | 3 | 3 | 2p01 p23 | | 1 |
| VRNDSCALE | | | | | | |
| SS/PS/SD/PD | v,v,v,i | 2 | 2 | 2p01/05 | 8 | 1 |
| VREDUCE | | | | | | |
| SS/PS/SD/PD | v,v,v,i | 1 | 1 | p01 | 4 | 0.5-1 |
| DPPS | x,x,i / v,v,v,i | 4 | 4 | 3p01 p5 | 13 | 2 |
| DPPS | x,m,i / v,v,m,i | 6 | 6 | 3p01 p23 p5 p6 | | 4 |
| DPPD | x,x,i | 3 | 3 | 2p01 p5 | 9 | 1 |
| DPPD | x,m128,i | 4 | 4 | 2p01 p23 p5 | | 1 |
| VDBPSADBW | v,v,v,i | 1 | 1 | p5 | 3 | 1 |
| VFIXUPIMMSS/SD | x,x,x,i | 1 | 1 | p01 | 4 | 2 |
| VFIXUPIMMPS/PD | v,v,v,i | 1 | 1 | p01/05 | 4 | 2 |
| VFPCLASSSS/SD | k,x,i | 1 | 1 | p5 | 3 | 1 |
| VFPCLASSPS/PD | k,v,i | 1 | 1 | p5 | 3 | 1 |
| VFMADD... | | | | | | |
| (all FMA instr.) | xy,xy,xy | 1 | 1 | p01 | 4 | 0.5 |
| VFMADD... | | | | | | |
| (all FMA instr.) | z,z,z | 1 | 1 | p05 | 4 | 0.5-1 |
| VFMADD... | | | | | | |
| (all FMA instr.) | v,v,m | 1 | 2 | p01/05 p23 | | 0.5-1 |
| Math | | | | | | |
| SQRTSS/PS | x,x | 1 | 1 | p0 | 12 | 3 |
| VSQRTPS | y,y | 1 | 1 | p0 | 12 | 6 |
| VSQRTPS | z,z | 3 | 3 | p0 | 20 | 12 |
| SQRTSS/PS | x,m128 | 1 | 2 | p0 p23 | | 3 |
| VSQRTPS | y,m256 | 4 | 4 | p0 p23 | | 6 |
| SQRTSD | x,x | 1 | 1 | p0 | 15-16 | 4-6 |
| SQRTPD | x,x | 1 | 1 | p0 | 15-16 | 4-6 |
| VSQRTPD | y,y | 1 | 1 | p0 | 15-16 | 9-12 |

SkylakeX

| | | | | | | | |
|--------------------------|-------------|-----|-----|---------------|-------|-------|----------------|
| VSQRTPD | z,z | 3 | 3 | 2p0 p5 | 28-29 | 18-24 | |
| SQRTPD/PD | x,m128 | 1 | 2 | p0 p23 | | 4-6 | |
| VSQRTPD | y,m256 | 4 | 4 | p0 p23 | | 9-12 | |
| RSQRTPS/PS | x,x | 1 | 1 | p0 | 4 | 1 | |
| RSQRTPS | y,y | 1 | 1 | p0 | 2 | 1 | |
| VRSQRTPS/PS | x,x | 1 | 1 | p0 | 4 | 1 | |
| VRSQRTPS | y,y | 1 | 1 | p0 | 2 | 1 | |
| VRSQRTPS | z,z | 3 | 3 | 2p0 p5 | 6 | 2 | |
| VGETEXPSS/SD | x,x,x | 1 | 1 | p01 | 4 | 0.5-1 | |
| VGETEXPSS/PD | v,v,v | 1 | 1 | p01/05 | 4 | 0.5-1 | |
| VGETMANTSS/SD | x,x,x | 1 | 1 | p01 | 4 | 0.5-1 | |
| VGETMANTPS/PD | v,v,v | 1 | 1 | p01/05 | 4 | 0.5-1 | |
| VSCALESS/SD/PS/PD | v,v,v | 1 | 1 | p01 | 4 | 0.5-1 | |
| Logic | | | | | | | |
| AND/ANDN/OR/ XORPS/PD | x,x / y,y,y | 1 | 1 | p015 | 1 | 0.33 | |
| AND/ANDN/OR/ XORPS/PD | z,z,z | 1 | 1 | p05 | 1 | 0.5 | |
| AND/ANDN/OR/ XORPS/PD | x,m / y,y,m | 1 | 2 | p015 p23 | | 0.5 | |
| AND/ANDN/OR/ XORPS/PD | z,z,m | 1 | 2 | p05 p23 | | 0.5 | |
| VTESTPS/PD | v,v | 1 | 1 | p0 | | 1 | |
| Other | | | | | | | |
| VZEROUPPER | | 4 | 4 | none | | 1 | |
| VZEROALL | | 25 | 25 | p0 p1 p5 p6 | | 12 | 32 bit mode |
| VZEROALL | | 34 | 34 | p0 p1 p5 p6 | | 12 | 64 bit mode |
| LDMXCSR | m32 | 4 | 4 | p0 p5 p6 p23 | 5 | 3 | |
| STMXCSR | m32 | 3 | 4 | p0 p4 p6 p237 | 4 | 1 | |
| FXSAVE | m | 106 | 136 | | 76 | 76 | 32 bit mode |
| FXSAVE | m | 114 | 136 | | 65 | 65 | 64 bit mode |
| FXRSTOR | m | 105 | | | 76 | 76 | 32 bit mode |
| FXRSTOR | m | 121 | | | 72 | 72 | 64 bit mode |
| XSAVE | m | 286 | | | 112 | 112 | 32 bit mode |
| XSAVE | m | 350 | | | 136 | 136 | 64 bit mode |
| XRSTOR | m | 199 | | | 83 | 83 | 32 bit mode |
| XRSTOR | m | 266 | | | 116 | 116 | 64 bit mode |
| XSAVEC | m | 226 | | | 83 | 83 | 64 bit mode |
| XSAVEOPT | m | 214 | | | 84 | 84 | 32/64 bit mode |

Mask register instructions

| Instruction | Operands | μops fused domain | μops unfused domain | μops each port | Latency | Reciprocal throughput | Comments |
|--------------------------|----------|-------------------|---------------------|----------------|---------|-----------------------|----------|
| Move instructions | | | | | | | |
| KMOVB/W/D/Q | k,k | 1 | 1 | p0 | 1 | 1 | |
| KMOVB/W/D/Q | k,m | 3 | | | 3 | | |
| KMOVB/W/D/Q | m,k | 2 | 1 | p237 p4 | 4 | 1 | |
| KMOVB/W/D/Q | k,r | 1 | 1 | p5 | 2 | 1 | |

SkyLakeX

| | | | | | | |
|--------------------------------|-------|---|---|-------|---|-----|
| KMOVB/W/D/Q | r,k | 1 | 1 | p0 | 2 | 1 |
| VPBROADCASTMB2Q | v,k | 2 | 2 | p0 p5 | 6 | 1 |
| VPBROADCASTMW2D | v,k | 2 | 2 | p0 p5 | 6 | 1 |
| KUNPCKBW | k,k,k | 1 | 1 | p5 | 4 | 1 |
| KUNPCKWD | k,k,k | 1 | 1 | p5 | 4 | 1 |
| KUNPCKDQ | k,k,k | 1 | 1 | p5 | 4 | 1 |
| VPMOVM2B/W | v,k | 1 | 1 | p05 | 3 | 0.5 |
| VPMOVM2D/Q | v,k | 1 | 1 | p05 | 2 | 0.5 |
| VPMOVB/W2M | k,v | 1 | 1 | p0 | 3 | 1 |
| VPMOVD/Q2M | k,v | 1 | 1 | p0 | 2 | 1 |
| Arithmetic instructions | | | | | | |
| KADDB/W/D/Q | k,k,k | 1 | 1 | p5 | 4 | 1 |
| KSHIFTLB/W/D/Q | k,k,i | 1 | 1 | p5 | 4 | 1 |
| KSHIFTRB/W/D/Q | k,k,i | 1 | 1 | p5 | 4 | 1 |
| Logic instructions | | | | | | |
| KANDB/W/D/Q | k,k,k | 1 | 1 | p0 | 1 | 1 |
| KANDNB/W/D/Q | k,k,k | 1 | 1 | p0 | 1 | 1 |
| KORB/W/D/Q | k,k,k | 1 | 1 | p0 | 1 | 1 |
| KXORB/W/D/Q | k,k,k | 1 | 1 | p0 | 1 | 1 |
| KXNORB/W/D/Q | k,k,k | 1 | 1 | p0 | 1 | 1 |
| KNOTB/W/D/Q | k,k | 1 | 1 | p0 | 1 | 1 |
| KTESTB/W/D/Q | k,k | 1 | 1 | p0 | 2 | 1 |
| KORTESTB/W/D/Q | k,k | 1 | 1 | p0 | 2 | 1 |

Intel Pentium 4

List of instruction timings and μ op breakdown

This list is measured for a Pentium 4, model 2. Timings for model 3 may be more like the values for P4E, listed on the next sheet

Explanation of column headings:

| | |
|-------------------------------|--|
| Instruction: | Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc. |
| Operands: | i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, xmm = 128 bit xmm register, sr = segment register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc. |
| μops: | Number of μ ops issued from instruction decoder and stored in trace cache. |
| Microcode: | Number of additional μ ops issued from microcode ROM. |
| Latency: | This is the delay that the instruction generates in a dependency chain if the next dependent instruction starts in the same execution unit. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latency of moves to and from memory cannot be measured accurately because of the problem with memory intermediates explained above under "How the values were measured". |
| Additional latency: | This number is added to the latency if the next dependent instruction is in a different execution unit. There is no additional latency between ALU0 and ALU1. |
| Reciprocal throughput: | This is also called issue latency. This value indicates the number of clock cycles from the execution of an instruction begins to a subsequent independent instruction can begin to execute in the same execution subunit. A value of 0.25 indicates 4 instructions per clock cycle in one thread. |
| Port: | The port through which each μ op goes to an execution unit. Two independent μ ops can start to execute simultaneously only if they are going through different ports. |
| Execution unit: | Use this information to determine additional latency. When an instruction with more than one μ op uses more than one execution unit, only the first and the last execution unit is listed. |
| Execution subunit: | Throughput measures apply only to instructions executing in the same subunit. |
| Instruction set | Indicates the compatibility of an instruction with other 80x86 family microprocessors. The instruction can execute on microprocessors that support the instruction set indicated. |

Integer instructions

Pentium 4

| Instruction | Operands | µops | Microcode | Latency | Additional latency | Reciprocal throughput | Port | Execution unit | Subunit | Instruction set | Notes |
|--------------------------|-------------|------|-----------|---------|--------------------|-----------------------|------|----------------|---------|-----------------|-------|
| Move instructions | | | | | | | | | | | |
| MOV | r,r | 1 | 0 | 0.5 | 0.5-1 | 0.25 | 0/1 | alu0/1 | | 86 | c |
| MOV | r,i | 1 | 0 | 0.5 | 0.5-1 | 0.25 | 0/1 | alu0/1 | | 86 | |
| MOV | r32,m | 1 | 0 | 2 | 0 | 1 | 2 | load | | 86 | |
| MOV | r8/16,m | 2 | 0 | 3 | 0 | 1 | 2 | load | | 86 | |
| MOV | m,r | 1 | 0 | 1 | | 2 | 0 | store | | 86 | b, c |
| MOV | m,i | 3 | 0 | | | 2 | 0,3 | store | | 86 | |
| MOV | r,sr | 4 | 2 | | | 6 | | | | 86 | |
| MOV | sr,r/m | 4 | 4 | 12 | 0 | 14 | | | | 86 | a, q |
| MOVNTI | m,r32 | 2 | 0 | | | ≈33 | | | | sse2 | |
| MOVZX | r,r | 1 | 0 | 0.5 | 0.5-1 | 0.25 | 0/1 | alu0/1 | | 386 | c |
| MOVZX | r,m | 1 | 0 | 2 | 0 | 1 | 2 | load | | 386 | |
| MOVSX | r,r | 1 | 0 | 0.5 | 0.5-1 | 0.5 | 0 | alu0 | | 386 | c |
| MOVSX | r,m | 2 | 0 | 3 | 0.5-1 | 1 | 2,0 | | | 386 | |
| CMOVcc | r,r/m | 3 | 0 | 6 | 0 | 3 | | | | ppro | a, e |
| XCHG | r,r | 3 | 0 | 1.5 | 0.5-1 | 1 | 0/1 | alu0/1 | | 86 | |
| XCHG | r,m | 4 | 8 | >100 | | | | | | 86 | |
| XLAT | | 4 | 0 | 3 | | | | | | 86 | |
| PUSH | r | 2 | 0 | 1 | | 2 | | | | 86 | |
| PUSH | i | 2 | 0 | 1 | | 2 | | | | 186 | |
| PUSH | m | 3 | 0 | | | 2 | | | | 86 | |
| PUSH | sr | 4 | 4 | | | 7 | | | | 86 | |
| PUSHF(D) | | 4 | 4 | | | 10 | | | | 86 | |
| PUSHA(D) | | 4 | 10 | | | 19 | | | | 186 | |
| POP | r | 2 | 0 | 1 | 0 | 1 | | | | 86 | |
| POP | m | 4 | 8 | | | 14 | | | | 86 | |
| POP | sr | 4 | 5 | | | 13 | | | | 86 | |
| POPF(D) | | 4 | 8 | | | 52 | | | | 86 | |
| POPA(D) | | 4 | 16 | | | 14 | | | | 186 | |
| LEA | r,[r+r/i] | 1 | 0 | 0.5 | 0.5-1 | 0.25 | 0/1 | alu0/1 | | 86 | |
| LEA | r,[r+r+i] | 2 | 0 | 1 | 0.5-1 | 0.5 | 0/1 | alu0/1 | | 86 | |
| LEA | r,[r*i] | 3 | 0 | 4 | 0.5-1 | 1 | 1 | int,alu | | 386 | |
| LEA | r,[r+r*i] | 2 | 0 | 4 | 0.5-1 | 1 | 1 | int,alu | | 386 | |
| LEA | r,[r+r*i+i] | 3 | 0 | 4 | 0.5-1 | 1 | 1 | int,alu | | 386 | |
| LAHF | | 1 | 0 | 4 | 0 | 4 | 1 | int | | 86 | |
| SAHF | | 1 | 0 | 0.5 | 0.5-1 | 0.5 | 0/1 | alu0/1 | | 86 | d |
| SALC | | 3 | 0 | 5 | 0 | 1 | 1 | int | | 86 | |
| LDS, LES, ... | r,m | 4 | 7 | | | 15 | | | | 86 | |
| BSWAP | r | 3 | 0 | 7 | 0 | 2 | | int,alu | | 486 | |
| IN, OUT | r,r/i | 8 | 64 | | | >1000 | | | 86 | | |
| PREFETCHNTA | m | 4 | 2 | | | 6 | | | | sse | |
| PREFETCHT0/1/2 | m | 4 | 2 | | | 6 | | | | sse | |
| SFENCE | | 4 | 2 | | | 40 | | | | sse | |
| LFENCE | | 4 | 2 | | | 38 | | | | sse2 | |
| MFENCE | | 4 | 2 | | | 100 | | | | sse2 | |

Pentium 4

| Arithmetic instructions | | | | | | | | | | |
|-------------------------|-----------|---|-----|-----|-------|------|-----|---------|-------|-----|
| ADD, SUB | r,r | 1 | 0 | 0.5 | 0.5-1 | 0.25 | 0/1 | alu0/1 | 86 | c |
| ADD, SUB | r,m | 2 | 0 | 1 | 0.5-1 | 1 | | | 86 | c |
| ADD, SUB | m,r | 3 | 0 | ≥ 8 | | ≥ 4 | | | 86 | c |
| ADC, SBB | r,r | 4 | 4 | 6 | 0 | 6 | 1 | int,alu | 86 | |
| ADC, SBB | r,i | 3 | 0 | 6 | 0 | 6 | 1 | int,alu | 86 | |
| ADC, SBB | r,m | 4 | 6 | 8 | 0 | 8 | 1 | int,alu | 86 | |
| ADC, SBB | m,r | 4 | 7 | ≥ 9 | | 8 | | | 86 | |
| CMP | r,r | 1 | 0 | 0.5 | 0.5-1 | 0.25 | 0/1 | alu0/1 | 86 | c |
| CMP | r,m | 2 | 0 | 1 | 0.5-1 | 1 | | | 86 | c |
| INC, DEC | r | 2 | 0 | 0.5 | 0.5-1 | 0.5 | 0/1 | alu0/1 | 86 | |
| INC, DEC | m | 4 | 0 | 4 | | ≥ 4 | | | 86 | |
| NEG | r | 1 | 0 | 0.5 | 0.5-1 | 0.5 | 0 | alu0 | 86 | |
| NEG | m | 3 | 0 | | | ≥ 3 | | | 86 | |
| AAA, AAS | | 4 | 27 | 90 | | | | | 86 | |
| DAA, DAS | | 4 | 57 | 100 | | | | | 86 | |
| AAD | | 4 | 10 | 22 | | | 1 | int | fpmul | 86 |
| AAM | | 4 | 22 | 56 | | | 1 | int | fpmul | 86 |
| MUL, IMUL | r8/32 | 4 | 6 | 16 | 0 | 8 | 1 | int | fpmul | 86 |
| MUL, IMUL | r16 | 4 | 7 | 17 | 0 | 8 | 1 | int | fpmul | 86 |
| MUL, IMUL | m8/32 | 4 | 7-8 | 16 | 0 | 8 | 1 | int | fpmul | 86 |
| MUL, IMUL | m16 | 4 | 10 | 16 | 0 | 8 | 1 | int | fpmul | 86 |
| IMUL | r32,r | 4 | 0 | 14 | 0 | 4.5 | 1 | int | fpmul | 386 |
| IMUL | r32,(r),i | 4 | 0 | 14 | 0 | 4.5 | 1 | int | fpmul | 386 |
| IMUL | r16,r | 4 | 5 | 16 | 0 | 9 | 1 | int | fpmul | 386 |
| IMUL | r16,r,i | 4 | 5 | 15 | 0 | 8 | 1 | int | fpmul | 186 |
| IMUL | r16,m16 | 4 | 7 | 15 | 0 | 10 | 1 | int | fpmul | 386 |
| IMUL | r32,m32 | 4 | 0 | 14 | 0 | 8 | 1 | int | fpmul | 386 |
| IMUL | r,m,i | 4 | 7 | 14 | 0 | 10 | 1 | int | fpmul | 186 |
| DIV | r8/m8 | 4 | 20 | 61 | 0 | 24 | 1 | int | fpmul | 86 |
| DIV | r16/m16 | 4 | 18 | 53 | 0 | 23 | 1 | int | fpmul | 86 |
| DIV | r32/m32 | 4 | 21 | 50 | 0 | 23 | 1 | int | fpmul | 386 |
| IDIV | r8/m8 | 4 | 24 | 61 | 0 | 24 | 1 | int | fpmul | 86 |
| IDIV | r16/m16 | 4 | 22 | 53 | 0 | 23 | 1 | int | fpmul | 86 |
| IDIV | r32/m32 | 4 | 20 | 50 | 0 | 23 | 1 | int | fpmul | 386 |
| CBW | | 2 | 0 | 1 | 0.5-1 | 1 | 0 | alu0 | 86 | |
| CWD, CDQ | | 2 | 0 | 1 | 0.5-1 | 0.5 | 0/1 | alu0/1 | 86 | |
| CWDE | | 1 | 0 | 0.5 | 0.5-1 | 0.5 | 0 | alu0 | 386 | |
| Logic instructions | | | | | | | | | | |
| AND, OR, XOR | r,r | 1 | 0 | 0.5 | 0.5-1 | 0.5 | 0 | alu0 | 86 | c |
| AND, OR, XOR | r,m | 2 | 0 | ≥ 1 | 0.5-1 | ≥ 1 | | | 86 | c |
| AND, OR, XOR | m,r | 3 | 0 | ≥ 8 | | ≥ 4 | | | 86 | c |
| TEST | r,r | 1 | 0 | 0.5 | 0.5-1 | 0.5 | 0 | alu0 | 86 | c |
| TEST | r,m | 2 | 0 | ≥ 1 | 0.5-1 | ≥ 1 | | | 86 | c |
| NOT | r | 1 | 0 | 0.5 | 0.5-1 | 0.5 | 0 | alu0 | 86 | |
| NOT | m | 4 | 0 | | | ≥ 4 | | | 86 | |
| SHL, SHR, SAR | r,i | 1 | 0 | 4 | 1 | 1 | 1 | int | mmxsh | 186 |
| SHL, SHR, SAR | r,CL | 2 | 0 | 6 | 0 | 1 | 1 | int | mmxsh | 86 |
| ROL, ROR | r,i | 1 | 0 | 4 | 1 | 1 | 1 | int | mmxsh | 186 |
| ROL, ROR | r,CL | 2 | 0 | 6 | 0 | 1 | 1 | int | mmxsh | 86 |
| RCL, RCR | r,1 | 1 | 0 | 4 | 1 | 1 | 1 | int | mmxsh | 86 |

Pentium 4

| | | | | | | | | | | | |
|--------------------------------------|------------|---|--------|-------|---|---------|---|------|--------|-----|---|
| RCL, RCR | r,i | 4 | 15 | 16 | 0 | 15 | 1 | int | mmxsh | 186 | d |
| RCL, RCR | r,CL | 4 | 15 | 16 | 0 | 14 | 1 | int | mmxsh | 86 | d |
| SHL,SHR,SAR,ROL, ROR | m,i/CL | 4 | 7-8 | 10 | 0 | 10 | 1 | int | mmxsh | 86 | d |
| RCL, RCR | m,1 | 4 | 7 | 10 | 0 | 10 | 1 | int | mmxsh | 86 | d |
| RCL, RCR | m,i/CL | 4 | 18 | 18-28 | | 14 | 1 | int | mmxsh | 86 | d |
| SHLD, SHRD | r,r,i/CL | 4 | 14 | 14 | 0 | 14 | 1 | int | mmxsh | 386 | |
| SHLD, SHRD | m,r,i/CL | 4 | 18 | 14 | 0 | 14 | 1 | int | mmxsh | 386 | |
| BT | r,i | 3 | 0 | 4 | 0 | 2 | 1 | int | mmxsh | 386 | d |
| BT | r,r | 2 | 0 | 4 | 0 | 1 | 1 | int | mmxsh | 386 | d |
| BT | m,i | 4 | 0 | 4 | 0 | 2 | 1 | int | mmxsh | 386 | d |
| BT | m,r | 4 | 12 | 12 | 0 | 12 | 1 | int | mmxsh | 386 | d |
| BTR, BTS, BTC | r,i | 3 | 0 | 6 | 0 | 2 | 1 | int | mmxsh | 386 | |
| BTR, BTS, BTC | r,r | 2 | 0 | 6 | 0 | 4 | 1 | int | mmxsh | 386 | |
| BTR, BTS, BTC | m,i | 4 | 7 | 18 | 0 | 8 | 1 | int | mmxsh | 386 | |
| BTR, BTS, BTC | m,r | 4 | 15 | 14 | 0 | 14 | 1 | int | mmxsh | 386 | |
| BSF, BSR | r,r | 2 | 0 | 4 | 0 | 2 | 1 | int | mmxsh | 386 | |
| BSF, BSR | r,m | 3 | 0 | 4 | 0 | 3 | 1 | int | mmxsh | 386 | |
| SETcc | r | 3 | 0 | 5 | 0 | 1 | 1 | int | | 386 | |
| SETcc | m | 4 | 0 | 5 | 0 | 3 | 1 | int | | 386 | |
| CLC, STC | | 3 | 0 | 10 | 0 | 2 | | | | 86 | d |
| CMC | | 3 | 0 | 10 | 0 | 2 | | | | 86 | |
| CLD | | 4 | 7 | 52 | 0 | 52 | | | | 86 | |
| STD | | 4 | 5 | 48 | 0 | 48 | | | | 86 | |
| CLI | | 4 | 5 | 35 | | 35 | | | | 86 | |
| STI | | 4 | 12 | 43 | | 43 | | | | 86 | |
| Control transfer instructions | | | | | | | | | | | |
| JMP | short/near | 1 | 0 | 0 | 0 | 1 | 0 | alu0 | branch | 86 | |
| JMP | far | 4 | 28 | 118 | | 118 | 0 | | | 86 | |
| JMP | r | 3 | 0 | 4 | | 4 | 0 | alu0 | branch | 86 | |
| JMP | m(near) | 3 | 0 | 4 | | 4 | 0 | alu0 | branch | 86 | |
| JMP | m(far) | 4 | 31 | 11 | | 11 | 0 | | | 86 | |
| Jcc | short/near | 1 | 0 | 0 | | 2-4 | 0 | alu0 | branch | 86 | |
| J(E)CXZ | short | 4 | 4 | 0 | | 2-4 | 0 | alu0 | branch | 86 | |
| LOOP | short | 4 | 4 | 0 | | 2-4 | 0 | alu0 | branch | 86 | |
| CALL | near | 3 | 0 | 2 | | 2 | 0 | alu0 | branch | 86 | |
| CALL | far | 4 | 34 | | | | 0 | | | 86 | |
| CALL | r | 4 | 4 | 8 | | | 0 | alu0 | branch | 86 | |
| CALL | m(near) | 4 | 4 | 9 | | | 0 | alu0 | branch | 86 | |
| CALL | m(far) | 4 | 38 | | | | 0 | | | 86 | |
| RETN | | 4 | 0 | 2 | | | 0 | alu0 | branch | 86 | |
| RETN | i | 4 | 0 | 2 | | | 0 | alu0 | branch | 86 | |
| RETF | | 4 | 33 | 11 | | | 0 | | | 86 | |
| RETF | i | 4 | 33 | 11 | | | 0 | | | 86 | |
| IRET | | 4 | 48 | 24 | | | 0 | | | 86 | |
| ENTER | i,0 | 4 | 12 | 26 | | 26 | | | | 186 | |
| ENTER | i,n | 4 | 45+24n | | | 128+16n | | | 186 | | |
| LEAVE | | 4 | 0 | 3 | | 3 | | | | 186 | |
| BOUND | m | 4 | 14 | 14 | | 14 | | | | 186 | |
| INTO | | 4 | 5 | 18 | | 18 | | | | 86 | |
| INT | i | 4 | 84 | 644 | | | | | | 86 | |

Pentium 4

| String instructions | | | | | | | | | | |
|---------------------|---|--------------------|-----------------|---------|------|-----|--------|----|------|--|
| LODS | 4 | 3 | 6 | | 6 | | | | 86 | |
| REP LODS | 4 | 5n | $\approx 4n+36$ | | | | 86 | | | |
| STOS | 4 | 2 | 6 | | 6 | | | | 86 | |
| REP STOS | 4 | 2n+3 | $\approx 3n+10$ | | | | 86 | | | |
| MOVS | 4 | 4 | 6 | | 4 | | | | 86 | |
| REP MOVS | 4 | $\approx 163+1.1n$ | | | | | 86 | | | |
| SCAS | 4 | 3 | | | 6 | | | | 86 | |
| REP SCAS | 4 | $\approx 40+6n$ | $\approx 4n$ | | | | | 86 | | |
| CMPS | 4 | 5 | | | 8 | | | | 86 | |
| REP CMPS | 4 | $\approx 50+8n$ | $\approx 4n$ | | | | | 86 | | |
| Other | | | | | | | | | | |
| NOP (90) | 1 | 0 | 0 | | 0.25 | 0/1 | alu0/1 | | 86 | |
| Long NOP (0F 1F) | 1 | 0 | 0 | | 0.25 | 0/1 | alu0/1 | | ppro | |
| PAUSE | 4 | 2 | | | | | | | sse2 | |
| CPUID | 4 | 39-81 | | 200-500 | | | p5 | | | |
| RDTSC | 4 | 7 | | | 80 | | | | p5 | |

Notes:

- a) Add 1 μ op if source is a memory operand.
- b) Uses an extra μ op (port 3) if SIB byte used. A SIB byte is needed if the memory operand has more than one pointer register, or a scaled index, or ESP is used as base pointer.
- c) Add 1 μ op if source or destination, but not both, is a high 8-bit register (AH, BH, CH, DH).
- d) Has (false) dependence on the flags in most cases.
- e) Not available on PMMX
- q) Latency is 12 in 16-bit real or virtual mode, 24 in 32-bit protected mode.

Floating point x87 instructions

| Instruction | Operands | μ ops | Microcode | Latency | Additional latency | Reciprocal throughput | Port | Execution unit | Subunit | Instruction set | Notes |
|-------------------|----------|-----------|-----------|--------------|--------------------|-----------------------|------|----------------|---------|-----------------|-------|
| Move instructions | | | | | | | | | | | |
| FLD | r | 1 | 0 | 6 | 0 | 1 | 0 | mov | | 87 | |
| FLD | m32/64 | 1 | 0 | ≈ 7 | 0 | 1 | 2 | load | | 87 | |
| FLD | m80 | 3 | 4 | | | 6 | 2 | load | | 87 | |
| FBLD | m80 | 3 | 75 | | | 90 | 2 | load | | 87 | |
| FST(P) | r | 1 | 0 | 6 | 0 | 1 | 0 | mov | | 87 | |
| FST(P) | m32/64 | 2 | 0 | ≈ 7 | | 2-3 | 0 | store | | 87 | |
| FSTP | m80 | 3 | 8 | | | 8 | 0 | store | | 87 | |
| FBSTP | m80 | 3 | 311 | | | 400 | 0 | store | | 87 | |
| FXCH | r | 1 | 0 | 0 | 0 | 1 | 0 | mov | | 87 | |
| FILD | m16 | 3 | 3 | ≈ 10 | | 6 | 2 | load | | 87 | |
| FILD | m32/64 | 2 | 0 | ≈ 10 | | 1 | 2 | load | | 87 | |
| FIST | m16 | 3 | 0 | ≈ 10 | | 2-4 | 0 | store | | 87 | |
| FIST | m32/64 | 2 | 0 | ≈ 10 | | 2-3 | 0 | store | | 87 | |

Pentium 4

| | | | | | | | | | | | |
|--------------------------------|-------|----|------|------|---|------|-----|-------|------|------|------|
| FISTP | m | 3 | 0 | ≈ 10 | | 2-4 | 0 | store | | 87 | |
| FLDZ | | 1 | 0 | | | 2 | 0 | mov | | 87 | |
| FLD1 | | 2 | 0 | | | 2 | 0 | mov | | 87 | |
| FCMOVcc | st0,r | 4 | 0 | 2-4 | 1 | 4 | 1 | fp | PPro | e | |
| FFREE | r | 3 | 0 | | | 4 | 0 | mov | | 87 | |
| FINCSTP, FDECSTP | | 1 | 0 | 0 | 0 | 1 | 0 | mov | | 87 | |
| FNSTSW | AX | 4 | 0 | 11 | 0 | 3 | 1 | | | 287 | |
| FSTSW | AX | 6 | 0 | 11 | 0 | 3 | 1 | | | 287 | |
| FNSTSW | m16 | 4 | 4 | | | 6 | 0 | | | 87 | |
| FNSTCW | m16 | 4 | 4 | | | 6 | 0 | | | 87 | |
| FLDCW | m16 | 4 | 7 | (3) | | (8) | 0,2 | | | 87 | f |
| Arithmetic instructions | | | | | | | | | | | |
| FADD(P),FSUB(R)(P) | r | 1 | 0 | 5 | 1 | 1 | 1 | fp | add | 87 | |
| FADD,FSUB(R) | m | 2 | 0 | 5 | 1 | 1 | 1 | fp | add | 87 | |
| FIADD,FISUB(R) | m16 | 3 | 4 | 6 | 0 | 6 | 1 | fp | add | 87 | |
| FIADD,FISUB(R) | m32 | 3 | 0 | 5 | 1 | 2 | 1 | fp | add | 87 | |
| FMUL(P) | r | 1 | 0 | 7 | 1 | 2 | 1 | fp | mul | 87 | |
| FMUL | m | 2 | 0 | 7 | 1 | 2 | 1 | fp | mul | 87 | |
| FIMUL | m16 | 3 | 4 | 7 | 1 | 6 | 1 | fp | mul | 87 | |
| FIMUL | m32 | 3 | 0 | 7 | 1 | 2 | 1 | fp | mul | 87 | |
| FDIV(R)(P) | r | 1 | 0 | 43 | 0 | 43 | 1 | fp | div | 87 | g, h |
| FDIV(R) | m | 2 | 0 | 43 | 0 | 43 | 1 | fp | div | 87 | g, h |
| FIDIV(R) | m16 | 3 | 4 | 43 | 0 | 43 | 1 | fp | div | 87 | g, h |
| FIDIV(R) | m32 | 3 | 0 | 43 | 0 | 43 | 1 | fp | div | 87 | g, h |
| FABS | | 1 | 0 | 2 | 1 | 1 | 1 | fp | misc | 87 | |
| FCHS | | 1 | 0 | 2 | 1 | 1 | 1 | fp | misc | 87 | |
| FCOM(P), FUCOM(P) | r | 1 | 0 | 2 | 0 | 1 | 1 | fp | misc | 87 | |
| FCOM(P) | m | 2 | 0 | 2 | 0 | 1 | 1 | fp | misc | 87 | |
| FCOMPP, FUCOMPP | | 2 | 0 | 2 | 0 | 1 | 1 | fp | misc | 87 | |
| FCOMI(P) | r | 3 | 0 | 10 | 0 | 3 | 0,1 | fp | misc | PPro | |
| FICOM(P) | m16 | 4 | 4 | | | 6 | 1 | fp | misc | 87 | |
| FICOM(P) | m32 | 3 | 0 | 2 | 0 | 2 | 1,2 | fp | misc | 87 | |
| FTST | | 1 | 0 | 2 | 0 | 1 | 1 | fp | misc | 87 | |
| FXAM | | 1 | 0 | 2 | 0 | 1 | 1 | fp | misc | 87 | |
| FRNDINT | | 3 | 15 | 23 | 0 | 15 | 0,1 | | | 87 | |
| FPREM | | 6 | 84 | 212 | | | 1 | fp | | 87 | |
| FPREM1 | | 6 | 84 | 212 | | | 1 | fp | | 387 | |
| Math | | | | | | | | | | | |
| FSQRT | | 1 | 0 | 43 | 0 | 43 | 1 | fp | div | 87 | g, h |
| FLDPI, etc. | | 2 | 0 | | | 3 | 1 | fp | | 87 | |
| FSIN | | 6 | ≈150 | ≈180 | | ≈170 | 1 | fp | | 387 | |
| FCOS | | 6 | ≈175 | ≈207 | | ≈207 | 1 | fp | | 387 | |
| FSINCOS | | 7 | ≈178 | ≈216 | | ≈211 | 1 | fp | | 387 | |
| FPTAN | | 6 | ≈160 | ≈230 | | ≈200 | 1 | fp | | 87 | |
| FPATAN | | 3 | 92 | ≈187 | | ≈153 | 1 | fp | | 87 | |
| FSCALE | | 3 | 24 | 57 | | 66 | 1 | fp | | 87 | |
| FTRACT | | 3 | 15 | 20 | | 20 | 1 | fp | | 87 | |
| F2XM1 | | 3 | 45 | ≈165 | | 63 | 1 | fp | | 87 | |
| FYL2X | | 3 | 60 | ≈200 | | 90 | 1 | fp | | 87 | |
| FYL2XP1 | | 11 | 134 | ≈242 | | ≈220 | 1 | fp | | 87 | |

Pentium 4

| Other | | | | | | | | | | | |
|---------|--|---|-----|-----|---|-----|-----|--|-----|-----|---|
| FNOP | | 1 | 0 | 1 | 0 | 1 | 0 | | mov | 87 | |
| (F)WAIT | | 2 | 0 | 0 | 0 | 1 | 0 | | mov | 87 | |
| FNCLEX | | 4 | 4 | | | 96 | 1 | | | 87 | |
| FNINIT | | 6 | 29 | | | 172 | | | | 87 | |
| FNSAVE | | 4 | 174 | 456 | | 420 | 0,1 | | | 87 | |
| FRSTOR | | 4 | 96 | 528 | | 532 | | | | 87 | |
| FXSAVE | | 4 | 69 | 132 | | 96 | | | | sse | i |
| FXRSTOR | | 4 | 94 | 208 | | 208 | | | | sse | i |

Notes:

- e) Not available on PMMX
- f) The latency for FLDCW is 3 when the new value loaded is the same as the value of the control word before the preceding FLDCW, i.e. when alternating between the same two values. In all other cases, the latency and reciprocal throughput is 143.
- g) Latency and reciprocal throughput depend on the precision setting in the F.P. control word. Single precision: 23, double precision: 38, long double precision (default): 43.
- h) Throughput of FP-MUL unit is reduced during the use of the FP-DIV unit.
- i) Takes 6 μ ops more and 40-80 clocks more when XMM registers are disabled.

