- Avoid resource conflicts. Interleaving instructions so that they don’t compete for the same port or execution unit can increase throughput. For example, alternate PADDQ and PMULUDQ (each has a throughput of one issue per two clock cycles). When interleaved, they can achieve an effective throughput of one instruction per cycle because they use the same port but different execution units. Selecting instructions with fast throughput also helps to preserve issue port bandwidth, hide latency and allows for higher software performance.
- Minimize the latency of dependency chains that are on the critical path. For example, an operation to shift left by two bits executes faster when encoded as two adds than when it is encoded as a shift. If latency is not an issue, the shift results in a denser byte encoding.

In addition to the general and specific rules, coding guidelines and the instruction data provided in this manual, you can take advantage of the software performance analysis and tuning toolset available at http://developer.intel.com/software/products/index.htm. The tools include the Intel VTune Performance Analyzer, with its performance-monitoring capabilities.

D.2 DEFINITIONS

The data is listed in several tables. The tables contain the following:

- **Instruction Name** — The assembly mnemonic of each instruction.
- **Latency** — The number of clock cycles that are required for the execution core to complete the execution of all of the μops that form an instruction.
- **Throughput** — The number of clock cycles required to wait before the issue ports are free to accept the same instruction again. For many instructions, the throughput of an instruction can be significantly less than its latency.
- The case of RDRAND instruction latency and throughput is an exception to the definitions above, because the hardware facility that executes the RDRAND instruction resides in the uncore and is shared by all processor cores and logical processors in a physical package. The software observable latency and throughput using the sequence of “rdrand followby jnc” in a single-thread scenario can be as low as ~100 cycles. In third generation Intel Core processors based on Ivy Bridge microarchitecture, the total bandwidth to deliver random numbers via RDRAND by the uncore is about 500 MBytes/sec. Within the same processor core microarchitecture and different uncore implementations, RDRAND latency/throughput can vary across Intel Core and Intel Xeon processors.

D.3 LATENCY AND THROUGHPUT

This section presents the latency and throughput information for commonly-used instructions including: MMX technology, Streaming SIMD Extensions, subsequent generations of SIMD instruction extensions, and most of the frequently used general-purpose integer and x87 floating-point instructions.

Due to the complexity of dynamic execution and out-of-order nature of the execution core, the instruction latency data may not be sufficient to accurately predict realistic performance of actual code sequences based on adding instruction latency data.

- Instruction latency data is useful when tuning a dependency chain. However, dependency chains limit the out-of-order core’s ability to execute micro-ops in parallel. Instruction throughput data are useful when tuning parallel code unencumbered by dependency chains.
- Numeric data in the tables is:
  - Approximate and subject to change in future implementations of the microarchitecture.
  - Not meant to be used as reference for instruction-level performance benchmarks. Comparison of instruction-level performance of microprocessors that are based on different microarchitectures is a complex subject and requires information that is beyond the scope of this manual.

Comparisons of latency and throughput data between different microarchitectures can be misleading.

Appendix D.3.1 provides latency and throughput data for the register-to-register instruction type. Appendix D.3.3 discusses how to adjust latency and throughput specifications for the register-to-memory and memory-to-register instructions.

In some cases, the latency or throughput figures given are just one half of a clock. This occurs only for the double-speed ALUs.

D.3.1 Latency and Throughput with Register Operands

Instruction latency and throughput data are presented in Table D-4 through Table D-18. Tables include AESNI, SSE4.2, SSE4.1, Supplemental Streaming SIMD Extension 3, Streaming SIMD Extension 3, Streaming SIMD Extension 2, Streaming SIMD Extension, MMX technology and most common Intel 64 and IA-32 instructions. Instruction latency and throughput for different processor microarchitectures are in separate columns.

Processor instruction timing data is implementation specific; it can vary between model encodings within the same family encoding (e.g. model = 3 vs model < 2). Separate sets of instruction latency and throughput are shown in the columns for CPUID signature 0xF2n and 0xF3n. The column represented by 0xF3n also applies to Intel processors with CPUID signature 0xF4n and 0xF6n. The notation 0xF2n represents the hex value of the lower 12 bits of the EAX register reported by CPUID instruction with input value of EAX = 1; 'F' indicates the family encoding value is 15, '2' indicates the model encoding is 2, 'n' indicates it applies to any value in the stepping encoding.

Intel Core Solo and Intel Core Duo processors are represented by 06_0EH. Processors bases on 65 nm Intel Core microarchitecture are represented by 06_0FH. Processors based on Enhanced Intel Core microarchitecture are represented by 06_17H and 06_1DH. CPUID family/Model signatures of processors based on Nehalem microarchitecture are represented by 06_1AH, 06_1EH, 06_1FH, and 06_2EH. Processors based on Westmere microarchitecture are represented by 06_25H, 06_2CH and 06_2FH. Processors based on Sandy Bridge microarchitecture are represented by 06_2AH, 06_2DH. Processors based on Ivy Bridge microarchitecture are represented by 06_3AH, 06_3EH. Processors based on Haswell microarchitecture are represented by 06_3CH, 06_45H and 06_46H.

Table D-1. CPUID Signature Values of Of Recent Intel Microarchitectures

DisplayFamily_DisplayModel	Recent Intel Microarchitectures
06_4EH, 06_5EH	Skylake microarchitecture
06_3DH, 06_47H, 06_56H	Broadwell microarchitecture
06_3CH, 06_45H, 06_46H, 06_3FH	Haswell microarchitecture
06_3AH, 06_3EH	Ivy Bridge microarchitecture
06_2AH, 06_2DH	Sandy Bridge microarchitecture
06_25H, 06_2CH, 06_2FH	Intel microarchitecture Westmere
06_1AH, 06_1EH, 06_1FH, 06_2EH	Intel microarchitecture Nehalem
06_17H, 06_1DH	Enhanced Intel Core microarchitecture
06_0FH	Intel Core microarchitecture

Instruction latency varies by microarchitectures. Table D-2 lists SIMD extensions introduction in recent microarchitectures. Each microarchitecture may be associated with more than one signature value given by the CPUID's "display_family" and "display_model". Not all instruction set extensions are enabled in all processors associated with a particular family/model designation. To determine whether a given instruction set extension is supported, software must use the appropriate CPUID feature flag as described in *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A*.

.

Table D-2. Instruction Extensions Introduction by Microarchitectures (CPUID Signature)

SIMD Instruction Extensions	DisplayFamily_DisplayModel							
	06_4EH, 06_5EH	06_3DH, 06_47H, 06_56H	06_3CH, 06_45H, 06_46H, 06_3FH	06_3AH, 06_3EH	06_2AH, 06_2DH	06_25H, 06_2CH, 06_2FH	06_1AH, 06_1EH, 06_1FH, 06_2EH	06_17H, 06_1DH
CLFLUSHOPT	Yes	No	No	No	No	No	No	No
ADX, RDSEED	Yes	Yes	No	No	No	No	No	No
AVX2, FMA, BMI1, BMI2	Yes	Yes	Yes	No	No	No	No	No
F16C, RDRAND, RWFSGSBASE	Yes	Yes	Yes	Yes	No	No	No	No
AVX	Yes	Yes	Yes	Yes	Yes	No	No	No
AESNI, PCLMULQDQ	Yes	Yes	Yes	Yes	Yes	Yes	No	No
SSE4.2, POPCNT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
SSE4.1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SSSE3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SSE3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SSE2	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SSE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
MMX	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table D-3. BMI1, BMI2 and General Purpose Instructions

Instruction	Latency [1]		Throughput	
DisplayFamily_DisplayModel	06_4E, 06_5E	06_3D, 06_47, 06_56	06_4E, 06_5E	06_3D, 06_47, 06_56
ADCX	1	1	1	1
ADOX	1	1	1	1
RESEED	Similar to RDRAND	Similar to RDRAND	Similar to RDRAND	Similar to RDRAND