Integer MMX and XMM instructions

| Instruction | Operands | μ ops | Microcode | Latency | Additional latency | Reciprocal throughput | Port | Execution unit | Subunit | Instruction set | Notes |
|--------------------------|----------|-----------|-----------|-------------|--------------------|-----------------------|------|----------------|---------|-----------------|-------|
| Move instructions | | | | | | | | | | | |
| MOVD | r32, mm | 2 | 0 | 5 | 1 | 1 | 0 | fp | | mmx | |
| MOVD | mm, r32 | 2 | 0 | 2 | 0 | 2 | 1 | mmx | alu | mmx | |
| MOVD | mm,m32 | 1 | 0 | ≈ 8 | 0 | 1 | 2 | load | | mmx | |
| MOVD | r32, xmm | 2 | 0 | 10 | 1 | 2 | 0 | fp | | sse2 | |
| MOVD | xmm, r32 | 2 | 0 | 6 | 1 | 2 | 1 | mmx | shift | sse2 | |
| MOVD | xmm,m32 | 1 | 0 | ≈ 8 | 0 | 1 | 2 | load | | sse2 | |
| MOVD | m32, r | 2 | 0 | ≈ 8 | | 2 | 0,1 | | | mmx | |
| MOVQ | mm,mm | 1 | 0 | 6 | 0 | 1 | 0 | mov | | mmx | |
| MOVQ | xmm,xmm | 1 | 0 | 2 | 1 | 2 | 1 | mmx | shift | sse2 | |
| MOVQ | r,m64 | 1 | 0 | ≈ 8 | | 1 | 2 | load | | mmx | |
| MOVQ | m64,r | 2 | 0 | ≈ 8 | | 2 | 0 | mov | | mmx | |
| MOVDQA | xmm,xmm | 1 | 0 | 6 | 0 | 1 | 0 | mov | | sse2 | |
| MOVDQA | xmm,m | 1 | 0 | ≈ 8 | | 1 | 2 | load | | sse2 | |
| MOVDQA | m,xmm | 2 | 0 | ≈ 8 | | 2 | 0 | mov | | sse2 | |
| MOVDQU | xmm,m | 4 | 0 | | | 2 | 2 | load | | sse2 | k |
| MOVDQU | m,xmm | 4 | 6 | | | 2 | 0 | mov | | sse2 | k |
| MOVDQ2Q | mm,xmm | 3 | 0 | 8 | 1 | 2 | 0,1 | mov-mmx | sse2 | | |
| MOVQ2DQ | xmm,mm | 2 | 0 | 8 | 1 | 2 | 0,1 | mov-mmx | sse2 | | |
| MOVNTQ | m,mm | 3 | 0 | | | 75 | 0 | mov | | sse | |
| MOVNTDQ | m,xmm | 2 | 0 | | | 18 | 0 | mov | | sse2 | |

Pentium 4

| | | | | | | | | | | | |
|--------------------------------|-----------|---|----|----|---|-----|-----|----------|-------|------|-----|
| PACKSSWB/DW | mm,r/m | 1 | 0 | 2 | 1 | 1 | 1 | mmx | shift | mmx | a |
| PACKUSWB | | | | | | | | | | | |
| PACKSSWB/DW | xmm,r/m | 1 | 0 | 4 | 1 | 2 | 1 | mmx | shift | mmx | a |
| PACKUSWB | | | | | | | | | | | |
| PUNPCKH/LBW/WD/DQ | mm,r/m | 1 | 0 | 2 | 1 | 1 | 1 | mmx | shift | mmx | a |
| PUNPCKHBW/WD/DQ/QDQ | xmm,r/m | 1 | 0 | 4 | 1 | 2 | 1 | mmx | shift | sse2 | a |
| PUNPCKLBW/WD/DQ/QDQ | xmm,r/m | 1 | 0 | 2 | 1 | 2 | 1 | mmx | shift | sse2 | a |
| PSHUFD | xmm,xmm,i | 1 | 0 | 4 | 1 | 2 | 1 | mmx | shift | sse2 | |
| PSHUFL/HW | xmm,xmm,i | 1 | 0 | 2 | 1 | 2 | 1 | mmx | shift | sse2 | |
| PSHUFW | mm,mm,i | 1 | 0 | 2 | 1 | 1 | 1 | mmx | shift | mmx | |
| MASKMOVQ | mm,mm | 4 | 4 | | | 7 | 0 | mov | | sse | |
| MASKMOVDQU | xmm,xmm | 4 | 6 | | | 10 | 0 | mov | | sse2 | |
| PMOVBMSKB | r32,r | 2 | 0 | 7 | 1 | 3 | 0,1 | mmx-alu0 | sse | | |
| PEXTRW | r32,mm,i | 3 | 0 | 8 | 1 | 2 | 1 | mmx-int | sse | | |
| PEXTRW | r32,xmm,i | 3 | 0 | 9 | 1 | 2 | 1 | mmx-int | sse2 | | |
| PINSRW | mm,r32,i | 2 | 0 | 3 | 1 | 2 | 1 | int-mmx | sse | | |
| PINSRW | xmm,r32,i | 2 | 0 | 4 | 1 | 2 | 1 | int-mmx | sse2 | | |
| Arithmetic instructions | | | | | | | | | | | |
| PADDB/W/D | | | | | | | | | | | |
| PADD(U)SB/W | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | mmx | a,j |
| PSUBB/W/D | | | | | | | | | | | |
| PSUB(U)SB/W | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | mmx | a,j |
| PADDQ, PSUBQ | mm,r/m | 1 | 0 | 2 | 1 | 1 | 1 | mmx | alu | sse2 | a |
| PADDQ, PSUBQ | xmm,r/m | 1 | 0 | 4 | 1 | 2 | 1 | fp | add | sse2 | a |
| PCMPEQB/W/D | | | | | | | | | | | |
| PCMPGTB/W/D | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | mmx | a,j |
| PMULLW PMULHW | r,r/m | 1 | 0 | 6 | 1 | 1,2 | 1 | fp | mul | mmx | a,j |
| PMULHUW | r,r/m | 1 | 0 | 6 | 1 | 1,2 | 1 | fp | mul | sse | a,j |
| PMADDWD | r,r/m | 1 | 0 | 6 | 1 | 1,2 | 1 | fp | mul | mmx | a,j |
| PMULUDQ | r,r/m | 1 | 0 | 6 | 1 | 1,2 | 1 | fp | mul | sse2 | a,j |
| PAVGB/W | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | sse | a,j |
| PMIN/MAXUB | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | sse | a,j |
| PMIN/MAXSW | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | sse | a,j |
| PSADBW | r,r/m | 1 | 0 | 4 | 1 | 1,2 | 1 | mmx | alu | sse | a,j |
| Logic | | | | | | | | | | | |
| PAND, PANDN | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | mmx | a,j |
| POR, PXOR | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | mmx | a,j |
| PSLL/RLW/D/Q, PSRAW/D | r,i/r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | shift | mmx | a,j |
| PSLLDQ, PSRLDQ | xmm,i | 1 | 0 | 4 | 1 | 2 | 1 | mmx | shift | sse2 | a |
| Other | | | | | | | | | | | |
| EMMS | | 4 | 11 | 12 | | 12 | 0 | | | mmx | |

Notes:

- a) Add 1 μ op if source is a memory operand.
j) Reciprocal throughput is 1 for 64 bit operands, and 2 for 128 bit operands.
k) It may be advantageous to replace this instruction by two 64-bit moves

Floating point XMM instructions

| Instruction | Operands | µops | Microcode | Latency | Additional latency | Reciprocal throughput | Port | Execution unit | Subunit | Instruction set | Notes |
|--------------------|----------|------|-----------|---------|--------------------|-----------------------|------|----------------|---------|-----------------|--------|
| Move instructions | | | | | | | | | | | |
| MOVAPS/D | r,r | 1 | 0 | 6 | 0 | 1 | 0 | mov | | sse | k k |
| MOVAPS/D | r,m | 1 | 0 | ≈ 7 | 0 | 1 | 2 | | | sse | |
| MOVAPS/D | m,r | 2 | 0 | ≈ 7 | | 2 | 0 | | | sse | |
| MOVUPS/D | r,r | 1 | 0 | 6 | 0 | 1 | 0 | mov | | sse | |
| MOVUPS/D | r,m | 4 | 0 | | | 2 | 2 | | | sse | |
| MOVUPS/D | m,r | 4 | 6 | | | 8 | 0 | | | sse | |
| MOVSS | r,r | 1 | 0 | 2 | 0 | 2 | 1 | mmx | shift | sse | |
| MOVSD | r,r | 1 | 0 | 2 | 1 | 2 | 1 | mmx | shift | sse | |
| MOVSS, MOVSD | r,m | 1 | 0 | ≈ 7 | 0 | 1 | 2 | | | sse | |
| MOVSS, MOVSD | m,r | 2 | 0 | | | 2 | 0 | | | sse | |
| MOVHPS | r,r | 1 | 0 | 4 | 0 | 2 | 1 | mmx | shift | sse | |
| MOVLHPS | r,r | 1 | 0 | 2 | 0 | 2 | 1 | mmx | shift | sse | |
| MOVHPS/D, MOVLPS/D | | | | | | | | | | | |
| | r,m | 3 | 0 | | | 4 | 2 | | | sse | |
| MOVHPS/D, MOVLPS/D | | | | | | | | | | | |
| | m,r | 2 | 0 | | | 2 | 0 | | | sse | |
| MOVNTPS/D | m,r | 2 | 0 | | | 4 | 0 | | | sse/2 | |
| MOVMSKPS/D | r32,r | 2 | 0 | 6 | 1 | 3 | 1 | fp | | sse | |
| SHUFPS/D | r,r/m,i | 1 | 0 | 4 | 1 | 2 | 1 | mmx | shift | sse | |
| UNPCKHPS/D | r,r/m | 1 | 0 | 4 | 1 | 2 | 1 | mmx | shift | sse | |
| UNPCKLPS/D | r,r/m | 1 | 0 | 2 | 1 | 2 | 1 | mmx | shift | sse | |
| Conversion | | | | | | | | | | | |
| CVTTPS2PD | r,r/m | 4 | 0 | 7 | 1 | 4 | 1 | mmx | shift | sse2 | a |
| CVTPD2PS | r,r/m | 2 | 0 | 10 | 1 | 2 | 1 | fp-mmx | sse2 | a | |
| CVTSD2SS | r,r/m | 4 | 0 | 14 | 1 | 6 | 1 | mmx | shift | sse2 | a |
| CVTSS2SD | r,r/m | 4 | 0 | 10 | 1 | 6 | 1 | mmx | shift | sse2 | a |
| CVTDQ2PS | r,r/m | 1 | 0 | 4 | 1 | 2 | 1 | fp | | sse2 | a |
| CVTDQ2PD | r,r/m | 3 | 0 | 9 | 1 | 4 | 1 | mmx-fp | sse2 | a | |
| CVT(T)PS2DQ | r,r/m | 1 | 0 | 4 | 1 | 2 | 1 | fp | | sse2 | a |
| CVT(T)PD2DQ | r,r/m | 2 | 0 | 9 | 1 | 2 | 1 | fp-mmx | sse2 | a | |
| CVTPI2PS | xmm,mm | 4 | 0 | 10 | 1 | 4 | 1 | mmx | | sse | a |
| CVTPI2PD | xmm,mm | 4 | 0 | 11 | 1 | 5 | 1 | fp-mmx | sse2 | a | |
| CVT(T)PS2PI | mm,xmm | 3 | 0 | 7 | 0 | 2 | 0,1 | fp-mmx | sse | a | |
| CVT(T)PD2PI | mm,xmm | 3 | 0 | 11 | 1 | 3 | 0,1 | fp-mmx | sse2 | a | |
| CVTSI2SS | xmm,r32 | 3 | 0 | 10 | 1 | 3 | 1 | fp-mmx | sse | a | |
| CVTSI2SD | xmm,r32 | 4 | 0 | 15 | 1 | 6 | 1 | fp-mmx | sse2 | a | |
| CVT(T)SD2SI | r32,xmm | 2 | 0 | 8 | 1 | 2.5 | 1 | fp | | sse2 | a |
| CVT(T)SS2SI | r32,xmm | 2 | 0 | 8 | 1 | 2.5 | 1 | fp | | sse | a |
| Arithmetic | | | | | | | | | | | |
| ADDPS/D ADDSS/D | r,r/m | 1 | 0 | 4 | 1 | 2 | 1 | fp | add | sse | a |
| SUBPS/D SUBSS/D | r,r/m | 1 | 0 | 4 | 1 | 2 | 1 | fp | add | sse | a |

Pentium 4

| | | | | | | | | | | | |
|------------------------------------|-------|---|---|----|---|-----|---|-----|-----|------|-----|
| MULPS/D MULSS/D | r,r/m | 1 | 0 | 6 | 1 | 2 | 1 | fp | mul | sse | a |
| DIVSS | r,r/m | 1 | 0 | 23 | 0 | 23 | 1 | fp | div | sse | a,h |
| DIVPS | r,r/m | 1 | 0 | 39 | 0 | 39 | 1 | fp | div | sse | a,h |
| DIVSD | r,r/m | 1 | 0 | 38 | 0 | 38 | 1 | fp | div | sse2 | a,h |
| DIVPD | r,r/m | 1 | 0 | 69 | 0 | 69 | 1 | fp | div | sse2 | a,h |
| RCPPS RCPSS | r,r/m | 2 | 0 | 4 | 1 | 4 | 1 | mmx | | sse | a |
| MAXPS/D MAXSS/ DMINPS/D MINSS/D | r,r/m | 1 | 0 | 4 | 1 | 2 | 1 | fp | add | sse | a |
| CMPccPS/D | r,r/m | 1 | 0 | 4 | 1 | 2 | 1 | fp | add | sse | a |
| CMPccSS/D | r,r/m | 2 | 0 | 6 | 1 | 3 | 1 | fp | add | sse | a |
| COMISS/D UCOMISS/D | r,r/m | 2 | 0 | 6 | 1 | 3 | 1 | fp | add | sse | a |
| Logic | | | | | | | | | | | |
| ANDPS/D ANDNPS/D ORPS/D XORPS/D | r,r/m | 1 | 0 | 2 | 1 | 2 | 1 | mmx | alu | sse | a |
| Math | | | | | | | | | | | |
| SQRTSS | r,r/m | 1 | 0 | 23 | 0 | 23 | 1 | fp | div | sse | a,h |
| SQRTPS | r,r/m | 1 | 0 | 39 | 0 | 39 | 1 | fp | div | sse | a,h |
| SQRTSD | r,r/m | 1 | 0 | 38 | 0 | 38 | 1 | fp | div | sse2 | a,h |
| SQRTPD | r,r/m | 1 | 0 | 69 | 0 | 69 | 1 | fp | div | sse2 | a,h |
| RSQRTSS | r,r/m | 2 | 0 | 4 | 1 | 3 | 1 | mmx | | sse | a |
| RSQRTPS | r,r/m | 2 | 0 | 4 | 1 | 4 | 1 | mmx | | sse | a |
| Other | | | | | | | | | | | |
| LDMXCSR | m | 4 | 8 | 98 | | 100 | 1 | | | sse | |
| STMXCSR | m | 4 | 4 | | | 6 | 1 | | | sse | |

Notes:

- a) Add 1 μ op if source is a memory operand.
- h) Throughput of FP-MUL unit is reduced during the use of the FP-DIV unit.
- k) It may be advantageous to replace this instruction by two 64-bit moves.

Intel Pentium 4 w. EM64T (Prescott)

List of instruction timings and μ op breakdown

Explanation of column headings:

| | |
|-------------------------------|--|
| Instruction: | Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc. |
| Operands: | i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, xmm = 128 bit xmm register, sr = segment register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc., mabs = memory operand with 64-bit absolute address. |
| μops: | Number of μ ops issued from instruction decoder and stored in trace cache. |
| Microcode: | Number of additional μ ops issued from microcode ROM. |
| Latency: | This is the delay that the instruction generates in a dependency chain if the next dependent instruction starts in the same execution unit. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latency of moves to and from memory cannot be measured accurately because of the problem with memory intermediates explained above under "How the values were measured". |
| Additional latency: | This number is added to the latency if the next dependent instruction is in a different execution unit. There is no additional latency between ALU0 and ALU1. |
| Reciprocal throughput: | This is also called issue latency. This value indicates the number of clock cycles from the execution of an instruction begins to a subsequent independent instruction can begin to execute in the same execution subunit. A value of 0.25 indicates 4 instructions per clock cycle in one thread. |
| Port: | The port through which each μ op goes to an execution unit. Two independent μ ops can start to execute simultaneously only if they are going through different ports. |
| Execution unit: | Use this information to determine additional latency. When an instruction with more than one μ op uses more than one execution unit, only the first and the last execution unit is listed. |
| Execution subunit: | Throughput measures apply only to instructions executing in the same subunit. |
| Instruction set | Indicates the compatibility of an instruction with other 80x86 family microprocessors. The instruction can execute on microprocessors that support the instruction set indicated. |

Integer instructions

| Instruction | Operands | μ ops | Microcode | Latency | Additional latency | Reciprocal throughput | Port | Execution unit | Subunit | Instruction set | Notes |
|--------------------------|------------|-----------|-----------|---------|--------------------|-----------------------|------|----------------|---------|-----------------|-------|
| Move instructions | | | | | | | | | | | |
| MOV | r,r | 1 | 0 | 1 | 0 | 0.25 | 0/1 | alu0/1 | | 86 | c |
| MOV | r8/16/32,i | 1 | 0 | 1 | 0 | 0.25 | 0/1 | alu0/1 | | 86 | |
| MOV | r64,i32 | 1 | 0 | | 0 | 0.5 | 0/1 | alu0/1 | | x64 | |
| MOV | r64,i64 | 2 | 0 | | 0 | 1 | 1 | alu1 | | x64 | |

Prescott

| | | | | | | | | | | | |
|---------------|--------------|---|----------|-----------|---|-------|-----|--------|----|------|-------|
| MOV | r8/16,m | 2 | 0 | 3 | 0 | 1 | 2 | load | | 86 | |
| MOV | r32/64,m | 1 | 0 | 2 | 0 | 1 | 2 | load | | 86 | |
| MOV | m,r | 1 | 0 | | | 2 | 0 | store | | 86 | b,c |
| MOV | m,i | 2 | 0 | | | 2 | 0,3 | store | | 86 | |
| MOV | m64,i32 | 2 | 0 | | | 2 | 0,3 | store | | x64 | |
| MOV | r,sr | 1 | 2 | | | 8 | | | | 86 | |
| MOV | sr,r/m | 1 | 8 | | | 27 | | | | 86 | a,q |
| MOV | r,mabs | 3 | 0 | | | 1 | | | | x64 | l |
| MOV | mabs,r | 3 | 0 | | | 2 | | | | x64 | l |
| MOVNTI | m,r32 | 2 | 0 | | | 2 | | | | sse2 | |
| MOVZX | r,r | 1 | 0 | 1 | 0 | 0.25 | 0/1 | alu0/1 | | 386 | c |
| MOVZX | r16,r8 | 2 | 0 | 2 | 0 | 1 | 0/1 | alu0/1 | | 386 | c |
| MOVZX | r,m | 1 | 0 | 2 | 0 | 1 | 2 | load | | 386 | |
| MOVSX | r16,r8 | 2 | 0 | 2 | 0 | 1 | 0 | alu0 | | 386 | a,c,o |
| MOVSX | r32/64,r8/16 | 1 | 0 | 1 | 0 | 0.5 | 0 | alu0 | | 386 | a,c,o |
| MOVSX | r,m | 2 | 0 | 3 | 0 | 1 | 2 | load | | 386 | |
| MOVSLD | r64,r32 | 1 | 0 | 1 | 0 | 0.5 | 0 | alu0 | | x64 | a |
| CMOVC | r,r/m | 3 | 0 | 9.5 | 0 | 3 | | | | PPro | a,e |
| XCHG | r,r | 3 | 0 | 2 | 0 | 1 | 0/1 | alu0/1 | | 86 | |
| XCHG | r,m | 2 | 6 | ≈100 | | | | | | 86 | |
| XLAT | | 4 | 0 | 6 | | | | | | 86 | |
| PUSH | r | 2 | 0 | 2 | | 2 | | | | 86 | |
| PUSH | i | 2 | 0 | 2 | | 2 | | | | 186 | |
| PUSH | m | 3 | 0 | 2 | | 2 | | | | 86 | |
| PUSH | sr | 1 | 3 | | | 9 | | | | 86 | |
| PUSHF(D/Q) | | 1 | 3 | | | 9 | | | | 86 | |
| PUSHA(D) | | 1 | 9 | | | 16 | | | | 186 | m |
| POP | r | 2 | 0 | 1 | 0 | 1 | | | | 86 | |
| POP | m | 2 | 6 | | | 10 | | | | 86 | |
| POP | sr | 1 | 8 | | | 30 | | | | 86 | |
| POPF(D/Q) | | 1 | 8 | | | 70 | | | | 86 | |
| POPA(D) | | 2 | 16 | | | 15 | | | | 186 | m |
| LEA | r,[m] | 1 | 0 | | | 0.25 | 0/1 | alu0/1 | | 86 | p |
| LEA | r,[r+r/i] | 1 | 0 | 2.5 | 0 | 0.25 | 0/1 | alu0/1 | | 86 | |
| LEA | r,[r+r+i] | 2 | 0 | 3.5 | 0 | 0.5 | 0/1 | alu0/1 | | 86 | |
| LEA | r,[r*i] | 3 | 0 | 3.5 | 0 | 1 | 1 | alu | | 386 | |
| LEA | r,[r+r*i] | 2 | 0 | 3.5 | 0 | 1 | 0,1 | alu0,1 | | 386 | |
| LEA | r,[r+r*i+i] | 3 | 0 | 3.5 | 0 | 1 | 1 | alu | | 386 | |
| LAHF | | 1 | 0 | 4 | 0 | | 1 | int | | 86 | n |
| SAHF | | 1 | 0 | 5 | 0 | | 0/1 | alu0/1 | | 86 | d,n |
| SALC | | 2 | 0 | | 0 | 1 | 1 | int | | 86 | m |
| LDS, LES, ... | r,m | 2 | 10 | | | 28 | | | | 86 | m |
| LODS | | 1 | 3 | 8 | | 8 | | | | 86 | |
| REP LODS | | 1 | 5n | ≈ 4n+50 | | | | 86 | | | |
| STOS | | 1 | 2 | 8 | | 8 | | | | 86 | |
| REP STOS | | 1 | 2.5n | ≈ 3n | | | | 86 | | | |
| MOVS | | 1 | 4 | 8 | | 8 | | | | 86 | |
| REP MOVSB | | 9 | ≈.3n | ≈.3n | | | | 86 | | | |
| REP MOVSW | | 1 | ≈.5-1.1n | ≈ .6-1.4n | | | | 86 | | | |
| REP MOVSD | | 1 | ≈1.1n | ≈ 1.4 n | | | | 86 | | | |
| REP MOVSQ | | 1 | ≈1.1n | ≈ 1.4 n | | | | x64 | | | |
| BSWAP | r | 1 | 0 | 1 | 0 | 1 | | alu | | 486 | |
| IN, OUT | r,r/i | 1 | 52 | | | >1000 | | | 86 | | |

Prescott

| | | | | | | | | | | | |
|--------------------------------|-------------|---|----|----|---|-------|-----|---------|-------|------|---|
| PREFETCHNTA | m | 1 | 0 | | | 1 | | | | sse | |
| PREFETCHT0/1/2 | m | 1 | 0 | | | 1 | | | | sse | |
| SFENCE | | 1 | 2 | | | 50 | | | | sse | |
| LFENCE | | 1 | 2 | | | 50 | | | | sse2 | |
| MFENCE | | 1 | 4 | | | 124 | | | | sse2 | |
| Arithmetic instructions | | | | | | | | | | | |
| ADD, SUB | r,r | 1 | 0 | 1 | 0 | 0.25 | 0/1 | alu0/1 | | 86 | c |
| ADD, SUB | r,m | 2 | 0 | 1 | 0 | 1 | | | | 86 | c |
| ADD, SUB | m,r | 3 | 0 | 5 | | 2 | | | | 86 | c |
| ADC, SBB | r,r/i | 3 | 0 | 10 | 0 | 10 | 1 | int,alu | | 86 | |
| ADC, SBB | r,m | 2 | 5 | 10 | 0 | 10 | 1 | int,alu | | 86 | |
| ADC, SBB | m,r | 2 | 6 | 20 | | 10 | | | | 86 | |
| ADC, SBB | m,i | 3 | 5 | 22 | | 10 | | | | 86 | |
| CMP | r,r | 1 | 0 | 1 | 0 | 0.25 | 0/1 | alu0/1 | | 86 | c |
| CMP | r,m | 2 | 0 | 1 | 0 | 1 | | | | 86 | c |
| INC, DEC | r | 2 | 0 | 1 | 0 | 0.5 | 0/1 | alu0/1 | | 86 | |
| INC, DEC | m | 4 | 0 | 5 | | 3 | | | | 86 | |
| NEG | r | 1 | 0 | 1 | 0 | 0.5 | 0 | alu0 | | 86 | |
| NEG | m | 3 | 0 | 5 | | 3 | | | | 86 | |
| AAA, AAS | | 1 | 10 | 26 | | | | | | 86 | m |
| DAA, DAS | | 1 | 16 | 29 | | | | | | 86 | m |
| AAD | | 2 | 5 | 13 | | | 1 | int | mul | 86 | m |
| AAM | | 2 | 17 | 71 | | | 1 | int | fpdiv | 86 | m |
| MUL, IMUL | r8 | 1 | 0 | 10 | 0 | | 1 | int | mul | 86 | |
| MUL, IMUL | r16 | 4 | 0 | 11 | 0 | | 1 | int | mul | 86 | |
| MUL, IMUL | r32 | 3 | 0 | 11 | 0 | | 1 | int | mul | 86 | |
| MUL, IMUL | r64 | 1 | 5 | 11 | 0 | | 1 | int | mul | x64 | |
| MUL, IMUL | m8 | 2 | 0 | 10 | 0 | | 1 | int | mul | 86 | |
| MUL, IMUL | m16 | 2 | 5 | 11 | 0 | | 1 | int | mul | 86 | |
| MUL, IMUL | m32 | 3 | 0 | 11 | 0 | | 1 | int | mul | 86 | |
| MUL, IMUL | m64 | 2 | 6 | 11 | 0 | | 1 | int | mul | x64 | |
| IMUL | r16,r16 | 1 | 0 | 10 | 0 | 2.5 | 1 | int | mul | 386 | |
| IMUL | r16,r16,i | 2 | 0 | 11 | 0 | 2.5 | 1 | int | mul | 186 | |
| IMUL | r32,r32 | 1 | 0 | 10 | 0 | 2.5 | 1 | int | mul | 386 | |
| IMUL | r32,(r32),i | 1 | 0 | 10 | 0 | 2.5 | 1 | int | mul | 386 | |
| IMUL | r64,r64 | 1 | 0 | 10 | 0 | 2.5 | 1 | int | mul | x64 | |
| IMUL | r64,(r64),i | 1 | 0 | 10 | 0 | 2.5 | 1 | int | mul | x64 | |
| IMUL | r16,m16 | 2 | 0 | 10 | 0 | 2.5 | 1 | int | mul | 386 | |
| IMUL | r32,m32 | 2 | 0 | 10 | 0 | 2.5 | 1 | int | mul | 386 | |
| IMUL | r64,m64 | 2 | 0 | 10 | 0 | 2.5 | 1 | int | mul | x64 | |
| IMUL | r,m,i | 3 | 0 | 10 | 0 | 1-2.5 | 1 | int | mul | 186 | |
| DIV | r8/m8 | 1 | 20 | 74 | 0 | 34 | 1 | int | fpdiv | 86 | a |
| DIV | r16/m16 | 1 | 19 | 73 | 0 | 34 | 1 | int | fpdiv | 86 | a |
| DIV | r32/m32 | 1 | 21 | 76 | 0 | 34 | 1 | int | fpdiv | 386 | a |
| DIV | r64/m64 | 1 | 31 | 63 | 0 | 52 | 1 | int | fpdiv | x64 | a |
| IDIV | r8/m8 | 1 | 21 | 76 | 0 | 34 | 1 | int | fpdiv | 86 | a |
| IDIV | r16/m16 | 1 | 19 | 79 | 0 | 34 | 1 | int | fpdiv | 86 | a |
| IDIV | r32/m32 | 1 | 19 | 79 | 0 | 34 | 1 | int | fpdiv | 386 | a |
| IDIV | r64/m64 | 1 | 58 | 96 | 0 | 91 | 1 | int | fpdiv | x64 | a |
| CBW | | 2 | 0 | 2 | 0 | 1 | 0 | alu0 | | 86 | |
| CWD | | 2 | 0 | 2 | 0 | 1 | 0/1 | alu0/1 | | 86 | |
| CDQ | | 1 | 0 | 1 | 0 | 1 | 0/1 | alu0/1 | | 386 | |

Prescott

| | | | | | | | | | | | |
|---------------|---------------|---|-----------------|----|--------------|-----|-----|--------|----|-----|---|
| CQO | | 1 | 0 | 7 | 0 | 1 | 0/1 | alu0/1 | | x64 | |
| CWDE | | 2 | 0 | 2 | 0 | 1 | 0/1 | alu0/1 | | 386 | |
| CDQE | | 1 | 0 | 1 | 0 | 1 | 0/1 | alu0/1 | | x64 | |
| SCAS | | 1 | 3 | | 0 | 8 | | | | 86 | |
| REP SCAS | | 1 | $\approx 54+6n$ | | $\approx 4n$ | | | | 86 | | |
| CMPS | | 1 | 5 | | | 10 | | | | 86 | |
| REP CMPS | | 1 | $\approx 81+8n$ | | $\approx 5n$ | | | | 86 | | |
| Logic | | | | | | | | | | | |
| AND, OR, XOR | r,r | 1 | 0 | 1 | 0 | 0.5 | 0 | alu0 | | 86 | c |
| AND, OR, XOR | r,m | 2 | 0 | 1 | 0 | 1 | | | | 86 | c |
| AND, OR, XOR | m,r | 3 | 0 | 5 | | 2 | | | | 86 | c |
| TEST | r,r | 1 | 0 | 1 | 0 | 0.5 | 0 | alu0 | | 86 | c |
| TEST | r,m | 2 | 0 | 1 | 0 | 1 | | | | 86 | c |
| NOT | r | 1 | 0 | 1 | 0 | 0.5 | 0 | alu0 | | 86 | |
| NOT | m | 3 | 0 | 5 | | 2 | | | | 86 | |
| SHL | r,i | 1 | 0 | 1 | 0 | 0.5 | 1 | alu1 | | 186 | |
| SHR, SAR | r8/16/32,i | 1 | 0 | 1 | 0 | 0.5 | 1 | alu1 | | 186 | |
| SHR, SAR | r64,i | 1 | 0 | 7 | 0 | 2 | 1 | alu1 | | x64 | |
| SHL | r,CL | 2 | 0 | 2 | 0 | 2 | 1 | alu1 | | 86 | |
| SHR, SAR | r8/16/32,CL | 2 | 0 | 2 | 0 | 2 | 1 | alu1 | | 86 | |
| SHR, SAR | r64,CL | 2 | 0 | 8 | 0 | | 1 | alu1 | | x64 | |
| ROL, ROR | r8/16/32,i | 1 | 0 | 1 | 0 | 1 | 1 | alu1 | | 186 | d |
| ROL, ROR | r64,i | 1 | 0 | 7 | 0 | 7 | 1 | alu1 | | x64 | d |
| ROL, ROR | r8/16/32,CL | 2 | 0 | 2 | 0 | 2 | 1 | alu1 | | 86 | d |
| ROL, ROR | r64,CL | 2 | 0 | 8 | 0 | 8 | 1 | alu1 | | x64 | d |
| RCL, RCR | r,1 | 1 | 0 | 7 | 0 | 7 | 1 | alu1 | | 86 | d |
| RCL | r,i | 2 | 11 | 31 | 0 | 31 | 1 | alu1 | | 186 | d |
| RCR | r,i | 2 | 11 | 25 | 0 | 25 | 1 | alu1 | | 186 | d |
| RCL | r,CL | 1 | 11 | 31 | 0 | 31 | 1 | alu1 | | 86 | d |
| RCR | r,CL | 1 | 11 | 25 | 0 | 25 | 1 | alu1 | | 86 | d |
| SHL, SHR, SAR | m8/16/32,i | 3 | 6 | 10 | 0 | | 1 | alu1 | | 86 | |
| ROL, ROR | m8/16/32,i | 3 | 6 | 10 | 0 | | 1 | alu1 | | 86 | d |
| SHL, SHR, SAR | m8/16/32,cl | 2 | 6 | 10 | 0 | | 1 | alu1 | | 86 | |
| ROL, ROR | m8/16/32,cl | 2 | 6 | 10 | 0 | | 1 | alu1 | | 86 | d |
| RCL, RCR | m8/16/32,1 | 2 | 5 | 27 | 0 | 27 | 1 | alu1 | | 86 | d |
| RCL, RCR | m8/16/32,i | 3 | 13 | 38 | 0 | 38 | 1 | alu1 | | 86 | d |
| RCL, RCR | m8/16/32,cl | 2 | 13 | 37 | 0 | 37 | 1 | alu1 | | 86 | d |
| SHLD, SHRD | r8/16/32,r,i | 3 | 0 | 8 | 0 | 7 | 1 | alu1 | | 386 | |
| SHLD | r64,r64,i | 4 | 5 | 10 | 0 | | 1 | alu1 | | x64 | |
| SHRD | r64,r64,i | 3 | 7 | 10 | 0 | | 1 | alu1 | | x64 | |
| SHLD, SHRD | r8/16/32,r,cl | 4 | 0 | 9 | 0 | 8 | 1 | alu1 | | 386 | |
| SHLD | r64,r64,cl | 4 | 5 | 14 | 0 | | 1 | alu1 | | x64 | |
| SHRD | r64,r64,cl | 3 | 8 | 12 | 0 | | 1 | alu1 | | x64 | |
| SHLD, SHRD | m,r,i | 3 | 8 | 20 | 0 | 10 | 1 | alu1 | | 386 | |
| SHLD, SHRD | m,r,CL | 2 | 8 | 20 | 0 | 10 | 1 | alu1 | | 386 | |
| BT | r,i | 1 | 0 | 8 | 0 | 8 | 1 | alu1 | | 386 | d |
| BT | r,r | 2 | 0 | 9 | 0 | 9 | 1 | alu1 | | 386 | d |
| BT | m,i | 3 | 0 | 8 | 0 | 8 | 1 | alu1 | | 386 | d |
| BT | m,r | 2 | 7 | 10 | 0 | 10 | 1 | alu1 | | 386 | d |
| BTR, BTS, BTC | r,i | 1 | 0 | 8 | 0 | 8 | 1 | alu1 | | 386 | |
| BTR, BTS, BTC | r,r | 2 | 0 | 9 | 0 | 9 | 1 | alu1 | | 386 | |
| BTR, BTS, BTC | m,i | 3 | 6 | 28 | 0 | 10 | 1 | alu1 | | 386 | |

Prescott

| | | | | | | | | | | | |
|--------------------------------------|------------|-----|-------|----|---|---------|-----|--------|--------|--------|---|
| BTR, BTS, BTC | m,r | 2 | 10 | 14 | 0 | 14 | 1 | alu1 | | 386 | |
| BSF, BSR | r,r/m | 2 | 0 | 16 | 0 | 4 | 1 | alu1 | | 386 | |
| SETcc | r | 2 | 0 | 9 | 0 | 1 | 1 | int | | 386 | |
| SETcc | m | 3 | 0 | 9 | 0 | 2 | 1 | int | | 386 | |
| CLC, STC | | 2 | 0 | | 0 | 8 | | | | 86 | d |
| CMC | | 3 | 0 | 15 | 0 | | | | | 86 | |
| CLD, STD | | 1 | 8 | | 0 | 53 | | | | 86 | |
| Control transfer instructions | | | | | | | | | | | |
| JMP | short/near | 1 | 0 | 0 | 0 | 1 | 0 | alu0 | branch | 86 | |
| JMP | far | 2 | 25 | | | 154 | 0 | | | 86 | m |
| JMP | r | 3 | 0 | | | 15 | 0 | alu0 | branch | 86 | |
| JMP | m(near) | 3 | 0 | | | 10 | 0 | alu0 | branch | 86 | |
| JMP | m(far) | 2 | 28 | | | 157 | 0 | | | 86 | |
| Jcc | short/near | 1 | 0 | | | 2-4 | 0 | alu0 | branch | 86 | |
| J(E)CXZ | short | 4 | 0 | | | 4 | 0 | alu0 | branch | 86 | |
| LOOP | short | 4 | 0 | | | 4 | 0 | alu0 | branch | 86 | |
| CALL | near | 3 | 0 | | | 7 | 0 | alu0 | branch | 86 | |
| CALL | far | 3 | 29 | | | 160 | 0 | | | 86 | m |
| CALL | r | 4 | 0 | | | 7 | 0 | alu0 | branch | 86 | |
| CALL | m(near) | 4 | 0 | | | 9 | 0 | alu0 | branch | 86 | |
| CALL | m(far) | 2 | 32 | | | 160 | 0 | | | 86 | |
| RETN | | 4 | 0 | | | 7 | 0 | alu0 | branch | 86 | |
| RETN | i | 4 | 0 | | | 7 | 0 | alu0 | branch | 86 | |
| RETF | | 1 | 30 | | | 160 | 0 | | | 86 | |
| RETF | i | 2 | 30 | | | 160 | 0 | | | 86 | |
| IRET | | 1 | 49 | | | 325 | 0 | | | 86 | |
| BOUND | m | 2 | 11 | | | 12 | | | | 186 | m |
| INT | i | 2 | 67 | | | 470 | | | | 86 | |
| INTO | | 1 | 4 | | | 26 | | | | 86 | m |
| Other | | | | | | | | | | | |
| NOP (90) | | 1 | 0 | 0 | | 0.25 | 0/1 | alu0/1 | | 86 | |
| Long NOP (0F 1F) | | 1 | 0 | 0 | | 0.25 | 0/1 | alu0/1 | | ppro | |
| PAUSE | | 1 | 2 | | | 50 | | | | sse2 | |
| LEAVE | | 4 | 0 | 5 | | 5 | | | | 186 | |
| CLI | | 1 | 5 | | | 52 | | | | 86 | |
| STI | | 1 | 11 | | | 64 | | | | 86 | |
| CPUID | | 1 | 49-90 | | | 300-500 | | p5 | | | |
| RDTSC | | 1 | 12 | | | 100 | | | | p5 | |
| RDPMC (bit 31 = 1) | 1 | 37 | | | | 100 | | | p5 | | |
| RDPMC (bit 31 = 0) | 4 | 154 | | | | 240 | | | p5 | | |
| MONITOR | | | | | | | | | | (sse3) | |
| MWAIT | | | | | | | | | | (sse3) | |

Notes:

- a) Add 1 μ op if source is a memory operand.
- b) Uses an extra μ op (port 3) if SIB byte used.
- c) Add 1 μ op if source or destination, but not both, is a high 8-bit register (AH, BH, CH, DH).
- d) Has (false) dependence on the flags in most cases.
- e) Not available on PMMX
- l) Move accumulator to/from memory with 64 bit absolute address (opcode A0 - A3).

- m) Not available in 64 bit mode.
- n) Not available in 64 bit mode on some processors.
- o) MOVSB uses an extra μop if the destination register is smaller than the biggest register size available. Use a 32 bit destination register in 16 bit and 32 bit mode, and a 64 bit destination register in 64 bit mode for optimal performance.
- p) LEA with a direct memory operand has 1 μop and a reciprocal throughput of 0.25. This also applies if there is a RIP-relative address in 64-bit mode. A sign-extended 32-bit direct memory operand in 64-bit mode without RIP-relative address takes 2 μops because of the SIB byte. The throughput is 1 in this case. You may use a MOV instead.
- q) These values are measured in 32-bit mode. In 16-bit real mode there is 1 microcode μop and a reciprocal throughput of 17.

Floating point x87 instructions

| Instruction | Operands | μops | Microcode | Latency | Additional latency | Reciprocal throughput | Port | Execution unit | Subunit | Instruction set | Notes |
|--------------------------------|----------|-----------------|-----------|---------|--------------------|-----------------------|------|----------------|---------|-----------------|-------|
| Move instructions | | | | | | | | | | | |
| FLD | r | 1 | 0 | 7 | 0 | 1 | 0 | mov | | 87 | |
| FLD | m32/64 | 1 | 0 | | 0 | 1 | 2 | load | | 87 | |
| FLD | m80 | 3 | 3 | | | 8 | 2 | load | | 87 | |
| FBLD | m80 | 3 | 74 | | | 90 | 2 | load | | 87 | |
| FST(P) | r | 1 | 0 | 7 | 0 | 1 | 0 | mov | | 87 | |
| FST(P) | m32/64 | 2 | 0 | 7 | | 2 | 0 | store | | 87 | |
| FSTP | m80 | 3 | 6 | | | 10 | 0 | store | | 87 | |
| FBSTP | m80 | 3 | 311 | | | 400 | 0 | store | | 87 | |
| FXCH | r | 1 | 0 | 0 | 0 | 1 | 0 | mov | | 87 | |
| FILD | m16 | 3 | 2 | | | 8 | 2 | load | | 87 | |
| FILD | m32/64 | 2 | 0 | | | 2 | 2 | load | | 87 | |
| FIST(P) | m | 3 | 0 | | | 2.5 | 0 | store | | 87 | |
| FISTTP | m | 3 | 0 | | | 2.5 | 0 | store | | sse3 | |
| FLDZ | | 1 | 0 | | | 2 | 0 | mov | | 87 | |
| FLD1 | | 2 | 0 | | | 2 | 0 | mov | | 87 | |
| FCMOVcc | st0,r | 4 | 0 | 5 | 1 | 4 | 1 | fp | | PPro | e |
| FFREE | r | 3 | 0 | | | 3 | 0 | mov | | 87 | |
| FINCSTP, FDECSTP | | 1 | 0 | 0 | 0 | 1 | 0 | mov | | 87 | |
| FNSTSW | AX | 4 | 0 | | 0 | 3 | 1 | | | 287 | |
| FSTSW | AX | 6 | 0 | | 0 | 3 | 1 | | | 287 | |
| FNSTSW | m16 | 2 | 3 | | | 8 | 0 | | | 87 | |
| FNSTCW | m16 | 4 | 0 | | | 3 | 0 | | | 87 | |
| FLDCW | m16 | 3 | 6 | | | 10 | 0,2 | | | 87 | f |
| Arithmetic instructions | | | | | | | | | | | |
| FADD(P),FSUB(R)(P) | r | 1 | 0 | 6 | 1 | 1 | 1 | fp | add | 87 | |
| FADD,FSUB(R) | m | 2 | 0 | 6 | 1 | 1 | 1 | fp | add | 87 | |
| FIADD,FISUB(R) | m16 | 3 | 3 | 7 | 1 | 6 | 1 | fp | add | 87 | |
| FIADD,FISUB(R) | m32 | 3 | 0 | 6 | 1 | 2 | 1 | fp | add | 87 | |

Prescott

| | | | | | | | | | | | |
|-------------------|-----|---|------|------|---|------|-----|----|------|------|-----|
| FMUL(P) | r | 1 | 0 | 8 | 1 | 2 | 1 | fp | mul | 87 | |
| FMUL | m | 2 | 0 | 8 | 1 | 2 | 1 | fp | mul | 87 | |
| FIMUL | m16 | 3 | 3 | 8 | 1 | 8 | 1 | fp | mul | 87 | |
| FIMUL | m32 | 3 | 0 | 8 | 1 | 3 | 1 | fp | mul | 87 | |
| FDIV(R)(P) | r | 1 | 0 | 45 | 1 | 45 | 1 | fp | div | 87 | g,h |
| FDIV(R) | m | 2 | 0 | 45 | 1 | 45 | 1 | fp | div | 87 | g,h |
| FIDIV(R) | m16 | 3 | 3 | 45 | 1 | 45 | 1 | fp | div | 87 | g,h |
| FIDIV(R) | m32 | 3 | 3 | 45 | 1 | 45 | 1 | fp | div | 87 | g,h |
| FABS | | 1 | 0 | 3 | 1 | 1 | 1 | fp | misc | 87 | |
| FCHS | | 1 | 0 | 3 | 1 | 1 | 1 | fp | misc | 87 | |
| FCOM(P), FUCOM(P) | r | 1 | 0 | 3 | 0 | 1 | 1 | fp | misc | 87 | |
| FCOM(P) | m | 2 | 0 | 3 | 0 | 1 | 1 | fp | misc | 87 | |
| FCOMPP, FUCOMPP | | 2 | 0 | 3 | 0 | 1 | 1 | fp | misc | 87 | |
| FCOMI(P) | r | 3 | 0 | | | 3 | 0,1 | fp | misc | PPro | |
| FICOM(P) | m16 | 3 | 3 | | | 8 | 1 | fp | misc | 87 | |
| FICOM(P) | m32 | 3 | 0 | | | 2 | 1,2 | fp | misc | 87 | |
| FTST | | 1 | 0 | | | 1 | 1 | fp | misc | 87 | |
| FXAM | | 1 | 0 | | | 1 | 1 | fp | misc | 87 | |
| FRNDINT | | 3 | 14 | 28 | 1 | 16 | 0,1 | | | 87 | |
| FPREM | | 8 | 86 | 220 | 1 | | 1 | fp | | 87 | |
| FPREM1 | | 9 | 92 | 220 | 1 | | 1 | fp | | 387 | |
| Math | | | | | | | | | | | |
| FSQRT | | 1 | 0 | 45 | 1 | 45 | 1 | fp | div | 87 | g,h |
| FLDPI, etc. | | 2 | 0 | | | 2 | 1 | fp | | 87 | |
| FSIN, FCOS | | 3 | ≈100 | ≈200 | | ≈200 | 1 | fp | | 387 | |
| FSINCOS | | 5 | ≈150 | ≈200 | | ≈200 | 1 | fp | | 387 | |
| FPTAN | | 8 | ≈170 | ≈270 | | ≈270 | 1 | fp | | 87 | |
| FPATAN | | 4 | 97 | ≈250 | | ≈250 | 1 | fp | | 87 | |
| FSCALE | | 3 | 25 | 96 | | | 1 | fp | | 87 | |
| FXTRACT | | 4 | 16 | 27 | | | 1 | fp | | 87 | |
| F2XM1 | | 3 | 190 | ≈270 | | | 1 | fp | | 87 | |
| FYL2X | | 3 | 63 | ≈170 | | | 1 | fp | | 87 | |
| FYL2XP1 | | 3 | 58 | ≈170 | | | 1 | fp | | 87 | |
| Other | | | | | | | | | | | |
| FNOP | | 1 | 0 | 1 | 0 | 1 | 0 | | mov | 87 | |
| (F)WAIT | | 2 | 0 | 0 | 0 | 1 | 0 | | mov | 87 | |
| FNCLEX | | 1 | 4 | | | 120 | 1 | | | 87 | |
| FNINIT | | 1 | 30 | | | 200 | | | | 87 | |
| FNSAVE | | 2 | 181 | 500 | | | 0,1 | | | 87 | |
| FRSTOR | | 2 | 96 | 570 | | | | | | 87 | |
| FXSAVE | | 2 | 121 | | | 160 | | | | sse | i |
| FXRSTOR | | 2 | 118 | | | 244 | | | | sse | i |

Notes:

- e) Not available on PMMX
- f) The latency for FLDCW is 3 when the new value loaded is the same as the value of the control word before the preceding FLDCW, i.e. when alternating between the same two values. In all other cases, the latency and reciprocal throughput is > 100.
- g) Latency and reciprocal throughput depend on the precision setting in the F.P. control word. Single precision: 32, double precision: 40, long double precision (default): 45.
- h) Throughput of FP-MUL unit is reduced during the use of the FP-DIV unit.