Table D-4. 256-bit AVX2 Instructions

Instruction	Latency [1]			Throughput		
DisplayFamily_DisplayModel	**06_4E, 06_5E**	**06_3D, 06_47, 06_56**	**06_3C, 06_45, 06_46, 06_3F**	**06_4E, 06_5E**	**06_3D, 06_47, 06_56**	**06_3C, 06_45, 06_46, 06_3F**
VEXTRACTI128 xmm1, ymm2, imm	1	1	1	1	1	1
VMPSADBW	4	6	6	2	2	2
VPACKUSDW/SSWB	1	1	1	1	1	1
VPADDB/D/W/Q	1	1	1	0.33	0.5	0.5
VPADDSB	1	1	1	0.5	0.5	0.5
VPADDUSB	1	1	1	0.5	0.5	0.5
VPALIGNR	1	1	1	1	1	1
VPAVGB	1	1	1	0.5	0.5	0.5
VPBLENDD	1	1	1	0.33	0.33	0.33
VPBLENDW	1	1	1	1	1	1
VPBLENDVB	1	2	2	1	2	2
VPBROADCASTB/D/SS/SD	3	3	3	1	1	1
VPCMPEQB/W/D	1	1	1	0.5	0.5	0.5
VPCMPEQQ	1	1	1	0.5	0.5	0.5
VPCMPGTQ	3	5	5	1	1	1
VPHADDW/D/SW	3	3	3	2	2	2
VINSERTI128 ymm1, ymm2, xmm, imm	3	3	3	1	1	1
VPMADDWD	5[b]	5	5	0.5	1	1
VPMADDUBSW	5[b]	5	5	0.5	1	1
VPMAXSD	1	1	1	0.5	0.5	0.5
VPMAXUD	1	1	1	0.5	0.5	0.5
VPMOVSX	3	3	3	1	1	1
VPMOVZX	3	3	3	1	1	1
VPMULDQ/UDQ	5[b]	5	5	0.5	1	1
VPMULHRSW	5[b]	5	5	0.5	1	1
VPMULHW/LW	5[b]	5	5	0.5	1	1
VPMULLD	10[b]	10	10	1	2	2
VPOR/VPXOR	1	1	1	0.33	0.33	0.33
VPSADBW	3	5	5	1	1	1
VPSHUFB	1	1	1	1	1	1
VPSHUFD	1	1	1	1	1	1
VPSHUFLW/HW	1	1	1	1	1	1
VPSIGNB/D/W/Q	1	1	1	0.5	0.5	0.5
VPERMD/PS	3	3	3	1	1	1
VPSLLVD/Q	2	2	2	0.5	2	2

Table D-4. 256-bit AVX2 Instructions (Contd.)

Instruction	**Latency** [1]			**Throughput**		
DisplayFamily_DisplayModel	**06_4E, 06_5E**	**06_3D, 06_47, 06_56**	**06_3C, 06_45, 06_46, 06_3F**	**06_4E, 06_5E**	**06_3D, 06_47, 06_56**	**06_3C, 06_45, 06_46, 06_3F**
VPSRAVD	2	2	2	0.5	2	2
VPSRAD/W ymm1, ymm2, imm8	1	1	1	1	1	1
VPSLLDQ ymm1, ymm2, imm8	1	1	1	1	1	1
VPSLLQ/D/W ymm1, ymm2, imm8	1	1	1	1	1	1
VPSLLQ/D/W ymm, ymm, ymm	4	4	4	1	1	1
VPUNPCKHBW/WD/DQ/QDQ	1	1	1	1	1	1
VPUNPCKLBW/WD/DQ/QDQ	1	1	1	1	1	1
ALL VFMA	4	5	5	0.5	0.5	0.5
VPMASKMOVD/Q mem, ymm[d], ymm				1	2	2
VPMASKMOVD/Q NUL, msk_0, ymm				>200[e]	2	2
VPMASKMOVD/Q ymm, ymm[d], mem	11	8	8	1	2	2
VPMASKMOVD/Q ymm, msk_0, [base+index][f]	>200	~200	~200	>200	~200	~200

b: includes 1-cycle bubble due to bypass.
c: includes two 1-cycle bubbles due to bypass
d: MASKMOV instruction timing measured with L1 reference and mask register selecting at least 1 or more elements.
e: MASKMOV store instruction with a mask value selecting 0 elements and illegal address (NUL or non-NUL) incurs delay due to assist.
f: MASKMOV Load instruction with a mask value selecting 0 elements and certain addressing forms incur delay due to assist.

Table D-5. Gather Timing Data from L1D*

Instruction	**Latency** [1]			**Throughput**		
DisplayFamily_DisplayModel	06_4E, 06_5E	06_3D, 06_47, 06_56	06_3C/45/46/3F	06_4E, 06_5E	06_3D, 06_47, 06_56	06_3C/45/46/3F
VPGATHERDD/PS xmm, [vi128], xmm	~20	~17	~14	~4	~5	~7
VPGATHERQQ/PD xmm, [vi128], xmm	~18	~15	~12	~3	~4	~5
VPGATHERDD/PS ymm, [vi256], ymm	~22	~19	~20	~5	~6	~10
VPGATHERQQ/PD ymm, [vi256], ymm	~20	~16	~15	~4	~5	~7

* Gather Instructions fetch data elements via memory references. The timing data shown applies to memory references that reside within the L1 data cache and all mask elements selected

Table D-6. BMI1, BMI2 and General Purpose Instructions

Instruction	Latency [1]			Throughput		
DisplayFamily_DisplayModel	06_4E, 06_5E	06_3D, 06_47, 06_56	06_3C/45/46/3F	06_4E, 06_5E	06_3D, 06_47, 06_56	06_3C/45/46/3F
ANDN	1	1	1	0.5	0.5	0.5
BEXTR	2	2	2	0.5	0.5	0.5
BLSI/BLSMSK/BLSR	1	1	1	0.5	0.5	0.5
BZHI	1	1	1	0.5	0.5	0.5
MULX r64, r64, r64	4	4	4	1	1	1
PDEP/PEXT r64, r64, r64	3	3	3	1	1	1
RORX r64, r64, r64	1	1	1	0.5	0.5	0.5
SALX/SARX/SHLX r64, r64, r64	1	1	1	0.5	0.5	0.5
LZCNT/TZCNT	3	3	3	1	1	1

Table D-7. F16C,RDRAND Instructions

Instruction	Latency [1]				Throughput			
DisplayFamily_DisplayModel	06_4E, 06_5E	06_3D, 06_47, 06_56	06_3C/45/46/3F	06_3A/3E	06_4E, 06_5E	06_3D, 06_47, 06_56	06_3C/45/46/3F	06_3A/3E
RDRAND* r64	Varies	Varies	Varies	<200	<300	~250	~250	<200
VCVTPH2PS ymm1, xmm2	7	6	6	7	1	1	1	1
VCVTPH2PS xmm1, xmm2	5	4	4	6	1	1	1	1
VCVTPS2PH ymm1, xmm2, imm	7	6	6	10	1	1	1	1
VCVTPS2PH xmm1, xmm2, imm	5	4	4	9	1	1	1	1

* See Section D.2

Table D-8. 256-bit AVX Instructions

Instruction	Latency [1]				Throughput			
DisplayFamily_DisplayModel	06_4E, 06_5E	06_3D/47/56	06_3C/45/46/3F	06_3A/3E	06_4E, 06_5E	06_3D/47/56	06_3C/45/46/3F	06_3A/3E
VADDPD/PS ymm1, ymm2, ymm3	4	3	3	3	0.5	1	1	1
VADDSUBPD/PS ymm1, ymm2, ymm3	4	3	3	3	0.5	1	1	1
VANDNPD/PS ymm1, ymm2, ymm3	1	1	1	1	0.33	1	1	1
VANDPD/PS ymm1, ymm2, ymm3	1	1	1	1	0.33	1	1	1
VBLENDPD/PS ymm1, ymm2, ymm3, imm	1	1	1	1	0.33	0.33	0.33	0.5
VBLENDVPD/PS ymm1, ymm2, ymm3, ymm	1	2	2	1	1	2	2	1
VCMPPD/PS ymm1, ymm2, ymm3	4	3	3	3	0.5	1	1	1

Table D-8. 256-bit AVX Instructions (Contd.)