- i) Takes fewer microcode pops when XMM registers are disabled, but the throughput is the same.

Integer MMX and XMM instructions

| Instruction | Operands | uops | Microcode | Latency | Additional latency | Reciprocal throughput | Port | Execution unit | Subunit | Instruction set | Notes |
|--------------------------|-----------|------|-----------|---------|--------------------|-----------------------|------|----------------|---------|-----------------|-------|
| Move instructions | | | | | | | | | | | |
| MOVD | r32, mm | 2 | 0 | 6 | 1 | 1 | 0 | fp | | mmx | |
| MOVD | mm, r32 | 1 | 0 | 3 | 1 | 1 | 1 | mmx | alu | mmx | |
| MOVD | mm,m32 | 1 | 0 | | | 1 | 2 | load | | mmx | |
| MOVD | r32, xmm | 1 | 0 | 7 | 1 | 1 | 0 | fp | | sse2 | |
| MOVD | xmm, r32 | 2 | 0 | 4 | 1 | 2 | 1 | mmx | shift | sse2 | |
| MOVD | xmm,m32 | 1 | 0 | | | 1 | 2 | load | | sse2 | |
| MOVD | m32, r | 2 | 0 | | | 2 | 0,1 | | | mmx | |
| MOVQ | mm,mm | 1 | 0 | 7 | 0 | 1 | 0 | mov | | mmx | |
| MOVQ | xmm,xmm | 1 | 0 | 2 | 1 | 2 | 1 | mmx | shift | sse2 | |
| MOVQ | r,m64 | 1 | 0 | | | 1 | 2 | load | | mmx | |
| MOVQ | m64,r | 2 | 0 | | | 2 | 0 | mov | | mmx | |
| MOVDQA | xmm,xmm | 1 | 0 | 7 | 0 | 1 | 0 | mov | | sse2 | |
| MOVDQA | xmm,m | 1 | 0 | | | 1 | 2 | load | | sse2 | |
| MOVDQA | m,xmm | 2 | 0 | | | 2 | 0 | mov | | sse2 | |
| MOVDQU | xmm,m | 4 | 0 | | | 23 | 2 | load | | sse2 | k |
| MOVDQU | m,xmm | 4 | 2 | | | 8 | 0 | mov | | sse2 | k |
| LDDQU | xmm,m | 4 | 0 | | | 2.5 | 2 | load | | sse3 | |
| MOVDQ2Q | mm,xmm | 3 | 0 | 10 | 1 | 2 | 0,1 | mov-mmx | sse2 | | |
| MOVQ2DQ | xmm,mm | 2 | 0 | 10 | 1 | 2 | 0,1 | mov-mmx | sse2 | | |
| MOVNTQ | m,mm | 3 | 0 | | | 4 | 0 | mov | | sse | |
| MOVNTDQ | m,xmm | 2 | 0 | | | 4 | 0 | mov | | sse2 | |
| MOVDDUP | xmm,xmm | 1 | 0 | 2 | 1 | 2 | 1 | mmx | shift | sse3 | |
| MOVSHDUP | | | | | | | | | | | |
| MOVSLDUP | xmm,xmm | 1 | 0 | 4 | 1 | 2 | 1 | mmx | shift | sse3 | |
| PACKSSWB/DW | | | | | | | | | | | |
| PACKUSWB | mm,r/m | 1 | 0 | 2 | 1 | 2 | 1 | mmx | shift | mmx | a |
| PACKSSWB/DW | | | | | | | | | | | |
| PACKUSWB | xmm,r/m | 1 | 0 | 4 | 1 | 4 | 1 | mmx | shift | mmx | a |
| PUNPCKH/LBW/WD/DQ | mm,r/m | 1 | 0 | 2 | 1 | 2 | 1 | mmx | shift | mmx | a |
| PUNPCKHBW/WD/DQ/QDQ | xmm,r/m | 1 | 0 | 4 | 1 | 4 | 1 | mmx | shift | sse2 | a |
| PUNPCKLBW/WD/DQ/QDQ | xmm,r/m | 1 | 0 | 2 | 1 | 2 | 1 | mmx | shift | sse2 | a |
| PSHUFD | xmm,xmm,i | 1 | 0 | 4 | 1 | 2 | 1 | mmx | shift | sse2 | |
| PSHUFL/HW | xmm,xmm,i | 1 | 0 | 2 | 1 | 2 | 1 | mmx | shift | sse | |
| PSHUFW | mm,mm,i | 1 | 0 | 2 | 1 | 1 | 1 | mmx | shift | sse | |
| MASKMOVQ | mm,mm | 1 | 4 | | | 10 | 0 | mov | | sse | |
| MASKMOVDQU | xmm,xmm | 1 | 6 | | | 12 | 0 | mov | | sse2 | |
| PMOVBMSKB | r32,r | 2 | 0 | 7 | | 3 | 0,1 | mmx-alu0 | sse | | |

Prescott

| | | | | | | | | | | | |
|--------------------------------|-----------|----|----|---|---|-----|---|---------|-------|------|-----|
| PEXTRW | r32,mm,i | 2 | 0 | 7 | | 2 | 1 | mmx-int | sse | | |
| PEXTRW | r32,xmm,i | 2 | 0 | 7 | | 3 | 1 | mmx-int | sse2 | | |
| PINSRW | r,r32,i | 2 | 0 | 4 | | 2 | 1 | int-mmx | sse | | |
| Arithmetic instructions | | | | | | | | | | | |
| PADDB/W/D | | | | | | | | | | | |
| PADD(U)SB/W | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | mmx | a,j |
| PSUBB/W/D | | | | | | | | | | | |
| PSUB(U)SB/W | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | mmx | a,j |
| PADDQ, PSUBQ | mm,r/m | 1 | 0 | 2 | 1 | 1 | 1 | mmx | alu | sse2 | a |
| PADDQ, PSUBQ | xmm,r/m | 1 | 0 | 5 | 1 | 2 | 1 | fp | add | sse2 | a |
| PCMPEQB/W/D | | | | | | | | | | | |
| PCMPGTB/W/D | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | mmx | a,j |
| PMULLW PMULHW | r,r/m | 1 | 0 | 7 | 1 | 1,2 | 1 | fp | mul | mmx | a,j |
| PMULHUW | r,r/m | 1 | 0 | 7 | 1 | 1,2 | 1 | fp | mul | sse | a,j |
| PMADDWD | r,r/m | 1 | 0 | 7 | 1 | 1,2 | 1 | fp | mul | mmx | a,j |
| PMULUDQ | r,r/m | 1 | 0 | 7 | 1 | 1,2 | 1 | fp | mul | sse2 | a,j |
| PAVGB/W | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | sse | a,j |
| PMIN/MAXUB | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | sse | a,j |
| PMIN/MAXSW | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | sse | a,j |
| PSADBW | r,r/m | 1 | 0 | 4 | 1 | 1,2 | 1 | mmx | alu | sse | a,j |
| Logic | | | | | | | | | | | |
| PAND, PANDN | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | mmx | a,j |
| POR, PXOR | r,r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | alu | mmx | a,j |
| PSLL/RLW/D/Q, | | | | | | | | | | | |
| PSRAW/D | r,i/r/m | 1 | 0 | 2 | 1 | 1,2 | 1 | mmx | shift | mmx | a,j |
| PSLLDQ, PSRLDQ | xmm,i | 1 | 0 | 4 | 1 | 2 | 1 | mmx | shift | sse2 | |
| Other | | | | | | | | | | | |
| EMMS | | 10 | 10 | | | 12 | 0 | | | mmx | |

Notes:

- a) Add 1 μ op if source is a memory operand.
- j) Reciprocal throughput is 1 for 64 bit operands, and 2 for 128 bit operands.
- k) It may be advantageous to replace this instruction by two 64-bit moves or LD-DQU.

Floating point XMM instructions

| Instruction | Operands | μ ops | Microcode | Latency | Additional latency | Reciprocal throughput | Port | Execution unit | Subunit | Instruction set | Notes |
|--------------------------|----------|-----------|-----------|---------|--------------------|-----------------------|------|----------------|---------|-----------------|-------|
| Move instructions | | | | | | | | | | | |
| MOVAPS/D | r,r | 1 | 0 | 7 | 0 | 1 | 0 | mov | | sse | |
| MOVAPS/D | r,m | 1 | 0 | | 0 | 1 | 2 | | | sse | |
| MOVAPS/D | m,r | 2 | 0 | | | 2 | 0 | | | sse | |
| MOVUPS/D | r,r | 1 | 0 | 7 | 0 | 1 | 0 | mov | | sse | |
| MOVUPS/D | r,m | 4 | 0 | | | 2 | 2 | | | sse | k |
| MOVUPS/D | m,r | 4 | 2 | | | 8 | 0 | | | sse | k |

Prescott

| | | | | | | | | | | | |
|------------------------------------|---------|---|---|----|---|-----|-----|--------|-------|------|-----|
| MOVSS | r,r | 1 | 0 | 2 | 1 | 2 | 1 | mmx | shift | sse | |
| MOVSD | r,r | 1 | 0 | 4 | 1 | 2 | 1 | mmx | shift | sse | |
| MOVSS, MOVSD | r,m | 1 | 0 | | 0 | 1 | 2 | | | sse | |
| MOVSS, MOVSD | m,r | 2 | 0 | | | 2 | 0 | | | sse | |
| MOVHLPS | r,r | 1 | 0 | 4 | 1 | 2 | 1 | mmx | shift | sse | |
| MOVLHPS | r,r | 1 | 0 | 2 | 1 | 2 | 1 | mmx | shift | sse | |
| MOVHPS/D, MOVLPS/D | r,m | 2 | 0 | | | 2 | 2 | | | sse | |
| MOVHPS/D, MOVLPS/D | m,r | 2 | 0 | | | 2 | 0 | | | sse | |
| MOVSH/LDUP | r,r | 1 | 0 | 4 | 1 | 2 | 1 | | | sse3 | |
| MOVDDUP | r,r | 1 | 0 | 2 | 1 | 2 | 1 | | | sse3 | |
| MOVNTPS/D | m,r | 2 | 0 | | | 4 | 0 | | | sse | |
| MOVMSKPS/D | r32,r | 2 | 0 | 5 | 1 | 3 | 1 | fp | | sse | |
| SHUFPS/D | r,r/m,i | 1 | 0 | 4 | 1 | 2 | 1 | mmx | shift | sse | |
| UNPCKHPS/D | r,r/m | 2 | 0 | 4 | 1 | 2 | 1 | mmx | shift | sse | |
| UNPCKLPS/D | r,r/m | 1 | 0 | 2 | 1 | 2 | 1 | mmx | shift | sse | |
| Conversion | | | | | | | | | | | |
| CVTTPS2PD | r,r/m | 1 | 0 | 4 | 1 | 4 | 1 | mmx | shift | sse2 | a |
| CVTPD2PS | r,r/m | 2 | 0 | 10 | 1 | 2 | 1 | fp-mmx | sse2 | a | |
| CVTSD2SS | r,r/m | 3 | 0 | 14 | 1 | 6 | 1 | mmx | shift | sse2 | a |
| CVTSS2SD | r,r/m | 2 | 0 | 8 | 1 | 6 | 1 | mmx | shift | sse2 | a |
| CVTDQ2PS | r,r/m | 1 | 0 | 5 | 1 | 2 | 1 | fp | | sse2 | a |
| CVTDQ2PD | r,r/m | 3 | 0 | 10 | 1 | 4 | 1 | mmx-fp | sse2 | a | |
| CVT(T)PS2DQ | r,r/m | 1 | 0 | 5 | 1 | 2 | 1 | fp | | sse2 | a |
| CVT(T)PD2DQ | r,r/m | 2 | 0 | 11 | 1 | 2 | 1 | fp-mmx | sse2 | a | |
| CVTPI2PS | xmm,mm | 4 | 0 | 12 | 1 | 6 | 1 | mmx | | sse | a |
| CVTPI2PD | xmm,mm | 4 | 0 | 12 | 1 | 5 | 1 | fp-mmx | sse2 | a | |
| CVT(T)PS2PI | mm,xmm | 3 | 0 | 8 | 0 | 2 | 0,1 | fp-mmx | sse | a | |
| CVT(T)PD2PI | mm,xmm | 4 | 0 | 12 | 1 | 3 | 0,1 | fp-mmx | sse2 | a | |
| CVTSI2SS | xmm,r32 | 3 | 0 | 20 | 1 | 4 | 1 | fp-mmx | sse | a | |
| CVTSI2SD | xmm,r32 | 4 | 0 | 20 | 1 | 5 | 1 | fp-mmx | sse2 | a | |
| CVT(T)SD2SI | r32,xmm | 2 | 0 | 12 | 1 | 4 | 1 | fp | | sse2 | a |
| CVT(T)SS2SI | r32,xmm | 2 | 0 | 17 | 1 | 4 | 1 | fp | | sse | a |
| Arithmetic | | | | | | | | | | | |
| ADDPS/D ADDSS/D | r,r/m | 1 | 0 | 5 | 1 | 2 | 1 | fp | add | sse | a |
| SUBPS/D SUBSS/D | r,r/m | 1 | 0 | 5 | 1 | 2 | 1 | fp | add | sse | a |
| ADDSUBPS/D | r,r/m | 1 | 0 | 5 | 1 | 2 | 1 | fp | add | sse3 | a |
| HADDPS/D HSUBPS/D | r,r/m | 3 | 0 | 13 | 1 | 5-6 | 1 | fp | add | sse3 | a |
| MULPS/D MULSS/D | r,r/m | 1 | 0 | 7 | 1 | 2 | 1 | fp | mul | sse | a |
| DIVSS | r,r/m | 1 | 0 | 32 | 1 | 23 | 1 | fp | div | sse | a,h |
| DIVPS | r,r/m | 1 | 0 | 41 | 1 | 41 | 1 | fp | div | sse | a,h |
| DIVSD | r,r/m | 1 | 0 | 40 | 1 | 40 | 1 | fp | div | sse2 | a,h |
| DIVPD | r,r/m | 1 | 0 | 71 | 1 | 71 | 1 | fp | div | sse2 | a,h |
| RCPPS RCPSS | r,r/m | 2 | 0 | 6 | 1 | 4 | 1 | mmx | | sse | a |
| MAXPS/D MAXSS/ DMINPS/D MINSS/D | r,r/m | 1 | 0 | 5 | 1 | 2 | 1 | fp | add | sse | a |
| CMPccPS/D | | | | | | | | | | | |
| CMPccSS/D | r,r/m | 1 | 0 | 5 | 1 | 2 | 1 | fp | add | sse | a |
| COMISS/D UCOMISS/D | r,r/m | 2 | 0 | 6 | 1 | 3 | 1 | fp | add | sse | a |
| Logic | | | | | | | | | | | |

Prescott

| | | | | | | | | | | | |
|------------------------------------|-------|---|----|----|---|----|---|-----|-----|------|-----|
| ANDPS/D ANDNPS/D ORPS/D XORPS/D | r,r/m | 1 | 0 | 2 | 1 | 2 | 1 | mmx | alu | sse | a |
| Math | | | | | | | | | | | |
| SQRTSS | r,r/m | 1 | 0 | 32 | 1 | 32 | 1 | fp | div | sse | a,h |
| SQRTPS | r,r/m | 1 | 0 | 41 | 1 | 41 | 1 | fp | div | sse | a,h |
| SQRTSD | r,r/m | 1 | 0 | 40 | 1 | 40 | 1 | fp | div | sse2 | a,h |
| SQRTPD | r,r/m | 1 | 0 | 71 | 1 | 71 | 1 | fp | div | sse2 | a,h |
| RSQRTSS | r,r/m | 2 | 0 | 5 | 1 | 3 | 1 | mmx | | sse | a |
| RSQRTPS | r,r/m | 2 | 0 | 6 | 1 | 4 | 1 | mmx | | sse | a |
| Other | | | | | | | | | | | |
| LDMXCSR | m | 2 | 11 | | | 13 | 1 | | | sse | |
| STMXCSR | m | 3 | 0 | | | 3 | 1 | | | sse | |

Notes:

- a) Add 1 μ op if source is a memory operand.
- h) Throughput of FP-MUL unit is reduced during the use of the FP-DIV unit.
- k) It may be advantageous to replace this instruction by two 64-bit moves or LDDQU.

Intel Atom

List of instruction timings and μ ops breakdown

Explanation of column headings:

| | |
|-------------------------------|---|
| Instruction: | Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc. |
| Operands: | i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, mm/x = mmx or xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc. |
| μops: | The number of μ ops from the decoder or ROM. |
| Unit: | Tells which execution unit is used. Instructions that use the same unit cannot execute simultaneously. ALU0 and ALU1 means integer unit 0 or 1, respectively. ALU0/1 means that either unit can be used. ALU0+1 means that both units are used. Mem means memory in/out unit. FP0 means floating point unit 0 (includes multiply, divide and other SIMD instructions). FP1 means floating point unit 1 (adder). MUL means multiplier, shared between FP and integer units. DIV means divider, shared between FP and integer units. np means not pairable: Cannot execute simultaneously with any other instruction. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay. |
| Reciprocal throughput: | The average number of clock cycles per instruction for a series of independent instructions of the same kind in the same thread. |

Integer instructions

| | Operands | μ ops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|----------|-----------|-----------|---------|-----------------------|-----------------|
| Move instructions | | | | | | |
| MOV | r,r | 1 | ALU0/1 | 1 | 1/2 | All addr. modes |
| MOV | r,i | 1 | ALU0/1 | 1 | 1/2 | |
| MOV | r,m | 1 | ALU0, Mem | 1-3 | 1 | |
| MOV | m,r | 1 | ALU0, Mem | 1 | 1 | |
| MOV | m,i | 1 | ALU0, Mem | | 1 | |
| MOV | r,sr | 1 | | 1 | 1 | |
| MOV | m,sr | 2 | | | 5 | |
| MOV | sr,r | 7 | | | 21 | |
| MOV | sr,m | 8 | | | 26 | |
| MOVNTI | m,r | 1 | ALU0, Mem | | 2.5 | |
| MOVSX MOVZX MOVSD | r,r/m | 1 | ALU0 | 1 | 1 | Implicit lock |
| CMOVcc | r,r | 1 | ALU0+1 | 2 | 2 | |
| CMOVcc | r,m | 1 | | | 3 | |
| XCHG | r,r | 3 | | 6 | 6 | |
| XCHG | r,m | 4 | | 6 | 6 | |
| XLAT | | 3 | | 6 | 6 | |
| PUSH | r | 1 | np | 1 | 1 | |

Atom

| | | | | | | |
|--------------------------------|-----------|----|-------------|-----|-----|-----------------------------------|
| PUSH | i | 1 | np | | 1 | |
| PUSH | m | 2 | | | 5 | |
| PUSH | sr | 3 | | | 6 | |
| PUSHF(D/Q) | | 14 | | | 12 | |
| PUSHA(D) | | 9 | | | 11 | Not in x64 mode |
| POP | r | 1 | np | 1 | 1 | |
| POP | (E/R)SP | 1 | np | 1 | 1 | |
| POP | m | 3 | | | 6 | |
| POP | sr | 7 | | | 31 | |
| POPF(D/Q) | | 19 | | | 28 | |
| POPA(D) | | 16 | | | 12 | Not in x64 mode |
| LAHF | | 1 | ALU0+1 | 2 | 2 | |
| SAHF | | 1 | ALU0/1 | 1 | 1/2 | |
| SALC | | 2 | | 7 | 5 | Not in x64 mode |
| LEA | r,m | 1 | AGU1 | 1-4 | 1 | 4 clock latency on input register |
| BSWAP | r | 1 | ALU0 | 1 | 1 | |
| LDS LES LFS LGS LSS | m | 10 | | 30 | 30 | |
| PREFETCHNTA | m | 1 | Mem | | 1 | |
| PREFETCHT0/1/2 | m | 1 | Mem | | 1 | |
| LFENCE | | 1 | | | 1/2 | |
| MFENCE | | 1 | | | 1 | |
| SFENCE | | 1 | | | 1 | |
| Arithmetic instructions | | | | | | |
| ADD SUB | r,r/i | 1 | ALU0/1 | 1 | 1/2 | |
| ADD SUB | r,m | 1 | ALU0/1, Mem | | 1 | |
| ADD SUB | m,r/i | 1 | | 2 | 1 | |
| ADC SBB | r,r/i | 1 | | 2 | 2 | |
| ADC SBB | r,m | 1 | | 2 | 2 | |
| ADC SBB | m,r/i | 1 | | 2 | 2 | |
| CMP | r,r/i | 1 | ALU0/1 | 1 | 1/2 | |
| CMP | m,r/i | 1 | | | 1 | |
| INC DEC NEG NOT | r | 1 | ALU0/1 | 1 | 1/2 | |
| INC DEC NEG NOT | m | 1 | | 1 | | |
| AAA | | 13 | | 16 | | Not in x64 mode |
| AAS | | 13 | | 12 | | Not in x64 mode |
| DAA | | 20 | | 20 | | Not in x64 mode |
| DAS | | 21 | | 25 | | Not in x64 mode |
| AAD | | 4 | | 7 | | Not in x64 mode |
| AAM | | 10 | | 24 | | Not in x64 mode |
| MUL IMUL | r8 | 3 | ALU0, Mul | 7 | 7 | |
| MUL IMUL | r16 | 4 | ALU0, Mul | 6 | 6 | |
| MUL IMUL | r32 | 3 | ALU0, Mul | 6 | 6 | |
| MUL IMUL | r64 | 8 | ALU0, Mul | 14 | 14 | |
| IMUL | r16,r16 | 2 | ALU0, Mul | 6 | 5 | |
| IMUL | r32,r32 | 1 | ALU0, Mul | 5 | 2 | |
| IMUL | r64,r64 | 6 | ALU0, Mul | 13 | 11 | |
| IMUL | r16,r16,i | 2 | ALU0, Mul | 5 | 5 | |
| IMUL | r32,r32,i | 1 | ALU0, Mul | 5 | 2 | |
| IMUL | r64,r64,i | 7 | ALU0, Mul | 14 | 14 | |
| MUL IMUL | m8 | 3 | ALU0, Mul | 6 | | |
| MUL IMUL | m16 | 5 | ALU0, Mul | 7 | | |

Atom

| | | | | | | |
|---------------------------|------------|-------|-------------|-------|-----|-----------------|
| MUL IMUL | m32 | 4 | ALU0, Mul | 7 | | |
| MUL IMUL | m64 | 8 | ALU0, Mul | 14 | | |
| DIV | r/m8 | 9 | ALU0, Div | 22 | 22 | |
| DIV | r/m16 | 12 | ALU0, Div | 33 | 33 | |
| DIV | r/m32 | 12 | ALU0, Div | 49 | 49 | |
| DIV | r/m 64 | 38 | ALU0, Div | 183 | 183 | |
| IDIV | r/m8 | 26 | ALU0, Div | 38 | 38 | |
| IDIV | r/m16 | 29 | ALU0, Div | 45 | 45 | |
| IDIV | r/m32 | 29 | ALU0, Div | 61 | 61 | |
| IDIV | r/m64 | 60 | ALU0, Div | 207 | 207 | |
| CBW | | 2 | ALU0 | 5 | | |
| CWDE | | 1 | ALU0 | 1 | | |
| CDQE | | 1 | ALU0 | 1 | | |
| CWD | | 2 | ALU0 | 5 | | |
| CDQ | | 1 | ALU0 | 1 | | |
| CQO | | 1 | ALU0 | 1 | | |
| Logic instructions | | | | | | |
| AND OR XOR | r,r/i | 1 | ALU0/1 | 1 | 1/2 | |
| AND OR XOR | r,m | 1 | ALU0/1, Mem | | 1 | |
| AND OR XOR | m,r/i | 1 | ALU0/1, Me | 1 | 1 | |
| TEST | r,r/i | 1 | ALU0/1 | 1 | 1/2 | |
| TEST | m,r/i | 1 | ALU0/1, Mem | | 1 | |
| SHR SHL SAR | r,i/cl | 1 | ALU0 | 1 | 1 | |
| SHR SHL SAR | m,i/cl | 1 | ALU0 | 1 | 1 | |
| ROR ROL | r,i/cl | 1 | ALU0 | 1 | 1 | |
| ROR ROL | m,i/cl | 1 | ALU0 | 1 | 1 | |
| RCR | r,1 | 5 | ALU0 | 7 | | |
| RCL | r,1 | 2 | ALU0 | 1 | | |
| RCR | r/m,i/cl | 12-17 | ALU0 | 12-15 | | |
| RCL | r/m,i/cl | 14-20 | ALU0 | 14-18 | | |
| SHLD | r16,r16,i | 10 | ALU0 | 10 | | 1-2 more if mem |
| SHLD | r32,r32,i | 2 | ALU0 | 5 | | 1-2 more if mem |
| SHLD | r64,r64,i | 10 | ALU0 | 11 | | 1-2 more if mem |
| SHLD | r16,r16,cl | 9 | ALU0 | 9 | | 1-2 more if mem |
| SHLD | r32,r32,cl | 2 | ALU0 | 5 | | 1-2 more if mem |
| SHLD | r64,r64,cl | 9 | ALU0 | 10 | | 1-2 more if mem |
| SHRD | r16,r16,i | 8 | ALU0 | 8 | | 1-2 more if mem |
| SHRD | r32,r32,i | 2 | ALU0 | 5 | | 1-2 more if mem |
| SHRD | r64,r64,i | 10 | ALU0 | 9 | | 1-2 more if mem |
| SHRD | r16,r16,cl | 7 | ALU0 | 8 | | 1-2 more if mem |
| SHRD | r32,r32,cl | 2 | ALU0 | 5 | | 1-2 more if mem |
| SHRD | r64,r64,cl | 9 | ALU0 | 9 | | 1-2 more if mem |
| BT | r,r/i | 1 | ALU1 | 1 | 1 | |
| BT | m,r | 9 | | 10 | | |
| BT | m,i | 2 | | 5 | | |
| BTR BTS BTC | r,r/i | 1 | ALU1 | 1 | 1 | |
| BTR BTS BTC | m,r | 10 | ALU1 | 11 | | |
| BTR BTS BTC | m,i | 3 | ALU1 | 6 | | |
| BSF BSR | r,r/m | 10 | | 16 | | |
| SETcc | r | 1 | ALU0+1 | 2 | 2 | |
| SETcc | m | 2 | | | 5 | |
| CLC STC | | 1 | ALU0/1 | | 1/2 | |

Atom

| | | | | | | |
|--------------------------------------|------------|-------|--------|---------|-----|--------------------|
| CMC | | 1 | | 2 | 2 | |
| CLD | | 5 | | | 7 | |
| STD | | 6 | | | 25 | |
| Control transfer instructions | | | | | | |
| JMP | short/near | 1 | ALU1 | | 2 | |
| JMP | far | 29 | | | 66 | Not in x64 mode |
| JMP | r | 1 | | | 4 | |
| JMP | m(near) | 2 | | | 7 | |
| JMP | m(far) | 30 | | | 78 | |
| Conditional jump | short/near | 1 | ALU1 | | 2 | |
| J(E/R)CXZ | short | 3 | | | 7 | |
| LOOP | short | 8 | | | 8 | |
| LOOP(N)E | short | 8 | | | 8 | |
| CALL | near | 1 | | | 3 | |
| CALL | far | 37 | | | 65 | Not in x64 mode |
| CALL | r | 1 | | | 18 | |
| CALL | m(near) | 2 | | | 20 | |
| CALL | m(far) | 38 | | | 64 | |
| RETN | | 1 | np | | 6 | |
| RETN | i | 1 | np | | 6 | |
| RETF | | 36 | | | 80 | |
| RETF | i | 36 | | | 80 | |
| BOUND | r,m | 11 | | | 10 | Not in x64 mode |
| INTO | | 4 | | | 6 | Not in x64 mode |
| String instructions | | | | | | |
| LODS | | 3 | | 6 | | |
| REP LODS | | 5n+11 | | 3n+50 | | |
| STOS | | 2 | | 5 | | |
| REP STOS | | 3n+10 | | 2n+4 | | |
| MOVS | | 4 | | 6 | | |
| REP MOVS | | 4n+11 | | 2n - 4n | | fastest for high n |
| SCAS | | 3 | | 6 | | |
| REP SCAS | | 5n+16 | | 3n+60 | | |
| CMPS | | 5 | | 7 | | |
| REP CMPS | | 6n+16 | | 4n+40 | | |
| Other | | | | | | |
| NOP (90) | | 1 | ALU0/1 | | 1/2 | |
| Long NOP (0F 1F) | | 1 | ALU0/1 | | 1/2 | |
| PAUSE | | 5 | | 24 | | |
| ENTER | a,0 | 14 | | 23 | | |
| ENTER | a,b | 20+6b | | | | |
| LEAVE | | 4 | | | 6 | |
| CPUID | | 40-80 | | 100-170 | | |
| RDTSC | | 16 | | 29 | | |
| RDPMSR | | 24 | | 48 | | |

Floating point x87 instructions

| | Operands | μops | Unit | Latency | Reciprocal throughput | Remarks |
|--|----------|------|------|---------|-----------------------|---------|
|--|----------|------|------|---------|-----------------------|---------|

Atom

| Move instructions | | | | | | | |
|-------------------------|---------|-----|-----|------|-----|--|------|
| FLD | r | 1 | | 1 | 1 | | |
| FLD | m32/m64 | 1 | | 3 | 1 | | |
| FLD | m80 | 4 | | 9 | 10 | | |
| FBLD | m80 | 52 | | 92 | 92 | | |
| FST(P) | r | 1 | | 1 | 1 | | |
| FST(P) | m32/m64 | 3 | | 7 | 9 | | |
| FSTP | m80 | 8 | | 12 | 13 | | |
| FBSTP | m80 | 189 | | 221 | 221 | | |
| FXCH | r | 1 | | 1 | 1 | | |
| FILD | m | 1 | | 7 | 6 | | |
| FIST(P) | m | 3 | | 11 | 9 | | |
| FISTTP | m | 3 | | 11 | 9 | | SSE3 |
| FLDZ | | 1 | | | 1 | | |
| FLD1 | | 2 | | | 8 | | |
| FLDPI FLDL2E etc. | | 2 | | | 10 | | |
| FCMOVcc | r | 3 | | 9 | 9 | | |
| FNSTSW | AX | 4 | | | 10 | | |
| FNSTSW | m16 | 4 | | | 10 | | |
| FLDCW | m16 | 2 | | | 8 | | |
| FNSTCW | m16 | 3 | | | 9 | | |
| FINCSTP FDECSTP | | 1 | | 1 | 1 | | |
| FFREE(P) | | 1 | | | 1 | | |
| FNSAVE | m | 166 | | 321 | 321 | | |
| FRSTOR | m | 83 | | 177 | 177 | | |
| Arithmetic instructions | | | | | | | |
| FADD(P) FSUB(R)(P) | r/m | 1 | | 5 | 1 | | |
| FMUL(P) | r/m | 1 | Mul | 5 | 2 | | |
| FDIV(R)(P) | r/m | 1 | Div | 71 | 71 | | |
| FABS | | 1 | | 1 | 1 | | |
| FCHS | | 1 | | 1 | 1 | | |
| FCOM(P) FUCOM | r/m | 1 | | 1 | 1 | | |
| FCOMPP FUCOMPP | | 1 | | 1 | 1 | | |
| FCOMI(P) FUCOMI(P) | r | 5 | | | 10 | | |
| FIADD FISUB(R) | m | 3 | | | 9 | | |
| FIMUL | m | 3 | Mul | | 9 | | |
| FIDIV(R) | m | 3 | Div | | 73 | | |
| FICOM(P) | m | 3 | | | 9 | | |
| FTST | | 1 | | 1 | 1 | | |
| FXAM | | 1 | | 1 | 1 | | |
| FPREM | | 26 | | ~110 | | | |
| FPREM1 | | 37 | | ~130 | | | |
| FRNDINT | | 19 | | 48 | | | |
| Math | | | | | | | |
| FSCALE | | 30 | | 56 | | | |
| FTRACT | | 15 | | 24 | | | |
| FSQRT | | 1 | Div | 71 | | | |
| FSIN FCOS | | 9 | | ~260 | | | |
| FSINCOS | | 112 | | ~260 | | | |
| F2XM1 | | 25 | | ~100 | | | |
| FYL2X FYL2XP1 | | 63 | | ~220 | | | |

Atom

| | | | |
|--------------|-----|------|----|
| FPTAN | 100 | ~300 | |
| FPATAN | 91 | ~300 | |
| Other | | | |
| FNOP | 1 | | 1 |
| WAIT | 2 | 5 | 5 |
| FNCLEX | 4 | | 26 |
| FNINIT | 23 | 74 | |

Integer MMX and XMM instructions

| | Operands | μops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------------|-------------|------|-------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| MOVD | r32/64,mm/x | 1 | | 4 | 2 | |
| MOVD | m32/64,mm/x | 1 | Mem | 5 | 1 | |
| MOVD | mm/x,r32/64 | 1 | | 3 | 1 | |
| MOVD | mm/x,m32/64 | 1 | Mem | 4 | 1 | |
| MOVQ | mm/x, mm/x | 1 | FP0/1 | 1 | 1/2 | |
| MOVQ | mm/x,m64 | 1 | Mem | 4 | 1 | |
| MOVQ | m64, mm/x | 1 | Mem | 5 | 1 | |
| MOVDQA | xmm, xmm | 1 | FP0/1 | 1 | 1/2 | |
| MOVDQA | xmm, m128 | 1 | Mem | 4 | 1 | |
| MOVDQA | m128, xmm | 1 | Mem | 5 | 1 | |
| MOVDQU | m128, xmm | 3 | Mem | 6 | 6 | |
| MOVDQU | xmm, m128 | 4 | Mem | 6 | 6 | |
| LDDQU | xmm, m128 | 4 | Mem | 6 | 6 | |
| MOVDQ2Q | mm, xmm | 1 | | 1 | 1 | |
| MOVQ2DQ | xmm,mm | 1 | | 1 | 1 | |
| MOVNTQ | m64,mm | 1 | Mem | ~400 | 1 | |
| MOVNTDQ | m128,xmm | 1 | Mem | ~450 | 3 | |
| PACKSSWB/DW | | | | | | |
| PACKUSWB | mm/x, mm/x | 1 | FP0 | 1 | 1 | |
| PUNPCKH/LBW/WD/DQ | mm/x, mm/x | 1 | FP0 | 1 | 1 | |
| PUNPCKH/LQDQ | mm/x, mm/x | 1 | FP0 | 1 | 1 | |
| PSHUFB | mm,mm | 1 | FP0 | 1 | 1 | |
| PSHUFB | xmm,xmm | 4 | | 6 | 6 | |
| PSHUFW | mm,mm,i | 1 | FP0 | 1 | 1 | |
| PSHUFL/HW | xmm,xmm,i | 1 | FP0 | 1 | 1 | |
| PSHUFD | xmm,xmm,i | 1 | FP0 | 1 | 1 | |
| PALIGNR | xmm, xmm,i | 1 | FP0 | 1 | 1 | |
| MASKMOVQ | mm,mm | 1 | Mem | | 2 | |
| MASKMOVDQU | xmm,xmm | 2 | Mem | | 7 | |
| PMOVMskB | r32,mm/x | 1 | | 4 | 2 | |
| PINSRW | mm/x,r32,i | 1 | | 3 | 1 | |
| PEXTRW | r32,mm/x,i | 2 | | 5 | 5 | |
| Arithmetic instructions | | | | | | |
| PADD/SUB(U)(S)B/W/D | mm/x, mm/x | 1 | FP0/1 | 1 | 1/2 | |
| PADDQ PSUBQ | mm/x, mm/x | 2 | | 5 | 5 | |
| PHADD(S)W PHSUB(S)W | mm/x, mm/x | 7 | | 8 | 8 | |

Atom

| | | | | | |
|---------------------------|------------|---|----------|---|-----|
| PHADDD PHSUBD | mm/x, mm/x | 3 | | 6 | |
| PCMPEQ/GTB/W/D | mm/x,mm/x | 1 | FP0/1 | 1 | 1/2 |
| PMULL/HW PMULHUW | mm,mm | 1 | FP0, Mul | 4 | 1 |
| PMULL/HW PMULHUW | xmm,xmm | 1 | FP0, Mul | 5 | 2 |
| PMULHRSW | mm,mm | 1 | FP0, Mul | 4 | 1 |
| PMULHRSW | xmm,xmm | 1 | FP0, Mul | 5 | 2 |
| PMULUDQ | mm,mm | 1 | FP0, Mul | 4 | 1 |
| PMULUDQ | xmm,xmm | 1 | FP0, Mul | 5 | 2 |
| PMADDWD | mm,mm | 1 | FP0, Mul | 4 | 1 |
| PMADDWD | xmm,xmm | 1 | FP0, Mul | 5 | 2 |
| PMADDUBSW | mm,mm | 1 | FP0, Mul | 4 | 1 |
| PMADDUBSW | xmm,xmm | 1 | FP0, Mul | 5 | 2 |
| PSADBW | mm,mm | 1 | FP0, Mul | 4 | 1 |
| PSADBW | xmm,xmm | 1 | FP0, Mul | 5 | 2 |
| PAVGB/W | mm/x,mm/x | 1 | FP0/1 | 1 | 1/2 |
| PMIN/MAXUB | mm/x,mm/x | 1 | FP0/1 | 1 | 1/2 |
| PMIN/MAXSW | mm/x,mm/x | 1 | FP0/1 | 1 | 1/2 |
| PABSB PABSW PABSD | mm/x,mm/x | 1 | FP0/1 | 1 | 1/2 |
| PSIGNB PSIGNW PSIGND | mm/x,mm/x | 1 | FP0/1 | 1 | 1/2 |
| Logic instructions | | | | | |
| PAND(N) POR PXOR | mm/x,mm/x | 1 | FP0/1 | 1 | 1/2 |
| PSLL/RL/RAW/D/Q | mm/x,mm/x | 2 | FP0 | 5 | 5 |
| PSLL/RL/RAW/D/Q | (x)xmm,i | 1 | FP0 | 1 | 1 |
| PSLL/RLDQ | xmm,i | 1 | FP0 | 1 | 1 |
| Other | | | | | |
| EMMS | | 9 | | | 9 |

Floating point XMM instructions

| | Operands | μops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|------------|------|-------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| MOVAPS/D | xmm,xmm | 1 | FP0/1 | 1 | 1/2 | |
| MOVAPS/D | xmm,m128 | 1 | Mem | 4 | 1 | |
| MOVAPS/D | m128,xmm | 1 | Mem | 5 | 1 | |
| MOVUPS/D | xmm,m128 | 4 | Mem | 6 | 6 | |
| MOVUPS/D | m128,xmm | 3 | Mem | 6 | 6 | |
| MOVSS/D | xmm,xmm | 1 | FP0/1 | 1 | 1/2 | |
| MOVSS/D | xmm,m32/64 | 1 | Mem | 4 | 1 | |
| MOVSS/D | m32/64,xmm | 1 | Mem | 5 | 1 | |
| MOVHPS/D MOVLPD/D | xmm,m64 | 1 | Mem | 5 | 1 | |
| MOVHPS/D | m64,xmm | 1 | Mem | 4 | 1 | |
| MOVLPS/D | m64,xmm | 1 | Mem | 4 | 1 | |
| MOVLHPS MOVHLPS | xmm,xmm | 1 | FP0 | 1 | 1 | |
| MOVMSKPS/D | r32,xmm | 1 | | 4 | 2 | |
| MOVNTPS/D | m128,xmm | 1 | Mem | ~500 | 3 | |
| SHUFPS | xmm,xmm,i | 1 | FP0 | 1 | 1 | |
| SHUFPD | xmm,xmm,i | 1 | FP0 | 1 | 1 | |
| MOVDDUP | xmm,xmm | 1 | FP0 | 1 | 1 | |
| MOVSH/LDUP | xmm,xmm | 1 | FP0 | 1 | 1 | |

Atom

| | | | | | |
|--------------------|---------|---|----------|-----|-----|
| UNPCKH/LPS | xmm,xmm | 1 | FP0 | 1 | 1 |
| UNPCKH/LPD | xmm,xmm | 1 | FP0 | 1 | |
| Conversion | | | | | |
| CVTPD2PS | xmm,xmm | 4 | | 11 | 11 |
| CVTSD2SS | xmm,xmm | 3 | | 10 | 10 |
| CVTPS2PD | xmm,xmm | 4 | | 7 | 6 |
| CVTSS2SD | xmm,xmm | 3 | | 6 | 6 |
| CVTDQ2PS | xmm,xmm | 3 | | 6 | 6 |
| CVT(T) PS2DQ | xmm,xmm | 3 | | 6 | 6 |
| CVTDQ2PD | xmm,xmm | 3 | | 7 | 6 |
| CVT(T)PD2DQ | xmm,xmm | 3 | | 6 | 6 |
| CVTPI2PS | xmm,mm | 1 | | 6 | 5 |
| CVT(T)PS2PI | mm,xmm | 1 | | 4 | 1 |
| CVTPI2PD | xmm,mm | 3 | | 7 | 6 |
| CVT(T) PD2PI | mm,xmm | 4 | | 7 | 7 |
| CVTSI2SS | xmm,r32 | 3 | | 7 | 6 |
| CVT(T)SS2SI | r32,xmm | 3 | | 10 | 8 |
| CVTSI2SD | xmm,r32 | 3 | | 8 | 6 |
| CVT(T)SD2SI | r32,xmm | 3 | | 10 | 8 |
| Arithmetic | | | | | |
| ADDSS SUBSS | xmm,xmm | 1 | FP1 | 5 | 1 |
| ADDSD SUBSD | xmm,xmm | 1 | FP1 | 5 | 1 |
| ADDPSS SUBPS | xmm,xmm | 1 | FP1 | 5 | 1 |
| ADDPD SUBPD | xmm,xmm | 3 | FP1 | 6 | 6 |
| ADDSUBPS | xmm,xmm | 1 | FP1 | 5 | 1 |
| ADDSUBPD | xmm,xmm | 3 | FP1 | 6 | 6 |
| HADDPSS HSUBPS | xmm,xmm | 5 | FP0+1 | 8 | 7 |
| HADDPD HSUBPD | xmm,xmm | 5 | FP0+1 | 8 | 7 |
| MULSS | xmm,xmm | 1 | FP0, Mul | 4 | 1 |
| MULSD | xmm,xmm | 1 | FP0, Mul | 5 | 2 |
| MULPS | xmm,xmm | 1 | FP0, Mul | 5 | 2 |
| MULPD | xmm,xmm | 6 | FP0, Mul | 9 | 9 |
| DIVSS | xmm,xmm | 3 | FP0, Div | 31 | 31 |
| DIVSD | xmm,xmm | 3 | FP0, Div | 60 | 60 |
| DIVPS | xmm,xmm | 6 | FP0, Div | 64 | 64 |
| DIVPD | xmm,xmm | 6 | FP0, Div | 122 | 122 |
| RCPSS | xmm,xmm | 1 | | 4 | 1 |
| RCPPS | xmm,xmm | 5 | | 9 | 8 |
| CMPccSS/D | xmm,xmm | 1 | FP0 | 5 | 1 |
| CMPccPS/D | xmm,xmm | 3 | FP0 | 6 | 6 |
| COMISS/D UCOMISS/D | xmm,xmm | 4 | FP0 | 9 | 9 |
| MAXSS/D MINSS/D | xmm,xmm | 1 | FP0 | 5 | 1 |
| MAXPS/D MINPS/D | xmm,xmm | 3 | FP0 | 6 | 6 |
| Math | | | | | |
| SQRTSS | xmm,xmm | 3 | FP0, Div | 31 | 31 |
| SQRTPS | xmm,xmm | 5 | FP0, Div | 63 | 63 |
| SQRTSD | xmm,xmm | 3 | FP0, Div | 60 | 60 |
| SQRTPD | xmm,xmm | 5 | FP0, Div | 121 | 121 |
| RSQRTSS | xmm,xmm | 1 | FP0 | 4 | 1 |
| RSQRTPS | xmm,xmm | 5 | FP0 | 9 | 8 |

Atom

| | | | | | | |
|--------------|---------|-----|-------|-----|-----|--|
| Logic | | | | | | |
| ANDPS/D | xmm,xmm | 1 | FP0/1 | 1 | 1/2 | |
| ANDNPS/D | xmm,xmm | 1 | FP0/1 | 1 | 1/2 | |
| ORPS/D | xmm,xmm | 1 | FP0/1 | 1 | 1/2 | |
| XORPS/D | xmm,xmm | 1 | FP0/1 | 1 | 1/2 | |
| Other | | | | | | |
| LDMXCSR | m32 | 4 | | 5 | 6 | |
| STMXCSR | m32 | 4 | | 14 | 15 | |
| FXSAVE | m4096 | 121 | | 142 | 144 | |
| FXRSTOR | m4096 | 116 | | 149 | 150 | |

Intel Silvermont

List of instruction timings and μ op breakdown

Explanation of column headings:

| | |
|-------------------------------|--|
| Instruction: | Name of instruction. Multiple names mean that these instructions have the same data. Instructions with or without V name prefix behave the same unless otherwise noted. |
| Operands: | i = immediate data, r = register, mm = 64 bit mmx register, x = 128 bit xmm register, mm/x = mmx or xmm register, m = memory, m32 = 32-bit memory operand, etc. |
| μops: | The number of μ ops from the decoder or ROM. A μ op that goes to multiple units is counted as one. |
| Unit: | Tells which execution unit is used. Instructions that use the same unit cannot execute simultaneously. IP0 and IP1 means integer port 0 or 1 and their associated pipelines IP0/1 means that either integer unit can be used. IP0+1 means that the μ op is split in two, using both units. Mem means memory execution cluster FP0 means floating point port 0 (includes multiply, divide, convert and shuffle). FP1 means floating point port 1 (adder). |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay. |
| Reciprocal throughput: | The average number of clock cycles per instruction for a series of independent instructions of the same kind in the same thread. Delays in the decoders are included in the latency and throughput timings. Values of 4 or more are often caused by bottlenecks in the decoders and microcode ROM rather than the execution units. |

Integer instructions

| | Operands | μ ops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|------------|-----------|-------|---------|-----------------------|-----------------|
| Move instructions | | | | | | |
| MOV | r,r | 1 | IP0/1 | 1 | 0.5 | All addr. modes |
| MOV | r,i | 1 | IP0/1 | 1 | 0.5 | |
| MOV | r,m | 1 | Mem | 4 | 1 | |
| MOV | m,r | 1 | Mem | 3 | 1 | |
| MOV | m,i | 1 | Mem | | 1 | |
| MOVNTI | m,r | 1 | Mem | | 2 | |
| MOVSX MOVZX MOVSD | r16,r8 | 2 | IP0 | | 4 | Implicit lock |
| MOVSX MOVZX MOVSD | r16,m8 | 3 | IP0 | | 10 | |
| MOVSX MOVZX MOVSD | r32/64,r/m | 1 | IP0 | 1 | 1 | |
| CMOVcc | r,r | 1 | IP0/1 | 2 | 1 | |
| CMOVcc | r,m | 1 | | | 1 | |
| XCHG | r,r | 3 | IP0/1 | 8 | 8 | |
| XCHG | r,m | 3 | | 24 | 24 | |
| XLAT | | 4 | | 8 | 8 | |
| PUSH | r | 1 | IP0+1 | | 1 | |
| PUSH | i | 1 | IP0+1 | | 1 | |

Silvermont

| | | | | | | |
|--------------------------------|---------------|----|------------|----|-----|-------------------|
| PUSH | m | 3 | IP0+1 | | 5 | |
| PUSHF(D/Q) | | 18 | IP0+1 | | 29 | |
| PUSHA(D) | | 10 | | | 10 | Not in x64 mode |
| POP | r | 2 | | | 1 | |
| POP | (E/R)SP | 2 | | | 3 | |
| POP | m | 6 | | | 6 | |
| POPF(D/Q) | | 21 | | | 47 | |
| POPA(D) | | 17 | | | 14 | Not in x64 mode |
| LAHF | | 1 | | 1 | 1 | |
| SAHF | | 1 | IP0 | 2 | 1 | |
| SALC | | 2 | | 6 | 4 | Not in x64 mode |
| LEA | r,[r+d] | 1 | IP0/1 | 1 | 1 | |
| LEA | r,[r+r*s] | 1 | IP1 | 1 | 1 | |
| LEA | r,[r+r*s+d] | 1 | IP0+1 | 2 | 2 | |
| LEA | r,[rip+d] | 1 | IP0/1 | | 0.5 | |
| LEA | r16,[m] | 2 | | 4 | 4 | |
| BSWAP | r | 1 | IP0 | 1 | 1 | |
| MOVBE | r16,m16 | 1 | | | 2 | |
| MOVBE | r32/64,m32/64 | 1 | | | 1 | |
| MOVBE | m,r | 1 | | | 1 | |
| PREFETCHNTA | m | 1 | | | 1 | |
| PREFETCHT0/1/2 | m | 1 | | | 1 | |
| PREFETCHNTW | m | 1 | | | 1 | |
| LFENCE | | 2 | | | 8 | |
| MFENCE | | 2 | | | 14 | |
| SFENCE | | 1 | | | 7 | |
| Arithmetic instructions | | | | | | |
| ADD SUB | r,r/i | 1 | IP0/1 | 1 | 0.5 | |
| ADD SUB | r,m | 1 | IP0/1, Mem | | 1 | |
| ADD SUB | m,r/i | 1 | IP0/1, Mem | 6 | 1 | |
| ADC SBB | r,r/i | 1 | IP0/1 | 2 | 2 | |
| ADC SBB | r,m | 1 | | | 2 | |
| ADC SBB | m,r/i | 1 | | 6 | 2 | |
| ADCX | r32,r32 | 1 | IP0+1 | 2 | 2 | |
| ADCX | r64,r64 | 1 | IP0+1 | 6 | 6 | |
| ADOX | r32,r32 | 1 | | 2 | 2 | |
| ADOX | r64,r64 | 1 | | 6 | 6 | |
| CMP | r,r/i | 1 | IP0/1 | 1 | 0.5 | |
| CMP | m,r/i | 1 | | | 1 | |
| INC DEC | r | 1 | IP0/1 | 1 | 1 | latency to flag=2 |
| NEG NOT | r | 1 | IP0/1 | 1 | 0.5 | |
| INC DEC NEG NOT | m | 1 | | 6 | 1 | |
| AAA | | 13 | | 12 | | Not in x64 mode |
| AAS | | 13 | | 12 | | Not in x64 mode |
| DAA | | 20 | | 16 | | Not in x64 mode |
| DAS | | 21 | | 16 | | Not in x64 mode |
| AAD | | 4 | | 5 | | Not in x64 mode |
| AAM | | 11 | | 24 | 16 | Not in x64 mode |
| MUL IMUL | r8 | 3 | IP0 | 5 | 5 | |
| MUL IMUL | r16 | 4 | IP0 | 5 | 5 | |
| MUL IMUL | r32 | 3 | IP0 | 5 | 5 | |
| MUL IMUL | r64 | 3 | IP0 | 7 | 7 | |

Silvermont

| | | | | | | |
|---------------------------|------------|-----|------------|--------|--------|---------------|
| IMUL | r16,r16 | 2 | IP0 | 4 | 4 | |
| IMUL | r32,r32 | 1 | IP0 | 3 | 1 | |
| IMUL | r64,r64 | 1 | IP0 | 5 | 2 | |
| IMUL | r16,r16,i | 2 | IP0 | 4 | 4 | |
| IMUL | r32,r32,i | 1 | IP0 | 3 | 1 | |
| IMUL | r64,r64,i | 1 | IP0 | 5 | 2 | |
| MUL IMUL | m8 | 3 | IP0 | | | |
| MUL IMUL | m16 | 5 | IP0 | | | |
| MUL IMUL | m32 | 4 | IP0 | | | |
| MUL IMUL | m64 | 4 | IP0 | 14 | | |
| MULX | r,r,r | 3-4 | IP0 | 8 | 8 | |
| MULX | r,r,m | 4 | IP0 | | 8-10 | |
| DIV | r/m8 | 9 | IP0, FP0 | 24 | 19 | |
| DIV | r/m16 | 12 | IP0, FP0 | 25-29 | 19-23 | |
| DIV | r/m32 | 12 | IP0, FP0 | 25-39 | 19-31 | |
| DIV | r/m 64 | 23 | IP0, FP0 | 34-94 | 25-94 | |
| IDIV | r/m8 | 26 | IP0, FP0 | 24-35 | 25 | |
| IDIV | r/m16 | 29 | IP0, FP0 | 37-41 | 30-32 | |
| IDIV | r/m32 | 29 | IP0, FP0 | 29-46 | 29-38 | |
| IDIV | r/m64 | 44 | IP0, FP0 | 47-107 | 47-107 | |
| CBW | | 2 | IP0 | 4 | | |
| CWDE | | 1 | IP0 | 1 | | |
| CDQE | | 1 | IP0 | 1 | | |
| CWD | | 2 | IP0 | 4 | | |
| CDQ | | 1 | IP0 | 1 | | |
| CQO | | 1 | IP0 | 1 | | |
| POPCNT | r16,r16 | 2 | | 4 | 4 | |
| POPCNT | r32,r32 | 1 | | 3 | 1 | |
| POPCNT | r64,r64 | 1 | | 3 | 1 | |
| CRC32 | r32,r8 | 2 | | 4 | 4 | |
| CRC32 | r32,r16 | 1 | | 6 | 6 | |
| CRC32 | r32,r32 | 1 | | 3 | 1 | |
| Logic instructions | | | | | | |
| AND OR XOR | r,r/i | 1 | IP0/1 | 1 | 0.5 | |
| AND OR XOR | r,m | 1 | IP0/1, Mem | | 1 | |
| AND OR XOR | m,r/i | 1 | IP0/1, Mem | 6 | 1 | |
| TEST | r,r/i | 1 | IP0/1 | 1 | 0.5 | |
| TEST | m,r/i | 1 | IP0/1, Mem | | 1 | |
| SHR SHL SAR | r,i/cl | 1 | IP0 | 1 | 1 | |
| SHR SHL SAR | m,i/cl | 1 | IP0 | | 1 | |
| ROR ROL | r,i/cl | 1 | IP0 | 1 | 1 | |
| ROR ROL | m,i/cl | 1 | IP0 | | 1 | |
| RCR | r,1 | 7 | IP0 | 9 | | |
| RCL | r,1 | 1 | IP0 | 2 | 2 | |
| RCR | r,i/cl | 11 | IP0 | 12 | | |
| RCR | m,i/cl | 14 | IP0 | 13 | | |
| RCL | r,i/cl | 13 | IP0 | 12 | | |
| RCL | m,i/cl | 16 | IP0 | 14 | | |
| SHLD | r16,r16,i | 10 | IP0 | 10 | | 2 more if mem |
| SHLD | r32,r32,i | 1 | IP0 | 2 | | 4 more if mem |
| SHLD | r64,r64,i | 10 | IP0 | 10 | | 2 more if mem |
| SHLD | r16,r16,cl | 9 | IP0 | 10 | | 2 more if mem |

Silvermont

| | | | | | | |
|--------------------------------------|------------|--------|-------|--------|-------|---------------------|
| SHLD | r32,r32,cl | 2 | IP0 | 4 | | 2 more if mem |
| SHLD | r64,r64,cl | 9 | IP0 | 10 | | 2 more if mem |
| SHRD | r16,r16,i | 8 | IP0 | 10 | | 3 more if mem |
| SHRD | r32,r32,i | 2 | IP0 | 4 | | 4 more if mem |
| SHRD | r64,r64,i | 8-10 | IP0 | 10 | | 3 more if mem |
| SHRD | r16,r16,cl | 7 | IP0 | 10 | | 2 more if mem |
| SHRD | r32,r32,cl | 2 | IP0 | 4 | | 2 more if mem |
| SHRD | r64,r64,cl | 2 | IP0 | 4 | | 2 more if mem |
| BT | r,r/i | 1 | IP0+1 | 1 | 1 | |
| BT | m,r | 7 | | 9 | | |
| BT | m,i | 1 | | 1 | | |
| BTR BTS BTC | r,r/i | 1 | IP0+1 | 1 | 1 | |
| BTR BTS BTC | m,r | 8 | | | 10 | |
| BTR BTS BTC | m,i | 1 | IP0+1 | | 1 | |
| BSF BSR | r,r/m | 10 | IP0+1 | 10 | 10 | |
| SETcc | r/m | 1 | IP0+1 | 2 | 1 | |
| CLC STC | | 1 | IP0/1 | | 1 | |
| CMC | | 1 | | 1 | 1 | |
| CLD | | 4 | IP0+1 | | 7 | |
| STD | | 5 | IP0+1 | | 35 | |
| Control transfer instructions | | | | | | |
| JMP | short/near | 1 | IP1 | | 2 | |
| JMP | r | 1 | | | 2 | |
| JMP | m(near) | 1 | | | 2 | |
| Conditional jump | short/near | 1 | IP1 | | 1-2 | |
| J(E/R)CXZ | short | 2 | | | 2-15 | |
| LOOP | short | 7 | | | 10-20 | |
| LOOP(N)E | short | 8 | | | | |
| CALL | near | 1 | | | 2 | |
| CALL | r | 1 | | | 9 | |
| CALL | m | 3 | | | 14 | |
| RET | | 1 | | | 3 | |
| RET | i | 1 | | | 3 | |
| BOUND | r,m | 10 | | | 10 | Not in x64 mode |
| INTO | | 4 | | | 7 | Not in x64 mode |
| String instructions | | | | | | |
| LODS | | 3 | | 5 | | |
| REP LODS | | ~4n | | ~2n | | |
| STOS | | 2 | | 4 | | |
| REP STOS | | ~0.12B | | ~0.1B | | per byte, best case |
| MOVS | | 5 | | 6 | | |
| REP MOVS | | ~ 0.2B | | ~0.15B | | per byte, best case |
| SCAS | | 3 | | 5 | | |
| REP SCAS | | ~5n | | ~3n | | |
| CMPS | | 6 | | 6 | | |
| REP CMPS | | ~6n | | ~3n | | |
| Synchronization instructions | | | | | | |
| XADD | m,r | 6 | | 6 | | |

Silvermont

| | | | | | |
|------------------|-----|-------|-------|--------|-------|
| LOCK XADD | m,r | 4 | | 10 | |
| LOCK ADD | m,r | 1 | | 10 | |
| CMPXCHG | m,r | 8 | | 10 | |
| LOCK CMPXCHG | m,r | 6 | | 11 | |
| CMPXCHG8B | m,r | 13 | | 14 | |
| LOCK CMPXCHG8B | m,r | 11 | | 14 | |
| CMPXCHG16B | m,r | 19 | | 24 | |
| LOCK CMPXCHG16B | m,r | 17 | | 27 | |
| Other | | | | | |
| NOP (90) | | 1 | IP0/1 | | 0.5 |
| Long NOP (0F 1F) | | 1 | IP0/1 | | 0.5 |
| PAUSE | | 6 | | 24 | |
| ENTER | a,0 | 15 | | 14 | |
| ENTER | a,b | 19+6b | | 59+5b | |
| LEAVE | | 4 | | | 5 |
| CPUID | | 31-80 | | 54-108 | |
| RDTSC | | 13 | | 29 | |
| RDTSCP | | 15 | | 25 | |
| RDPMC | | 19 | | 19 | |
| RDRAND | r | 15 | | | ~1472 |

Floating point x87 instructions

| | Operands | μops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------------|----------|------|-------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| FLD | r | 1 | | 1 | 0.5 | |
| FLD | m32/m64 | 1 | | 4 | 1 | |
| FLD | m80 | 5 | | 9 | 8 | |
| FBLD | m80 | 59 | | 68 | 68 | |
| FST(P) | r | 1 | | 1 | 0.5 | |
| FST(P) | m32/m64 | 1 | | 3 | 2 | |
| FSTP | m80 | 8 | | 9 | 9 | |
| FBSTP | m80 | 204 | | 239 | 239 | |
| FXCH | r | 2 | FP0+1 | 1 | 1 | |
| FILD | m | 1 | | 6 | 2 | |
| FIST(P) | m | 6 | | 9 | 9 | |
| FISTTP | m | 7 | | 6 | 13 | |
| FLDZ | | 1 | | | 1 | |
| FLD1 | | 1 | | | 7 | |
| FLDPI FLDL2E etc. | | 2 | | | 7 | |
| FCMOVcc | r | 3 | | 6 | 6 | |
| FNSTSW | AX | 2 | | ~9 | 9 | |
| FNSTSW | m16 | 4 | | | 11 | |
| FLDCW | m16 | 2 | | ~6 | 4 | |
| FNSTCW | m16 | 4 | | ~5 | 5 | |
| FINCSTP FDECSTP | | 1 | | 1 | 0.5 | |
| FFREE(P) | | 1 | | | 0.5 | |
| FNSAVE | m | 166 | | 240 | 240 | |
| FRSTOR | m | 82 | | 174 | 174 | |
| Arithmetic instructions | | | | | | |

Silvermont

| | | | | | |
|--------------------|-----|-----|-----|--------|--------|
| FADD(P) FSUB(R)(P) | r/m | 1 | FP1 | 3 | 1 |
| FMUL(P) | r/m | 1 | FP0 | 5 | 2 |
| FDIV(R)(P) | r/m | 1 | FP0 | 39 | 37 |
| FABS | | 1 | | 1 | 1 |
| FCHS | | 1 | | 1 | 1 |
| FCOM(P) FUCOM | r/m | 1 | | 5 | 1 |
| FCOMPP FUCOMPP | | 1 | | 5 | 1 |
| FCOMI(P) FUCOMI(P) | r | 1 | | 5 | 1 |
| FIADD FISUB(R) | m | 3 | | | 5 |
| FIMUL | m | 3 | | | 6 |
| FIDIV(R) | m | 3 | | | 39 |
| FICOM(P) | m | 3 | | | 5 |
| FTST | | 1 | | 6 | 1 |
| FXAM | | 1 | | 7 | 1 |
| FPREM | | 27 | | 32-57 | 32-57 |
| FPREM1 | | 27 | | 32-57 | 32-57 |
| FRNDINT | | 18 | | 26 | 26 |
| Math | | | | | |
| FSCALE | | 27 | | | 66 |
| FXTRACT | | 15 | | 20 | 20 |
| FSQRT | | 1 | | 13-40 | 13-40 |
| FSIN FCOS | | 18 | | 40-170 | 40-170 |
| FSINCOS | | 110 | | 40-170 | |
| F2XM1 | | 9 | | 39-90 | |
| FYL2X | | 34 | | 80-140 | |
| FYL2XP1 | | 61 | | 154 | |
| FPTAN | | 101 | | 45-200 | |
| FPATAN | | 63 | | 85-190 | |
| Other | | | | | |
| FNOP | | 1 | | | 0.5 |
| WAIT | | 2 | | | 4 |
| FNCLEX | | 4 | | | 24 |
| FNINIT | | 19 | | | 65 |

Integer MMX and XMM instructions

| | Operands | μops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|-------------|------|-------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| MOVD MOVQ | r32/64,mm/x | 1 | | 4 | 1 | |
| MOVD MOVQ | m,mm/x | 1 | Mem | 3 | 1 | |
| MOVD MOVQ | mm/x,r32/64 | 1 | | 3 | 1 | |
| MOVD MOVQ | mm/x,m | 1 | Mem | 4 | 1 | |
| MOVQ | mm/x, mm/x | 1 | FP0/1 | 1 | 0.5 | |
| MOVDQA | x, x | 1 | FP0/1 | 1 | 0.5 | |
| MOVDQA MOVDQU | x, m128 | 1 | Mem | 4 | 1 | |
| MOVDQA MOVDQU | m128, x | 1 | Mem | 3 | 1 | |
| LDDQU | x, m128 | 1 | Mem | 4 | 1 | |
| MOVDQ2Q | mm, x | 1 | | 1 | 1 | |

Silvermont

| | | | | | | |
|--------------------------------|------------|-----|-------|------|-----|-----------|
| MOVQ2DQ | x,mm | 1 | | 1 | 1 | |
| MOVNTQ | m64,mm | 1 | Mem | ~370 | 1 | |
| MOVNTDQ | m128,x | 1 | Mem | ~370 | 1 | |
| MOVNTDQA | x, m128 | 1 | | 4 | 1 | |
| PACKSSWB/DW | | | | | | |
| PACKUSWB | mm/x, mm/x | 1 | FP0 | 1 | 1 | |
| PACKUSDW | x,x | 1 | FP0 | 1 | 1 | |
| PUNPCKH/LBW/WD/DQ | mm/x, mm/x | 1 | FP0 | 1 | 1 | |
| PUNPCKH/LQDQ | mm/x, mm/x | 1 | FP0 | 1 | 1 | |
| PMOVSX/ZX BW BD BQ DW | | | | | | |
| DQ | x,x | 1 | | 1 | 1 | |
| PMOVSX/ZX BW BD BQ DW | | | | | | |
| DQ | x,m | 1 | | 1 | 1 | |
| PSHUFB | mm,mm | 1 | FP0 | 1 | 1 | |
| PSHUFB | x,x | 4 | FP0 | 5 | 5 | |
| PSHUFW | mm,mm,i | 1 | FP0 | 1 | 1 | |
| PSHUFL/HW | x,x,i | 1 | FP0 | 1 | 1 | |
| PSHUFD | x,x,i | 1 | FP0 | 1 | 1 | |
| PALIGNR | x,x,i | 1 | FP0 | 1 | 1 | |
| PBLENDVB | x,x,xmm0 | 2 | FP0 | 4 | 4 | |
| PBLENDVB | x,m,xmm0 | 3 | FP0 | | 5 | |
| PBLENDW | x,x/m,i | 1 | FP0 | 1 | 1 | |
| MASKMOVQ | mm,mm | 1 | Mem | ~370 | 1 | |
| MASKMOVDQU | x,x | 3 | Mem | ~370 | 5 | |
| PMOVMSKB | r32,mm/x | 1 | | 4 | 1 | |
| PINSRW | mm/x,r32,i | 1 | | 3 | 1 | |
| PINSRB/D/Q | x,r32,i | 1 | | 3 | 1 | |
| PINSRB/D/Q | x,m8,i | 1 | | | 1 | |
| PEXTRW | r32,mm/x,i | 2 | | 5 | 4 | |
| PEXTRB/W | r32,x,i | 2 | | 5 | 4 | |
| PEXTRQ | r64,x,i | 2 | | 7 | 7 | |
| PEXTRB/W | m8/16,x,i | 5 | | | 6 | |
| PEXTRD | m32,x,i | 4 | | | 5 | |
| PEXTRQ | m64,x,i | 4 | | | 8 | |
| Arithmetic instructions | | | | | | |
| PADD/SUB(U)(S)B/W/D | mm/x, mm/x | 1 | FP0/1 | 1 | 0.5 | |
| PADDQ PSUBQ | mm/x, mm/x | 2 | | 4 | 4 | |
| PADDQ PSUBQ | mm/x, m | 3 | | | 5 | |
| PHADD(S)W PHSUB(S)W | mm, mm | 5 | | 6 | 6 | |
| PHADD(S)W PHSUB(S)W | x, x/m | 7-8 | | 9 | 9 | |
| PHADDD PHSUBD | mm/x, mm/x | 3-4 | | 5-6 | 5-6 | |
| PCMPEQ/GTB/W/D | mm/x,mm/x | 1 | FP0/1 | 1 | 0.5 | |
| PCMPEQQ | x, x | 2 | | 4 | 4 | +1 if mem |
| PCMPGTQ | x, x | 1 | FP0 | 5 | 2 | |
| PMULL/HW PMULHUW | mm,mm | 1 | FP0 | 4 | 1 | |
| PMULL/HW PMULHUW | x, x | 1 | FP0 | 5 | 2 | |
| PMULHSW | mm,mm | 1 | FP0 | 4 | 1 | |
| PMULHSW | x, x | 1 | FP0 | 5 | 2 | |
| PMULLD | x, x | 7 | FP0 | 11 | 11 | +1 if mem |
| PMULDQ | x, x | 1 | FP0 | 5 | 2 | |
| PMULUDQ | mm,mm | 1 | FP0 | 4 | 1 | |
| PMULUDQ | x, x | 1 | FP0 | 5 | 2 | |
| PMADDWD | mm,mm | 1 | FP0 | 4 | 1 | |

Silvermont

| | | | | | | |
|--------------------------------|-----------|---|-------|----|-------|-----------|
| PMADDWD | x, x | 1 | FP0 | 5 | 2 | |
| PMADDUBSW | mm,mm | 1 | FP0 | 4 | 1 | |
| PMADDUBSW | x, x | 1 | FP0 | 5 | 2 | |
| PSADBW | mm,mm | 1 | FP0 | 4 | 1 | |
| PSADBW | x, x | 1 | FP0 | 5 | 2 | |
| MPSADBW | x,x,i | 3 | | 7 | 6 | |
| MPSADBW | x,m,i | 4 | | | 6 | |
| PAVGB/W | mm/x,mm/x | 1 | FP0/1 | 1 | 0.5 | |
| PMIN/MAXUB | mm/x,mm/x | 1 | FP0/1 | 1 | 0.5 | |
| PMIN/MAXSW | mm/x,mm/x | 1 | FP0/1 | 1 | 0.5 | |
| PMIN/PMAX | | | | | | |
| SB/SW/SD | | | | | | |
| UB/UW/UD | x,x | 1 | | 1 | 1 | |
| PHMINPOSUW | x,x | 1 | FP0 | 5 | 2 | |
| PASB PASW PABSD | mm/x,mm/x | 1 | FP0/1 | 1 | 0.5-1 | |
| PSIGNB PSIGNW PSIGND | mm/x,mm/x | 1 | FP0/1 | 1 | 0.5-1 | |
| Logic instructions | | | | | | |
| PAND(N) POR PXOR | mm/x,mm/x | 1 | FP0/1 | 1 | 0.5 | |
| PTEST | x,x | 1 | | 1 | 1 | |
| PSLL/RL/RAW/D/Q | mm/x,mm/x | 2 | FP0 | 2 | 2 | |
| PSLL/RL/RAW/D/Q | mm/x,i | 1 | FP0 | 1 | 1 | |
| PSLL/RDQ | x,i | 1 | FP0 | 1 | 1 | |
| String instructions | | | | | | |
| PCMPESTRI | x,x,i | 9 | FP0 | 21 | 21 | +1 if mem |
| PCMPESTRM | x,x,i | 8 | FP0 | 17 | 17 | +1 if mem |
| PCMPISTRI | x,x,i | 6 | FP0 | 17 | 17 | +1 if mem |
| PCMPISTRM | x,x,i | 5 | FP0 | 13 | 13 | +1 if mem |
| Encryption instructions | | | | | | |
| PCLMULQDQ | x,x,i | 8 | FP0 | 10 | 10 | +1 if mem |
| Other | | | | | | |
| EMMS | | 9 | | | 10 | |

Floating point XMM instructions

| | Operands | μops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|----------|------|-------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| MOVAPS/D | x, x | 1 | FP0/1 | 1 | 0.5 | |
| MOVAPS/D | x,m128 | 1 | Mem | 4 | 1 | |
| MOVAPS/D | m128,x | 1 | Mem | 3 | 1 | |
| MOVUPS/D | x,m128 | 1 | Mem | 4 | 1 | |
| MOVUPS/D | m128,x | 1 | Mem | 3 | 1 | |
| MOVSS/D | x, x | 1 | FP0/1 | 1 | 0.5 | |
| MOVSS/D | x,m32/64 | 1 | Mem | 4 | 1 | |
| MOVSS/D | m32/64,x | 1 | Mem | 3 | 1 | |
| MOVHPS/D MOVLPS/D | x,m64 | 1 | Mem | 4 | 1 | |
| MOVHPS/D MOVLPS/D | m64,x | 1 | Mem | 3 | 1 | |
| MOVLHPS MOVHLPS | x,x | 1 | FP0 | 1 | 1 | |
| BLENDPS/PD | x,x/m,i | 1 | | 1 | 1 | |

Silvermont

| | | | | | | |
|-------------------|----------|---|-------|------|----|-----------|
| BLENDVPS/PD | x,x,xmm0 | 2 | FP0+1 | 4 | 4 | |
| BLENDVPS/PD | x,m,xmm0 | 3 | FP0+1 | 5 | 5 | |
| INSERTPS | x,x,i | 1 | | 1 | 1 | |
| INSERTPS | x,m32,i | 3 | | 5 | 5 | |
| EXTRACTPS | r32,x,i | 2 | | | 4 | |
| EXTRACTPS | m32,x,i | 4 | | 4 | 5 | |
| MOVMSKPS/D | r32,x | 1 | FP0 | 4 | 1 | |
| MOVNTPS/D | m128,x | 1 | Mem | ~370 | 1 | |
| SHUFPS | x,x,i | 1 | FP0 | 1 | 1 | |
| SHUFPD | x,x,i | 1 | FP0 | 1 | 1 | |
| MOVDDUP | x, x | 1 | FP0 | 1 | 1 | |
| MOVSH/LDUP | x, x | 1 | FP0 | 1 | 1 | |
| UNPCKH/LPS | x, x | 1 | FP0 | 1 | 1 | |
| UNPCKH/LPD | x, x | 1 | FP0 | 1 | 1 | |
| Conversion | | | | | | |
| CVTPD2PS | x, x | 1 | FP0 | 5 | 2 | |
| CVTSD2SS | x, x | 1 | FP0 | 4 | 2 | |
| CVTPS2PD | x, x | 1 | FP0 | 5 | 2 | |
| CVTSS2SD | x, x | 1 | FP0 | 4 | 2 | |
| CVTDQ2PS | x, x | 1 | FP0 | 5 | 2 | |
| CVT(T) PS2DQ | x, x | 1 | FP0 | 5 | 2 | |
| CVTDQ2PD | x, x | 1 | FP0 | 5 | 2 | |
| CVT(T)PD2DQ | x, x | 1 | FP0 | 5 | 2 | |
| CVTPI2PS | x,mm | 1 | FP0 | 4 | 2 | |
| CVT(T)PS2PI | mm,x | 1 | FP0 | 4 | 2 | |
| CVTPI2PD | x,mm | 1 | FP0 | 5 | 2 | |
| CVT(T) PD2PI | mm,x | 1 | FP0 | 5 | 2 | |
| CVTSI2SS | x,r32 | 1 | FP0 | 5 | 2 | |
| CVT(T)SS2SI | r32,x | 1 | FP0 | 5 | 1 | |
| CVTSI2SD | xm,r32 | 1 | FP0 | 5 | 2 | |
| CVT(T)SD2SI | r32,x | 3 | FP0 | 5 | 1 | |
| Arithmetic | | | | | | |
| ADDSS SUBSS | x, x | 1 | FP1 | 3 | 1 | |
| ADDSD SUBSD | x, x | 1 | FP1 | 3 | 1 | |
| ADDPSS SUBPS | x, x | 1 | FP1 | 3 | 1 | |
| ADDPD SUBPD | x, x | 1 | FP1 | 4 | 2 | |
| ADDSSUBPS | x, x | 1 | FP1 | 3 | 1 | |
| ADDSSUBPD | x, x | 1 | FP1 | 4 | 2 | |
| HADDPSS HSUBPS | x, x | 4 | | 6 | 6 | +1 if mem |
| HADDPD HSUBPD | x, x | 4 | | 6 | 5 | +1 if mem |
| MULSS | x, x | 1 | FP0 | 4 | 1 | |
| MULSD | x, x | 1 | FP0 | 5 | 2 | |
| MULPS | x, x | 1 | FP0 | 5 | 2 | |
| MULPD | x, x | 1 | FP0 | 7 | 4 | |
| DIVSS | x, x | 1 | FP0 | 19 | 17 | |
| DIVSD | x, x | 1 | FP0 | 34 | 32 | |
| DIVPS | x, x | 6 | FP0 | 39 | 39 | |
| DIVPD | x, x | 6 | FP0 | 69 | 69 | |
| RCPSS | x, x | 1 | FP0 | 4 | 1 | |
| RCPPS | x, x | 5 | FP0 | 9 | 8 | |
| CMPccSS/D PS/D | x, x | 1 | FP1 | 3 | 1 | |

Silvermont

| | | | | | | |
|--------------------|-------|-----|-------|-----|-----|-------------|
| COMISS/D UCOMISS/D | x, x | 1 | FP1 | | 1 | |
| MAXSS/D MINSS/D | x, x | 1 | FP1 | 3 | 1 | |
| MAXPS MINPS | x, x | 1 | FP1 | 3 | 1 | |
| MAXPD MINPD | x, x | 1 | FP1 | 4 | 2 | |
| ROUNDSS/D | x,x,i | 1 | FP0 | 4 | 2 | |
| ROUNDPS/D | x,x,i | 1 | FP0 | 5 | 2 | |
| DPPS | x,x,i | 9 | FP0 | 15 | 12 | +1 if mem |
| DPPD | x,x,i | 5 | FP0 | 12 | 8 | +1 if mem |
| Math | | | | | | |
| SQRTSS | x, x | 1 | FP0 | 20 | 18 | |
| SQRTPS | x, x | 5 | FP0 | 40 | 40 | |
| SQRTSD | x, x | 1 | FP0 | 35 | 33 | |
| SQRTPD | x, x | 5 | FP0 | 70 | 70 | |
| RSQRTSS | x, x | 1 | FP0 | 4 | 1 | |
| RSQRTPS | x, x | 5 | FP0 | 9 | 8 | |
| Logic | | | | | | |
| ANDPS/D | x, x | 1 | FP0/1 | 1 | 0.5 | |
| ANDNPS/D | x, x | 1 | FP0/1 | 1 | 0.5 | |
| ORPS/D | x, x | 1 | FP0/1 | 1 | 0.5 | |
| XORPS/D | x, x | 1 | FP0/1 | 1 | 0.5 | |
| Other | | | | | | |
| LDMXCSR | m32 | 5 | | 10 | 8 | |
| STMXCSR | m32 | 4 | | 12 | 11 | |
| FXSAVE | m4096 | 115 | | 132 | 132 | 32 bit mode |
| FXSAVE | m4096 | 123 | | 143 | 143 | 64 bit mode |
| FXRSTOR | m4096 | 114 | | 118 | 118 | 32 bit mode |
| FXRSTOR | m4096 | 123 | | 122 | 122 | 64 bit mode |

Intel Goldmont

List of instruction timings and μ op breakdown

Explanation of column headings:

| | |
|-------------------------------|---|
| Instruction: | Name of instruction. Multiple names mean that these instructions have the same data. |
| Operands: | i = immediate data, r = register, mm = 64 bit mmx register, x = 128 bit xmm register, v = mmx or xmm vector register, m = memory, m32 = 32-bit memory operand, etc. |
| μops: | The number of μ ops from the decoder or ROM. A μ op that goes to multiple units is counted as one. |
| Unit: | Tells which execution unit is used. Instructions that use the same unit cannot execute simultaneously. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay. |
| Reciprocal throughput: | The average number of clock cycles per instruction for a series of independent instructions of the same kind in the same thread. Delays in the decoders are included in the latency and throughput timings. Values of 4 or more are often caused by bottlenecks in the decoders and microcode ROM rather than the execution units. |

Integer instructions

| | Operands | μ ops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|----------|-----------|------|---------|-----------------------|-----------------|
| Move instructions | | | | | | |
| MOV | r,r | 1 | | 1 | 0.33 | |
| MOV | r,i | 1 | | 1 | 0.33 | |
| MOV | r,m | 1 | | 3 | 1 | All addr. modes |
| MOV | m,r | 1 | | 2 | 1 | All addr. modes |
| MOV | m,i | 1 | | | 1 | |
| MOVNTI | m,r | 2 | | 2 | 2 | |
| MOVSX MOVZX | r16,r8 | 2 | | | 4 | |
| MOVSX MOVZX | r16,m8 | 3 | | | 4 | |
| MOVSX MOVZX MOVSX | r32/64,r | 1 | | 1 | 1 | |
| MOVSX MOVXSD | r32/64,m | 1 | | 3 | 1 | |
| MOVZX | r32/64,m | 1 | | 2 | 1 | |
| CMOVcc | r,r | 1 | | 2 | 1 | |
| CMOVcc | r,m | 1 | | | 1 | |
| XCHG | r,r | 3 | | 4 | 4 | |
| XCHG | r,m | 3 | | 13 | 13 | Implicit lock |
| XLAT | | 4 | | 5 | 5 | |
| PUSH | r | 1 | | | 1 | |
| PUSH | i | 1 | | | 1 | |
| PUSH | m | 2 | | | 1 | |
| PUSH | sp | 2 | | | 1 | |
| PUSHF(D/Q) | | 16 | | | 26 | |
| PUSHA(D) | | 10 | | | 8 | Not in x64 mode |
| POP | r | 1 | | | 1 | |
| POP | m | 6 | | | 5 | |

Goldmont

| | | | | | | |
|--------------------------------|-------------|----|--|----|------|-------------------|
| POP | sp | 1 | | | 2.5 | |
| POPF(D/Q) | | 22 | | | 42 | |
| POPA(D) | | 17 | | | 10 | Not in x64 mode |
| LAHF | | 1 | | 2 | 2 | |
| SAHF | | 1 | | 1 | 1 | |
| SALC | | 2 | | 4 | 4 | Not in x64 mode |
| LEA | r16,[m] | 2 | | 5 | 4 | |
| LEA | r,[r+d] | 1 | | 1 | 1 | |
| LEA | r,[r+r*1] | 1 | | 1 | 1 | |
| LEA | r32,[r+r*s] | 1 | | 2 | 2 | |
| LEA | r64,[r+r*s] | 1 | | 1 | 1 | |
| LEA | r,[r+r*s+d] | 1 | | 2 | 2 | |
| LEA | r,[rip+d] | 1 | | | 0.5 | |
| BSWAP | r | 1 | | 1 | 1 | |
| MOVBE | r,m | 1 | | | 1 | |
| MOVBE | m,r | 1 | | | 1 | |
| PREFETCHNTA | m | 1 | | | 1 | |
| PREFETCHT0/1/2 | m | 1 | | | 1 | |
| PREFETCHNTW | m | 1 | | | 1 | |
| LFENCE | | 3 | | | 8 | |
| MFENCE | | 3 | | | 22 | |
| SFENCE | | 1 | | | 2 | |
| CLFLUSH | m | 1 | | | 165 | |
| CLFLUSHOPT | m | 4 | | | 165 | |
| Arithmetic instructions | | | | | | |
| ADD SUB | r,r/i | 1 | | 1 | 0.33 | |
| ADD SUB | r,m | 1 | | | 1 | |
| ADD SUB | m,r/i | 1 | | 5 | 1 | |
| ADC SBB | r,r/i | 1 | | 2 | 2 | |
| ADC SBB | r,m | 1 | | | 2 | |
| ADC SBB | m,r/i | 1 | | 6 | 2 | |
| CMP | r,r/i | 1 | | 1 | 0.33 | |
| CMP | m,r/i | 1 | | | 1 | |
| INC DEC | r | 1 | | 1 | 1 | latency to flag=2 |
| NEG NOT | r | 1 | | 1 | 0.33 | |
| INC DEC NEG NOT | m | 1 | | 5 | 1 | |
| AAA | | 14 | | 10 | | Not in x64 mode |
| AAS | | 14 | | 10 | | Not in x64 mode |
| DAA | | 21 | | 13 | | Not in x64 mode |
| DAS | | 22 | | 14 | | Not in x64 mode |
| AAD | | 4 | | 5 | | Not in x64 mode |
| AAM | | 5 | | 14 | 11 | Not in x64 mode |
| MUL IMUL | r8 | 2 | | 4 | 4 | |
| MUL IMUL | r16 | 4 | | 5 | 5 | |
| MUL IMUL | r32 | 2 | | 4 | 2 | |
| MUL IMUL | r64 | 2 | | 6 | 2 | |
| IMUL | r16,r16 | 2 | | 4 | 4 | |
| IMUL | r32,r32 | 1 | | 3 | 1 | |
| IMUL | r64,r64 | 1 | | 5 | 2 | |
| IMUL | r16,r16,i | 2 | | 4 | 4 | |
| IMUL | r32,r32,i | 1 | | 3 | 1 | |
| IMUL | r64,r64,i | 1 | | 5 | 2 | |