Instruction	Latency [1]				Throughput			
DisplayFamily_DisplayModel	06_4E, 06_5E	06_3D/ 47/56	06_3C/4 5/46/3F	06_3A /3E	06_4E, 06_5E	06_3D/ 47/56	06_3C/4 5/46/3F	06_3A /3E
VCVTDQ2PD ymm1, ymm2	7	6	6	4	1	1	1	1
VCVTDQ2PS ymm1, ymm2	4	3	3	3	0.5	1	1	1
VCVT(T)PD2DQ ymm1, ymm2	7	6	6	4	1	1	1	1
VCVTPD2PS ymm1, ymm2	7	6	6	4	1	1	1	1
VCVT(T)PS2DQ ymm1, ymm2	4	3	3	3	1	1	1	1
VCVTPS2PD ymm1, xmm2	7	4	4	2	1	1	1	1
VDIVPD ymm1, ymm2, ymm3	14	16-23	25-35	27-35	8	16	27	28
VDIVPS ymm1, ymm2, ymm3	11	13-17	17-21	18-21	5	10	13	14
VDPPS ymm1, ymm2, ymm3	13	12	14	12	1.5	2	2	2
VEXTRACTF128 xmm1, ymm2, imm	3	3	3	3	1	1	1	1
VINSERTF128 ymm1, xmm2, imm	3	3	3	3	1	1	1	1
VMAXPD/PS ymm1, ymm2, ymm3	4	3	3	3	0.5	1	1	1
VMINPD/PS ymm1, ymm2, ymm3	4	3	3	3	0.5	1	1	1
VMOVAPD/PS ymm1, ymm2	1	1	1	1	0.25	0.5	0.5	1
VMOVDDUP ymm1, ymm2	1	1	1	1	1	1	1	1
VMOVDQA/U ymm1, ymm2	1	1	1	1	0.25	0.25	0.25	0.5
VMOVMSKPD/PS ymm1, ymm2	2	2	2	1	1	1	1	1
VMOVQ xmm1, xmm2	1	1	1	1	0.33	0.33	0.33	0.33
VMOVD/Q xmm1, r32/r64	2	1	1	1	1	1	1	1
VMOVD/Q r32/r64, xmm	2	1	1	1	1	1	1	1
VMOVNTDQ/PS/PD					1	1	1	1
VMOVSHDUP ymm1, ymm2	1	1	1	1	1	1	1	1
VMOVSLDUP ymm1, ymm2	1	1	1	1	1	1	1	1
VMOVUPD/PS ymm1, ymm2	1	1	1	1	0.25	0.5	0.5	1
VMULPD/PS ymm1, ymm2, ymm3	4	3	5	5	0.5	0.5	0.5	1
VORPD/PS ymm1, ymm2, ymm3	1	1	1	1	0.33	1	1	1
VPERM2F128 ymm1, ymm2, ymm3, imm	3	3	3	2	1	1	1	1
VPERMILPD/PS ymm1, ymm2, ymm3	1	1	1	1	1	1	1	1
VRCPPS ymm1, ymm2	4	7	7	7	1	2	2	2
VROUNDPD/PS ymm1, ymm2, imm	8	6	6	3	1	2	2	1
VRSQRTPS ymm1, ymm2	4	7	7	7	1	2	2	2
VSHUFPD/PS ymm1, ymm2, ymm3, imm	1	1	1	1	1	1	1	1
VSQRTPD ymm1, ymm2	<18	19-35	19-35	19-35	<12	16-27	16-27	28
VSQRTPS ymm1, ymm2	12	18-21	18-21	18-21	<6	13	13	14
VSUBPD/PS ymm1, ymm2, imm	4	3	3	3	0.5	1	1	1

Table D-8. 256-bit AVX Instructions (Contd.)

Instruction	Latency [1]				Throughput			
DisplayFamily_DisplayModel	06_4E, 06_5E	06_3D/ 47/56	06_3C/4 5/46/3F	06_3A /3E	06_4E, 06_5E	06_3D/ 47/56	06_3C/4 5/46/3F	06_3A /3E
VTESTPS ymm1, ymm2	3	2	2	2	1	1	1	1
VUNPCKHPD/PS ymm1, ymm2, ymm3	1	1	1	1	1	1	1	1
VUNPCKLPD/PS ymm1, ymm2, ymm3	1	1	1	1	1	1	1	1
VXORPD/PS ymm1, ymm2, ymm3	1	1	1	1	0.33	1	1	1
VZEROUPPER	0	0	0	0	1	1	1	1
VZEROALL					12	8	8	9
VEXTRACTPS reg, xmm2, imm	3	2	2	2	1	1	1	1
VINSERTPS xmm1, xmm2, reg, imm	1	1	1	1	1	1	1	1
VMASKMOVPD/PS mem[a], ymm, ymm					1	2	2	2
VMASKMOVPD/PS NUL, msk_0, ymm					>200[b]	2	2	2
VMASKMOVPD/PS ymm, ymm[a], mem	11	8	8	9	1	2	2	2
VMASKMOVPD/PS ymm, msk_0, [base+index][c]	>200	~200	~200	~200	>200	~200	~200	~200

Latency and Throughput data for CPUID signature 06_3AH are generally the same as those of 06_2AH, only those that differ from 06_2AH are shown in the 06_3AH column.
a: MASKMOV instruction timing measured with L1 reference and mask register selecting at least 1 or more elements.
b: MASKMOV store instruction with a mask value selecting 0 elements and illegal address (NUL or non-NUL) incurs delay due to assist.
c: MASKMOV Load instruction with a mask value selecting 0 elements and certain addressing forms incur delay due to assist.

Latency of VEX.128 encoded AVX instructions should refer to corresponding legacy 128-bit instructions.

Table D-9. AESNI and PCLMULQDQ Instructions

Instruction	Latency [1]				Throughput			
DisplayFamily_DisplayModel	06_4E, 06_5E	06_3D/ 47/56	06_3C/4 5/46/3F	06_3A /3E	06_4E, 06_5E	06_3D/ 47/56	06_3C/4 5/46/3F	06_3A /3E
AESDEC/AESDECLAST xmm1, xmm2	4	7	7	8	1	1	1	1
AESENC/AESENCLAST xmm1, xmm2	4	7	7	8	1	1	1	1
AESIMC xmm1, xmm2	8	14	14	14	2	2	2	2
AESKEYGENASSIST xmm1, xmm2, imm	12	10	10	10	12	8	8	8
PCLMULQDQ xmm1, xmm2, imm	7[b]	5	7	14	1	1	2	8

b: includes 1-cycle bubble due to bypass.

Table D-10. SSE4.2 Instructions

Instruction	Latency [1]				Throughput			
DisplayFamily_DisplayModel	06_4E, 06_5E	06_3D /47/56	06_3C /45/46 /3F	06_3A /3E/2A /2D	06_4E, 06_5E	06_3D /47/56	06_3C/ 45/46/ 3F	06_3A /3E/2A /2D
CRC32 r32, r32	3	3	3	3	1	1	1	1
PCMPESTRI xmm1, xmm2, imm	15	10	10	11	5	4	4	4
PCMPESTRM xmm1, xmm2, imm	10	10	10	11	6	5	5	4
PCMPISTRI xmm1, xmm2, imm	15	10	10	11	3	3	3	3
PCMPISTRM xmm1, xmm2, imm	15	11	11	11	3	3	3	3
PCMPGTQ xmm1, xmm2	3	5	5	5	0.33	1	1	1
POPCNT r32, r32	3	3	3	3	1	1	1	1
POPCNT r64, r64	3	3	3	3	1	1	1	1

Table D-11. SSE4.1 Instructions

Instruction	Latency [1]				Throughput			
DisplayFamily_DisplayModel	06_4E, 06_5E	06_3D /47/56	06_3C/ 45/46/ 3F	06_3A /3E/2A /2D	06_4E, 06_5E	06_3D/ 47/56	06_3C/ 45/46/ 3F	06_3A /3E/2A /2D
BLENDPD/S xmm1, xmm2, imm	1	1	1	1	0.33	0.33	0.33	0.5
BLENDVPD/S xmm1, xmm2	1	2	2	2	1	2	2	1
DPPD xmm1, xmm2	9	7	9	9	1	1	1	1
DPPS xmm1, xmm2	13	12	14	13	2	2	2	2
EXTRACTPS xmm1, xmm2, imm	3	2	2	2	1	1	1	1
INSERTPS xmm1, xmm2, imm	1	1	1	1	1	1	1	1
MPSADBW xmm1, xmm2, imm	4	6	6	6	2	2	2	1
PACKUSDW xmm1, xmm2	1	1	1	1	1	1	1	0.5
PBLENVB xmm1, xmm2	2	2	2	2	2	2	2	1
PBLENDW xmm1, xmm2, imm	1	1	1	1	1	1	1	0.5
PCMPEQQ xmm1, xmm2	1	1	1	1	0.5	0.5	0.5	0.5
PEXTRB/W/D reg, xmm1, imm	3	3	3	3	1	1	1	1
PHMINPOSUW xmm1,xmm2	4	5	5	5	1	1	1	1
PINSRB/W/D xmm1,reg, imm	2	2	2	2	1	1	1	1
PMAXSB/SD xmm1, xmm2	1	1	1	1	0.5	0.5	0.5	0.5
PMAXUW/UD xmm1, xmm2	1	1	1	1	0.5	0.5	0.5	0.5
PMINSB/SD xmm1, xmm2	1	1	1	1	0.5	0.5	0.5	0.5
PMINUW/UD xmm1, xmm2	1	1	1	1	0.5	0.5	0.5	0.5
PMOVSXBD/BW/BQ xmm1, xmm2	1	1	1	1	1	1	1	0.5
PMOVSXWD/WQ/DQ xmm1, xmm2	1	1	1	1	1	1	1	0.5

Table D-11. SSE4.1 Instructions (Contd.)

Instruction	Latency [1]				Throughput			
DisplayFamily_DisplayModel	06_4E, 06_5E	06_3D /47/56	06_3C/ 45/46/ 3F	06_3A /3E/2A /2D	06_4E, 06_5E	06_3D/ 47/56	06_3C/ 45/46/ 3F	06_3A /3E/2A /2D
PMOVZXBD/BW/BQ xmm1, xmm2	1	1	1	1	1	1	1	0.5
PMOVZXWD/WQ/DQ xmm1, xmm2	1	1	1	1	1	1	1	0.5
PMULDQ xmm1, xmm2	5^b	5	5	5	0.5	1	1	1
PMULLD xmm1, xmm2	10^c	10	10	5	2	2	2	1
PTEST xmm1, xmm2	3	2	2	2	1	1	1	1
ROUNDPD/PS xmm1, xmm2, imm	6	6	6	3	2	2	2	1
ROUNDSD/SS xmm1, xmm2, imm	6	6	6	3	2	2	2	1

b: includes 1-cycle bubble due to bypass
c: includes two 1-cycle bubbles due to bypass

Table D-12. Supplemental Streaming SIMD Extension 3 Instructions

Instruction	Latency [1]				Throughput			
DisplayFamily_DisplayModel	06_4E, 06_5E	06_3D/ 47/56	06_3C/ 45/46/ 3F	06_3A/ 3E/2A/ 2D	06_4E, 06_5E	06_3D/ 47/56	06_3C/ 45/46/ 3F	06_3A/ 3E/2A/ 2D
PALIGNR xmm1, xmm2, imm	1	1	1	1	1	1	1	0.5
PHADDD xmm1, xmm2	3	3	3	3	2	2	2	1.5
PHADDW xmm1, xmm2	3	3	3	3	2	2	2	1.5
PHADDSW xmm1, xmm2	3	3	3	3	2	2	2	1.5
PHSUBD xmm1, xmm2	3	3	3	3	2	2	2	1.5
PHSUBW xmm1, xmm2	3	3	3	3	2	2	2	1.5
PHSUBSW xmm1, xmm2	3	3	3	3	2	2	2	1.5
PMADDUBSW xmm1, xmm2	5^b	5	5	5	0.5	1	1	1
PMULHRSW xmm1, xmm2	5^b	5	5	5	0.5	1	1	1
PSHUFB xmm1, xmm2	1	1	1	1	1	1	1	0.5
PSIGNB/D/W xmm1, xmm2	1	1	1	1	0.5	0.5	0.5	0.5
PABSB/D/W xmm1, xmm2	1	1	1	1	0.5	0.5	0.5	0.5

b: includes 1-cycle bubble due to bypass

Table D-13. Streaming SIMD Extension 3 SIMD Floating-point Instructions

Instruction	Latency[1]				Throughput			
DisplayFamily_DisplayModel	06_4E,06_5E	06_3D/47/56	06_3C/45/46/3F	06_3A/3E/2A/2D	06_4E,06_5E	06_3D/47/56	06_3C/45/46/3F	06_3A/3E/2A/2D
ADDSUBPD/ADDSUBPS	4	3	3	3	0.5	1	1	1
HADDPD xmm1, xmm2	6	5	5	5	2	2	2	2
HADDPS xmm1, xmm2	6	5	5	5	2	2	2	2
HSUBPD xmm1, xmm2	6	5	5	5	2	2	2	2
HSUBPS xmm1, xmm2	6	5	5	5	2	2	2	2
MOVDDUP xmm1, xmm2	1	1	1	1	1	1	1	1
MOVSHDUP xmm1, xmm2	1	1	1	1	1	1	1	1
MOVSLDUP xmm1, xmm2	1	1	1	1	1	1	1	1

Table D-14. Streaming SIMD Extension 2 128-bit Integer Instructions

Instruction	Latency[1]				Throughput			
CPUID	06_4E, 06_5E	06_3D/47/56	06_3C/45/46/3F	06_3A/3E/2A/2D	06_4E, 06_5E	06_3D/47/56	06_3C/45/46/3F	06_3A/3E/2A/2D
CVTPS2DQ xmm, xmm	3	3	3	3	1	1	1	1
CVTTPS2DQ xmm, xmm	3	3	3	3	1	1	1	1
MASKMOVDQU xmm, xmm					7	6	6	6
MOVD xmm, r64/r32	2	1	1	1	1	1	1	1
MOVD r64/r32, xmm	2	1	1	1	1	1	1	1
MOVDQA xmm, xmm	1	1	1	1	0.25	0.33	0.33	0.5
MOVDQU xmm, xmm	1	1	1	1	0.25	0.33	0.33	0.5
MOVQ xmm, xmm	1	1	1	1	0.33	0.33	0.33	0.33
PACKSSWB/PACKSSDW/PACKUSWB xmm, xmm	1	1	1	1	1	1	1	0.5
PADDB/PADDW/PADDD xmm, xmm	1	1	1	1	0.33	0.5	0.5	0.5
PADDSB/PADDSW/PADDUSB/PADDUSW xmm, xmm	1	1	1	1	0.5	0.5	0.5	0.5
PADDQ/ PSUBQ[3] xmm, xmm	1	1	1	1	0.33	0.5	0.5	0.5
PAND xmm, xmm	1	1	1	1	0.33	0.33	0.33	0.33
PANDN xmm, xmm	1	1	1	1	0.33	0.33	0.33	0.33
PAVGB/PAVGW xmm, xmm	1	1	1	1	0.5	0.5	0.5	0.5
PCMPEQB/PCMPEQD/PCMPEQW xmm, xmm	1	1	1	1	0.5	0.5	0.5	0.5
PCMPGTB/PCMPGTD/PCMPGTW xmm, xmm	1	1	1	1	0.5	0.5	0.5	0.5
PEXTRW r32, xmm, imm8	3	3	3	3	1	1	1	1