Goldmont

| | | | | | | |
|---------------------------|------------|----|--|-------|-------|---------------|
| MUL IMUL | m8 | 3 | | | | |
| MUL IMUL | m16 | 5 | | | | |
| MUL IMUL | m32 | 2 | | | 1 | |
| MUL IMUL | m64 | 2 | | | 2 | |
| DIV | r8 | 3 | | 13-14 | 11-12 | |
| DIV | r16 | 6 | | 14-19 | 13-18 | |
| DIV | r32 | 6 | | 14-27 | 13-26 | |
| DIV | r64 | 6 | | 14-43 | 13-42 | |
| IDIV | r8 | 3 | | 13-14 | 11-12 | |
| IDIV | r16 | 6 | | 14-19 | 13-18 | |
| IDIV | r32 | 6 | | 14-27 | 13-26 | |
| IDIV | r64 | 6 | | 14-43 | 13-42 | |
| CBW | | 2 | | 4 | | |
| CWDE | | 1 | | 1 | | |
| CDQE | | 1 | | 1 | | |
| CWD | | 2 | | 4 | | |
| CDQ | | 1 | | 1 | | |
| CQO | | 1 | | 1 | | |
| POPCNT | r16,r16 | 2 | | 4 | 4 | |
| POPCNT | r32,r32 | 1 | | 3 | 1 | |
| POPCNT | r64,r64 | 1 | | 3 | 1 | |
| CRC32 | r32,r8 | 1 | | 3 | 1 | |
| CRC32 | r32,r16 | 1 | | 3 | 2 | |
| CRC32 | r32,r32 | 1 | | 3 | 1 | |
| Logic instructions | | | | | | |
| AND OR XOR | r,r/i | 1 | | 1 | 0.33 | |
| AND OR XOR | r,m | 1 | | | 1 | |
| AND OR XOR | m,r/i | 1 | | 5 | 1 | |
| TEST | r,r/i | 1 | | 1 | 0.33 | |
| TEST | m,r/i | 1 | | | 1 | |
| SHR SHL SAR | r,i/cl | 1 | | 1 | 1 | |
| SHR SHL SAR | m,i/cl | 1 | | | 1 | |
| ROR ROL | r,i/cl | 1 | | 1 | 1 | |
| ROR ROL | m,i/cl | 1 | | | 1 | |
| RCL | r,1 | 1 | | 2 | 2 | |
| RCL | r,i | 16 | | 14 | 14 | 2 more if mem |
| RCL | r,cl | 16 | | 16 | 16 | 2 more if mem |
| RCR | r,1 | 10 | | 11 | 11 | 3 more if mem |
| RCR | r,i | 14 | | 12 | 12 | 2 more if mem |
| RCR | r,cl | 14 | | 14 | 14 | 2 more if mem |
| SHLD | r16,r16,i | 10 | | 15 | 15 | 4 more if mem |
| SHLD | r32,r32,i | 2 | | 2 | 2 | 6 more if mem |
| SHLD | r64,r64,i | 13 | | 12 | 12 | 2 more if mem |
| SHLD | r16,r16,cl | 10 | | 17 | 17 | 4 more if mem |
| SHLD | r32,r32,cl | 2 | | 4 | 4 | 6 more if mem |
| SHLD | r64,r64,cl | 12 | | 14 | 14 | 2 more if mem |
| SHRD | r16,r16,i | 10 | | 11 | 11 | 3 more if mem |
| SHRD | r32,r32,i | 2 | | 2 | 2 | 6 more if mem |
| SHRD | r64,r64,i | 13 | | 12 | 12 | 2 more if mem |
| SHRD | r16,r16,cl | 10 | | 13 | 13 | 2 more if mem |
| SHRD | r32,r32,cl | 2 | | 4 | 4 | 6 more if mem |
| SHRD | r64,r64,cl | 12 | | 14 | 14 | 2 more if mem |

Goldmont

| | | | | | | |
|--------------------------------------|------------|--------|--|--------|-------|---------------------|
| BT | r,r/i | 1 | | 1 | 1 | |
| BT | m,r | 10 | | | 12 | |
| BT | m,i | 1 | | | 1 | |
| BTR BTS BTC | r,r/i | 1 | | 1 | 1 | |
| BTR BTS BTC | m,r | 10 | | 12 | 10 | |
| BTR BTS BTC | m,i | 1 | | | 2 | |
| BSF BSR | r,r/m | 11 | | 10 | 8 | |
| SETcc | r/m | 1 | | 2 | 1 | |
| CLC STC | | 1 | | | 1 | |
| CMC | | 1 | | 1 | 1 | |
| CLD | | 3 | | | 7 | |
| STD | | 4 | | | 35 | |
| Control transfer instructions | | | | | | |
| JMP | short/near | 1 | | | 2 | |
| JMP | r | 1 | | | 2 | |
| JMP | m(near) | 1 | | | 2 | |
| Conditional jump | short/near | 1 | | | 1-2 | |
| J(E/R)CXZ | short | 2 | | | 4-18 | |
| LOOP | short | 9 | | | 10-23 | |
| LOOP(N)E | short | 9 | | | 12-24 | |
| CALL | near | 1 | | | 2 | |
| CALL | r | 1 | | | 8 | |
| CALL | m | 3 | | | 11 | |
| RET | | 1 | | | 2-3 | |
| RET | i | 2 | | | 2 | |
| BOUND | r,m | 9 | | | 9 | Not in x64 mode |
| INTO | | 4 | | | 7 | Not in x64 mode |
| String instructions | | | | | | |
| LODS | | 3 | | 4 | | |
| REP LODS | | ~5n | | ~2n | | |
| STOS | | 2 | | 4 | | |
| REP STOS | | ~0.13B | | ~0.07B | | per byte, best case |
| MOVS | | 5 | | 5 | | |
| REP MOVS | | ~ 0.2B | | ~0.07B | | per byte, best case |
| SCAS | | 3 | | 4 | | |
| REP SCAS | | ~6n | | ~2n | | |
| CMPS | | 6 | | 4 | | |
| REP CMPS | | ~8n | | ~3n | | |
| Synchronization instructions | | | | | | |
| XADD | m,r | 6 | | 5 | | |
| LOCK XADD | m,r | 6 | | 13 | | |
| LOCK ADD | m,r | 1 | | 13 | | |
| CMPXCHG | m,r | 8 | | 8 | | |
| LOCK CMPXCHG | m,r | 8 | | 14 | | |
| CMPXCHG8B | m,r | 14 | | 9 | | |
| LOCK CMPXCHG8B | m,r | 14 | | 16 | | |
| CMPXCHG16B | m,r | 19 | | 23 | | |
| LOCK CMPXCHG16B | m,r | 19 | | 30 | | |

Goldmont

| | | | | | | |
|------------------|-----|-------|--|---------|-------|--|
| Other | | | | | | |
| NOP (90) | | 1 | | | 0.33 | |
| Long NOP (0F 1F) | | 1 | | | 1 | |
| PAUSE | | 3 | | 147 | | |
| ENTER | a,0 | 14 | | 10 | | |
| ENTER | a,b | 17+6b | | ~75+3b | | |
| LEAVE | | 2 | | | 3 | |
| CPUID | | 37-78 | | 69-2800 | | |
| RDTSC | | 19 | | | 20 | |
| RDTSCP | | 22 | | | 31 | |
| RDPMC | | 13 | | | 9 | |
| RDRAND | r | 17 | | | ~3100 | |
| RDSEED | r | 17 | | | ~3100 | |

Floating point x87 instructions

| | Operands | μops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------------|----------|------|------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| FLD | r | 1 | | 1 | 0.5 | |
| FLD | m32/m64 | 1 | | 3 | 1 | |
| FLD | m80 | 1 | | 3 | 1 | |
| FBLD | m80 | 54 | | 56 | 50 | |
| FST(P) | r | 1 | | 1 | 0.5 | |
| FST(P) | m32/m64 | 1 | | 3 | 2 | |
| FSTP | m80 | 1 | | 3 | 2 | |
| FBSTP | m80 | 195 | | 190 | 190 | |
| FXCH | r | 2 | | 1 | 1 | |
| FILD | m | 1 | | 7 | 1 | |
| FIST(P) | m | 8 | | 6 | 10 | |
| FISTTP | m | 8 | | 6 | 13 | |
| FLDZ | | 1 | | | 1 | |
| FLD1 | | 2 | | | 6 | |
| FLDPI FLDL2E etc. | | 2 | | | 3 | |
| FCMOVcc | r | 4 | | 5 | 5 | |
| FNSTSW | AX | 2 | | | 10 | |
| FNSTSW | m16 | 3 | | | 11 | |
| FLDCW | m16 | 4 | | | 15 | |
| FNSTCW | m16 | 3 | | | 4 | |
| FINCSTP FDECSTP | | 1 | | 1 | 0.5 | |
| FFREE(P) | | 1 | | | 0.5 | |
| FNSAVE | m | 151 | | 173 | 173 | |
| FRSTOR | m | 85 | | 155 | 155 | |
| Arithmetic instructions | | | | | | |
| FADD(P) FSUB(R)(P) | r | 1 | | 3 | 1 | |
| FADD(P) FSUB(R)(P) | m | 2 | | | 2 | |
| FMUL(P) | r | 1 | | 5 | 2 | |
| FMUL(P) | m | 2 | | | 2 | |
| FDIV(R)(P) | r | 1 | | 39 | 38 | |
| FDIV(R)(P) | m | 2 | | | 38 | |
| FABS | | 1 | | 1 | 1 | |

Goldmont

| | | | | | | |
|--------------------|-----|--------|--|--------|--------|--|
| FCHS | | 1 | | 1 | 1 | |
| FCOM(P) FUCOM | r/m | 1 | | | 1 | |
| FCOMPP FUCOMPP | | 1 | | | 1 | |
| FCOMI(P) FUCOMI(P) | r | 1 | | | 1 | |
| FIADD FISUB(R) | m | 3 | | | 5 | |
| FIMUL | m | 3 | | | 5 | |
| FIDIV(R) | m | 4 | | | 38 | |
| FICOM(P) | m | 3 | | | 4 | |
| FTST | | 1 | | | 1 | |
| FXAM | | 1 | | | 1 | |
| FPREM | | 29 | | 37-42 | 37-42 | |
| FPREM1 | | 29 | | 37-42 | 37-42 | |
| FRNDINT | | 19 | | 41 | 41 | |
| Math | | | | | | |
| FSCALE | | 30 | | 32 | 32 | |
| FXTRACT | | 16 | | 22 | 22 | |
| FSQRT | | 1 | | 10-40 | 40-40 | |
| FSIN FCOS | | 17-100 | | 45-150 | 45-150 | |
| FSINCOS | | 17-110 | | 48-135 | 48-135 | |
| F2XM1 | | 9-27 | | 40-90 | 40-90 | |
| FYL2X | | 34-61 | | 88-130 | 88-130 | |
| FYL2XP1 | | 61 | | 140 | 140 | |
| FPTAN | | 16-100 | | 45-180 | 45-180 | |
| FPATAN | | 33-65 | | 85-190 | 85-190 | |
| Other | | | | | | |
| FNOP | | 1 | | | 0.5 | |
| WAIT | | 2 | | | 6 | |
| FNCLEX | | 4 | | | 24 | |
| FNINIT | | 14 | | | 49 | |

Integer MMX and XMM instructions

| | Operands | µops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|----------|------|------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| MOVD MOVQ | r32/64,v | 1 | | 4 | 1 | |
| MOVD MOVQ | m,v | 1 | | 3 | 1 | |
| MOVD MOVQ | v,r32/64 | 1 | | 4 | 1 | |
| MOVD MOVQ | v,m | 1 | | 3 | 1 | |
| MOVQ | v,v | 1 | | 1 | 0.5 | |
| MOVDQA | x, x | 1 | | 1 | 0.5 | |
| MOVDQA MOVDQU | x, m128 | 1 | | 3 | 1 | |
| MOVDQA MOVDQU | m128, x | 1 | | 3 | 1 | |
| LDDQU | x, m128 | 1 | | 3 | 1 | |
| MOVDQ2Q | mm, x | 1 | | 1 | 0.5 | |
| MOVQ2DQ | x,mm | 1 | | 1 | 0.5 | |
| MOVNTQ | m64,mm | 1 | | 3 | 1 | |
| MOVNTDQ | m128,x | 1 | | 3 | 1 | |
| MOVNTDQA | x, m128 | 1 | | 3 | 1 | |

Goldmont

| | | | | | | |
|--------------------------------|----------|---|--|------|-----|-----------|
| PACKSSWB/DW | | | | | | |
| PACKUSWB | v,v | 1 | | 1 | 0.5 | |
| PACKUSDW | x,x | 1 | | 1 | 0.5 | |
| PUNPCKH/LBW/WD/DQ | v,v | 1 | | 1 | 0.5 | |
| PUNPCKH/LQDQ | v,v | 1 | | 1 | 0.5 | |
| PMOVsx/ZX BW BD BQ DW DQ | x,x | 1 | | 1 | 0.5 | |
| PMOVsx/ZX BW BD BQ DW DQ | x,m | 1 | | 1 | 1 | |
| PSHUFb | mm,mm | 1 | | 1 | 0.5 | |
| PSHUFb | x,x | 1 | | 1 | 1 | |
| PSHUFw | mm,mm,i | 1 | | 1 | 0.5 | |
| PSHUFL/HW | x,x,i | 1 | | 1 | 0.5 | |
| PSHUFD | x,x,i | 1 | | 1 | 0.5 | |
| PALIGNr | x,x,i | 1 | | 1 | 0.5 | |
| PBLENdvB | x,x,xmm0 | 2 | | 4 | 4 | |
| PBLENdvB | x,m,xmm0 | 3 | | | 4 | |
| PBLENdw | x,x/m,i | 1 | | 1 | 0.5 | |
| MASKMOVQ | mm,mm | 1 | | ~350 | 1 | |
| MASKMOVDQU | x,x | 3 | | ~360 | 1 | |
| PMOVMSKB | r32,v | 1 | | 4 | 1 | |
| PINSRW | v,r32,i | 1 | | 4 | 1 | |
| PINSRB/D/Q | x,r32,i | 1 | | 4 | 1 | |
| PINSRB/D/Q | x,m8,i | 1 | | | 1 | |
| PEXTRB/W/D/Q | r,v,i | 1 | | 4 | 1 | |
| PEXTRB/W/D/Q | m,v,i | 4 | | | 4 | |
| Arithmetic instructions | | | | | | |
| PADD/SUB(U)(S)B/W/D | v,v | 1 | | 1 | 0.5 | |
| PADDQ PSUBQ | v,v | 1 | | 2 | 1 | |
| PHADD(S)W PHSUB(S)W | mm, mm | 5 | | 7 | 7 | |
| PHADD(S)W PHSUB(S)W | x, x/m | 7 | | 6 | 6 | +1 if mem |
| PHADDd PHSUBd | v,v | 3 | | 4 | 4 | +1 if mem |
| PCMPEQ/GTB/W/D | v,v | 1 | | 1 | 0.5 | |
| PCMPEQQ | x, x | 1 | | 2 | 1 | |
| PCMPGTQ | x, x | 1 | | 2 | 1 | |
| PMULL/HW PMULHUW | v,v | 1 | | 4 | 1 | |
| PMULHRsw | v,v | 1 | | 4 | 1 | |
| PMULLD | x, x | 1 | | 5 | 2 | |
| PMULDQ | x, x | 1 | | 4 | 1 | |
| PMULUDQ | v,v | 1 | | 4 | 1 | |
| PMADDWD | v,v | 1 | | 4 | 1 | |
| PMADDUBSW | v,v | 1 | | 4 | 1 | |
| PSADBw | v,v | 1 | | 4 | 1 | |
| MPSADBw | x,x,i | 3 | | 5 | 4 | +1 if mem |
| PAVGB/W | v,v | 1 | | 1 | 0.5 | |
| PMIN/MAXUB | v,v | 1 | | 1 | 0.5 | |
| PMIN/PMAX SB/SW/SD | | | | | | |
| UB/UW/UD | x,x | 1 | | 1 | 0.5 | |
| PHMINPOSUW | x,x | 1 | | 5 | 2 | |
| PABSB PABSW PABSD | v,v | 1 | | 1 | 0.5 | |
| PSIGNB PSIGNW | | | | | | |
| PSIGND | v,v | 1 | | 1 | 0.5 | |

Goldmont

| | | | | | | |
|--------------------------------|-------|----|--|----|-----|-----------|
| Logic instructions | | | | | | |
| PAND(N) POR PXOR | v,v | 1 | | 1 | 0.5 | |
| PTEST | x,x | 1 | | 1 | 1 | |
| PSLL/RL/RAW/D/Q | v,v | 1 | | 2 | 1 | |
| PSLL/RL/RAW/D/Q | v,i | 1 | | 1 | 0.5 | |
| PSLL/RLDQ | x,i | 1 | | 1 | 0.5 | |
| String instructions | | | | | | |
| PCMPESTRI | x,x,i | 10 | | 13 | 13 | +1 if mem |
| PCMPESTRM | x,x,i | 9 | | 14 | 14 | +1 if mem |
| PCMPISTRI | x,x,i | 6 | | 8 | 8 | +1 if mem |
| PCMPISTRM | x,x,i | 5 | | 12 | 12 | +1 if mem |
| Encryption instructions | | | | | | |
| PCLMULQDQ | x,x,i | 3 | | 6 | 4 | |
| AESDEC | x,x | 1 | | 6 | 2 | +1 if mem |
| AESDECLAST | x,x | 1 | | 6 | 2 | +1 if mem |
| AESENC | x,x | 1 | | 6 | 2 | +1 if mem |
| AESENCLAST | x,x | 1 | | 6 | 2 | +1 if mem |
| AESIMC | x,x | 1 | | 5 | 2 | |
| AESKEYGENASSIST | x,x,i | 1 | | 5 | 2 | |
| SHA1RND4S | x,x,i | 1 | | 5 | 2 | |
| SHA1NEXTE | x,x | 1 | | 3 | 1 | |
| SHA1MSG1 | x,x | 1 | | 3 | 1 | |
| SHA1MSG2 | x,x | 1 | | 3 | 1 | |
| SHA256RND4S | x,x | 3 | | 8 | 4 | |
| SHA256MSG1 | x,x | 1 | | 3 | 1 | |
| SHA256MSG2 | x,x | 1 | | 3 | 1 | |
| Other | | | | | | |
| EMMS | | 13 | | | 23 | |

Floating point XMM instructions

| | Operands | μops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|----------|------|------|---------|-----------------------|-----------|
| Move instructions | | | | | | |
| MOVAPS/D | x, x | 1 | | 0-1 | 0.5 | |
| MOVAPS/D | x,m128 | 1 | | 3 | 1 | |
| MOVAPS/D | m128,x | 1 | | 3 | 1 | |
| MOVUPS/D | x,m128 | 1 | | 3 | 1 | |
| MOVUPS/D | m128,x | 1 | | 3 | 1 | |
| MOVSS/D | x, x | 1 | | 1 | 0.5 | |
| MOVSS/D | x,m32/64 | 1 | | 3 | 1 | |
| MOVSS/D | m32/64,x | 1 | | 3 | 1 | |
| MOVHPS/D MOVLPS/D | x,m64 | 1 | | 4 | 1 | |
| MOVHPS/D MOVLPS/D | m64,x | 1 | | 3 | 1 | |
| MOVLHPS MOVHLPS | x,x | 1 | | 1 | 0.5 | |
| BLENDPS/PD | x,x/m,i | 1 | | 1 | 0.5 | |
| BLENDVPS/PD | x,x,xmm0 | 2 | | 4 | 4 | +1 if mem |
| INSERTPS | x,x,i | 1 | | 1 | 0.5 | |
| INSERTPS | x,m32,i | 3 | | 4 | 4 | |

Goldmont

| | | | | | | |
|-------------------|---------|----|--|----|-----|-----------|
| EXTRACTPS | r32,x,i | 1 | | 4 | 1 | |
| EXTRACTPS | m32,x,i | 1 | | 4 | 2 | |
| MOVMSKPS/D | r32,x | 1 | | 4 | 1 | |
| MOVNTPS/D | m128,x | 1 | | 3 | 1 | |
| SHUFPS | x,x,i | 1 | | 1 | 0.5 | |
| SHUFPD | x,x,i | 1 | | 1 | 0.5 | |
| MOVDDUP | x, x | 1 | | 1 | 0.5 | |
| MOVSH/LDUP | x, x | 1 | | 1 | 0.5 | |
| UNPCKH/LPS/PD | x, x | 1 | | 1 | 0.5 | |
| Conversion | | | | | | |
| CVTPD2PS | x, x | 1 | | 4 | 1 | |
| CVTPS2PD | x, x | 1 | | 4 | 1 | |
| CVTSD2SS | x, x | 1 | | 4 | 2 | |
| CVTSS2SD | x, x | 1 | | 4 | 2 | |
| CVTDQ2PS | x, x | 1 | | 4 | 1 | |
| CVT(T) PS2DQ | x, x | 1 | | 4 | 1 | |
| CVTDQ2PD | x, x | 1 | | 4 | 1 | |
| CVT(T)PD2DQ | x, x | 1 | | 4 | 1 | |
| CVTPI2PS | x,mm | 1 | | 4 | 2 | |
| CVT(T)PS2PI | mm,x | 1 | | 4 | 1 | |
| CVTPI2PD | x,mm | 1 | | 4 | 1 | |
| CVT(T) PD2PI | mm,x | 1 | | 4 | 1 | |
| CVTSI2SS | x,r32 | 1 | | 6 | 2 | |
| CVT(T)SS2SI | r32,x | 1 | | 5 | 1 | |
| CVTSI2SD | xm,r32 | 1 | | 6 | 2 | |
| CVT(T)SD2SI | r32,x | 1 | | 5 | 1 | |
| Arithmetic | | | | | | |
| ADDSS/SD/PS/PD | x, x | 1 | | 3 | 1 | |
| SUBSS/SD/PS/PD | x, x | 1 | | 3 | 1 | |
| ADDSUBPS/PD | x, x | 1 | | 3 | 1 | |
| HADDPS HSUBPS | x, x | 4 | | 6 | 6 | +1 if mem |
| HADDPD HSUBPD | x, x | 4 | | 5 | 5 | +1 if mem |
| MULSS/SD/PS/PD | x, x | 1 | | 4 | 1 | |
| DIVSS | x, x | 1 | | 19 | 18 | |
| DIVSD | x, x | 1 | | 34 | 33 | |
| DIVPS | x, x | 1 | | 36 | 35 | |
| DIVPD | x, x | 1 | | 66 | 65 | |
| RCPSS | x, x | 1 | | 4 | 1 | |
| RCPPS | x, x | 5 | | 9 | 6 | +1 if mem |
| CMPccSS/SD/PS/PD | x, x | 1 | | 3 | 1 | |
| (U)COMISS/SD | x, x | 1 | | 4 | 1 | |
| MAXSS/SD/PS/PD | x, x | 1 | | 3 | 1 | |
| MINSS/SD/PS/PD | x, x | 1 | | 3 | 1 | |
| ROUNDSS/SD/PS/PD | x,x,i | 1 | | 4 | 1 | |
| DPPS | x,x,i | 9 | | 14 | 10 | +1 if mem |
| DPPD | x,x,i | 10 | | 8 | 8 | +1 if mem |
| Math | | | | | | |
| SQRTSS | x, x | 1 | | 20 | 19 | |
| SQRTPS | x, x | 1 | | 38 | 37 | |
| SQRTSD | x, x | 1 | | 35 | 34 | |

Goldmont

| | | | | | | |
|--------------|-------|-----|-------|-----|-----|--|
| SQRTPD | x, x | 1 | | 68 | 67 | |
| RSQRTSS | x, x | 1 | | 4 | 1 | |
| RSQRTPS | x, x | 5 | | 9 | 6 | |
| Logic | | | | | | |
| ANDPS/D | x, x | 1 | FP0/1 | 1 | 0.5 | |
| ANDNPS/D | x, x | 1 | | 1 | 0.5 | |
| ORPS/D | x, x | 1 | | 1 | 0.5 | |
| XORPS/D | x, x | 1 | | 1 | 0.5 | |
| Other | | | | | | |
| LDMXCSR | m32 | 6 | | 18 | 18 | 32 bit mode 64 bit mode 32 bit mode 64 bit mode |
| STMXCSR | m32 | 3 | | 12 | 12 | |
| FXSAVE | m4096 | 202 | | 130 | 130 | |
| FXSAVE | m4096 | 123 | | 143 | 143 | |
| FXRSTOR | m4096 | 160 | | 218 | 218 | |
| FXRSTOR | m4096 | 128 | | 155 | 155 | |
| XSAVEC | m | 241 | | 160 | 160 | |
| XSAVEOPT | m | 227 | | 144 | 144 | |

Intel Goldmont Plus

List of instruction timings and μ op breakdown

Explanation of column headings:

| | |
|-------------------------------|---|
| Instruction: | Name of instruction. Multiple names mean that these instructions have the same data. |
| Operands: | i = immediate data, r = register, mm = 64 bit mmx register, x = 128 bit xmm register, v = mmx or xmm vector register, m = memory, m32 = 32-bit memory operand, etc. |
| μops: | The number of μ ops from the decoder or ROM. A μ op that goes to multiple units is counted as one. |
| Unit: | Tells which execution unit is used. Instructions that use the same unit cannot execute simultaneously. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay. |
| Reciprocal throughput: | The average number of clock cycles per instruction for a series of independent instructions of the same kind in the same thread. Delays in the decoders are included in the latency and throughput timings. Values of 4 or more are often caused by bottlenecks in the decoders and microcode ROM rather than the execution units. |

Integer instructions

| | Operands | μ ops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|----------|-----------|------|---------|-----------------------|-----------------|
| Move instructions | | | | | | |
| MOV | r,r | 1 | | 1 | 0.33 | |
| MOV | r,i | 1 | | 1 | 0.33 | |
| MOV | r,m | 1 | | 3 | 1 | All addr. modes |
| MOV | m,r | 1 | | 2 | 1 | All addr. modes |
| MOV | m,i | 1 | | | 1 | |
| MOVNTI | m,r | 2 | | 2 | 2 | |
| MOVSX MOVZX | r16,r8 | 2 | | | 4 | |
| MOVSX MOVZX | r16,m8 | 3 | | | 4 | |
| MOVSX MOVZX MOVSX | r32/64,r | 1 | | 1 | 1 | |
| MOVSX MOVXSD | r32/64,m | 1 | | 3 | 1 | |
| MOVZX | r32/64,m | 1 | | 2 | 1 | |
| CMOVcc | r,r | 1 | | 2 | 1 | |
| CMOVcc | r,m | 1 | | | 1 | |
| XCHG | r,r | 3 | | 4 | 4 | |
| XCHG | r,m | 3 | | 13 | 13 | Implicit lock |
| XLAT | | 4 | | 5 | 5 | |
| PUSH | r | 1 | | | 1 | |
| PUSH | i | 1 | | | 1 | |
| PUSH | m | 2 | | | 1 | |
| PUSH | sp | 2 | | | 1 | |
| PUSHF(D/Q) | | 11 | | | 16 | |
| PUSHA(D) | | 10 | | | 8 | Not in x64 mode |
| POP | r | 1 | | | 1 | |
| POP | m | 7 | | | 6 | |

Goldmont Plus

| | | | | | | |
|--------------------------------|-------------|----|--|----|------|-------------------|
| POP | sp | 1 | | | 2 | |
| POPF(D/Q) | | 23 | | | 32 | |
| POPA(D) | | 17 | | | 10 | Not in x64 mode |
| LAHF | | 1 | | 1 | 1 | |
| SAHF | | 1 | | 2 | 1 | |
| SALC | | 2 | | 4 | 4 | Not in x64 mode |
| LEA | r16,[m] | 2 | | 5 | 4 | |
| LEA | r,[r+d] | 1 | | 1 | 1 | |
| LEA | r,[r+r*1] | 1 | | 1 | 1 | |
| LEA | r32,[r+r*s] | 1 | | 2 | 2 | |
| LEA | r64,[r+r*s] | 1 | | 1 | 1 | |
| LEA | r,[r+r*s+d] | 1 | | 2 | 2 | |
| LEA | r,[rip+d] | 1 | | | 0.5 | |
| BSWAP | r | 1 | | 1 | 1 | |
| MOVBE | r,m | 1 | | | 1 | |
| MOVBE | m,r | 1 | | | 1 | |
| PREFETCHNTA | m | 1 | | | 1 | |
| PREFETCHT0/1/2 | m | 1 | | | 1 | |
| PREFETCHNTW | m | 1 | | | 1 | |
| LFENCE | | 2 | | | 6 | |
| MFENCE | | 2 | | | 18 | |
| SFENCE | | 1 | | | 2 | |
| CLFLUSH | m | 1 | | | 165 | |
| CLFLUSHOPT | m | 4 | | | 165 | |
| Arithmetic instructions | | | | | | |
| ADD SUB | r,r/i | 1 | | 1 | 0.33 | |
| ADD SUB | r,m | 1 | | | 1 | |
| ADD SUB | m,r/i | 1 | | 5 | 1 | |
| ADC SBB | r,r/i | 1 | | 2 | 2 | |
| ADC SBB | r,m | 1 | | | 2 | |
| ADC SBB | m,r/i | 1 | | 6 | 2 | |
| CMP | r,r/i | 1 | | 1 | 0.33 | |
| CMP | m,r/i | 1 | | | 1 | |
| INC DEC | r | 1 | | 1 | 1 | latency to flag=2 |
| NEG NOT | r | 1 | | 1 | 0.33 | |
| INC DEC NEG NOT | m | 1 | | 5 | 1 | |
| AAA | | 14 | | 10 | | Not in x64 mode |
| AAS | | 14 | | 10 | | Not in x64 mode |
| DAA | | 21 | | 13 | | Not in x64 mode |
| DAS | | 22 | | 14 | | Not in x64 mode |
| AAD | | 4 | | 5 | | Not in x64 mode |
| AAM | | 5 | | 14 | 11 | Not in x64 mode |
| MUL IMUL | r8 | 2 | | 4 | 4 | |
| MUL IMUL | r16 | 4 | | 5 | 5 | |
| MUL IMUL | r32 | 2 | | 4 | 2 | |
| MUL IMUL | r64 | 2 | | 6 | 2 | |
| IMUL | r16,r16 | 2 | | 4 | 4 | |
| IMUL | r32,r32 | 1 | | 3 | 1 | |
| IMUL | r64,r64 | 1 | | 5 | 2 | |
| IMUL | r16,r16,i | 2 | | 4 | 4 | |
| IMUL | r32,r32,i | 1 | | 3 | 1 | |
| IMUL | r64,r64,i | 1 | | 5 | 2 | |

Goldmont Plus

| | | | | | |
|---------------------------|------------|----|--|-------|-------|
| MUL IMUL | m8 | 3 | | | |
| MUL IMUL | m16 | 5 | | | |
| MUL IMUL | m32 | 2 | | | 1 |
| MUL IMUL | m64 | 2 | | | 2 |
| DIV | r8 | 3 | | 13-14 | 11-12 |
| DIV | r16 | 6 | | 14-19 | 13-18 |
| DIV | r32 | 6 | | 14-27 | 13-26 |
| DIV | r64 | 6 | | 14-43 | 13-42 |
| IDIV | r8 | 3 | | 13-14 | 11-12 |
| IDIV | r16 | 6 | | 14-19 | 13-18 |
| IDIV | r32 | 6 | | 14-27 | 13-26 |
| IDIV | r64 | 6 | | 14-43 | 13-42 |
| CBW | | 2 | | 4 | |
| CWDE | | 1 | | 1 | |
| CDQE | | 1 | | 1 | |
| CWD | | 2 | | 4 | |
| CDQ | | 1 | | 1 | |
| CQO | | 1 | | 1 | |
| POPCNT | r16,r16 | 2 | | 4 | 4 |
| POPCNT | r32,r32 | 1 | | 3 | 1 |
| POPCNT | r64,r64 | 1 | | 3 | 1 |
| CRC32 | r32,r8 | 1 | | 3 | 1 |
| CRC32 | r32,r16 | 1 | | 3 | 2 |
| CRC32 | r32,r32 | 1 | | 3 | 1 |
| Logic instructions | | | | | |
| AND OR XOR | r,r/i | 1 | | 1 | 0.33 |
| AND OR XOR | r,m | 1 | | | 1 |
| AND OR XOR | m,r/i | 1 | | 5 | 1 |
| TEST | r,r/i | 1 | | 1 | 0.33 |
| TEST | m,r/i | 1 | | | 1 |
| SHR SHL SAR | r,i/cl | 1 | | 1 | 1 |
| SHR SHL SAR | m,i/cl | 1 | | | 1 |
| ROR ROL | r,i/cl | 1 | | 1 | 1 |
| ROR ROL | m,i/cl | 1 | | | 1 |
| RCL | r,1 | 1 | | 2 | 2 |
| RCL | r,i | 16 | | 14 | 14 |
| RCL | r,cl | 16 | | 16 | 16 |
| RCR | r,1 | 10 | | 11 | 11 |
| RCR | r,i | 14 | | 12 | 12 |
| RCR | r,cl | 14 | | 14 | 14 |
| SHLD | r16,r16,i | 10 | | 15 | 15 |
| SHLD | r32,r32,i | 2 | | 2 | 2 |
| SHLD | r64,r64,i | 12 | | 11 | 11 |
| SHLD | r16,r16,cl | 10 | | 16 | 16 |
| SHLD | r32,r32,cl | 2 | | 4 | 4 |
| SHLD | r64,r64,cl | 12 | | 13 | 13 |
| SHRD | r16,r16,i | 10 | | 10 | 10 |
| SHRD | r32,r32,i | 2 | | 2 | 2 |
| SHRD | r64,r64,i | 12 | | 11 | 11 |
| SHRD | r16,r16,cl | 10 | | 12 | 12 |
| SHRD | r32,r32,cl | 2 | | 4 | 4 |
| SHRD | r64,r64,cl | 12 | | 13 | 13 |

Goldmont Plus

| | | | | | |
|--------------------------------------|------------|--------|--------|-------|---------------------|
| BT | r,r/i | 1 | 1 | 1 | |
| BT | m,r | 8 | | 9 | |
| BT | m,i | 1 | | 1 | |
| BTR BTS BTC | r,r/i | 1 | 1 | 1 | |
| BTR BTS BTC | m,r | 9 | 7 | 7 | |
| BTR BTS BTC | m,i | 1 | | 2 | |
| BSF BSR | r,r/m | 11 | 9 | 8 | |
| SET _{cc} | r/m | 1 | 2 | 1 | |
| CLC STC | | 1 | | 1 | |
| CMC | | 1 | 1 | 1 | |
| CLD | | 3 | | 7 | |
| STD | | 4 | | 35 | |
| Control transfer instructions | | | | | |
| JMP | short/near | 1 | | 1-6 | when predicted |
| JMP | r | 1 | | 1 | |
| JMP | m(near) | 1 | | 1 | |
| Conditional jump | short/near | 1 | | 1-6 | when predicted |
| J(E/R)CXZ | short | 2 | | 4-18 | |
| LOOP | short | 9 | | 10-23 | |
| LOOP(N)E | short | 9 | | 12-24 | |
| CALL | near | 1 | | 2 | |
| CALL | r | 1 | | 8 | |
| CALL | m | 3 | | 11 | |
| RET | | 1 | | 2-3 | |
| RET | i | 2 | | 2 | |
| BOUND | r,m | 9 | | 9 | Not in x64 mode |
| INTO | | 4 | | 7 | Not in x64 mode |
| String instructions | | | | | |
| LODS | | 3 | 4 | | |
| REP LODS | | ~5n | ~2n | | |
| STOS | | 2 | 4 | | |
| REP STOS | | ~0.13B | ~0.07B | | per byte, best case |
| MOVS | | 5 | 5 | | |
| REP MOVS | | ~ 0.2B | ~0.07B | | per byte, best case |
| SCAS | | 3 | 4 | | |
| REP SCAS | | ~6n | ~2n | | |
| CMPS | | 6 | 4 | | |
| REP CMPS | | ~8n | ~3n | | |
| Synchronization instructions | | | | | |
| XADD | m,r | 6 | 5 | | |
| LOCK XADD | m,r | 6 | 13 | | |
| LOCK ADD | m,r | 1 | 13 | | |
| CMPXCHG | m,r | 8 | 6 | | |
| LOCK CMPXCHG | m,r | 8 | 14 | | |
| CMPXCHG8B | m,r | 14 | 9 | | |
| LOCK CMPXCHG8B | m,r | 14 | 15 | | |
| CMPXCHG16B | m,r | 19 | 24 | | |
| LOCK CMPXCHG16B | m,r | 19 | 31 | | |

Goldmont Plus

| | | | | | | |
|------------------|-----|--------|--|--------|--------|--|
| Other | | | | | | |
| NOP (90) | | 1 | | | 0.33 | |
| Long NOP (0F 1F) | | 1 | | | 0.33 | |
| PAUSE | | 3 | | 150 | | |
| ENTER | a,0 | 14 | | 10 | | |
| ENTER | a,b | 17+6b | | ~75+3b | | |
| LEAVE | | 2 | | | 3 | |
| CPUID | | 63-104 | | 80-200 | 80-200 | |
| RDTSC | | 21 | | | 21 | |
| RDTSCP | | 24 | | | 33 | |
| RDPMC | | 13 | | | 9 | |
| RDRAND | r | 16 | | | ~2300 | |
| RDSEED | r | 16 | | | ~2300 | |

Floating point x87 instructions

| | Operands | μops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------------|----------|------|------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| FLD | r | 1 | | 1 | 0.5 | |
| FLD | m32/m64 | 1 | | 3 | 1 | |
| FLD | m80 | 1 | | 3 | 1 | |
| FBLD | m80 | 54 | | 56 | 50 | |
| FST(P) | r | 1 | | 1 | 0.5 | |
| FST(P) | m32/m64 | 1 | | 3 | 2 | |
| FSTP | m80 | 1 | | 3 | 2 | |
| FBSTP | m80 | 195 | | 190 | 190 | |
| FXCH | r | 2 | | 1 | 1 | |
| FILD | m | 1 | | 7 | 1 | |
| FIST(P) | m | 8 | | 6 | 10 | |
| FISTTP | m | 8 | | 6 | 13 | |
| FLDZ | | 1 | | | 1 | |
| FLD1 | | 2 | | | 6 | |
| FLDPI FLDL2E etc. | | 2 | | | 3 | |
| FCMOVcc | r | 4 | | 5 | 5 | |
| FNSTSW | AX | 2 | | | 10 | |
| FNSTSW | m16 | 3 | | | 11 | |
| FLDCW | m16 | 4 | | | 15 | |
| FNSTCW | m16 | 3 | | | 4 | |
| FINCSTP FDECSTP | | 1 | | 1 | 0.5 | |
| FFREE(P) | | 1 | | | 0.5 | |
| FNSAVE | m | 146 | | 183 | 183 | |
| FRSTOR | m | 78 | | 162 | 162 | |
| Arithmetic instructions | | | | | | |
| FADD(P) FSUB(R)(P) | r | 1 | | 3 | 1 | |
| FADD(P) FSUB(R)(P) | m | 2 | | | 2 | |
| FMUL(P) | r | 1 | | 5 | 2 | |
| FMUL(P) | m | 2 | | | 2 | |
| FDIV(R)(P) | r | 1 | | 15 | 11 | |
| FDIV(R)(P) | m | 2 | | | 11 | |
| FABS | | 1 | | 1 | 1 | |

Goldmont Plus

| | | | | | | |
|--------------------|-----|--------|--|--------|--------|--|
| FCHS | | 1 | | 1 | 1 | |
| FCOM(P) FUCOM | r/m | 1 | | | 1 | |
| FCOMPP FUCOMPP | | 1 | | | 1 | |
| FCOMI(P) FUCOMI(P) | r | 1 | | | 1 | |
| FIADD FISUB(R) | m | 3 | | | 5 | |
| FIMUL | m | 3 | | | 5 | |
| FIDIV(R) | m | 4 | | | 11 | |
| FICOM(P) | m | 3 | | | 4 | |
| FTST | | 1 | | | 1 | |
| FXAM | | 1 | | | 1 | |
| FPREM | | 29 | | 35 | 35 | |
| FPREM1 | | 29 | | 35 | 35 | |
| FRNDINT | | 19 | | 41 | 41 | |
| Math | | | | | | |
| FSCALE | | 30 | | 32 | 32 | |
| FXTRACT | | 16 | | 22 | 22 | |
| FSQRT | | 1 | | 12-20 | 6-14 | |
| FSIN FCOS | | 17-100 | | 45-150 | 45-150 | |
| FSINCOS | | 16-110 | | 48-135 | 48-135 | |
| F2XM1 | | 9-27 | | 40-90 | 40-90 | |
| FYL2X | | 34-61 | | 88-130 | 88-130 | |
| FYL2XP1 | | 61 | | 140 | 140 | |
| FPTAN | | 16-100 | | 45-180 | 45-180 | |
| FPATAN | | 33-65 | | 85-190 | 85-190 | |
| Other | | | | | | |
| FNOP | | 1 | | | 0.5 | |
| WAIT | | 2 | | | 6 | |
| FNCLEX | | 4 | | | 21 | |
| FNINIT | | 12 | | | 50 | |