Table D-14. Streaming SIMD Extension 2 128-bit Integer Instructions (Contd.)

Instruction	Latency[1]				Throughput			
CPUID	06_4E, 06_5E	06_3D/47/56	06_3C/45/46/3F	06_3A/3E/2A/2D	06_4E, 06_5E	06_3D/47/56	06_3C/45/46/3F	06_3A/3E/2A/2D
PINSRW xmm, r32, imm8	2	2	2	2	2	2	2	1
PMADDWD xmm, xmm	5[b]	5	5	5	0.5	1	1	1
PMAX xmm, xmm	1	1	1	1	0.5	0.5	0.5	0.5
PMIN xmm, xmm	1	1	1	1	0.5	0.5	0.5	0.5
PMOVMSKB[3] r32, xmm	2	2	2	2	1	1	1	1
PMULHUW/PMULHW/PMULLW xmm, xmm	5[b]	5	5	5	0.5	1	1	1
PMULUDQ xmm, xmm	5[b]	5	5	5	0.5	1	1	1
POR xmm, xmm	1	1	1	1	0.33	0.33	0.33	0.33
PSADBW xmm, xmm	3	5	5	5	1	1	1	1
PSHUFD xmm, xmm, imm8	1	1	1	1	1	1	1	0.5
PSHUFHW xmm, xmm, imm8	1	1	1	1	1	1	1	0.5
PSHUFLW xmm, xmm, imm8	1	1	1	1	1	1	1	0.5
PSLLDQ xmm, imm8	1	1	1	1	1	1	1	0.5
PSLLW/PSLLD/PSLLQ xmm, imm8	1	1	1	1	1	1	1	1
PSLL/PSRL xmm, xmm	2	2	2	2	1	1	1	1
PSRAW/PSRAD xmm, imm8	1	1	1	1	1	1	1	1
PSRAW/PSRAD xmm, xmm	2	2	2	2	1	1	1	1
PSRLDQ xmm, imm8	1	1	1	1	1	1	1	0.5
PSRLW/PSRLD/PSRLQ xmm, imm8	1	1	1	1	1	1	1	1
PSUBB/PSUBW/PSUBD xmm, xmm	1	1	1	1	0.33	0.5	0.5	0.5
PSUBSB/PSUBSW/PSUBUSB/PSUBUSW xmm, xmm	1	1	1	1	0.5	0.5	0.5	0.5
PUNPCKHBW/PUNPCKHWD/PUNPCKHDQ xmm, xmm	1	1	1	1	1	1	1	0.5
PUNPCKHQDQ xmm, xmm	1	1	1	1	1	1	1	0.5
PUNPCKLBW/PUNPCKLWD/PUNPCKLDQ xmm, xmm	1	1	1	1	1	1	1	0.5
PUNPCKLQDQ xmm, xmm	1	1	1	1	1	1	1	0.5
PXOR xmm, xmm	1	1	1	1	0.33	0.33	0.33	0.33

b: includes 1-cycle bubble due to bypass

Table D-15. Streaming SIMD Extension 2 Double-precision Floating-point Instructions

Instruction	**Latency**[1]				**Throughput**			
CPUID	06_4E, 06_5E	06_3D/47/56	06_3C/45/46/3F	06_2A/2D(06_3A/3E)	06_4E, 06_5E	06_3D/47/56	06_3C/45/46/3F	06_2A/2D(06_3A/3E)
ADDPD xmm, xmm	4	3	3	3	0.5	1	1	1
ADDSD xmm, xmm	4	3	3	3	0.5	1	1	1
ANDNPD xmm, xmm	1	1	1	1	0.33	1	1	1
ANDPD xmm, xmm	1	1	1	1	0.33	1	1	1
CMPPD xmm, xmm, imm8	4	3	3	3	0.5	1	1	1
CMPSD xmm, xmm, imm8	4	3	3	3	0.5	1	1	1
COMISD xmm, xmm	2	2	2	2	1	1	1	1
CVTDQ2PD xmm, xmm	5	4	4	4	1	1	1	1
CVTDQ2PS xmm, xmm	4	3	3	3	1	1	1	1
CVTPD2DQ xmm, xmm	5	4	4	4	1	1	1	1
CVTPD2PS xmm, xmm	5	4	4	4	1	1	1	1
CVT[T]PS2DQ xmm, xmm	4	3	3	3	1	1	1	1
CVTPS2PD xmm, xmm	5	2	2	2	1	1	1	1
CVT[T]SD2SI r64/r32, xmm	6	4	4	5	1	1	1	1
CVTSD2SS xmm, xmm	5	4	4	4	1	1	1	1
CVTSI2SD xmm, r64/r32	5	3	3	4	1	1	1	1
CVTSS2SD xmm, xmm	5	2	2	2	1	1	1	1
CVTTPD2DQ xmm, xmm	5	4	4	4	1	1	1	1
CVTTSD2SI r32, xmm	6	4	4	5	1	1	1	1
DIVPD xmm, xmm[1]	14	<14	14-20	16-22 (15-20)	4	8	13	22(14)
DIVSD xmm, xmm	14	<14	14-20	16-22 (15-20)	4	5	13	22(14)
MAXPD xmm, xmm	4	3	3	3	0.5	1	1	1
MAXSD xmm, xmm	4	3	3	3	0.5	1	1	1
MINPD xmm, xmm	4	3	3	3	0.5	1	1	1
MINSD xmm, xmm	4	3	3	3	0.5	1	1	1
MOVAPD xmm, xmm	1	1	1	1	0.33	0.5	0.5	1
MOVMSKPD r64/r32, xmm	2	2	2	2	1	1	1	1
MOVSD xmm, xmm	1	1	1	1	1	1	1	1
MOVUPD xmm, xmm	1	1	1	1	0.33	0.5	0.5	1
MULPD xmm, xmm	3	5	5	5	0.5	0.5	0.5	1
MULSD xmm, xmm	3	5	5	5	0.5	0.5	0.5	1
ORPD xmm, xmm	1	1	1	1	0.33	1	1	1
SHUFPD xmm, xmm, imm8	1	1	1	1	1	1	1	1
SQRTPD xmm, xmm[2]	18	20	20	22(21)	6	13	13	22(14)
SQRTSD xmm, xmm	18	20	20	22(21)	6	7	13	22(14)
SUBPD xmm, xmm	4	3	3	3	0.5	1	1	1

Table D-15. Streaming SIMD Extension 2 Double-precision Floating-point Instructions (Contd.)