Integer MMX and XMM instructions

| | Operands | µops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|----------|------|------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| MOVD MOVQ | r32/64,v | 1 | | 4 | 1 | |
| MOVD MOVQ | m,v | 1 | | 3 | 1 | |
| MOVD MOVQ | v,r32/64 | 1 | | 5 | 1 | |
| MOVD MOVQ | v,m | 1 | | 3 | 1 | |
| MOVQ | v,v | 1 | | 1 | 0.5 | |
| MOVDQA | x, x | 1 | | 0-1 | 0.5 | |
| MOVDQA MOVDQU | x, m128 | 1 | | 3 | 1 | |
| MOVDQA MOVDQU | m128, x | 1 | | 3 | 1 | |
| LDDQU | x, m128 | 1 | | 3 | 1 | |
| MOVDQ2Q | mm, x | 1 | | 1 | 0.5 | |
| MOVQ2DQ | x,mm | 1 | | 1 | 0.5 | |
| MOVNTQ | m64,mm | 1 | | 3 | 1 | |
| MOVNTDQ | m128,x | 1 | | 3 | 1 | |
| MOVNTDQA | x, m128 | 1 | | 3 | 1 | |

Goldmont Plus

| | | | | | | |
|--------------------------------|----------|---|--|------|-----|-----------|
| PACKSSWB/DW | | | | | | |
| PACKUSWB | v,v | 1 | | 1 | 0.5 | |
| PACKUSDW | x,x | 1 | | 1 | 0.5 | |
| PUNPCKH/LBW/WD/DQ | v,v | 1 | | 1 | 0.5 | |
| PUNPCKH/LQDQ | v,v | 1 | | 1 | 0.5 | |
| PMOVSX/ZX BW BD BQ DW DQ | x,x | 1 | | 1 | 0.5 | |
| PMOVSX/ZX BW BD BQ DW DQ | x,m | 1 | | 1 | 1 | |
| PSHUFB | mm,mm | 1 | | 1 | 0.5 | |
| PSHUFB | x,x | 1 | | 1 | 1 | |
| PSHUFW | mm,mm,i | 1 | | 1 | 0.5 | |
| PSHUFL/HW | x,x,i | 1 | | 1 | 0.5 | |
| PSHUFD | x,x,i | 1 | | 1 | 0.5 | |
| PALIGNR | x,x,i | 1 | | 1 | 0.5 | |
| PBLENDVB | x,x,xmm0 | 2 | | 4 | 4 | |
| PBLENDVB | x,m,xmm0 | 3 | | | 4 | |
| PBLENDW | x,x/m,i | 1 | | 1 | 0.5 | |
| MASKMOVQ | mm,mm | 1 | | ~300 | 1 | |
| MASKMOVDQU | x,x | 3 | | ~330 | 4 | |
| PMOVMSKB | r32,v | 1 | | 4 | 1 | |
| PINSRW | v,r32,i | 1 | | 4 | 1 | |
| PINSRB/D/Q | x,r32,i | 1 | | 4 | 1 | |
| PINSRB/D/Q | x,m8,i | 1 | | | 1 | |
| PEXTRB/W/D/Q | r,v,i | 1 | | 4 | 1 | |
| PEXTRB/W/D/Q | m,v,i | 4 | | | 4 | |
| Arithmetic instructions | | | | | | |
| PADD/SUB(U)(S)B/W/D | v,v | 1 | | 1 | 0.5 | |
| PADDQ PSUBQ | v,v | 1 | | 2 | 1 | |
| PHADD(S)W PHSUB(S)W | mm, mm | 5 | | 7 | 7 | |
| PHADD(S)W PHSUB(S)W | x, x/m | 7 | | 6 | 6 | +1 if mem |
| PHADDD PHSUBD | v,v | 3 | | 4 | 4 | +1 if mem |
| PCMPEQ/GTB/W/D | v,v | 1 | | 1 | 0.5 | |
| PCMPEQQ | x, x | 1 | | 2 | 1 | |
| PCMPGTQ | x, x | 1 | | 2 | 1 | |
| PMULL/HW PMULHUW | v,v | 1 | | 4 | 1 | |
| PMULHRW | v,v | 1 | | 4 | 1 | |
| PMULLD | x, x | 1 | | 5 | 2 | |
| PMULDQ | x, x | 1 | | 4 | 1 | |
| PMULUDQ | v,v | 1 | | 4 | 1 | |
| PMADDWD | v,v | 1 | | 4 | 1 | |
| PMADDUBSW | v,v | 1 | | 4 | 1 | |
| PSADBW | v,v | 1 | | 4 | 1 | |
| MPSADBW | x,x,i | 3 | | 5 | 4 | +1 if mem |
| PAVGB/W | v,v | 1 | | 1 | 0.5 | |
| PMIN/MAXUB | v,v | 1 | | 1 | 0.5 | |
| PMIN/PMAX SB/SW/SD | | | | | | |
| UB/UW/UD | x,x | 1 | | 1 | 0.5 | |
| PHMINPOSUW | x,x | 1 | | 5 | 2 | |
| PABSB PABSW PABSD | v,v | 1 | | 1 | 0.5 | |
| PSIGNB PSIGNW PSIGND | v,v | 1 | | 1 | 0.5 | |

Goldmont Plus

| | | | | | | |
|--------------------------------|-------|----|--|----|-----|-----------|
| Logic instructions | | | | | | |
| PAND(N) POR PXOR | v,v | 1 | | 1 | 0.5 | |
| PTEST | x,x | 1 | | 1 | 1 | |
| PSLL/RL/RAW/D/Q | v,v | 1 | | 2 | 1 | |
| PSLL/RL/RAW/D/Q | v,i | 1 | | 1 | 0.5 | |
| PSLL/RLDQ | x,i | 1 | | 1 | 0.5 | |
| String instructions | | | | | | |
| PCMPESTRI | x,x,i | 11 | | 26 | 10 | +1 if mem |
| PCMPESTRM | x,x,i | 10 | | 17 | 12 | +1 if mem |
| PCMPISTRI | x,x,i | 6 | | 17 | 8 | +1 if mem |
| PCMPISTRM | x,x,i | 5 | | 12 | 7 | +1 if mem |
| Encryption instructions | | | | | | |
| PCLMULQDQ | x,x,i | 3 | | 6 | 4 | |
| AESDEC | x,x | 1 | | 4 | 1 | |
| AESDECLAST | x,x | 1 | | 4 | 1 | |
| AESENC | x,x | 1 | | 4 | 1 | |
| AESENCLAST | x,x | 1 | | 4 | 1 | |
| AESIMC | x,x | 1 | | 4 | 1 | |
| AESKEYGENASSIST | x,x,i | 1 | | 4 | 1 | |
| SHA1RND4S | x,x,i | 1 | | 5 | 2 | |
| SHA1NEXTE | x,x | 1 | | 3 | 1 | |
| SHA1MSG1 | x,x | 1 | | 3 | 1 | |
| SHA1MSG2 | x,x | 1 | | 3 | 1 | |
| SHA256RND4S | x,x | 3 | | 8 | 4 | |
| SHA256MSG1 | x,x | 1 | | 3 | 1 | |
| SHA256MSG2 | x,x | 1 | | 3 | 1 | |
| Other | | | | | | |
| EMMS | | 19 | | | 38 | |

Floating point XMM instructions

| | Operands | μops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|----------|------|------|---------|-----------------------|-----------|
| Move instructions | | | | | | |
| MOVAPS/D | x, x | 1 | | 0-1 | 0.5 | |
| MOVAPS/D | x,m128 | 1 | | 3 | 1 | |
| MOVAPS/D | m128,x | 1 | | 3 | 1 | |
| MOVUPS/D | x,m128 | 1 | | 3 | 1 | |
| MOVUPS/D | m128,x | 1 | | 3 | 1 | |
| MOVSS/D | x, x | 1 | | 1 | 0.5 | |
| MOVSS/D | x,m32/64 | 1 | | 3 | 1 | |
| MOVSS/D | m32/64,x | 1 | | 3 | 1 | |
| MOVHPS/D MOVLPS/D | x,m64 | 1 | | 4 | 1 | |
| MOVHPS/D MOVLPS/D | m64,x | 1 | | 3 | 1 | |
| MOVLHPS MOVHLPS | x,x | 1 | | 1 | 0.5 | |
| BLENDPS/PD | x,x/m,i | 1 | | 1 | 0.5 | |
| BLENDVPS/PD | x,x,xmm0 | 2 | | 4 | 4 | +1 if mem |
| INSERTPS | x,x,i | 1 | | 1 | 0.5 | |
| INSERTPS | x,m32,i | 3 | | 4 | 4 | |

Goldmont Plus

| | | | | | | |
|-------------------|---------|---|--|----|-----|-----------|
| EXTRACTPS | r32,x,i | 1 | | 4 | 1 | |
| EXTRACTPS | m32,x,i | 1 | | 4 | 2 | |
| MOVMSKPS/D | r32,x | 1 | | 4 | 1 | |
| MOVNTPS/D | m128,x | 1 | | 3 | 1 | |
| SHUFPS | x,x,i | 1 | | 1 | 0.5 | |
| SHUFPD | x,x,i | 1 | | 1 | 0.5 | |
| MOVDDUP | x, x | 1 | | 1 | 0.5 | |
| MOVSH/LDUP | x, x | 1 | | 1 | 0.5 | |
| UNPCKH/LPS/PD | x, x | 1 | | 1 | 0.5 | |
| Conversion | | | | | | |
| CVTPD2PS | x, x | 1 | | 4 | 1 | |
| CVTPS2PD | x, x | 1 | | 4 | 1 | |
| CVTSD2SS | x, x | 1 | | 4 | 2 | |
| CVTSS2SD | x, x | 1 | | 4 | 2 | |
| CVTDQ2PS | x, x | 1 | | 4 | 1 | |
| CVT(T) PS2DQ | x, x | 1 | | 4 | 1 | |
| CVTDQ2PD | x, x | 1 | | 4 | 1 | |
| CVT(T)PD2DQ | x, x | 1 | | 4 | 1 | |
| CVTPI2PS | x,mm | 1 | | 4 | 2 | |
| CVT(T)PS2PI | mm,x | 1 | | 4 | 1 | |
| CVTPI2PD | x,mm | 1 | | 4 | 1 | |
| CVT(T) PD2PI | mm,x | 1 | | 4 | 1 | |
| CVTSI2SS | x,r32 | 1 | | 6 | 2 | |
| CVT(T)SS2SI | r32,x | 1 | | 5 | 1 | |
| CVTSI2SD | xm,r32 | 1 | | 6 | 2 | |
| CVT(T)SD2SI | r32,x | 1 | | 5 | 1 | |
| Arithmetic | | | | | | |
| ADDSS/SD/PS/PD | x, x | 1 | | 3 | 1 | |
| SUBSS/SD/PS/PD | x, x | 1 | | 3 | 1 | |
| ADDSUBPS/PD | x, x | 1 | | 3 | 1 | |
| HADDPS HSUBPS | x, x | 4 | | 6 | 6 | +1 if mem |
| HADDPD HSUBPD | x, x | 4 | | 5 | 5 | +1 if mem |
| MULSS/SD/PS/PD | x, x | 1 | | 4 | 1 | |
| DIVSS | x, x | 1 | | 11 | 7 | |
| DIVSD | x, x | 1 | | 14 | 10 | |
| DIVPS | x, x | 1 | | 16 | 12 | |
| DIVPD | x, x | 1 | | 22 | 18 | |
| RCPSS | x, x | 1 | | 4 | 1 | |
| RCPPS | x, x | 1 | | 5 | 2 | |
| CMPccSS/SD/PS/PD | x, x | 1 | | 3 | 1 | |
| (U)COMISS/SD | x, x | 1 | | 4 | 1 | |
| MAXSS/SD/PS/PD | x, x | 1 | | 3 | 1 | |
| MINSS/SD/PS/PD | x, x | 1 | | 3 | 1 | |
| ROUNDSS/SD/PS/PD | x,x,i | 1 | | 4 | 1 | |
| DPPS | x,x,i | 9 | | 17 | 10 | +1 if mem |
| DPPD | x,x,i | 6 | | 5 | 5 | +1 if mem |
| Math | | | | | | |
| SQRTSS | x, x | 1 | | 12 | 8 | |
| SQRTPS | x, x | 1 | | 18 | 14 | |
| SQRTSD | x, x | 1 | | 18 | 14 | |

Goldmont Plus

| | | | | | | |
|--------------|-------|-----|--|----|-----|-------------|
| SQRTPD | x, x | 1 | | 30 | 26 | |
| RSQRTSS | x, x | 1 | | 4 | 1 | |
| RSQRTPS | x, x | 1 | | 5 | 2 | |
| Logic | | | | | | |
| ANDPS/D | x, x | 1 | | 1 | 0.5 | |
| ANDNPS/D | x, x | 1 | | 1 | 0.5 | |
| ORPS/D | x, x | 1 | | 1 | 0.5 | |
| XORPS/D | x, x | 1 | | 1 | 0.5 | |
| Other | | | | | | |
| LDMXCSR | m32 | 6 | | 18 | 18 | |
| STMXCSR | m32 | 3 | | 12 | 12 | |
| FXSAVE | m4096 | 130 | | | 106 | 32 bit mode |
| FXSAVE | m4096 | 156 | | | 116 | 64 bit mode |
| FXRSTOR | m4096 | 110 | | | 136 | 32 bit mode |
| FXRSTOR | m4096 | 136 | | | 146 | 64 bit mode |
| XSAVE | m | 193 | | | 119 | 32 bit mode |
| XSAVE | m | 452 | | | 315 | 64 bit mode |
| XRSTOR | m | 156 | | | 220 | 32 bit mode |
| XRSTOR | m | 173 | | | 237 | 64 bit mode |
| XSAVEC | m | 447 | | | 350 | 32 bit mode |
| XSAVEC | m | 449 | | | 370 | 64 bit mode |
| XSAVEOPT | m | 193 | | | 118 | 32 bit mode |
| XSAVEOPT | m | 219 | | | 128 | 64 bit mode |

Intel Knights Landing

List of instruction timings and μ op breakdown

Explanation of column headings:

| | |
|-------------------------------|--|
| Instruction: | Name of instruction. Multiple names mean that these instructions have the same data. Instructions with or without V name prefix behave the same unless otherwise noted. |
| Operands: | i = immediate data, r = register, mm = 64 bit mmx register, x = 128 bit xmm register, mm/x = mmx or xmm register, y = 256 bit ymm register, z = 512 bit zmm register, v = any vector register (mmx, xmm, ymm, zmm), k = mask register. same = same register for both operands. m = memory operand, m32 = 32-bit memory operand, etc. |
| μops: | The number of μ ops from the decoder or ROM. A μ op that goes to multiple units is counted as one. |
| Unit: | Tells which execution unit is used. Instructions that use the same unit cannot execute simultaneously. IP0 and IP1 means integer port 0 or 1 and their associated pipelines IP0/1 means that either integer unit can be used. IP0+1 means that the μ op is split in two, using both units. Mem means memory execution cluster FP0 means floating point port 0 (includes multiply, divide, convert and shuffle). FP1 means floating point port 1. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay. Some instructions have a range of latencies. For example VPSHUFD has a latency of 3-6. The short latency is measured in a chain of similar instructions. The long latency is measured when the input comes from an instruction of a different type and the output goes to an instruction of a different type, for example a move instruction. The long latency will apply in most cases. Division and some square root instructions have latencies that depend on the values of the operands. |
| Reciprocal throughput: | The average number of clock cycles per instruction for a series of independent instructions of the same kind in the same thread. Delays in the decoders are included in the latency and throughput timings. Values of 4 or more are often caused by bottlenecks in the decoders and microcode ROM rather than the execution units. |

Integer instructions

| | Operands | μ ops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|----------|-----------|-------|---------|-----------------------|-----------------|
| Move instructions | | | | | | |
| MOV | r,r | 1 | IP0/1 | 1 | 0.5 | All addr. modes |
| MOV | r,i | 1 | IP0/1 | 1 | 0.5 | |
| MOV | r,m | 1 | Mem | 4 | 1 | |
| MOV | m,r | 1 | Mem | 3 | 1 | |
| MOV | m,i | 1 | Mem | | 1 | |
| MOVNTI | m,r | 1 | Mem | | 2 | |
| MOVSX MOVZX MOVSXD | r16,r8 | 2 | IP0 | 7 | 7 | |
| MOVSX MOVZX MOVSXD | r16,m8 | 3 | IP0 | 7 | 8 | |

Knights Landing

| | | | | | | |
|--------------------------------|-------------|----|------------|-----|-----|-----------------|
| MOVSX MOVZX MOVSD | r32/64,r | 1 | IP0 | 1 | 1 | |
| MOVSX MOVZX MOVSD | r32/64,m | 1 | IP0 | 4 | 1 | |
| CMOVcc | r,r | 1 | IP0/1 | 2 | 1 | |
| CMOVcc | r,m | 1 | | | 1 | |
| XCHG | r,r | 3 | IP0/1 | 8 | 8 | |
| XCHG | r,m | 3 | | 24 | 24 | Implicit lock |
| XLAT | | 4 | | 8 | | |
| PUSH | r | 1 | | | 1 | |
| PUSH | i | 1 | IP0+1 | | 1 | |
| PUSH | m | 3 | IP0+1 | | 8 | |
| PUSHF(D/Q) | | 18 | IP0+1 | | 28 | |
| PUSHA(D) | | 10 | | | 10 | Not in x64 mode |
| POP | r | 2 | | | 1 | |
| POP | (E/R)SP | 2 | | | 4 | |
| POP | m | 6 | | | 9 | |
| POPF(D/Q) | | 21 | | | 48 | |
| POPA(D) | | 17 | | | 15 | Not in x64 mode |
| LAHF | | 1 | IP0 | 1 | 1 | |
| SAHF | | 1 | IP0 | 2 | 1 | |
| SALC | | 2 | | 9 | 7 | Not in x64 mode |
| LEA | r,[r+d] | 1 | IP0/1 | 1 | 0.5 | |
| LEA | r,[r+r*s] | 1 | IP1 | 1 | 1 | |
| LEA | r,[r+r*s+d] | 1 | IP0+1 | 2 | 1 | |
| LEA | r,[rip+d] | 1 | IP0/1 | | 0.5 | |
| LEA | r16,[m] | 2 | | 7 | | |
| BSWAP | r | 1 | IP0 | 1 | 1 | |
| MOVBE | r16,m16 | 1 | | | 1 | |
| MOVBE | r/m32/64 | 1 | | | 1 | |
| MOVBE | m,r | 1 | | | 1 | |
| PREFETCHNTA | m | 1 | | | 0.5 | |
| PREFETCHT0/1/2 | m | 1 | | | 0.5 | |
| PREFETCHNTW | m | 1 | | | 0.5 | |
| LFENCE | | 2 | | | 8 | |
| MFENCE | | 2 | | | 17 | |
| SFENCE | | 1 | | | 10 | |
| Arithmetic instructions | | | | | | |
| ADD SUB | r,r/i | 1 | IP0/1 | 1 | 0.5 | |
| ADD SUB | r,m | 1 | IP0/1, Mem | | 1 | |
| ADD SUB | m,r/i | 1 | IP0/1, Mem | 7 | 1 | |
| ADC SBB | r,r/i | 1 | IP0+1 | 2 | 2 | |
| ADC SBB | r,m | 1 | | | 2 | |
| ADC SBB | m,r/i | 1 | | 7 | 2 | |
| ADCX ADOX | r32,r32 | 1 | IP0+1 | 2 | 2 | |
| ADCX ADOX | r64,r64 | 1 | IP0+1 | 2 | 6 | due to decoder |
| CMP | r,r/i | 1 | IP0/1 | 1 | 0.5 | |
| CMP | m,r/i | 1 | | | 1 | |
| INC DEC | r | 1 | IP0/1 | 1-2 | 1 | |
| NEG NOT | r | 1 | IP0/1 | 1 | 0.5 | |
| INC DEC NEG NOT | m | 1 | | 7 | 1 | |
| AAA AAS | | 13 | | 13 | | Not in x64 mode |
| DAA | | 20 | | 17 | | Not in x64 mode |
| DAS | | 21 | | 17 | | Not in x64 mode |

Knights Landing

| | | | | | | |
|---------------------------|-----------|----|------------|--------|--------|-----------------|
| AAD | | 4 | | 8 | | Not in x64 mode |
| AAM | | 10 | | 30 | 14 | Not in x64 mode |
| MUL IMUL | r8 | 3 | IP0 | 8 | | |
| MUL IMUL | r16 | 4 | IP0 | 8 | | |
| MUL IMUL | r32 | 3 | IP0 | 8 | | |
| MUL IMUL | r64 | 3 | IP0 | 8 | | |
| IMUL | r16,r16 | 2 | IP0 | 7 | 7 | |
| IMUL | r32,r32 | 1 | IP0 | 3 | 1 | |
| IMUL | r64,r64 | 1 | IP0 | 5 | 2 | |
| IMUL | r16,r16,i | 2 | IP0 | 7 | 7 | |
| IMUL | r32,r32,i | 1 | IP0 | 3 | 1 | |
| IMUL | r64,r64,i | 1 | IP0 | 5 | 2 | |
| MUL IMUL | m8 | 3 | IP0 | | | |
| MUL IMUL | m16 | 3 | IP0 | | | |
| MUL IMUL | m32 | 4 | IP0 | | | |
| MUL IMUL | m64 | 3 | IP0 | | | |
| DIV | r/m8 | 9 | IP0, FP0 | 30 | 12 | |
| DIV | r/m16 | 12 | IP0, FP0 | 30-35 | 13-15 | |
| DIV | r/m32 | 12 | IP0, FP0 | 29-42 | 13-23 | |
| DIV | r/m 64 | 23 | IP0, FP0 | 39-95 | 22-95 | |
| IDIV | r/m8 | 26 | IP0, FP0 | 39 | 20 | |
| IDIV | r/m16 | 29 | IP0, FP0 | 38-42 | 22 | |
| IDIV | r/m32 | 29 | IP0, FP0 | 37-49 | 22-26 | |
| IDIV | r/m64 | 44 | IP0, FP0 | 53-108 | 36-107 | |
| CBW | | 2 | IP0 | 7 | | |
| CWDE | | 1 | IP0 | 1 | | |
| CDQE | | 1 | IP0 | 1 | | |
| CWD | | 2 | IP0 | 7 | | |
| CDQ | | 1 | IP0 | 1 | | |
| CQO | | 1 | IP0 | 1 | | |
| POPCNT | r16,r16 | 2 | | 7 | 7 | |
| POPCNT | r32,r32 | 1 | | 3 | 1 | |
| POPCNT | r64,r64 | 1 | | 3 | 1 | |
| CRC32 | r32,r8 | 2 | | 7 | 2 | |
| CRC32 | r32,r16 | 1 | | 6 | 6 | |
| CRC32 | r32,r32 | 1 | | 3 | 1 | |
| CRC32 | r64,r64 | 1 | | 6 | 1 | |
| Logic instructions | | | | | | |
| AND OR XOR | r,r/i | 1 | IP0/1 | 1 | 0.5 | |
| AND OR XOR | r,m | 1 | IP0/1, Mem | | 1 | |
| AND OR XOR | m,r/i | 1 | IP0/1, Mem | 6 | 1 | |
| TEST | r,r/i | 1 | IP0/1 | 1 | 0.5 | |
| TEST | m,r/i | 1 | IP0/1, Mem | | 1 | |
| SHR SHL SAR | r,i/cl | 1 | IP0 | 1 | 1 | |
| SHR SHL SAR | m,i/cl | 1 | IP0 | | 1 | |
| ROR ROL | r,i/cl | 1 | IP0 | 1 | 1 | |
| ROR ROL | m,i/cl | 1 | IP0 | | 1 | |
| RCR | r,1 | 7 | IP0 | 10 | 10 | |
| RCL | r,1 | 1 | IP0 | 2 | 2 | |
| RCR | r,i/cl | 11 | IP0 | 13 | 13 | |
| RCR | m,i/cl | 14 | IP0 | 13 | | |
| RCL | r,i/cl | 13 | IP0 | 13 | 13 | |

Knights Landing

| | | | | | |
|--------------------------------------|------------|----|-------|----|-----|
| RCL | m,i/cl | 16 | IP0 | 16 | 16 |
| SHLD | r16,r16,i | 10 | IP0 | 11 | 11 |
| SHLD | r16,m16,i | 13 | IP0 | 13 | 13 |
| SHLD | r32,r32,i | 1 | IP0 | 2 | 2 |
| SHLD | r32,m32,i | 6 | IP0 | 9 | 9 |
| SHLD | r64,r64,i | 10 | IP0 | 11 | 11 |
| SHLD | r64,m64,i | 13 | IP0 | 13 | 13 |
| SHLD | r16,r16,cl | 9 | IP0 | 11 | 11 |
| SHLD | r16,m16,cl | 12 | IP0 | 13 | 13 |
| SHLD | r32,r32,cl | 2 | IP0 | 7 | 7 |
| SHLD | r32,m32,cl | 6 | IP0 | 9 | 9 |
| SHLD | r64,r64,cl | 9 | IP0 | 11 | 11 |
| SHLD | r64,m64,cl | 12 | IP0 | 13 | 13 |
| SHRD | r16,r16,i | 8 | IP0 | 11 | 11 |
| SHRD | r16,m16,i | 11 | IP0 | 12 | 12 |
| SHRD | r32,r32,i | 2 | IP0 | 7 | 7 |
| SHRD | r32,m32,i | 6 | IP0 | 9 | 9 |
| SHRD | r64,r64,i | 10 | IP0 | 11 | 11 |
| SHRD | r64,m64,i | 13 | IP0 | 15 | 15 |
| SHRD | r16,r16,cl | 7 | IP0 | 11 | 11 |
| SHRD | r16,m16,cl | 10 | IP0 | 12 | 12 |
| SHRD | r32,r32,cl | 2 | IP0 | 7 | 7 |
| SHRD | r32,r32,cl | 6 | IP0 | 9 | 9 |
| SHRD | r64,r64,cl | 9 | IP0 | 11 | 11 |
| SHRD | r64,m64,cl | 12 | IP0 | 14 | 14 |
| SHLX SHRX SARX | r,r,r | 1 | IP0 | 2 | 1 |
| RORX | r,r,i | 1 | IP0 | 1 | 1 |
| BT | r,r/i | 1 | IP0+1 | 1 | 1 |
| BT | m,r | 7 | | 10 | 10 |
| BT | m,i | 1 | | 1 | 1 |
| BTR BTS BTC | r,r/i | 1 | IP0+1 | 1 | 1 |
| BTR BTS BTC | m,r | 8 | | 11 | 11 |
| BTR BTS BTC | m,i | 1 | IP0+1 | 1 | 1 |
| BSF BSR | r,r/m | 10 | IP0/1 | 11 | 11 |
| SETcc | r/m | 1 | IP0+1 | 2 | 1 |
| CLC STC | | 1 | IP0 | | 1 |
| CMC | | 1 | IP0 | 1 | 1 |
| CLD | | 4 | IP0/1 | | 8 |
| STD | | 5 | IP0/1 | | 36 |
| LZCNT | r,r/m | 1 | | 3 | 1 |
| TZCNT | r,r/m | 1 | | 3 | 1 |
| ANDN | r,r,r | 1 | | 1 | 0.5 |
| ANDN | r,r,m | 1 | | 1 | 1 |
| BLSI BLSMSK BLSR | r,r/m | 1 | | 1 | 1 |
| BEXTR | r,r,r | 2 | | 7 | 7 |
| BEXTR | r,r,m | 3 | | | 8 |
| BZHI | r,r,r | 1 | | 3 | 1 |
| PDEP | r,r,r | 1 | | 3 | 1 |
| PEXT | r,r,r | 1 | | 3 | 1 |
| Control transfer instructions | | | | | |
| JMP | short/near | 1 | IP1 | | 2 |
| JMP | r | 1 | IP1 | | 2 |

Knights Landing

| | | | | | | |
|-------------------------------------|------------|--------|-------|--------|---------|---------------------|
| JMP | m(near) | 1 | IP1 | | 2 | |
| Conditional jump | short/near | 1 | IP1 | | 1-2 | |
| J(E/R)CXZ | short | 2 | | | 7-18 | |
| LOOP | short | 7 | | | 14-23 | |
| LOOP(N)E | short | 8 | | | 14-23 | |
| CALL | near | 1 | | | 2 | |
| CALL | r | 1 | | | 2 | |
| CALL | m | 3 | | | 14 | |
| RET | | 1 | | | 2 | |
| RET | i | 1 | | | 2 | |
| BOUND | r,m | 10 | | | 11 | Not in x64 mode |
| INTO | | 4 | | | 8 | Not in x64 mode |
| String instructions | | | | | | |
| LODS | | 3 | | | 8 | |
| REP LODS | | ~4n | | | ~2n | |
| STOS | | 2 | | | 7 | |
| REP STOS | | ~0.07B | | | ~0.054B | per byte, best case |
| MOVS | | 5 | | | 9 | |
| REP MOVS | | ~ 0.1B | | | ~0.08B | per byte, best case |
| SCAS | | 3 | | | 8 | |
| REP SCAS | | ~5n | | | ~3n | |
| CMPS | | 6 | | | 9 | |
| REP CMPS | | ~6n | | | ~3n | |
| Synchronization instructions | | | | | | |
| XADD | m,r | 6 | | 9 | | |
| LOCK XADD | m,r | 4 | | 24 | | |
| LOCK ADD | m,r | 1 | | 13 | | |
| CMPXCHG | m,r | 8 | | 11 | | |
| LOCK CMPXCHG | m,r | 6 | | 26 | | |
| CMPXCHG8B | m,r | 13 | | 15 | | |
| LOCK CMPXCHG8B | m,r | 11 | | 29 | | |
| CMPXCHG16B | m,r | 19 | | 31 | | |
| LOCK CMPXCHG16B | m,r | 17 | | 48 | | |
| Other | | | | | | |
| NOP (90) | | 1 | IP0/1 | | 0.5 | |
| Long NOP (0F 1F) | | 1 | IP0/1 | | 0.5 | |
| PAUSE | | 6 | | 25 | | |
| ENTER | a,0 | 15 | | 14 | | |
| ENTER | a,b | 19+6b | | 66+4b | | |
| LEAVE | | 4 | | 8 | 5 | |
| XGETBV | | 7 | | | 14 | |
| CPUID | | 40-83 | | 63-270 | | |
| RDTSC | | 15 | | 30 | | |
| RDTSCP | | 17 | | 36 | | |
| RDPNC | | 19 | | 20 | | |
| RDRAND | r | 17 | | | 200 | |
| RDSEED | r | 17 | | | 200 | |

Floating point x87 instructions

Knights Landing

| | Operands | μops | Unit | Latency | Reciprocal throughput | Remarks |
|-------------------------|----------|------|-------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| FLD | r | 1 | FP0+1 | 2 | 1 | |
| FLD | m32/m64 | 1 | | 8 | 1 | |
| FLD | m80 | 5 | | 9 | 12 | |
| FBLD | m80 | 52 | | | 66 | |
| FST(P) | r | 1 | | 2 | 1 | |
| FST | m32/m64 | 5 | | 14 | 13 | |
| FSTP | m32/m64 | 6 | | 14 | 13 | |
| FSTP | m80 | 8 | | 11 | 14 | |
| FBSTP | m80 | 189 | | | 264 | |
| FXCH | r | 3 | | 9 | 9 | |
| FILD | m | 1 | | 5 | 2 | |
| FIST(P) | m | 6 | | 18 | 12 | |
| FISTTP | m | 6 | | | 14 | |
| FLDZ | | 1 | | | 1 | |
| FLD1 | | 2 | | | 10 | |
| FLDPI FLDL2E etc. | | 2 | | | 10 | |
| FCMOVcc | r | 3 | | 9 | | |
| FNSTSW | AX | 3 | | | 12 | |
| FNSTSW | m16 | 4 | | | 13 | |
| FLDCW | m16 | 3 | | | 15 | |
| FNSTCW | m16 | 5 | | | 15 | |
| FINCSTP FDECSTP | | 1 | | 1 | 1 | |
| FFREE(P) | | 1 | | | 1 | |
| Arithmetic instructions | | | | | | |
| FADD(P) FSUB(R)(P) | r | 1 | FP0 | 6 | 1.5 | |
| FADD(P) FSUB(R)(P) | m | 1 | FP0 | | 12 | |
| FMUL(P) | r | 1 | FP0 | 7 | 2 | |
| FMUL(P) | m | 1 | FP0 | 7 | 12 | |
| FDIV(R)(P) | r | 1 | FP0 | 41 | 37 | |
| FDIV(R)(P) | m | 1 | FP0 | 41 | 44 | |
| FABS | | 1 | | 2 | | |
| FCHS | | 1 | | 2 | | |
| FCOM(P) FUCOM | r | 1 | | | 1 | |
| FCOM(P) FUCOM | m | 1 | | | 2 | |
| FCOMPP FUCOMPP | | 1 | | | 1 | |
| FCOMI(P) FUCOMI(P) | r | 3 | | | 9 | |
| FIADD FISUB(R) | m | 3 | | | 17 | |
| FIMUL | m | 3 | | | 17 | |
| FIDIV(R) | m | 3 | | | 41 | |
| FICOM(P) | m | 3 | | | 8 | |
| FTST | | 1 | | | 1 | |
| FXAM | | 1 | | | 1 | |
| FPREM | | 27 | | 26-47 | 25-32 | |
| FPREM1 | | 27 | | 33-72 | 25-32 | |
| FRNDINT | | 18 | | 36 | 36 | |
| Math | | | | | | |
| FSCALE | | 30 | | 31 | | |
| FXTRACT | | 15 | | 19 | 18 | |

Knights Landing

| | | | |
|--------------|--------|---------|---------|
| FSQRT | 1 | 15-42 | 11-38 |
| FSIN FCOS | 16-100 | 40-250 | 40-250 |
| FSINCOS | 17-110 | 50-250 | 50-250 |
| F2XM1 | 9-24 | 100-400 | |
| FYL2X | 34-61 | 126-190 | 98-190 |
| FYL2XP1 | 61 | 190 | 190 |
| FPTAN | 17-100 | 50-280 | 50-280 |
| FPATAN | 33-63 | 125-265 | 125-265 |
| Other | | | |
| FNOP | 1 | | 1 |
| WAIT | 2 | | 7 |
| FNCLEX | 5 | | 26 |
| FNINIT | 15 | | 63 |

Integer MMX and XMM instructions

| | Operands | μops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|-------------|------|-------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| MOVD MOVQ | r32/64,mm/x | 1 | | 4 | 1 | |
| MOVD MOVQ | mm/x,r32/64 | 1 | | 5 | 1 | |
| MOVD MOVQ | m32/64,mm/x | 1 | Mem | 5 | 1 | |
| MOVD MOVQ | mm/x,m32/64 | 1 | Mem | 5 | 0.5 | |
| MOVQ | mm/x, mm/x | 1 | FP0/1 | 2 | 0.5 | |
| (V)MOVDQA/U | v,v | 1 | FP0/1 | 2 | 0.5 | |
| (V)MOVDQA/U | v,m | 1 | Mem | 5 | 0.5 | |
| VMOVDQA/U | v{k},m | 1 | Mem | 7 | 0.5 | |
| (V)MOVDQA/U | m,v | 1 | Mem | 5 | 1 | |
| VMOVDQA/U | m{k},v | 1 | Mem | 9 | 1 | |
| LDDQU | x, m128 | 1 | Mem | 5 | 0.5 | |
| MOVDQ2Q | mm, x | 1 | FP0/1 | 2 | 0.5 | |
| MOVQ2DQ | x,mm | 1 | FP0/1 | 2 | 0.5 | |
| MOVNTQ | m64,mm | 1 | Mem | ~650 | 1 | |
| MOVNTDQ | m128,x | 1 | Mem | ~550 | 1 | |
| (V)MOVNTDQA | v, m | 1 | Mem | 5 | 0.5 | |
| MASKMOVQ | mm,mm | 6 | Mem | ~550 | 12 | |
| MASKMOVDQU | x,x | 6 | Mem | ~550 | 12 | |
| VPMASKMOVD/Q | v,v,m | 5 | Mem | 7 | 9 | |
| VPMASKMOVD/Q | m,v,v | 4 | Mem | 6 | 8 | |
| VPACKSSWB/DW | | | | | | |
| VPACKUSWB/DW | mm/x, mm/x | 1 | FP0 | 2-6 | 1 | |
| VPACKSSWB/DW | | | | | | |
| VPACKUSWB/DW | y/z,y/z,y/z | 5 | | 11-14 | 9 | |
| VPACKSSWB/DW | | | | | | |
| VPACKUSWB/DW | y/z,y/z,m | 6 | | | 9 | |
| PUNPCKH/LBW/WD/DQ | mm/x, mm/x | 1 | FP0 | 2-6 | 1 | |
| VPUNPCKH/LBW/WD | y/z,y/z,y/z | 5 | | 11-14 | 9 | |
| VPUNPCKH/LBW/WD | y/z,y/z,m | 6 | | | 9 | |
| PUNPCKH/LQDQ | mm/x, mm/x | 1 | FP0 | 2-6 | 1 | |
| VPUNPCKH/L(Q)DQ | y/z,y/z,y/z | 1 | FP0 | 4-7 | 2 | |
| (V)PMOVSX BW BD BQ DW DQ | v,v | 2 | | 8 | 7-8 | |
| (V)PMOVZX BW BD BQ DW DQ | v,v | 1 | | 3 | 2 | |

Knights Landing

| | | | | | |
|---------------------------|-----------|----|-------|-------|-----|
| VPMOV QB QW QD DB DW | v,v | 1 | FP0 | 3 | 1 |
| VPMOV(U)S QB QW QD DB DW | v,v | 2 | | 8 | 7 |
| PSHUFB | mm,mm | 1 | FP0 | 2-6 | 1 |
| PSHUFB | x,x | 5 | FP0 | 11-13 | 10 |
| PSHUFB | x,m | 6 | FP0 | | 10 |
| VPSHUFB | y,y,y | 12 | FP0 | 23-25 | 12 |
| VPSHUFB | y,y,m | 13 | FP0 | | 13 |
| PSHUFW | mm,mm,i | 1 | FP0 | 2-6 | 1 |
| PSHUFL/HW | x,x,i | 1 | FP0 | 2-6 | 1 |
| VPSHUFL/HW | y,y,i | 4 | FP0 | 11-14 | 8 |
| VPSHUFL/HW | y,m,i | 5 | FP0 | | 9 |
| (V)PSHUFD | v,v,i | 1 | FP0 | 3-6 | 1 |
| PALIGNR | mm,mm,i | 1 | FP0 | 2-6 | 1 |
| PALIGNR | x,x,i | 1 | FP0 | 2-6 | 2 |
| VPALIGNR | y,y,y,i | 5 | FP0 | 11-14 | 9 |
| VPALIGNR | y,y,m,i | 6 | FP0 | | 9 |
| VALIGND/Q | z,z,z,i | 1 | FP0 | 3-6 | 1 |
| VPCOMPRESSD/Q | z{k},z | 1 | | 3-6 | 3 |
| VPEXPANDD/Q | z{k},z | 1 | | 3-6 | 3 |
| PBLENDVB | x,x,xmm0 | 5 | FP0 | 9-10 | 9 |
| PBLENDVB | x,m,xmm0 | 6 | FP0 | | 9 |
| VPBLENDVB | v,v,v,v | 4 | | 8-10 | 8 |
| PBLENDW | x,x/m,i | 1 | | 2 | 2 |
| VPBLENDW | y,y,y/m,i | 1 | | 2 | 0.5 |
| VPBLEND | v,v,v/m,i | 1 | FP0/1 | 2 | 0.5 |
| VPBLENDMD/Q | z{k},z,z | 1 | FP0/1 | 2 | 0.5 |
| VPERMD | v,v,v | 1 | FP0 | 3-6 | 1 |
| VPERMQ | v,v,i | 1 | FP0 | 3-6 | 1 |
| VPERM2I128 | v,v,v,i | 1 | FP0 | 4-7 | 2 |
| VPERM2D VPERMT2D | v,v,v | 1 | FP0 | 4-7 | 2 |
| VPERM2Q VPERMT2Q | v,v,v | 1 | FP0 | 4-7 | 2 |
| VSHUFI32X4 | z,z,z,i | 1 | FP0 | 4-7 | 2 |
| VSHUFI64X2 | z,z,z,i | 1 | FP0 | 4-7 | 2 |
| PMOVMASKB | r32,mm | 4 | | 14 | 8 |
| PMOVMASKB | r32,x | 5 | | 19 | 8 |
| PMOVMASKB | r32,y | 12 | | 26 | 12 |
| PEXTRB/W/D | r32,x,i | 2 | | 8 | 7 |
| PEXTRQ | r64,x,i | 2 | | 8 | 10 |
| VEEXTRACTI128 | x,y,i | 1 | FP0 | 3-6 | 1 |
| VEEXTRACTI128 | m128,y,i | 4 | | 7 | 8 |
| PINSRB/W | x,r32,i | 1 | FP0 | 5 | 1 |
| PINSRD | x,r32,i | 1 | FP0 | 4 | 1.5 |
| PINSRQ | x,r64,i | 1 | FP0 | 4 | 6 |
| VINSERTI128 | y,y,x,i | 1 | FP0 | 3-6 | 1 |
| VINSERTI32X4 | z,z,x,i | 1 | FP0 | 3-6 | 1 |
| VINSERTI64X4 | z,z,y,i | 1 | FP0 | 3-6 | 1 |
| VPBROADCASTB/W | v,x | 2 | | 8 | 7 |
| VPBROADCASTD/Q | v,x | 1 | FP0 | 3 | 1 |
| VBROADCASTI128 | y,m128 | 1 | | 5 | 0.5 |
| VBROADCASTI32X4 | z,m128 | 1 | | 5 | 0.5 |
| VBROADCASTI64X4 | z,m256 | 1 | | 5 | 0.5 |
| Gather and scatter | | | | | |

Knights Landing

| | | | | | |
|--------------------------------|-------------|---|-------|----|-----|
| VPGATHERDD | x,[r+s*x],x | 6 | | | 12 |
| VPGATHERDD | y,[r+s*y],y | 6 | | | 12 |
| VPGATHERDD | z,[r+s*z],z | 1 | | | 11 |
| VPGATHERQD | x,[r+s*x],x | 6 | | | 12 |
| VPGATHERQD | x,[r+s*y],x | 6 | | | 12 |
| VPGATHERQD | y,[r+s*z],y | 1 | | | 7 |
| VPGATHERDQ | x,[r+s*x],x | 6 | | | 12 |
| VPGATHERDQ | y,[r+s*x],y | 6 | | | 12 |
| VPGATHERDQ | z,[r+s*y],z | 1 | | | 7 |
| VPGATHERQQ | x,[r+s*x],x | 6 | | | 12 |
| VPGATHERQQ | y,[r+s*y],y | 6 | | | 12 |
| VPGATHERQQ | z,[r+s*z],z | 1 | | | 7 |
| VPSCATTERDD | z,[r+s*z],z | 4 | | | 17 |
| VPSCATTERQD | y,[r+s*z],y | 4 | | | 11 |
| VPSCATTERDQ | z,[r+s*y],z | 4 | | | 11 |
| VPSCATTERQQ | z,[r+s*z],z | 4 | | | 11 |
| Arithmetic instructions | | | | | |
| PADD/SUB(U,S)B/W/D/Q | mm/x, mm/x | 1 | FP0/1 | 2 | 0.5 |
| VPADD/SUB(U,S)B/W/D/Q | v,v,v | 1 | FP0/1 | 2 | 0.5 |
| PHADD(S)W PHSUB(S)W | mm, mm | 5 | | 18 | 9 |
| PHADD(S)W PHSUB(S)W | x, x | 6 | FP0 | 28 | 28 |
| PHADD(S)W PHSUB(S)W | x, m | 7 | | | 28 |
| PHADD(S)W PHSUB(S)W | y, y | 7 | FP0 | 23 | 9 |
| PHADDD PHSUBD | mm, mm | 4 | | 14 | 8 |
| PHADDD PHSUBD | x, x | 3 | | 11 | 9 |
| VPHADDD VPHSUBD | y,y,y | 3 | | 11 | 8 |
| PCMPEQ/GTB/W/D | mm/x,mm/x | 1 | FP0/1 | 2 | 0.5 |
| VPCMPEQ/GTB/W/D | y,y,y | 1 | FP0/1 | 2 | 0.5 |
| PCMPEQ/GTQ | x, x | 1 | | 2 | 2 |
| VPCMPEQQ | y,y,y | 1 | FP0/1 | 2 | 0.5 |
| VPCMP(U)D/Q | k,z,z,i | 1 | FP0/1 | 2 | 0.5 |
| VPTSTMD/Q | k,z,z | 1 | FP0/1 | 2 | 0.5 |
| PMULL/HW PMULHUW | mm,mm | 1 | FP0 | 6 | 1 |
| PMULL/HW PMULHUW | x, x | 1 | FP0 | 7 | 2 |
| VPMULL/HW VPMULHUW | y,y,y | 5 | FP0 | 16 | 9 |
| PMULHSW | mm,mm | 1 | FP0 | | 1 |
| PMULHSW | x, x | 1 | FP0 | 7 | 2 |
| PMULLD | x, x | 1 | FP0 | 7 | 2 |
| VPMULLD | v,v,v | 1 | FP0 | 7 | 1 |
| PMULDQ | x, x | 1 | FP0 | 6 | 2 |
| VPMULDQ | v,v,v | 1 | FP0/1 | 6 | 0.5 |
| PMULUDQ | mm/x,mm/x | 1 | FP0/1 | 6 | 0.5 |
| VPMULUDQ | v,v,v | 1 | FP0/1 | 6 | 0.5 |
| PMADDWD | mm,mm | 1 | FP0 | 6 | 1 |
| PMADDWD | x, x | 1 | FP0 | 7 | 2 |
| PMADDUBSW | mm,mm | 1 | FP0 | 6 | 1 |
| PMADDUBSW | x, x | 1 | FP0 | 7 | 2 |
| PMADDUBSW | y,y,y | 5 | FP0 | 16 | 9 |
| PSADBW | mm,mm | 1 | FP0 | 6 | 1 |
| PSADBW | x, x | 1 | FP0 | 7 | 2 |
| PSADBW | y,y,y | 5 | FP0 | 16 | 9 |
| MPSADBW | x,x,i | 3 | | 9 | 9 |

Knights Landing

| | | | | | | |
|--|-----------|----|-------|-----|-----|-----------|
| VMPSADBWB | y,y,y,i | 9 | | 19 | 13 | |
| PAVGB/W | mm/x,mm/x | 1 | FP0/1 | 2 | 0.5 | |
| PAVGB/W | y,y,y | 1 | FP0/1 | 2 | 0.5 | |
| PMIN/MAXUB/SW | mm/x,mm/x | 1 | FP0/1 | 2 | 2 | |
| VPMIN/MAXUB/SW/D/Q | y,y,y | 1 | FP0/1 | 2 | 0.5 | |
| PHMINPOSUW | x,x | 1 | FP0 | 3 | 2 | |
| PABSB/W/D | mm,mm | 1 | FP0/1 | 2 | 0.5 | |
| PABSB/W/D | x,x | 1 | FP0/1 | 2 | 2 | |
| VPABSB/W/D/Q | y,y | 1 | FP0/1 | 2 | 0.5 | |
| PSIGNB/W/D | mm,mm | 1 | FP0/1 | 2 | 0.5 | |
| PSIGNB/W/D | x,x | 1 | FP0/1 | 2 | 2 | |
| VPSIGNB/W/D | y,y | 1 | FP0/1 | 2 | 0.5 | |
| Logic instructions | | | | | | |
| PAND(N)/OR/XOR | mm/x,mm/x | 1 | FP0/1 | 2 | 0.5 | |
| VPAND(N)/OR/XOR | y,y,y | 1 | FP0/1 | 2 | 0.5 | |
| VPAND(N)/OR/XORD/Q | z,z,z | 1 | FP0/1 | 2 | 0.5 | |
| VPTERNLOGD/Q | z,z,z,i | 1 | FP0+1 | 2 | 1 | |
| PTEST | x,x | 4 | | 9 | 9 | |
| VPTEST(N)MD/Q | k,z,z | 1 | FP0/1 | 2 | 0.5 | |
| VPTEST | y,y | 4 | | 9 | 8 | |
| PSLL/RL/RAW/D/Q | mm,mm | 1 | FP0 | 2 | 1 | |
| PSLL/RL/RAW/D/Q | x,x | 2 | FP0 | 13 | 13 | |
| PSLL/RL/RAW/D/Q | mm/x,i | 1 | FP0 | 2 | 1 | |
| VPSLL/RL/RAW/D/Q | y,y,i | 4 | FP0 | 11 | 8 | |
| VPSLL/RA/RLVD/Q | z,z,z | 1 | FP0 | 2 | 1 | |
| VPROL/RD/Q | z,z,i | 1 | FP0 | 2 | 1 | |
| VPROL/RVD/Q | z,z,z | 1 | FP0 | 2 | 1 | |
| VPLZCNTD/Q | z,z | 1 | FP0 | 2 | 1 | |
| VPCONFLICTD/Q | z,z | 1 | FP0 | 3 | 1 | |
| String instructions | | | | | | |
| PCMPESTRI | x,x,i | 9 | FP0 | 21 | 21 | +1 if mem |
| PCMPESTRM | x,x,i | 8 | FP0 | 17 | 17 | +1 if mem |
| PCMPISTRI | x,x,i | 6 | FP0 | 17 | 17 | +1 if mem |
| PCMPISTRM | x,x,i | 5 | FP0 | 13 | 13 | +1 if mem |
| Encryption instructions | | | | | | |
| PCLMULQDQ | x,x,i | 1 | FP0 | 3-6 | 2 | +1 if mem |
| AESDEC, AESDECLAST, AESENC, AESENCLAST, AESIMC, AESKEYGENASSIST | x,x | 1 | | 3-6 | 2 | |
| Other | | | | | | |
| EMMS | | 10 | | | 13 | |

Floating point XMM instructions

| | Operands | μops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|----------|------|-------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| (V)MOVAPS/D | v,v | 1 | FP0/1 | 2 | 0.5 | |
| (V)MOVAPS/D | v,m | 1 | Mem | 5 | 0.5 | |

Knights Landing

| | | | | | |
|---------------------|----------|---|-------|------|-----|
| (V)MOVAPS/D | m,v | 1 | Mem | 5 | 1 |
| (V)MOVUPS/D | v,m | 1 | Mem | 5 | 0.5 |
| (V)MOVUPS/D | m,v | 1 | Mem | 5 | 1 |
| VMOVAPS/D VMOVUPS/D | z{k},m | 1 | Mem | 7 | 0.5 |
| VMOVAPS/D VMOVUPS/D | m{k},z | 1 | Mem | 9 | 1 |
| MOVSS/D | x, x | 1 | FP0/1 | 2 | 0.5 |
| MOVSS/D | x,m | 1 | Mem | 5 | 0.5 |
| MOVSS/D | m,x | 1 | Mem | 5 | 1 |
| MOVHPS/D | x,m64 | 1 | Mem | 6 | 1.5 |
| MOVHPS/D | m64,x | 4 | Mem | 9 | 1 |
| MOVLPS/D | x,m64 | 1 | Mem | 6 | 1.5 |
| MOVLPS/D | m64,x | 1 | Mem | 6 | 1 |
| (V)MOVNTPS/D | m,v | 1 | Mem | ~500 | |
| MOVMSKPS/D | r32,x | 2 | FP0 | 6 | 7 |
| MOVLHPS MOVHLPS | x,x | 1 | FP0 | 3-6 | 1 |
| MOVDDUP | x,x | 1 | FP0 | 3-6 | 1 |
| MOVDDUP | x,m | 1 | | 14 | |
| VMOVDDUP | v,v | 1 | | 3-6 | 1 |
| (V)MOVSH/LDUP | v,v | 1 | FP0 | 3-6 | 1 |
| VBROADCASTSS/D | v,x | 1 | | 3-6 | 1 |
| VBROADCASTSS/D | v,m | 1 | | 5 | 0.5 |
| VBROADCASTF128 | y,m128 | 1 | | 5 | 0.5 |
| VBROADCASTF32X4 | v,m128 | 1 | | 5 | 0.5 |
| VBROADCASTF64X4 | z,m256 | 1 | | 5 | 0.5 |
| UNPCKH/LPS/D | x,x | 1 | FP0 | 2 | 2 |
| VUNPCKH/LPS/D | v,v,v | 1 | FP0 | 4-7 | 2 |
| INSERTPS | x,x,i | 2 | | 8 | 7 |
| INSERTPS | x,m32,i | 4 | | 17 | 8 |
| INSERTF128 | y,x | 1 | | 3-6 | 1 |
| INSERTF128 | y,m128 | 1 | | 7 | 1 |
| VINSERTF32X4 | z,z,x | 1 | | 3-6 | 1 |
| VINSERTF32X4 | z,z,m128 | 1 | | 7 | 1 |
| VINSERTF64X4 | z,z,y | 1 | | 3-6 | 1 |
| VINSERTF64X4 | z,z,m256 | 1 | | 7 | 1 |
| EXTRACTPS | r32,x,i | 2 | | 8 | 7 |
| EXTRACTPS | m32,x,i | 4 | | 7 | 8 |
| VEXTRACTF128 | x,y,i | 1 | | 3-6 | 1 |
| VEXTRACTF128 | m128,y | 4 | | 7 | 8 |
| VEXTRACTF32X4 | x,z | 1 | | 3-6 | 1 |
| VEXTRACTF32X4 | m128,z | 4 | | 7 | 8 |
| VEXTRACTF64X4 | y,z | 1 | | 3-6 | 1 |
| VEXTRACTF64X4 | m256,z | 4 | | 7 | 8 |
| BLENDPS/PD | x,x/m,i | 1 | FP0/1 | 2 | 2 |
| VBLENDPS/PD | v,v,v,i | 1 | FP0/1 | 2 | 0.5 |
| (V)BLENDVPS/PD | v,v,v | 2 | | 7 | 7 |
| BLENDVPS/PD | x,m,xmm0 | 3 | | | 8 |
| VBLENDMPS/D | z{k},z,z | 1 | FP0/1 | 2 | 0.5 |
| SHUFPS/D | x,x,i | 1 | FP0 | 4 | 2 |
| VSHUFPS/D | v,v,v,i | 1 | FP0 | 4-7 | 2 |
| VSHUFF32X4 | z,z,z,i | 1 | FP0 | 4-7 | 2 |
| VSHUFF64X2 | z,z,z,i | 1 | FP0 | 4-7 | 2 |
| VPERMILPS/PD | v,v/m,i | 1 | FP0 | 3-6 | 1 |
| VPERMILPS/PD | v,v,v/m | 1 | FP0 | 3-6 | 1 |

Knights Landing

| | | | | | |
|---------------------------|-------------|---|-----|-----|------|
| VPERM2F128 | y,y,y/m,i | 1 | FP0 | 4-7 | 2 |
| VPERMPS/PD | v,v,v/m | 1 | FP0 | 3-6 | 1 |
| VPERMI2PS/PD | z,z,z/m | 1 | FP0 | 4-7 | 2 |
| VCOMPRESSPS/D | z{k},z | 1 | | 3-6 | 3 |
| VEXPANDPS/D | z{k},z | 1 | | 3-6 | 3 |
| Gather and scatter | | | | | |
| VPGATHERDPS | x,[r+s*x],x | 6 | | | 12 |
| VPGATHERDPS | y,[r+s*y],y | 6 | | | 12 |
| VPGATHERDPS | z,[r+s*z],z | 1 | | | 11 |
| VPGATHERQPS | x,[r+s*x],x | 6 | | | 12 |
| VPGATHERQPS | x,[r+s*y],x | 6 | | | 12 |
| VPGATHERQPS | y,[r+s*z],y | 1 | | | 7 |
| VPGATHERDPD | x,[r+s*x],x | 6 | | | 12 |
| VPGATHERDPD | y,[r+s*x],y | 6 | | | 12 |
| VPGATHERDPD | z,[r+s*y],z | 1 | | | 7 |
| VPGATHERQPD | x,[r+s*x],x | 6 | | | 12 |
| VPGATHERQPD | y,[r+s*y],y | 6 | | | 12 |
| VPGATHERQPD | z,[r+s*z],z | 1 | | | 7 |
| VGATHERPF0DPS | z,[r+s*z],z | 1 | | | ~200 |
| VGATHERPF0QPS | y,[r+s*z],y | 1 | | | ~100 |
| VGATHERPF0DPD | z,[r+s*y],z | 1 | | | ~100 |
| VGATHERPF0QPD | z,[r+s*z],z | 1 | | | ~100 |
| VGATHERPF1DPS | z,[r+s*z],z | 1 | | | ~200 |
| VGATHERPF1QPS | y,[r+s*z],y | 1 | | | ~100 |
| VGATHERPF1DPD | z,[r+s*y],z | 1 | | | ~100 |
| VGATHERPF1QPD | z,[r+s*z],z | 1 | | | ~100 |
| VPSCATTERDPS | z,[r+s*z],z | 4 | | | 17 |
| VPSCATTERQPS | y,[r+s*z],y | 4 | | | 11 |
| VPSCATTERDPD | z,[r+s*y],z | 4 | | | 11 |
| VPSCATTERQPD | z,[r+s*z],z | 4 | | | 11 |
| VSCATTERPF0DPS | z,[r+s*z],z | 1 | | | ~200 |
| VSCATTERPF0QPS | y,[r+s*z],y | 1 | | | ~100 |
| VSCATTERPF0DPD | z,[r+s*y],z | 1 | | | ~100 |
| VSCATTERPF0QPD | z,[r+s*z],z | 1 | | | ~100 |
| VSCATTERPF1DPS | z,[r+s*z],z | 1 | | | ~200 |
| VSCATTERPF1QPS | y,[r+s*z],y | 1 | | | ~100 |
| VSCATTERPF1DPD | z,[r+s*y],z | 1 | | | ~100 |
| VSCATTERPF1QPD | z,[r+s*z],z | 1 | | | ~100 |
| Conversion | | | | | |
| (V)CVTSD2SS | x,x | 1 | FP0 | 2 | 1 |
| (V)CVTSS2SD | x,x | 1 | FP0 | 2 | 1 |
| (V)CVTPD2PS | v,v | 2 | FP0 | 7 | 7 |
| (V)CVTPS2PD | v,v | 2 | FP0 | 7 | 7 |
| VCVTPS2PH | v,v | 2 | | 7 | 7 |
| VCVTPS2PH | m,v | 5 | | | 9 |
| VCVTPH2PS | v,v | 2 | | 7 | 7 |
| VCVTPH2PS | v,m | 3 | | | 8 |
| (V)CVT(T)SS2(U)SI | r32/64,x | 2 | FP0 | 6 | 7 |
| (V)CVT(U)SI2SS | x,r32/64 | 1 | FP0 | 5 | 1 |
| (V)CVT(T)SD2(U)SI | r32/64,x | 2 | FP0 | 6 | 7 |
| (V)CVT(U)SI2SD | x,r32/64 | 1 | FP0 | 5 | 1 |

Knights Landing

| | | | | | |
|---------------------|---------|----|-------|----|-----|
| CVT(T)PS2PI | mm,x | 1 | FP0 | 3 | 1 |
| CVTPI2PS | x,mm | 2 | FP0 | 7 | 7 |
| CVT(T) PD2PI | mm,x | 2 | FP0 | 7 | 7 |
| CVTPI2PD | x,mm | 2 | FP0 | 7 | 7 |
| (V)CVT(T) PS2DQ | v,v | 1 | FP0 | 2 | 1 |
| (V)CVTDQ2PS | v,v | 1 | FP0 | 2 | 1 |
| (V)CVT(T)PD2DQ | v,v | 2 | FP0 | 7 | 7 |
| (V)CVTDQ2PD | v,v | 2 | FP0 | 7 | 7 |
| VCVT(T)PS2UDQ | z,z | 1 | FP0 | 2 | 1 |
| VCVTUDQ2PS | z,z | 1 | FP0 | 2 | 1 |
| VCVT(T)PD2UDQ | z,z | 2 | FP0 | 7 | 7 |
| VCVTUDQ2PD | z,z | 2 | FP0 | 7 | 7 |
| Arithmetic | | | | | |
| ADDSS SUBSS | x,x | 1 | FP0/1 | 6 | 0.5 |
| ADDSD SUBSD | x,x | 1 | FP0/1 | 6 | 0.5 |
| ADDPS SUBPS | x,x | 1 | FP0/1 | 6 | 0.5 |
| VADDPS VSUBPS | v,v,v | 1 | FP0/1 | 6 | 0.5 |
| ADDPD SUBPD | x,x | 1 | FP0/1 | 6 | 0.5 |
| VADDPD VSUBPD | v,v,v | 1 | FP0/1 | 6 | 0.5 |
| ADDSUBPS/D | x,x | 1 | FP0/1 | 6 | 0.5 |
| VADDSUBPS/D | v,v,v | 1 | FP0/1 | 6 | 0.5 |
| HADDPS/D HSUBPS/D | x,x | 3 | | 15 | 8 |
| VHADDPS/D VHSUBPS/D | yy,y, | 3 | | 15 | 8 |
| MULSS/D | x,x | 1 | FP0/1 | 6 | 0.5 |
| MULPS/D | x,x | 1 | FP0/1 | 6 | 0.5 |
| VMULPS/D | v,v,v | 1 | FP0/1 | 6 | 0.5 |
| DIVSS | x,x | 3 | FP0 | 27 | 17 |
| DIVSD | x,x | 3 | FP0 | 42 | 42 |
| DIVPS | x,x | 18 | FP0 | 32 | 20 |
| VDIVPS | v,v,v | 18 | FP0 | 32 | 32 |
| DIVPD | x,x | 18 | FP0 | 32 | 20 |
| VDIVPD | v,v,v | 18 | FP0 | 32 | 32 |
| RCPSS | x,x | 1 | FP0 | 7 | 2 |
| (V)RCPPS | v,v | 1 | FP0 | 8 | 3 |
| VRCP14SS | x,x,x | 1 | FP0 | 7 | 2 |
| VRCP14PS | v,v | 1 | FP0 | 8 | 3 |
| VRCP28SS | x,x,x | 1 | FP0 | 7 | 2 |
| VRCP28PS | v,v | 1 | FP0 | 8 | 3 |
| VRCP28SD | x,x,x | 1 | FP0 | 7 | 2 |
| VRCP28PD | v,v | 1 | FP0 | 7 | 2 |
| CMPccSS/D PS/D | x,x | 1 | FP0/1 | 2 | 0.5 |
| VCMPccPS/D | k,z,z | 1 | FP0/1 | 2 | 0.5 |
| COMISS/D UCOMISS/D | x,x | 2 | | 7 | 7 |
| COMISS/D UCOMISS/D | x,m | 3 | | | 8 |
| MAXSS/D MINSS/D | x,x | 1 | FP0/1 | 2 | 0.5 |
| MAXPS/D MINPS/D | x,x | 1 | FP0/1 | 2 | 0.5 |
| VMAXPS/D VMINPS/D | v,v,v | 1 | FP0/1 | 2 | 0.5 |
| ROUNDSS/D | x,x,i | 1 | | 6 | 2 |
| (V)ROUNDPS/D | v,v,i | 1 | | 6 | 0.5 |
| VRNDSCALESS/D | x,x,x,i | 1 | | 6 | 0.5 |
| VRNDSCALEPS/D | v,v,i | 1 | | 6 | 0.5 |
| VSCALEFSS/D | x,x,x | 1 | FP0/1 | 6 | 0.5 |

Knights Landing

| | | | | | | |
|----------------------------|---------|-----|-------|-----|-----|-------------|
| VSCALEFPS/D | z,z,z | 1 | FP0/1 | 6 | 0.5 | |
| DPPS | x,x,i | 14 | | 36 | 14 | |
| VDPPS | y,y,y,i | 14 | | 36 | 13 | |
| DPPD | x,x,i | 12 | | 24 | 13 | |
| VFMADD... (all FMA instr.) | v,v,v | 1 | | 6 | 0.5 | |
| Math | | | | | | |
| SQRTSS | x,x | 3 | FP0 | 28 | 18 | |
| SQRTPS | x,x | 18 | FP0 | 38 | 16 | |
| VSQRTPS | v,v | 18 | FP0 | 38 | 16 | |
| SQRTSD | x,x | 30 | FP0 | 43 | 35 | |
| SQRTPD | x,x | 18 | FP0 | 37 | 16 | |
| VSQRTPD | v,v | 18 | FP0 | 37 | 16 | |
| RSQRTSS | x,x | 1 | FP0 | 7 | 2 | |
| RSQRTPS | x,x | 1 | FP0 | 8 | 3 | |
| VRSQRTPS | v,v | 1 | FP0 | 7 | 3 | |
| VRSQRT14SS | v,v,v | 1 | FP0 | 7 | 2 | |
| VRSQRT14PS | v,v | 1 | FP0 | 7 | 3 | |
| VRSQRT28SS | v,v,v | 1 | FP0 | 7 | 2 | |
| VRSQRT28PS | v,v | 1 | FP0 | 7 | 3 | |
| VRSQRT28SD | v,v,v | 1 | FP0 | 7 | 2 | |
| VRSQRT28PD | v,v | 1 | FP0 | 6 | 2 | |
| VEXP2PS | v,v | 2 | | 10 | 7 | |
| VEXP2PD | v,v | 2 | | 9 | 7 | |
| VFIXUPIMMSS/D/PS/D | v,v,v,i | 1 | FP0 | 2 | 1 | |
| VGETEXPSS/D | v,v,v | 1 | FP0/1 | 6 | 0.5 | |
| VGETEXPPS/D | v,v | 1 | FP0/1 | 6 | 0.5 | |
| VGETMANTSS/D | v,v,v | 1 | FP0/1 | 6 | 0.5 | |
| VGETMANTPS/D | v,v | 1 | FP0/1 | 6 | 0.5 | |
| VFIXUPIMMSS/SD/PS/PD | v,v,v | 1 | FP0 | 2 | 1 | |
| Logic | | | | | | |
| ANDPS/D ANDNPS/D | x,x | 1 | FP0/1 | 2 | 0.5 | |
| ORPS/D XORPS/D | x,x | 1 | FP0/1 | 2 | 0.5 | |
| VANDPS/D VANDNPS/D | v,v,v | 1 | FP0/1 | 2 | 0.5 | |
| VORPS/D VXORPS/D | v,v,v | 1 | FP0/1 | 2 | 0.5 | |
| Other | | | | | | |
| VZEROUPPER | | 11 | | | 30 | 32 bit mode |
| VZEROUPPER | | 19 | | | 36 | 64 bit mode |
| VZEROALL | | 11 | | | 30 | 32 bit mode |
| VZEROALL | | 19 | | | 36 | 64 bit mode |
| LDMXCSR | m32 | 6 | | | 21 | |
| STMXCSR | m32 | 5 | | | 15 | |
| FXSAVE | m | 90 | | | 113 | 32 bit mode |
| FXSAVE | m | 98 | | | 119 | 64 bit mode |
| FXRSTOR | m | 98 | | | 122 | 32 bit mode |
| FXRSTOR | m | 114 | | | 130 | 64 bit mode |
| FNSAVE | m | 135 | | 205 | 205 | |
| FRSTOR | m | 78 | | 191 | 191 | |
| XSAVE | m | 251 | | | 396 | 32 bit mode |
| XSAVE | m | 291 | | | 430 | 64 bit mode |
| XRSTOR | m | 116 | | | 231 | 32 bit mode |

Knights Landing

| | | | | | | |
|----------|---|-----|--|--|-----|-------------|
| XRSTOR | m | 157 | | | 273 | 64 bit mode |
| XSAVEOPT | m | 251 | | | 396 | 32 bit mode |
| XSAVEOPT | m | 291 | | | 428 | 64 bit mode |

Mask register instructions

| | Operands | μops | Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|----------|------|-------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| KMOVW | k,k | 1 | FP0/1 | 2 | 0.5 | |
| KMOVW | k,m | 1 | | 7 | 0.5 | |
| KMOVW | m,k | 1 | | 7 | 1 | |
| KMOVW | k,r | 1 | FP0 | 5 | 1 | |
| KMOVW | r,k | 1 | | 4 | 1 | |
| KUNPCKBW | k,k,k | 1 | FP0/1 | 2 | 0.5 | |
| VPBROADCASTMB2Q | v,k | 1 | FP0 | 6 | 1 | |
| VPBROADCASTMW2D | v,k | 1 | FP0 | 6 | 1 | |
| Arithmetic | | | | | | |
| KSHIFTLW | k,k,i | 1 | FP0/1 | 2 | 0.5 | |
| KSHIFTRW | k,k,i | 1 | FP0/1 | 2 | 0.5 | |
| Logic | | | | | | |
| KANDW KANDNW | k,k,k | 1 | FP0/1 | 2 | 0.5 | |
| KORW KXORW KXNORW | k,k,k | 1 | FP0/1 | 2 | 0.5 | |
| KNOTW | k,k | 1 | FP0/1 | 2 | 0.5 | |
| KORTESTW | k,k | 1 | FP1 | 5 | 1 | |

VIA Nano 2000 series

List of instruction timings and μ ops breakdown

Explanation of column headings:

| | |
|-----------------------------|---|
| Operands: | i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, mm/x = mmx or xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc. |
| μops: | The number of micro-operations from the decoder or ROM. Note that the VIA Nano 2000 processor has no reliable performance monitor counter for μ ops. Therefore the number of μ ops cannot be determined except in simple cases. |
| Port: | Tells which execution port or unit is used. Instructions that use the same port cannot execute simultaneously. |
| I1: | Integer add, Boolean, shift, etc. |
| I2: | Integer add, Boolean, move, jump. |
| I12: | Can use either I1 or I2, whichever is vacant first. |
| MA: | Multiply, divide and square root on all operand types. |
| MB: | Various Integer and floating point SIMD operations. |
| MBfadd: | Floating point addition subunit under MB. |
| SA: | Memory store address. |
| ST: | Memory store. |
| LD: | Memory load. |
| Latency: | This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay. |

Note: There is an additional latency for moving data from one unit or subunit to another. A table of these latencies is given in manual 3: "The microarchitecture of Intel, AMD and VIA CPUs". These additional latencies are not included in the listings below where the source and destination operands are of the same type.

Reciprocal throughput: The average number of clock cycles per instruction for a series of independent instructions of the same kind in the same thread.

Integer instructions

| | Operands | μ ops | Port | Latency | Reciprocal throughput | Remarks |
|--------------------------|----------|-----------|--------|---------|-----------------------|-------------------------------|
| Move instructions | | | | | | |
| MOV | r,r | 1 | I2 | 1 | 1 | Latency 4 on pointer register |
| MOV | r,i | 1 | I2 | 1 | 1 | |
| MOV | r,m | 1 | LD | 2 | 1 | |
| MOV | m,r | 1 | SA, ST | 2 | 1.5 | |
| MOV | m,i | 1 | SA, ST | | 1.5 | |
| MOV | r,sr | | | | 1 | |
| MOV | m,sr | | | | 2 | |
| MOV | sr,r | | | 20 | 20 | |
| MOV | sr,m | | | 20 | 20 | |
| MOVNTI | m,r | | SA, ST | 2 | 1.5 | |

VIA Nano 2000

| | | | | | | |
|--------------------------------|---------|---|--------------|-----|------|-----------------------------------|
| MOV SX MOV SX D | | | | | | |
| MOV ZX | r,r | 1 | I2 | 1 | 1 | |
| MOV SX MOV SX D | r,m | 2 | LD, I2 | 3 | 1 | |
| MOV ZX | r,m | 1 | LD | 2 | 1 | |
| CMOVcc | r,r | 2 | I1, I2 | 2 | 1 | |
| CMOVcc | r,m | | LD, I1 | 5 | 2 | |
| XCHG | r,r | 3 | I2 | 3 | 3 | |
| XCHG | r,m | | | 20 | 20 | Implicit lock |
| XLAT | m | | | 6 | | |
| PUSH | r | | SA, ST | | 1-2 | |
| PUSH | i | | SA, ST | | 1-2 | |
| PUSH | m | | Ld, SA, ST | | 2 | |
| PUSH | sr | | | | 17 | |
| PUSHF(D/Q) | | | | 8 | 8 | |
| PUSHA(D) | | | | | 15 | Not in x64 mode |
| POP | r | | LD | | 1.25 | |
| POP | (E/R)SP | | | | 4 | |
| POP | m | | | | 5 | |
| POP | sr | | | | 20 | |
| POPF(D/Q) | | | | 9 | 9 | |
| POPA(D) | | | | | 12 | Not in x64 mode |
| LAHF | | 1 | I1 | 1 | 1 | |
| SAHF | | 1 | I1 | 1 | 1 | |
| SALC | | | | 9 | 6 | Not in x64 mode |
| LEA | r,m | 1 | SA | 1 | 1 | 3 clock latency on input register |
| BSWAP | r | 1 | I2 | 1 | 1 | |
| LDS LES LFS LGS LSS | m | | | 30 | 30 | |
| PREFETCHNTA | m | | LD | | 1-2 | |
| PREFETCHT0/1/2 | m | | LD | | 1-2 | |
| LFENCE | | | | | 14 | |
| MFENCE | | | | | 14 | |
| SFENCE | | | | | 14 | |
| Arithmetic instructions | | | | | | |
| ADD SUB | r,r/i | 1 | I12 | 1 | 1/2 | |
| ADD SUB | r,m | 2 | LD I12 | | 1 | |
| ADD SUB | m,r/i | 3 | LD I12 SA ST | 5 | 2 | |
| ADC SBB | r,r/i | 1 | I1 | 1 | 1 | |
| ADC SBB | r,m | 2 | LD I1 | | 1 | |
| ADC SBB | m,r/i | 3 | LD I1 SA ST | 5 | 2 | |
| CMP | r,r/i | 1 | I12 | 1 | 1/2 | |
| CMP | m,r/i | 2 | LD I12 | | 1 | |
| INC DEC NEG NOT | r | 1 | I12 | 1 | 1/2 | |
| INC DEC NEG NOT | m | 3 | LD I12 SA ST | 5 | | |
| AAA | | | | | 37 | Not in x64 mode |
| AAS | | | | | 37 | Not in x64 mode |
| DAA | | | | | 22 | Not in x64 mode |
| DAS | | | | | 24 | Not in x64 mode |
| AAD | | | | | 23 | Not in x64 mode |
| AAM | | | | | 30 | Not in x64 mode |
| MUL IMUL | r8 | | MA | 7-9 | | Extra latency to other ports |

VIA Nano 2000

| | | | | | | |
|--------------------------------------|------------|---|--------------|---------|---------|---------------------------------|
| MUL IMUL | r16 | | MA | 7-9 | | do. |
| MUL IMUL | r32 | | MA | 7-9 | | do. |
| MUL IMUL | r64 | | MA | 8-10 | | do. |
| IMUL | r16,r16 | | MA | 4-6 | 1 | do. |
| IMUL | r32,r32 | | MA | 4-6 | 1 | do. |
| IMUL | r64,r64 | | MA | 5-7 | 2 | do. |
| IMUL | r16,r16,i | | MA | 4-6 | 1 | do. |
| IMUL | r32,r32,i | | MA | 4-6 | 1 | do. |
| IMUL | r64,r64,i | | MA | 5-7 | 2 | do. |
| DIV | r8 | | MA | 26 | 26 | do. |
| DIV | r16 | | MA | 27-35 | 27-35 | do. |
| DIV | r32 | | MA | 25-41 | 25-41 | do. |
| DIV | r64 | | MA | 148-183 | 148-183 | do. |
| IDIV | r8 | | MA | 26 | 26 | do. |
| IDIV | r16 | | MA | 27-35 | 27-35 | do. |
| IDIV | r32 | | MA | 23-39 | 23-39 | do. |
| IDIV | r64 | | MA | 187-222 | 187-222 | do. |
| CBW CWDE CDQE | | 1 | I1 | 1 | 1 | |
| CWD CDQ CQO | | 1 | I1 | 1 | 1 | |
| Logic instructions | | | | | | |
| AND OR XOR | r,r/i | 1 | I12 | 1 | 1/2 | |
| AND OR XOR | r,m | 2 | LD I12 | | 1 | |
| AND OR XOR | m,r/i | 3 | LD I12 SA ST | 5 | 2 | |
| TEST | r,r/i | 1 | I12 | 1 | 1/2 | |
| TEST | m,r/i | 2 | LD I12 | | 1 | |
| SHR SHL SAR | r,i/cl | 1 | I1 | 1 | 1 | |
| ROR ROL | r,i/cl | 1 | I1 | 1 | 1 | |
| RCR RCL | r,1 | 1 | I1 | 1 | 1 | |
| RCR RCL | r,i/cl | | I1 | 28+3n | 28+3n | |
| SHLD SHRD | r16,r16,i | | I1 | 11 | 11 | |
| SHLD SHRD | r32,r32,i | | I1 | 7 | 7 | |
| SHLD | r64,r64,i | | I1 | 33 | 33 | |
| SHRD | r64,r64,i | | I1 | 43 | 43 | |
| SHLD SHRD | r16,r16,cl | | I1 | 11 | 11 | |
| SHLD SHRD | r32,r32,cl | | I1 | 7 | 7 | |
| SHLD | r64,r64,cl | | I1 | 33 | 33 | |
| SHRD | r64,r64,cl | | I1 | 43 | 43 | |
| BT | r,r/i | 1 | I1 | 1 | 1 | |
| BT | m,r | | I1 | | 8 | |
| BT | m,i | 2 | I1 | | 1 | |
| BTR BTS BTC | r,r/i | 2 | I1 | 2 | 2 | |
| BTR BTS BTC | m,r | | I1 | 10 | 10 | |
| BTR BTS BTC | m,i | | I1 | 8 | 8 | |
| BSF BSR | r,r | | I1 | 3 | 2 | |
| SETcc | r | | I1 | 2 | 1 | |
| SETcc | m | | | | 1 | |
| CLC STC CMC | | | I1 | 3 | 3 | |
| CLD STD | | | | 3 | 3 | |
| Control transfer instructions | | | | | | |
| JMP | short/near | 1 | I2 | 3 | 3 | 8 if >2 jumps in 16 bytes block |

VIA Nano 2000

| | | | | | | |
|----------------------------|------------|---|-----|--------|--|---|
| JMP | far | | | 58 | | Not in x64 mode |
| JMP | r | | 12 | 3 | 3 | 8 if >2 jumps in 16 bytes block |
| JMP | m(near) | | | 3 | 3 | do. |
| JMP | m(far) | | | 55 | | |
| Conditional jump | short/near | | | 1-3-8 | 1-3-8 | 1 if not jumping. 3 if jumping. 8 if >2 jumps in 16 bytes block |
| J(E/R)CXZ | short | | | 1-3-8 | 1-3-8 | do. |
| LOOP | short | | | 1-3-8 | 1-3-8 | do. |
| LOOP(N)E | short | | | 25 | 25 | |
| CALL | near | | | 3 | 3 | 8 if >2 jumps in 16 bytes block |
| CALL | far | | | 72 | 72 | Not in x64 mode |
| CALL | r | | | 3 | 3 | 8 if >2 jumps in 16 bytes block |
| CALL | m(near) | | | 4 | 3 | do. |
| CALL | m(far) | | | 72 | 72 | |
| RETN | | | | 3 | 3 | 8 if >2 jumps in 16 bytes block |
| RETN | i | | | 3 | 3 | do. |
| RETF | | | | 39 | 39 | |
| RETF | i | | | 39 | 39 | |
| BOUND | r,m | | | | 13 | Not in x64 mode |
| INTO | | | | | 7 | Not in x64 mode |
| String instructions | | | | | | |
| LODSB/W/D/Q | | | | | 1 | |
| REP LODSB/W/D/Q | | | | | 3n+22 | |
| STOSB/W/D/Q | | | | | 1-2 | |
| REP STOSB/W/D/Q | | | | | Small: 2n+2, Big: 6 bytes per clock | |
| MOVSB/W/D/Q | | | | | 2 | |
| REP MOVSB/W/D/Q | | | | | Small: 2n+45, Big: 6 bytes per clock | |
| SCASB/W/D/Q | | | | | 1 | |
| REP SCASB | | | | | 2.2n | |
| REP SCASW/D/Q | | | | | Small: 2n+50 Big: 5 bytes per clock | |
| CMPSB/W/D/Q | | | | | 6 | |
| REP CMPSB/W/D/Q | | | | | 2.4n+24 | |
| Other | | | | | | |
| NOP (90) | | 1 | All | | 1 | Blocks all ports |
| Long NOP (0F 1F) | | 1 | 112 | | 1/2 | |
| PAUSE | | | | | 25 | |
| ENTER | a,0 | | | | 23 | |
| ENTER | a,b | | | | 52+5b | |
| LEAVE | | | | 4 | 4 | |
| CPUID | | | | 53-173 | | |

| | | | | | | |
|-------|--|--|--|----|----|--|
| RDTSC | | | | | 39 | |
| RDPMC | | | | 40 | 40 | |

Floating point x87 instructions

| | Operands | μops | Port and Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------------|----------|------|---------------|---------|-----------------------|------------------------------------|
| Move instructions | | | | | | |
| FLD | r | 1 | MB | 1 | 1 | |
| FLD | m32/m64 | 2 | LD MB | 4 | 1 | |
| FLD | m80 | 2 | LD MB | 4 | 1 | |
| FBLD | m80 | | | 54 | 54 | |
| FST(P) | r | 1 | MB | 1 | 1 | |
| FST(P) | m32/m64 | 3 | MB SA ST | 5 | 1-2 | |
| FSTP | m80 | 3 | MB SA ST | 5 | 1-2 | |
| FBSTP | m80 | | | 125 | 125 | |
| FXCH | r | 1 | I2 | 0 | 1 | |
| FILD | m16 | | | 7 | | |
| FILD | m32 | | | 5 | | |
| FILD | m64 | | | 5 | | |
| FIST(T)(P) | m16 | | | 6 | | |
| FIST(T)(P) | m32 | | | 5 | | |
| FIST(T)(P) | m64 | | | 5 | | |
| FLDZ FLD1 | | 1 | MB | | 1 | |
| FLDPI FLDL2E etc. | | | | | 10 | |
| FCMOVcc | r | | | 2 | 2 | |
| FNSTSW | AX | | | | 5 | |
| FNSTSW | m16 | | | | 3 | |
| FLDCW | m16 | | | 13 | 13 | |
| FNSTCW | m16 | | | | 2 | |
| FINCSTP FDECSTP | | 1 | I2 | 0 | 1 | |
| FFREE(P) | | 1 | MB | | 1 | |
| FNSAVE | m | | | 321 | 321 | |
| FRSTOR | m | | | 195 | 195 | |
| Arithmetic instructions | | | | | | |
| FADD(P) FSUB(R)(P) | r/m | 1 | MB | 2 | 1 | Lower precision: Lat: 4, Thr: 2 |
| FMUL(P) | r/m | 1 | MA | 4 | 2 | |
| FDIV(R)(P) | r/m | | MA | 15-42 | 15-42 | |
| FABS | | 1 | MB | 1 | 1 | |
| FCHS | | 1 | MB | 1 | 1 | |
| FCOM(P) FUCOM | r/m | 1 | MB | | 1 | |
| FCOMPP FUCOMPP | | 1 | MB | | 1 | |
| FCOMI(P) FUCOMI(P) | r | 1 | MB | | 1 | |
| FIADD FISUB(R) | m | | MB | | 2 | |
| FIMUL | m | | | | 4 | |
| FIDIV(R) | m | | | | 42 | |
| FICOM(P) | m | 1 | | | 2 | |
| FTST | | 1 | MB | | 1 | |
| FXAM | | | | | 41 | |
| FPREM | | | | 151-171 | | |
| FPREM1 | | | | 106-155 | | |

| | | | | | | |
|--------------|--|---|-----|---------|-----|--|
| FRNDINT | | | | 29 | | |
| Math | | | | | | |
| FSCALE | | | | 39 | | |
| FXTRACT | | | | 36-57 | | |
| FSQRT | | | | 73 | | |
| FSIN FCOS | | | | 51-159 | | |
| FSINCOS | | | | 270-360 | | |
| F2XM1 | | | | 50-200 | | |
| FYL2X | | | | ~60 | | |
| FYL2XP1 | | | | ~170 | | |
| FPTAN | | | | 300-370 | | |
| FPATAN | | | | ~170 | | |
| Other | | | | | | |
| FNOP | | 1 | MB | | 1 | |
| WAIT | | 1 | 112 | 0 | 1/2 | |
| FNCLEX | | | | | 57 | |
| FNINIT | | | | | 85 | |