Instruction	Latency[1]				Throughput			
CPUID	06_4E, 06_5E	06_3D/47/56	06_3C/45/46/3F	06_2A/2D(06_3A/3E)	06_4E, 06_5E	06_3D/47/56	06_3C/45/46/3F	06_2A/2D(06_3A/3E)
SUBSD xmm, xmm	4	3	3	3	0.5	1	1	1
UCOMISD xmm, xmm	2	2	2	2	1	1	1	1
UNPCKHPD xmm, xmm	1	1	1	1	1	1	1	1
UNPCKLPD xmm, xmm	1	1	1	1	1	1	1	1
XORPD[3] xmm, xmm	1	1	1	1	0.33	1	1	1

NOTES:

1. The latency and throughput of DIVPD/DIVSD can vary with input values. For certain values, hardware can complete quickly, throughput may be as low as ~ 6 cycles. Similarly, latency for certain input values may be as low as less than 10 cycles.
2. The latency throughput of SQRTPD/SQRTSD can vary with input value. For certain values, hardware can complete quickly, throughput may be as low as ~ 6 cycles. Similarly, latency for certain input values may be as low as less than10 cycles.

Table D-16. Streaming SIMD Extension Single-precision Floating-point Instructions

Instruction	Latency[1]				Throughput			
CPUID	06_4E, 06_5E	06_3D/47/56	06_3C/45/46/3F	06_2A/2D(06_3A/3E)	06_4E, 06_5E	06_3D/47/56	06_3C/45/46/3F	06_2A/2D(06_3A/3E)
ADDPS xmm, xmm	4	3	3	3	0.5	1	1	1
ADDSS xmm, xmm	4	3	3	3	0.5	1	1	1
ANDNPS xmm, xmm	1	1	1	1	0.33	1	1	1
ANDPS xmm, xmm	1	1	1	1	0.33	1	1	1
CMPPS xmm, xmm	4	3	3	3	0.5	1	1	1
CMPSS xmm, xmm	4	3	3	3	0.5	1	1	1
COMISS xmm, xmm	2	2	2	2	1	1	1	1
CVTSI2SS xmm, r32	6	4	4	5	1	1	1	1
CVTSS2SI r32, xmm	6	4	4	5	1	1	1	1
CVT[T]SS2SI r64, xmm	6	4	4	5	1	1	1	1
CVTTSS2SI r32, xmm	6	4	4	5	1	1	1	1
DIVPS xmm, xmm[1]	11	<11	<13	10-14	3	4	6	14(6)
DIVSS xmm, xmm	11	<11	<13	10-14	3	2.5	6	14(6)
MAXPS xmm, xmm	4	3	3	3	0.5	1	1	1
MAXSS xmm, xmm	4	3	3	3	0.5	1	1	1
MINPS xmm, xmm	4	3	3	3	0.5	1	1	1
MINSS xmm, xmm	4	3	3	3	0.5	1	1	1
MOVAPS xmm, xmm	1	1	1	1	0.25	0.5	0.5	1
MOVHLPS xmm, xmm	1	1	1	1	1	1	1	1
MOVLHPS xmm, xmm	1	1	1	1	1	1	1	1
MOVMSKPS r64/r32, xmm	2	2	2	2	1	1	1	1

Table D-16. Streaming SIMD Extension Single-precision Floating-point Instructions (Contd.)

Instruction	Latency[1]				Throughput			
CPUID	06_4E, 06_5E	06_3D/47/56	06_3C/45/46/3F	06_2A/2D(06_3A/3E)	06_4E, 06_5E	06_3D/47/56	06_3C/45/46/3F	06_2A/2D(06_3A/3E)
MOVSS xmm, xmm	1	1	1	1	1	1	1	1
MOVUPS xmm, xmm	1	1	1	1	0.25	0.5	0.5	1
MULPS xmm, xmm	4	3	5	5	0.5	0.5	0.5	1
MULSS xmm, xmm	4	3	5	5	0.5	0.5	0.5	1
ORPS xmm, xmm	1	1	1	1	0.33	1	1	1
RCPPS xmm, xmm	4	5	5	5	1	1	1	1
RCPSS xmm, xmm	4	5	5	5	1	1	1	1
RSQRTPS xmm, xmm	4	5	5	5	1	1	1	1
RSQRTSS xmm, xmm	4	5	5	5	1	1	1	1
SHUFPS xmm, xmm, imm8	1	1	1	1	1	1	1	1
SQRTPS xmm, xmm[2]	13	13	13	14	3	7	7	14(7)
SQRTSS xmm, xmm	13	13	13	14	3	4	7	14(7)
SUBPS xmm, xmm	4	3	3	3	0.5	1	1	1
SUBSS xmm, xmm	4	3	3	3	0.5	1	1	1
UCOMISS xmm, xmm	2	2	2	2	1	1	1	1
UNPCKHPS xmm, xmm	1	1	1	1	1	1	1	1
UNPCKLPS xmm, xmm	1	1	1	1	1	1	1	1
XORPS xmm, xmm	1	1	1	1	1	1	1	1
LFENCE[3]					6	5	5	4
MFENCE[3]					~40	~35	~35	~35
SFENCE[3]					7	6	6	5
STMXCSR[3]					1	1	1	1
FXSAVE[3]					~90	~71	~75	~78

NOTES:

1. The latency and throughput of DIVPS/DIVSS can vary with input values. For certain values, hardware can complete quickly, throughput may be as low as ~ 6 cycles. Similarly, latency for certain input values may be as low as less than 10 cycles.
2. The latency and throughput of SQRTPS/SQRTSS can vary with input values. For certain values, hardware can complete quickly, throughput may be as low as ~ 6 cycles. Similarly, latency for certain input values may be as low as less than 10 cycles
3. The throughputs of FXSAVE/LFENCE/MFENCE/SFENCE/STMXCSR are measured with the destination in L1 Data Cache.

Table D-17. General Purpose Instructions

Instruction	Latency[1]				Throughput			
CPUID	06_4E,06_5E	06_3D/47/56	06_3C/45/46/3F	06_3A, 06_3E	06_4E,06_5E	06_3D/47/56	06_3C/45/46/3F	06_3A, 06_3E
ADC/SBB reg, reg	1	2	2	2	0.5	1	1	1
ADC/SBB reg, imm	1	2	2	2	0.5	1	1	1
ADD/SUB	1	1	1	1	0.25	0.25	0.25	0.33
AND/OR/XOR	1	1	1	1	0.25	0.25	0.25	0.33
BSF/BSR	3	3	3	3	1	1	1	1
BSWAP	2	2	2	2	0.5	0.5	0.5	1
BT	1	1	1	1	0.5	0.5	0.5	0.5
BTC/BTR/BTS	1	1	1	1	0.5	0.5	0.5	0.5
CBW/CWDE/CDQE	1	1	1	1	1	1	1	1
CDQ	1	1	1	1	1	1	1	1
CQO	1	1	1	1	0.5	0.5	0.5	0.5
CLC					0.25	0.33	0.33	0.33
CMC					0.25	0.33	0.33	0.33
STC					0.25	0.33	0.33	0.33
CLFLUSH[12]					~2 to 50	~3 to 50	~3 to 50	~5 to 50
CLFLUSHOPT[13]					~2to 10	NA	NA	NA
CMOVE/CMOVcc	1	1	2	2	0.5	0.5	0.5	0.5
CMOVBE/NBE/A/NA	2	2	3	3	1	1	1	1
CMP/TEST	1	1	1	1	0.25	0.25	0.25	0.33
CPUID (EAX = 0)					~100	~100	~100	~95
CPUID (EAX != 0)					>200	>200	>200	>200
CMPXCHG r64, r64	5	5	5	5	5	5	5	5
CMPXCHG8B m64	15	8	8	8	15	8	8	8
CMPXCHG16B m128	19	10	10	10	19	10	10	10
Lock CMPXCHG8B m64	22	19	19	24	22	19	19	24
Lock CMPXCHG16B m128	32	28	28	29	32	28	28	29
DEC/INC	1	2	2	2	0.25	0.25	0.25	0.33
IMUL r64, r64	3	3	3	3	1	1	1	1
IMUL r64[10]	4, 5	3, 4	3, 4	3, 4	1	1	1	1
IMUL r32	5	4	4	4	1	1	1	1
IDIV r64 (RDX!= 0)[8]					~85-100	~85-100	~85-100	~85-100
IDIV r32[9]					~20-26	~20-26	~20-26	~19-25
LEA	1	1	1	1	0.5	0.5	0.5	0.5
LEA [base+index]disp	3	3	3	3	1	1	1	1
MOVSB/MOVSW	1	1	1	1	0..25	0..25	0..25	0.33
MOVZB/MOVZW	1	1	1	1	0.25	0.25	0.25	0.33
DIV r64 (RDX!= 0)[8]					~80-95	~80-95	~80-95	~80-95
DIV r32[9]					~20-26	~20-26	~20-26	~19-25

Table D-17. General Purpose Instructions (Contd.)

Instruction	Latency[1]				Throughput			
CPUID	06_4E,06_5E	06_3D/47/56	06_3C/45/46/3F	06_3A, 06_3E	06_4E,06_5E	06_3D/47/56	06_3C/45/46/3F	06_3A, 06_3E
MUL r64[10]	4, 5	3, 4	3, 4	3, 4	1	1	1	1
NEG/NOT	1	2	2	2	0.25	0.25	0.25	0.33
PAUSE					~140	~10	~10	~10
RCL/RCR reg, 1	2	2	2	2	2	1.5	1.5	1.5
RCL/RCR	6	6	6	6	6	6	6	6
RDTSC					~13	~10	~10	~20
RDTSCP					~20	~30	~30	~30
ROL/ROR reg 1	1 (2 flg)	1 (2 flg)	1 (2 flg)	1 (2 flg)	1	1	1	1
ROL/ROR reg imm	1	1	1	1	0.5	0.5	0.5	0.5
ROL/ROR reg, cl	2	2	2	2	1.5	1.5	1.5	1.5
LAHF/SAHF	3	2	2	2				
SAL/SAR/SHL/SHR reg, imm	1	1	1	1	0.5	0.5	0.5	0.5
SAL/SAR/SHL/SHR reg, cl	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
SETBE	2	2	2	2	1	1	1	1
SETE	1	1	1	1	0.5	0.5	0.5	0.5
SHLD/RD reg, reg, cl	6	4	4	2 (4 flg)	1.5	1	1	1.5
SHLD/RD reg, reg, imm	3	3	3	1	0.5	0.5	0.5	0.5
XSAVE[11]					~98	~100	~100	~100
XSAVEOPT[11]					~86	~90	~90	~90
XADD	2	2	2	2	1	1	1	1
XCHG reg, reg	1	1	1	2	1	1	1	1
XCHG reg, mem	22	19	19	19	22	19	19	19

D.3.2 Table Footnotes

The following footnotes refer to all tables in this appendix.

1. Latency information for many instructions that are complex (> 4 μops) are estimates based on conservative (worst-case) estimates. Actual performance of these instructions by the out-of-order core execution unit can range from somewhat faster to significantly faster than the latency data shown in these tables.
2. Latency and Throughput of transcendental instructions can vary substantially in a dynamic execution environment. Only an approximate value or a range of values are given for these instructions.
3. It may be possible to construct repetitive calls to some Intel 64 and IA-32 instructions in code sequences to achieve latency that is one or two clock cycles faster than the more realistic number listed in this table.
4. The FXCH instruction has 0 latency in code sequences. However, it is limited to an issue rate of one instruction per clock cycle.
5. The load constant instructions, FINCSTP, and FDECSTP have 0 latency in code sequences.
6. Selection of conditional jump instructions should be based on the recommendation of Section 3.4.1, "Branch Prediction Optimization," to improve the predictability of branches. When branches are predicted successfully, the latency of jcc is effectively zero.

7. RCL/RCR with shift count of 1 are optimized. Using RCL/RCR with shift count other than 1 will be executed more slowly. This applies to the Pentium 4 and Intel Xeon processors.
8. The throughput of "DIV/IDIV r64" varies with the number of significant digits in the input RDX:RAX. The throughput is significantly higher if RDX input is 0, similar to those of "DIV/IDIV r32". If RDX is not zero, the throughput is significantly lower, as shown in the range. The throughput decreases (increasing numerical value in cycles) with increasing number of significant bits in the input RDX:RAX (relative to the number of significant bits of the divisor) or the output quotient. The latency of "DIV/IDIV r64" also varies with the significant bits of input values. For a given set of input values, the latency is about the same as the throughput in cycles.
9. The throughput of "DIV/IDIV r32" varies with the number of significant digits in the input EDX:EAX and/or of the quotient of the division for a given size of significant bits in the divisor r32. The throughput decreases (increasing numerical value in cycles) with increasing number of significant bits in the input EDX:EAX or the output quotient. The latency of "DIV/IDIV r32" also varies with the significant bits of the input values. For a given set of input values, the latency is about the same as the throughput in cycles.
10. The latency of MUL r64 into 128-bit result has two sets of numbers, the read-to-use latency of the low 64-bit result (RAX) is smaller. The latency of the high 64-bit of the 128 bit result (RDX) is larger.
11. The throughputs of XSAVE and XSAVEOPT are measured with the destination in L1 Data Cache and includes the YMM states.
12. CLFLUSH throughput is representative from clean cache lines for a range of buffer sizes. CLFLUSH throughput can decrease significantly by factors including: (a) the number of back-to-back CLFLUSH being executed, (b) flushing modified cache lines incurs additional cost than cache lines in other coherent state. See Section 9.4.6.
13. CLFLUSHOPT throughput is representative from clean cache lines for a range of buffer sizes. CLFLUSHOPT throughput can decrease by factors including: (a) flushing modified cache lines incurs additional cost than cache lines in other coherent state, (b) the number of cache lines back-to-back. See Section 9.4.7.

D.3.3 Instructions with Memory Operands

The latency of an Instruction with memory operand can vary greatly due to a number of factors, including data locality in the memory/cache hierarchy and characteristics that are unique to each microarchitecture. Generally, software can approach tuning for locality and instruction selection independently. Thus Table D-4 through Table D-18 can be used for the purpose of instruction selection. Latency and throughput of data movement in the cache/memory hierarchy can be dealt with independent of instruction latency and throughput. Load-to-use Latency of the cache hierarchy can be found in Chapter 2.

D.3.3.1 Software Observable Latency of Memory References

When measuring latency of memory references of individual instructions, many factors can influence the observed latency exposure. Aside from access patterns, cache locality, effect of the hardware prefetchers, different microarchitectures may expose variability such register domains of the destination or memory addressing form with respect to the instruction encoding.

The table below gives a few selected sampling of the variability of L1D cache hit latency that software may observe using pointer-chasing constructs, due to memory reference encoding details, on recent Intel microarchitectures.

Table D-18. Pointer-Chasing Variability of Software Measurable Latency of L1 Data Cache Latency

Pointer Chase Construct	L1D latency Observation
MOV rax, [rax]	4
MOV rax, disp32[rax] , disp32 < 2048	4
MOV rax, [rcx+rax]	5
MOV rax, disp32[rcx+rax] , disp32 < 2048	5

APPENDIX E EARLIER GENERATIONS OF INTEL® 64 AND IA-32 PROCESSOR ARCHITECTURES

E.1 HASWELL MICROARCHITECTURE

The Haswell microarchitecture builds on the successes of the Sandy Bridge and Ivy Bridge microarchitectures. The basic pipeline functionality of the Haswell microarchitecture is depicted in Figure E-1. In general, most of the features described in Section E.1.1 - Section E.1.4 also apply to the Broadwell microarchitecture. Enhancements of the Broadwell microarchitecture are summarized in Section E.1.7.