Integer MMX and XMM instructions

| | Operands | μops | Port and Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|-------------|------|---------------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| MOVD | r32/64,mm/x | 1 | | 3 | 1 | |
| MOVD | m32/64,mm/x | 1 | SA ST | 2-3 | 1-2 | |
| MOVD | mm/x,r32/64 | | | 4 | 1 | |
| MOVD | mm/x,m32/64 | 1 | LD | 2-3 | 1 | |
| MOVQ | mm/x, mm/x | 1 | MB | 1 | 1 | |
| MOVQ | mm/x,m64 | 1 | LD | 2-3 | 1 | |
| MOVQ | m64, mm/x | 1 | SA ST | 2-3 | 1-2 | |
| MOVDQA | xmm, xmm | 1 | MB | 1 | 1 | |
| MOVDQA | xmm, m128 | 1 | LD | 2-3 | 1 | |
| MOVDQA | m128, xmm | 1 | SA ST | 2-3 | 1-2 | |
| MOVDQU | m128, xmm | 1 | SA ST | 2-3 | 1-2 | |
| MOVDQU | xmm, m128 | 1 | LD | 2-3 | 1 | |
| LDDQU | xmm, m128 | 1 | LD | 2-3 | 1 | |
| MOVDQ2Q | mm, xmm | 1 | MB | 1 | 1 | |
| MOVQ2DQ | xmm,mm | 1 | MB | 1 | 1 | |
| MOVNTQ | m64,mm | 3 | | ~300 | 2 | |
| MOVNTDQ | m128,xmm | 3 | | ~300 | 2 | |
| PACKSSWB/DW | | | | | | |
| PACKUSWB | v,v | 1 | MB | 1 | 1 | |
| PUNPCKH/LBW/WD/DQ | v,v | 1 | MB | 1 | 1 | |
| PUNPCKH/LQDQ | v,v | 1 | MB | 1 | 1 | |
| PSHUFB | v,v | 1 | MB | 1 | 1 | |
| PSHUFW | mm,mm,i | 1 | MB | 1 | 1 | |
| PSHUFL/HW | x,x,i | 1 | MB | 1 | 1 | |
| PSHUFD | x,x,i | 1 | MB | 1 | 1 | |
| PALIGNR | x,x,i | 1 | MB | 1 | 1 | |
| MASKMOVQ | mm,mm | | | | 1-3 | |
| MASKMOVDQU | xmm,xmm | | | | 1-3 | |

VIA Nano 2000

| | | | | | | |
|--------------------------------|-------------|---|----|----|---|--|
| PMOVMSKB | r32,mm/x | | | 3 | 1 | |
| PEXTRW | r32 ,mm/x,i | | | 3 | 1 | |
| PINSRW | mm/x,r32,i | | | 9 | 9 | |
| Arithmetic instructions | | | | | | |
| PADD/SUB(U)(S)B/W/D | v,v | 1 | MB | 1 | 1 | |
| PADDQ PSUBQ | v,v | 1 | MB | 1 | 1 | |
| PHADD(S)W | | | | | | |
| PHSUB(S)W | v,v | 3 | MB | 3 | 3 | |
| PHADDD PHSUBD | v,v | 3 | MB | 3 | 3 | |
| PCMPEQ/GTB/W/D | v,v | 1 | MB | 1 | 1 | |
| PMULL/HW PMULHUW | v,v | 1 | MA | 3 | 1 | |
| PMULHSW | v,v | 1 | MA | 3 | 1 | |
| PMULUDQ | v,v | 1 | MA | 3 | 1 | |
| PMADDWD | v,v | | | 4 | 2 | |
| PMADDUBSW | v,v | | | 10 | 8 | |
| PSADBW | v,v | | MB | 2 | 1 | |
| PAVGB/W | v,v | 1 | MB | 1 | 1 | |
| PMIN/MAXUB | v,v | 1 | MB | 1 | 1 | |
| PMIN/MAXSW | v,v | 1 | MB | 1 | 1 | |
| PABSB PABSW PABSD | v,v | 1 | MB | 1 | 1 | |
| PSIGNB PSIGNW | | | | | | |
| PSIGND | v,v | 1 | MB | 1 | 1 | |
| Logic instructions | | | | | | |
| PAND(N) POR PXOR | v,v | 1 | MB | 1 | 1 | |
| PSLL/RL/RAW/D/Q | v,v | 1 | MB | 1 | 1 | |
| PSLL/RL/RAW/D/Q | v,i | 1 | MB | 1 | 1 | |
| PSLL/RLDQ | x,i | 1 | MB | 1 | 1 | |
| Other | | | | | | |
| EMMS | | 1 | MB | | 1 | |

Floating point XMM instructions

| | Operands | μops | Port and Unit | Latency | Reciprocal throughput | Remarks |
|--------------------------|----------|------|---------------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| MOVAPS/D | xmm,xmm | 1 | MB | 1 | 1 | |
| MOVAPS/D | xmm,m128 | 1 | LD | 2-3 | 1 | |
| MOVAPS/D | m128,xmm | 1 | SA ST | 2-3 | 1-2 | |
| MOVUPS/D | xmm,m128 | 1 | LD | 2-3 | 1 | |
| MOVUPS/D | m128,xmm | 1 | SA ST | 2-3 | 1-2 | |
| MOVSS/D | xmm,xmm | 1 | MB | 1 | 1 | |
| MOVSS/D | x,m32/64 | 1 | LD | 2-3 | 1 | |
| MOVSS/D | m32/64,x | 1 | SA ST | 2-3 | 1-2 | |
| MOVHPS/D | xmm,m64 | | | 6 | 1 | |
| MOVLPS/D | xmm,m64 | | | 6 | 1 | |
| MOVHPS/D | m64,xmm | | | 6 | 1-2 | |
| MOVLPS/D | m64,xmm | | | 2 | 1-2 | |
| MOVLHPS MOVHLPS | xmm,xmm | 1 | MB | 1 | 1 | |
| MOVMSKPS/D | r32,xmm | | | 3 | 1 | |
| MOVNTPS/D | m128,xmm | | | ~300 | 2.5 | |

VIA Nano 2000

| | | | | | |
|--------------------|-----------|---|--------|-------|-------|
| SHUFPS | xmm,xmm,i | 1 | MB | 1 | 1 |
| SHUFPD | xmm,xmm,i | 1 | MB | 1 | 1 |
| MOVDDUP | xmm,xmm | 1 | MB | 1 | 1 |
| MOVSH/LDUP | xmm,xmm | 1 | MB | 1 | 1 |
| UNPCKH/LPS | xmm,xmm | 1 | MB | 1 | 1 |
| UNPCKH/LPD | xmm,xmm | 1 | MB | 1 | 1 |
| Conversion | | | | | |
| CVTPD2PS | xmm,xmm | | | 3-4 | |
| CVTSD2SS | xmm,xmm | | | 15 | |
| CVTPS2PD | xmm,xmm | | | 3-4 | |
| CVTSS2SD | xmm,xmm | | | 15 | |
| CVTDQ2PS | xmm,xmm | | | 3 | |
| CVT(T) PS2DQ | xmm,xmm | | | 2 | |
| CVTDQ2PD | xmm,xmm | | | 4 | |
| CVT(T)PD2DQ | xmm,xmm | | | 3 | |
| CVTPI2PS | xmm,mm | | | 4 | |
| CVT(T)PS2PI | mm,xmm | | | 3 | |
| CVTPI2PD | xmm,mm | | | 4 | |
| CVT(T) PD2PI | mm,xmm | | | 3 | |
| CVTSI2SS | xmm,r32 | | | 5 | |
| CVT(T)SS2SI | r32,xmm | | | 4 | |
| CVTSI2SD | xmm,r32 | | | 5 | |
| CVT(T)SD2SI | r32,xmm | | | 4 | |
| Arithmetic | | | | | |
| ADDSS SUBSS | xmm,xmm | 1 | MBfadd | 2-3 | 1 |
| ADDSD SUBSD | xmm,xmm | 1 | MBfadd | 2-3 | 1 |
| ADDPS SUBPS | xmm,xmm | 1 | MBfadd | 2-3 | 1 |
| ADDPD SUBPD | xmm,xmm | 1 | MBfadd | 2-3 | 1 |
| ADDSUBPS | xmm,xmm | 1 | MBfadd | 2-3 | 1 |
| ADDSUBPD | xmm,xmm | 1 | MBfadd | 2-3 | 1 |
| HADDPS HSUBPS | xmm,xmm | | MBfadd | 5 | 3 |
| HADDPD HSUBPD | xmm,xmm | | MBfadd | 5 | 3 |
| MULSS | xmm,xmm | 1 | MA | 3 | 1 |
| MULSD | xmm,xmm | 1 | MA | 4 | 2 |
| MULPS | xmm,xmm | | MA | 3 | 1 |
| MULPD | xmm,xmm | | MA | 4 | 2 |
| DIVSS | xmm,xmm | | MA | 15-22 | 15-22 |
| DIVSD | xmm,xmm | | MA | 15-36 | 15-36 |
| DIVPS | xmm,xmm | | MA | 42-82 | 42-82 |
| DIVPD | xmm,xmm | | MA | 24-70 | 24-70 |
| RCPSS | xmm,xmm | | | 5 | 5 |
| RCPPS | xmm,xmm | | | 14 | 11 |
| CMPccSS/D | xmm,xmm | 1 | MBfadd | 2 | 1 |
| CMPccPS/D | xmm,xmm | 1 | MBfadd | 2 | 1 |
| COMISS/D UCOMISS/D | xmm,xmm | | | 3 | 1 |
| MAXSS/D MINSS/D | xmm,xmm | 1 | MBfadd | 2 | 1 |
| MAXPS/D MINPS/D | xmm,xmm | 1 | MBfadd | 2 | 1 |
| Math | | | | | |
| SQRTSS | xmm,xmm | | MA | 33 | 33 |

VIA Nano 2000

| | | | | | |
|--------------|---------|---|----|-----|-----|
| SQRTPS | xmm,xmm | | MA | 126 | 126 |
| SQRTSD | xmm,xmm | | MA | 62 | 62 |
| SQRTPD | xmm,xmm | | MA | 122 | 122 |
| RSQRTSS | xmm,xmm | | | 5 | 5 |
| RSQRTPS | xmm,xmm | | | 14 | 11 |
| Logic | | | | | |
| ANDPS/D | xmm,xmm | 1 | MB | 1 | 1 |
| ANDNPS/D | xmm,xmm | 1 | MB | 1 | 1 |
| ORPS/D | xmm,xmm | 1 | MB | 1 | 1 |
| XORPS/D | xmm,xmm | 1 | MB | 1 | 1 |
| Other | | | | | |
| LDMXCSR | m32 | | | 45 | 29 |
| STMXCSR | m32 | | | 13 | 13 |
| FXSAVE | m4096 | | | 208 | 208 |
| FXRSTOR | m4096 | | | 232 | 232 |

VIA-specific instructions

| Instruction | Conditions | Clock cycles, approximately |
|---------------|--------------------|------------------------------|
| XSTORE | Data available | 160-400 clock giving 8 bytes |
| XSTORE | No data available | 50-80 clock giving 0 bytes |
| REP XSTORE | Quality factor = 0 | 4800 clock per 8 bytes |
| REP XSTORE | Quality factor > 0 | 19200 clock per 8 bytes |
| REP XCRYPTECB | 128 bits key | 44 clock per 16 bytes |
| REP XCRYPTECB | 192 bits key | 46 clock per 16 bytes |
| REP XCRYPTECB | 256 bits key | 48 clock per 16 bytes |
| REP XCRYPTCBC | 128 bits key | 54 clock per 16 bytes |
| REP XCRYPTCBC | 192 bits key | 59 clock per 16 bytes |
| REP XCRYPTCBC | 256 bits key | 63 clock per 16 bytes |
| REP XCRYPTCTR | 128 bits key | 43 clock per 16 bytes |
| REP XCRYPTCTR | 192 bits key | 46 clock per 16 bytes |
| REP XCRYPTCTR | 256 bits key | 48 clock per 16 bytes |
| REP XCRYPTCFB | 128 bits key | 54 clock per 16 bytes |
| REP XCRYPTCFB | 192 bits key | 59 clock per 16 bytes |
| REP XCRYPTCFB | 256 bits key | 63 clock per 16 bytes |
| REP XCRYPTOFB | 128 bits key | 54 clock per 16 bytes |
| REP XCRYPTOFB | 192 bits key | 59 clock per 16 bytes |
| REP XCRYPTOFB | 256 bits key | 63 clock per 16 bytes |
| REP XSHA1 | | 3 clock per byte |
| REP XSHA256 | | 4 clock per byte |

VIA Nano 3000 series

List of instruction timings and μ ops breakdown

Explanation of column headings:

Operands: i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, mm/x = mmx or xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc.

μ ops: The number of micro-operations from the decoder or ROM. Note that the VIA Nano 3000 processor has no reliable performance monitor counter for μ ops. Therefore the number of μ ops cannot be determined except in simple cases.

Port: Tells which execution port or unit is used. Instructions that use the same port cannot execute simultaneously.

I1: Integer add, Boolean, shift, etc.

I2: Integer add, Boolean, move, jump.

I12: Can use either I1 or I2, whichever is vacant first.

MA: Multiply, divide and square root on all operand types.

MB: Various Integer and floating point SIMD operations.

MBfadd: Floating point addition subunit under MB.

SA: Memory store address.

ST: Memory store.

LD: Memory load.

Latency: This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay.

Note: There is an additional latency for moving data from one unit or subunit to another. A table of these latencies is given in manual 3: "The microarchitecture of Intel, AMD and VIA CPUs". These additional latencies are not included in the listings below where the source and destination operands are of the same type.

Reciprocal throughput: The average number of clock cycles per instruction for a series of independent instructions of the same kind in the same thread.

Integer instructions

| | Operands | μ ops | Port | Latency | Reciprocal throughput | Remarks |
|--------------------------|----------|-----------|--------|---------|-----------------------|-------------------------------|
| Move instructions | | | | | | |
| MOV | r,r | 1 | I2 | 1 | 1 | Latency 4 on pointer register |
| MOV | r,i | 1 | I12 | 1 | 1/2 | |
| MOV | r,m | 1 | LD | 2 | 1 | |
| MOV | m,r | 1 | SA, ST | 2 | 1.5 | |
| MOV | m,i | 1 | SA, ST | | 1.5 | |
| MOV | r,sr | | I12 | | 1/2 | |
| MOV | m,sr | | | | 1.5 | |
| MOV | sr,r | | | 20 | 20 | |
| MOV | sr,m | | | 20 | 20 | |
| MOVNTI | m,r | | SA, ST | 2 | 1.5 | |

Nano 3000

| | | | | | | |
|--------------------------------|---------|----|--------------|----|------|------------------------------|
| MOVSX MOVZX | r,r | 1 | I12 | 1 | 1/2 | |
| MOVSD | r64,r32 | 1 | | 1 | 1 | |
| MOVSX MOVSD | r,m | 2 | LD, I12 | 3 | 1 | |
| MOVZX | r,m | 1 | LD | 2 | 1 | |
| CMOVcc | r,r | 1 | I12 | 1 | 1/2 | |
| CMOVcc | r,m | | LD, I12 | 5 | 1 | |
| XCHG | r,r | 3 | I12 | 3 | 1.5 | |
| XCHG | r,m | | | 18 | 18 | Implicit lock |
| XLAT | m | 3 | LD, I1 | 6 | 2 | |
| PUSH | r | 1 | SA, ST | | 1-2 | |
| PUSH | i | 1 | SA, ST | | 1-2 | |
| PUSH | m | | LD, SA, ST | | 2 | |
| PUSH | sr | | | | 6 | |
| PUSHF(D/Q) | | 3 | | 2 | 2 | |
| PUSHA(D) | | 9 | | | 15 | Not in x64 mode |
| POP | r | 2 | LD | | 1.25 | |
| POP | (E/R)SP | | | | 4 | |
| POP | m | 3 | | | 2 | |
| POP | sr | | | | 11 | |
| POPF(D/Q) | | 3 | | | 1 | |
| POPA(D) | | 16 | | | 12 | Not in x64 mode |
| LAHF | | 1 | I1 | 1 | 1 | |
| SAHF | | 1 | I1 | 1 | 1 | |
| SALC | | 2 | | 10 | 6 | Not in x64 mode |
| LEA | r,m | 1 | SA | 1 | 1 | Extra latency to other ports |
| BSWAP | r | 1 | I2 | 1 | 1 | |
| LDS LES LFS LGS LSS | m | 12 | | 28 | 28 | |
| PREFETCHNTA | m | 1 | LD | | 1 | |
| PREFETCHT0/1/2 | m | 1 | LD | | 1 | |
| LFENCE MFENCE SFENCE | | | | | 15 | |
| Arithmetic instructions | | | | | | |
| ADD SUB | r,r/i | 1 | I12 | 1 | 1/2 | |
| ADD SUB | r,m | 2 | LD I12 | | 1 | |
| ADD SUB | m,r/i | 3 | LD I12 SA ST | 5 | 2 | |
| ADC SBB | r,r/i | 1 | I1 | 1 | 1 | |
| ADC SBB | r,m | 2 | LD I1 | | 1 | |
| ADC SBB | m,r/i | 3 | LD I1 SA ST | 5 | 2 | |
| CMP | r,r/i | 1 | I12 | 1 | 1/2 | |
| CMP | m,r/i | 2 | LD I12 | | 1 | |
| INC DEC NEG NOT | r | 1 | I12 | 1 | 1/2 | |
| INC DEC NEG NOT | m | 3 | LD I12 SA ST | 5 | | |
| AAA | | 12 | | | 37 | Not in x64 mode |
| AAS | | 12 | | | 22 | Not in x64 mode |
| DAA | | 14 | | | 22 | Not in x64 mode |
| DAS | | 14 | | | 24 | Not in x64 mode |
| AAD | | 7 | | | 24 | Not in x64 mode |
| AAM | | 13 | | | 31 | Not in x64 mode |
| MUL IMUL | r8 | 1 | I2 | 2 | | |
| MUL IMUL | r16 | 3 | I2 | 3 | | |
| MUL IMUL | r32 | 3 | I2 | 3 | | |

Nano 3000

| | | | | | | |
|--------------------------------------|--------------|------|--------------|---------|---------|---------------------------------|
| MUL IMUL | r64 | 3 | MA | 8 | 8 | Extra latency to other ports |
| IMUL | r16,r16 | 1 | I2 | 2 | 1 | |
| IMUL | r32,r32 | 1 | I2 | 2 | 1 | |
| IMUL | r64,r64 | 1 | MA | 5 | 2 | Extra latency to other ports |
| IMUL | r16,r16,i | 1 | I2 | 2 | 1 | |
| IMUL | r32,r32,i | 1 | I2 | 2 | 1 | |
| IMUL | r64,r64,i | 1 | MA | 5 | 2 | Extra latency to other ports |
| DIV | r8 | | MA | 22-24 | 22-24 | |
| DIV | r16 | | MA | 24-28 | 24-28 | |
| DIV | r32 | | MA | 22-30 | 22-30 | |
| DIV | r64 | | MA | 145-162 | 145-162 | |
| IDIV | r8 | | MA | 21-24 | 21-24 | |
| IDIV | r16 | | MA | 24-28 | 24-28 | |
| IDIV | r32 | | MA | 18-26 | 18-26 | |
| IDIV | r64 | | MA | 182-200 | 182-200 | |
| CBW CWDE CDQE | | 1 | I2 | 1 | 1 | |
| CWD CDQ CQO | | 1 | I2 | 1 | 1 | |
| Logic instructions | | | | | | |
| AND OR XOR | r,r/i | 1 | I12 | 1 | 1/2 | |
| AND OR XOR | r,m | 2 | LD I12 | | 1 | |
| AND OR XOR | m,r/i | 3 | LD I12 SA ST | 5 | 2 | |
| TEST | r,r/i | 1 | I12 | 1 | 1/2 | |
| TEST | m,r/i | 2 | LD I12 | | 1 | |
| SHR SHL SAR | r,i/cl | 1 | I12 | 1 | 1/2 | |
| ROR ROL | r,i/cl | 1 | I1 | 1 | 1 | |
| RCR RCL | r,1 | 1 | I1 | 1 | 1 | |
| RCR RCL | r,i/cl | 5+2n | I1 | 28+3n | 28+3n | |
| SHLD SHRD | r16,r16,i/cl | 2 | I1 | 2 | 2 | |
| SHLD SHRD | r32,r32,i/cl | 2 | I1 | 2 | 2 | |
| SHLD | r64,r64,i/cl | 16 | I1 | 32 | 32 | |
| SHRD | r64,r64,i/cl | 23 | I1 | 42 | 42 | |
| BT | r,r/i | 1 | I1 | 1 | 1 | |
| BT | m,r | 6 | I1 | | 8 | |
| BT | m,i | 2 | I1 | | 1 | |
| BTR BTS BTC | r,r/i | 2 | I1 | 2 | 2 | |
| BTR BTS BTC | m,r | 8 | I1 | 10 | 10 | |
| BTR BTS BTC | m,i | 5 | I1 | 8 | 8 | |
| BSF BSR | r,r | 2 | I1 | 2 | 2 | |
| SETcc | r8 | 1 | I1 | 1 | 1 | |
| SETcc | m | 2 | | | 2 | |
| CLC STC CMC | | 3 | I1 | 3 | 3 | |
| CLD STD | | 3 | I1 | 3 | 3 | |
| Control transfer instructions | | | | | | |
| JMP | short/near | 1 | I2 | 3 | 3 | 8 if >2 jumps in 16 bytes block |
| JMP | far | 14 | | | 50 | Not in x64 mode |
| JMP | r | 2 | I2 | 3 | 3 | 8 if >2 jumps in 16 bytes block |
| JMP | m(near) | 2 | | 3 | 3 | do. |

Nano 3000

| | | | | | | |
|----------------------------|------------|-----|-----|--------|---|---|
| JMP | m(far) | 17 | | | 42 | 1 if not jumping. 3 if jumping. 8 if >2 jumps in 16 bytes block |
| Conditional jump | short/near | 1 | 12 | 1-3-8 | 1-3-8 | |
| J(E/R)CXZ | short | 2 | | 1-3-8 | 1-3-8 | |
| LOOP | short | 2 | | 1-3-8 | 1-3-8 | |
| LOOP(N)E | short | 5 | | 24 | 24 | |
| CALL | near | 2 | | 3 | 3 | 8 if >2 jumps in 16 bytes block |
| CALL | far | 17 | | | 58 | Not in x64 mode |
| CALL | r | 2 | | 3 | 3 | 8 if >2 jumps in 16 bytes block |
| CALL | m(near) | 3 | | 4 | 3 | do. |
| CALL | m(far) | 19 | | | 54 | |
| RETN | | 3 | | 3 | 3 | 8 if >2 jumps in 16 bytes block |
| RETN | i | 4 | | 3 | 3 | do. |
| RETF | | 20 | | | 49 | |
| RETF | i | 20 | | | 49 | |
| BOUND | r,m | 9 | | | 13 | Not in x64 mode |
| INTO | | 3 | | | 7 | Not in x64 mode |
| String instructions | | | | | | |
| LODSB/W/D/Q | | 2 | | | 1 | |
| REP LODSB/W/D/Q | | 3n | | | 3n+27 | |
| STOSB/W/D/Q | | 1 | | | 1-2 | |
| REP STOSB/W/D/Q | | | | | Small: n+40, Big: 6-7 bytes/ clk | |
| MOVSB/W/D/Q | | 3 | | | 2 | |
| REP MOVSB/W/D/Q | | | | | Small: 2n+20, Big: 6-7 bytes/clk | |
| SCASB/W/D/Q | | 3 | | | 1 | |
| REP SCASB | | | | | 2.4n | |
| REP SCASW/D/Q | | | | | Small: 2n+31, Big: 5 bytes/clk | |
| CMPSB/W/D/Q | | 5 | | | 6 | |
| REP CMPSB/W/D/Q | | | | | 2.2n+30 | |
| Other | | | | | | |
| NOP (90) | | 0-1 | 112 | 0 | 1/2 | Sometimes fused |
| long NOP (0F 1F) | | 0-1 | 112 | 0 | 1/2 | |
| PAUSE | | 2 | | | 6 | |
| ENTER | a,0 | 10 | | | 21 | |
| ENTER | a,b | | | | 52+5b | |
| LEAVE | | 3 | | 2 | 2 | |
| CPUID | | | | 55-146 | | |
| RDTSC | | | | | 37 | |
| RDPMSR | | | | | 40 | |
| | | | | | | |

Floating point x87 instructions

| | Operands | μops | Port | Latency | Reciprocal through- put | Remarks |
|--------------------------------|----------|------|----------|---------|-------------------------------|---------|
| Move instructions | | | | | | |
| FLD | r | 1 | MB | 1 | 1 | |
| FLD | m32/m64 | 2 | LD MB | 4 | 1 | |
| FLD | m80 | 2 | LD MB | 4 | 1 | |
| FBLD | m80 | 36 | | 54 | 54 | |
| FST(P) | r | 1 | MB | 1 | 1 | |
| FST(P) | m32/m64 | 3 | MB SA ST | 5 | 1-2 | |
| FSTP | m80 | 3 | MB SA ST | 5 | 1-2 | |
| FBSTP | m80 | 80 | | 125 | 125 | |
| FXCH | r | 1 | I2 | 0 | 1 | |
| FILD | m16 | 3 | | 7 | | |
| FILD | m32 | 2 | | 5 | | |
| FILD | m64 | 2 | | 5 | | |
| FIST(T)(P) | m16 | 3 | | 6 | | |
| FIST(T)(P) | m32 | 3 | | 5 | | |
| FIST(T)(P) | m64 | 3 | | 5 | | |
| FLDZ FLD1 | | 1 | MB | | 1 | |
| FLDPI FLDL2E etc. | | 3 | | | 10 | |
| FCMOVcc | r | 1 | MB | 2 | 2 | |
| FNSTSW | AX | 1 | | | 1 | |
| FNSTSW | m16 | 3 | | | 2 | |
| FLDCW | m16 | 5 | | | 8 | |
| FNSTCW | m16 | 3 | | | 2 | |
| FINCSTP FDECSTP | | 1 | I2 | 0 | 1 | |
| FFREE(P) | | 1 | MB | | 1 | |
| FNSAVE | m | 122 | | 319 | 319 | |
| FRSTOR | m | 115 | | 196 | 196 | |
| Arithmetic instructions | | | | | | |
| FADD(P) FSUB(R)(P) | r/m | 1 | MB | 2 | 1 | |
| FMUL(P) | r/m | 1 | MA | 4 | 2 | |
| FDIV(R)(P) | r/m | | MA | 14-23 | 14-23 | |
| FABS | | 1 | MB | 1 | 1 | |
| FCHS | | 1 | MB | 1 | 1 | |
| FCOM(P) FUCOM | r/m | 1 | MB | | 1 | |
| FCOMPP FUCOMPP | | 1 | MB | | 1 | |
| FCOMI(P) FUCOMI(P) | r | 1 | MB | 2 | 1 | |
| FIADD FISUB(R) | m | 3 | MB | | 2 | |
| FIMUL | m | 3 | | | 4 | |
| FIDIV(R) | m | 3 | | | 16 | |
| FICOM(P) | m | 3 | | | 2 | |
| FTST | | 1 | MB | 2 | 1 | |
| FXAM | | 15 | | 38 | 38 | |
| FPREM | | | | ~130 | | |
| FPREM1 | | | | ~130 | | |
| FRNDINT | | 11 | | 27 | | |

Nano 3000

| | | | | | | |
|--------------|--|----|-----|---------|-----|--|
| Math | | | | | | |
| FSCALE | | 22 | | 37 | | |
| EXTRACT | | 13 | | 57 | | |
| FSQRT | | | | 73 | | |
| FSIN FCOS | | | | ~150 | | |
| FSINCOS | | | | 270-360 | | |
| F2XM1 | | | | 50-200 | | |
| FYL2X | | | | ~50 | | |
| FYL2XP1 | | | | ~50 | | |
| FPTAN | | | | 300-370 | | |
| FPATAN | | | | ~180 | | |
| Other | | | | | | |
| FNOP | | 1 | MB | | 1 | |
| WAIT | | 1 | I12 | 0 | 1/2 | |
| FNCLEX | | | | | 59 | |
| FNINIT | | | | | 84 | |

Less at lower precision

Integer MMX and XMM instructions

| | Operands | μops | Port | Latency | Reciprocal throughput | Remarks |
|--------------------------|-----------|------|-------|---------|-----------------------|---------|
| Move instructions | | | | | | |
| MOVD | r,mm/x | 1 | MB | 3 | 1 | |
| MOVD | m,mm/x | 1 | SA ST | 2 | 1-2 | |
| MOVD | mm/x,r | 1 | I2 | 4 | 1 | |
| MOVD | mm/x,m | 1 | LD | 2 | 1 | |
| MOVQ | v,v | 1 | MB | 1 | 1 | |
| MOVQ | mm/x,m64 | 1 | LD | 2 | 1 | |
| MOVQ | m64, mm/x | 1 | SA ST | 2 | 1-2 | |
| MOVDQA | x,x | 1 | MB | 1 | 1 | |
| MOVDQA | x, m128 | 1 | LD | 2 | 1 | |
| MOVDQA | m128, x | 1 | SA ST | 2 | 1-2 | |
| MOVDQU | m128, x | 1 | SA ST | 2 | 1-2 | |
| MOVDQU | x, m128 | 1 | LD | 2 | 1 | |
| LDDQU | x, m128 | 1 | LD | 2 | 1 | |
| MOVDQ2Q | mm, x | 1 | MB | 1 | 1 | |
| MOVQ2DQ | x,mm | 1 | MB | 1 | 1 | |
| MOVNTQ | m64,mm | 2 | | ~360 | 2 | |
| MOVNTDQ | m128,x | 2 | | ~360 | 2 | |
| MOVNTDQA | x,m128 | 1 | | 2 | 1 | |
| PACKSSWB/DW | | | | | | |
| PACKUSWB | v,v | 1 | MB | 1 | 1 | |
| PACKUSDW | x,x | 1 | MB | 1 | 1 | |
| PUNPCKH/LBW/WD/DQ | v,v | 1 | MB | 1 | 1 | |
| PUNPCKH/LQDQ | v,v | 1 | MB | 1 | 1 | |
| PSHUFB | v,v | 1 | MB | 1 | 1 | |
| PSHUFW | mm,mm,i | 1 | MB | 1 | 1 | |
| PSHUFL/HW | x,x,i | 1 | MB | 1 | 1 | |
| PSHUFD | x,x,i | 1 | MB | 1 | 1 | |
| PBLENDVB | x,x,xmm0 | 1 | MB | 2 | 2 | |

Nano 3000

| | | | | | |
|--------------------------------|-------------|---|----|----|-----|
| PBLENDW | x,x,i | 1 | MB | 1 | 1 |
| PALIGNR | x,x,i | 1 | MB | 1 | 1 |
| MASKMOVQ | mm,mm | | | | 1-2 |
| MASKMOVDQU | x,x | | | | 1-2 |
| PMOVMSKB | r32,mm/x | | | 3 | 1 |
| PEXTRW | r32 ,mm/x,i | 1 | MB | 3 | 1 |
| PEXTRB/D/Q | r32/64,x,i | 1 | MB | 3 | 1 |
| PINSRW | mm/x,r32,i | 2 | MB | 5 | 1 |
| PINSRB/D/Q | x,r32/64,i | 2 | MB | 5 | 1 |
| PMOVSX/ZXBW/BD/ BQ/WD/WQ/DQ | x,x | 1 | MB | 1 | 1 |
| Arithmetic instructions | | | | | |
| PADD/SUB(U)(S)B/W/D | v,v | 1 | MB | 1 | 1 |
| PADDQ PSUBQ | v,v | 1 | MB | 1 | 1 |
| PHADD(S)W | | | | | |
| PHSUB(S)W | v,v | 3 | MB | 3 | 3 |
| PHADDD PHSUBD | v,v | 3 | MB | 3 | 3 |
| PCMPEQ/GTB/W/D | v,v | 1 | MB | 1 | 1 |
| PCMPEQQ | x,x | 1 | MB | 1 | 1 |
| PMULL/HW PMULHUW | v,v | 1 | MA | 3 | 1 |
| PMULHRW | v,v | 1 | MA | 3 | 1 |
| PMULLD | x,x | 1 | MA | 3 | 1 |
| PMULUDQ | v,v | 1 | MA | 3 | 1 |
| PMULDQ | x,x | 1 | MA | 3 | 1 |
| PMADDWD | v,v | 1 | MA | 4 | 2 |
| PMADDUBSW | v,v | 7 | | 10 | 8 |
| PSADBW | v,v | 1 | MB | 2 | 1 |
| MPSADBW | x,x,i | 1 | MB | 2 | 1 |
| PAVGB/W | v,v | 1 | MB | 1 | 1 |
| PMIN/MAXSW | v,v | 1 | MB | 1 | 1 |
| PMIN/MAXUB | v,v | 1 | MB | 1 | 1 |
| PMIN/MAXSB/D | x,x | 1 | MB | 1 | 1 |
| PMIN/MAXUW/D | x,x | 1 | MB | 1 | 1 |
| PHMINPOSUW | x,x | 1 | MB | 2 | 1 |
| PABSB PABSW PABSD | v,v | 1 | MB | 1 | 1 |
| PSIGNB PSIGNW | | | | | |
| PSIGND | v,v | 1 | MB | 1 | 1 |
| Logic instructions | | | | | |
| PAND(N) POR PXOR | v,v | 1 | MB | 1 | 1 |
| PTEST | v,v | 1 | MB | 3 | 1 |
| PSLL/RL/RAW/D/Q | v,v | 1 | MB | 1 | 1 |
| PSLL/RL/RAW/D/Q | (x)xmm,i | 1 | MB | 1 | 1 |
| PSLL/RDQ | x,i | 1 | MB | 1 | 1 |
| Other | | | | | |
| EMMS | | 1 | MB | | 1 |

Floating point XMM instructions

| | Operands | μops | Port | Latency | Reciprocal through- put | Remarks |
|--------------------------|----------|------|--------|---------|-------------------------------|---------|
| Move instructions | | | | | | |
| MOVAPS/D | x,x | 1 | MB | 1 | 1 | |
| MOVAPS/D | x,m128 | 1 | LD | 2 | 1 | |
| MOVAPS/D | m128,x | 1 | SA ST | 2 | 1 | |
| MOVUPS/D | x,m128 | 1 | LD | 2 | 1 | |
| MOVUPS/D | m128,x | 2 | SA ST | 2 | 1 | |
| MOVSS/D | x,x | 1 | MB | 1 | 1 | |
| MOVSS/D | x,m32/64 | 1 | LD | 2-3 | 1 | |
| MOVSS/D | m32/64,x | 2 | SA ST | 2-3 | 1-2 | |
| MOVHPS/D | x,m64 | 2 | | 6 | 1 | |
| MOVLPS/D | x,m64 | 2 | | 6 | 1 | |
| MOVHPS/D | m64,x | 3 | | 6 | 1-2 | |
| MOVLPS/D | m64,x | 1 | | 2 | 1-2 | |
| MOVLHPS MOVHLPS | x,x | 1 | | 1 | 1 | |
| MOVMSKPS/D | r32,x | | | 3 | 1 | |
| MOVNTPS/D | m128,x | 2 | | ~360 | 1-2 | |
| SHUFPS | x,x,i | 1 | MB | 1 | 1 | |
| SHUFPD | x,x,i | 1 | MB | 1 | 1 | |
| MOVDDUP | x,x | 1 | MB | 1 | 1 | |
| MOVSH/LDUP | x,x | 1 | MB | 1 | 1 | |
| UNPCKH/LPS | x,x | 1 | MB | 1 | 1 | |
| UNPCKH/LPD | x,x | 1 | MB | 1 | 1 | |
| Conversion | | | | | | |
| CVTPD2PS | x,x | 2 | | 5 | 2 | |
| CVTSD2SS | x,x | 1 | | 2 | | |
| CVTPS2PD | x,x | 2 | | 5 | 1 | |
| CVTSS2SD | x,x | 1 | | 2 | | |
| CVTDQ2PS | x,x | 1 | MB | 3 | 1 | |
| CVT(T) PS2DQ | x,x | 1 | | 2 | 1 | |
| CVTDQ2PD | x,x | 2 | | 5 | 1 | |
| CVT(T)PD2DQ | x,x | | | 4 | 2 | |
| CVTPI2PS | x,mm | 2 | | 5 | 2 | |
| CVT(T)PS2PI | mm,x | 1 | | 4 | 1 | |
| CVTPI2PD | x,mm | 2 | | 4 | 1 | |
| CVT(T) PD2PI | mm,x | 2 | | 4 | 2 | |
| CVTSI2SS | x,r32 | 2 | | 5 | | |
| CVT(T)SS2SI | r32,x | 1 | | 4 | 1 | |
| CVTSI2SD | x,r32 | 2 | | 5 | | |
| CVT(T)SD2SI | r32,x | 1 | | 4 | 1 | |
| Arithmetic | | | | | | |
| ADDSS SUBSS | x,x | 1 | MBfadd | 2 | 1 | |
| ADDSD SUBSD | x,x | 1 | MBfadd | 2 | 1 | |
| ADDPs SUBPS | x,x | 1 | MBfadd | 2 | 1 | |
| ADDPD SUBPD | x,x | 1 | MBfadd | 2 | 1 | |
| ADDSubPS | x,x | 1 | MBfadd | 2 | 1 | |
| ADDSubPD | x,x | 1 | MBfadd | 2 | 1 | |
| HADDPs HSubPS | x,x | 3 | MBfadd | 5 | 3 | |
| HADDPD HSubPD | x,x | 3 | MBfadd | 5 | 3 | |

Nano 3000

| | | | | | |
|--------------------|-------|---|--------|-------|-------|
| MULSS | x,x | 1 | MA | 3 | 1 |
| MULSD | x,x | 1 | MA | 4 | 2 |
| MULPS | x,x | 1 | MA | 3 | 1 |
| MULPD | x,x | 1 | MA | 4 | 2 |
| DIVSS | x,x | 1 | MA | 13 | 13 |
| DIVSD | x,x | 1 | MA | 13-20 | 13-20 |
| DIVPS | x,x | 1 | MA | 24 | 24 |
| DIVPD | x,x | 1 | MA | 21-38 | 21-38 |
| RCPSS | x,x | 1 | MA | 5 | 5 |
| RCPPS | x,x | 3 | MA | 14 | 11 |
| CMPccSS/D | x,x | 1 | MBfadd | 2 | 1 |
| CMPccPS/D | x,x | 1 | MBfadd | 2 | 1 |
| COMISS/D UCOMISS/D | x,x | 1 | MBfadd | 3 | 1 |
| MAXSS/D MINSS/D | x,x | 1 | MBfadd | 2 | 1 |
| MAXPS/D MINPS/D | x,x | 1 | MBfadd | 2 | 1 |
| Math | | | | | |
| SQRTSS | x,x | 1 | MA | 33 | 33 |
| SQRTPS | x,x | 1 | MA | 64 | 64 |
| SQRTSD | x,x | 1 | MA | 62 | 62 |
| SQRTPD | x,x | 1 | MA | 122 | 122 |
| RSQRTSS | x,x | 1 | | 5 | 5 |
| RSQRTPS | x,x | 3 | | 14 | 11 |
| Logic | | | | | |
| ANDPS/D | x,x | 1 | MB | 1 | 1 |
| ANDNPS/D | x,x | 1 | MB | 1 | 1 |
| ORPS/D | x,x | 1 | MB | 1 | 1 |
| XORPS/D | x,x | 1 | MB | 1 | 1 |
| Other | | | | | |
| LDMXCSR | m32 | | | | 31 |
| STMXCSR | m32 | | | | 13 |
| FXSAVE | m4096 | | | | 97 |
| FXRSTOR | m4096 | | | | 201 |

VIA-specific instructions

| Instruction | Conditions | Clock cycles, approximately |
|---------------|--------------------|------------------------------|
| XSTORE | Data available | 160-400 clock giving 8 bytes |
| XSTORE | No data available | 50-80 clock giving 0 bytes |
| REP XSTORE | Quality factor = 0 | 1300 clock per 8 bytes |
| REP XSTORE | Quality factor > 0 | 5455 clock per 8 bytes |
| REP XCRYPTECB | 128 bits key | 15 clock per 16 bytes |
| REP XCRYPTECB | 192 bits key | 17 clock per 16 bytes |
| REP XCRYPTECB | 256 bits key | 18 clock per 16 bytes |
| REP XCRYPTCBC | 128 bits key | 29 clock per 16 bytes |
| REP XCRYPTCBC | 192 bits key | 33 clock per 16 bytes |
| REP XCRYPTCBC | 256 bits key | 37 clock per 16 bytes |
| REP XCRYPTCTR | 128 bits key | 23 clock per 16 bytes |
| REP XCRYPTCTR | 192 bits key | 26 clock per 16 bytes |
| REP XCRYPTCTR | 256 bits key | 27 clock per 16 bytes |
| REP XCRYPTCFB | 128 bits key | 29 clock per 16 bytes |

Nano 3000

| | | |
|---------------|--------------|-----------------------|
| REP XCRYPTCFB | 192 bits key | 33 clock per 16 bytes |
| REP XCRYPTCFB | 256 bits key | 37 clock per 16 bytes |
| REP XCRYPTOFB | 128 bits key | 29 clock per 16 bytes |
| REP XCRYPTOFB | 192 bits key | 33 clock per 16 bytes |
| REP XCRYPTOFB | 256 bits key | 37 clock per 16 bytes |
| REP XSHA1 | | 5 clock per byte |
| REP XSHA256 | | 5 clock per byte |