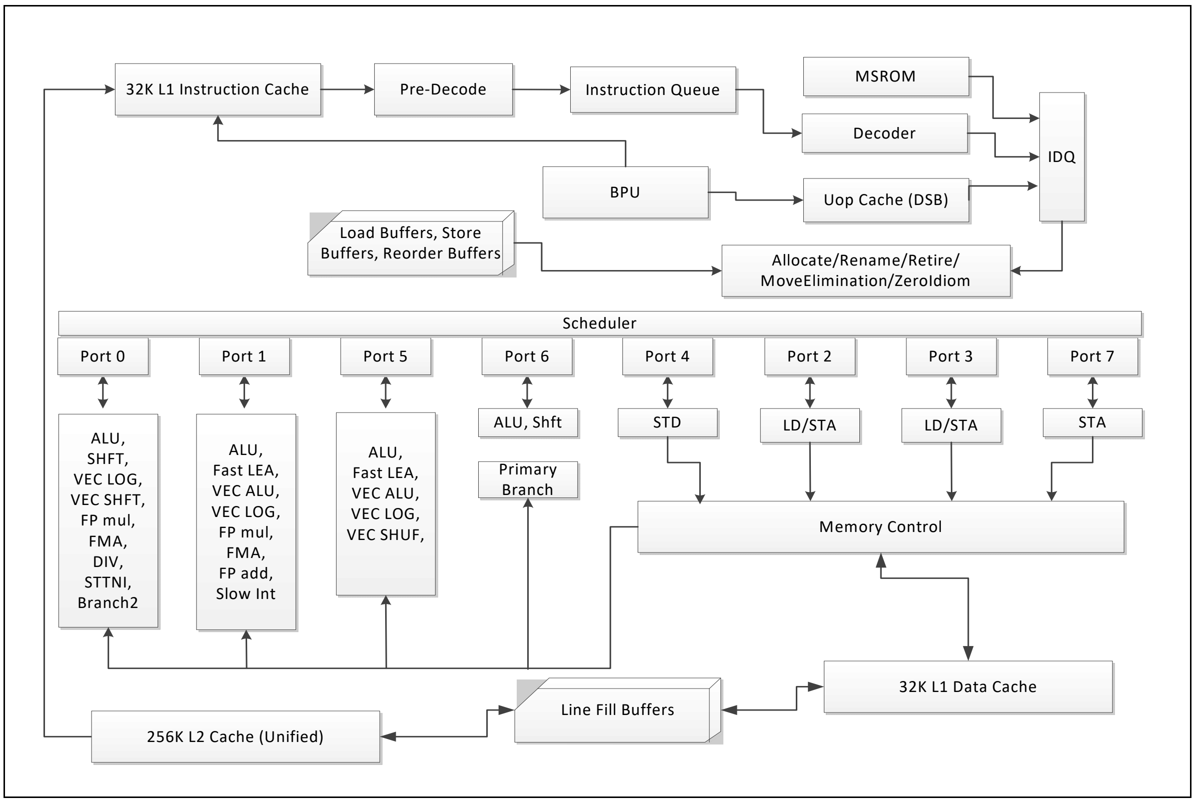
32K L1 Instruction Cache
Pre-Decode
Instruction Queue
MSROM
Decoder
IDQ
BPU
Uop Cache (DSB)
Load Buffers, Store Buffers, Reorder Buffers
Allocate/Rename/Retire/ MoveElimination/ZeroIdiom
Scheduler
Port 0
Port 1
Port 5
Port 6
Port 4
Port 2
Port 3
Port 7
ALU, SHFT, VEC LOG, VEC SHFT, FP mul, FMA, DIV, STTNI, Branch2
ALU, Fast LEA, VEC ALU, VEC LOG, FP mul, FMA, FP add, Slow Int
ALU, Fast LEA, VEC ALU, VEC LOG, VEC SHUF,
ALU, Shft
Primary Branch
STD
LD/STA
LD/STA
STA
Memory Control
32K L1 Data Cache
Line Fill Buffers
256K L2 Cache (Unified)

Figure E-1. CPU Core Pipeline Functionality of the Haswell Microarchitecture

The Haswell microarchitecture offers the following innovative features:

- Support for Intel Advanced Vector Extensions 2 (Intel AVX2), FMA.
- Support for general-purpose, new instructions to accelerate integer numeric encryption.
- Support for Intel® Transactional Synchronization Extensions (Intel® TSX).
- Each core can dispatch up to 8 micro-ops per cycle.
- 256-bit data path for memory operation, FMA, AVX floating-point and AVX2 integer execution units.
- Improved L1D and L2 cache bandwidth.
- Two FMA execution pipelines.
- Four arithmetic logical units (ALUs).

- Three store address ports.
- Two branch execution units.
- Advanced power management features for IA processor core and uncore sub-systems.
- Support for optional fourth level cache.

The microarchitecture supports flexible integration of multiple processor cores with a shared uncore sub-system consisting of a number of components including a ring interconnect to multiple slices of L3 (an off-die L4 is optional), processor graphics, integrated memory controller, interconnect fabrics, etc. An example of the system integration view of four CPU cores with uncore components is illustrated in Figure E-2.

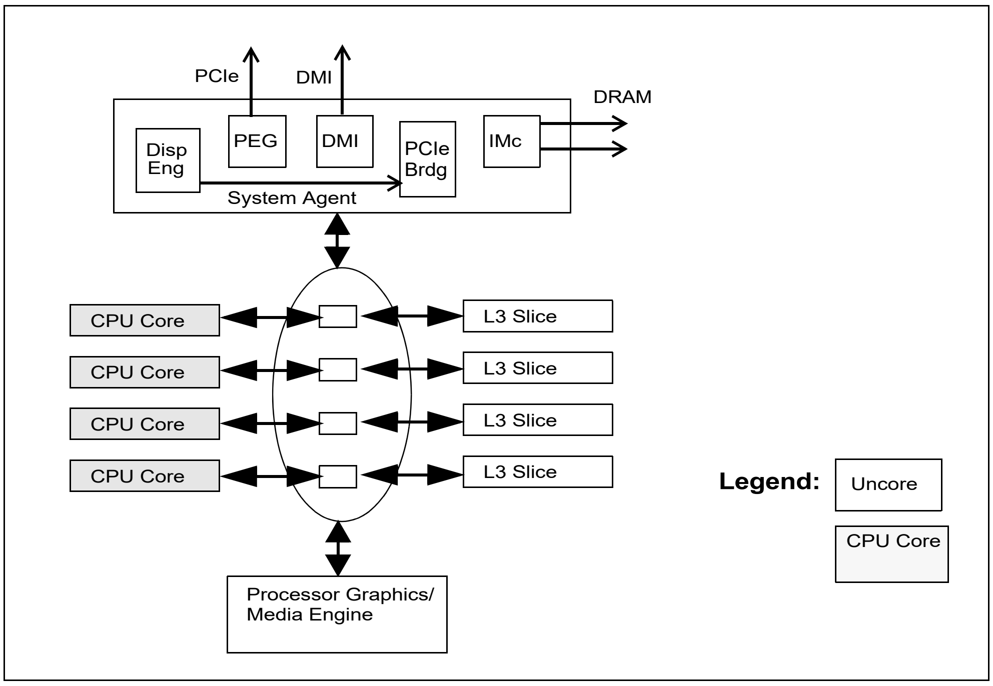

Figure E-2. Four Core System Integration of the Haswell Microarchitecture

E.1.1 The Front End

The front end of Haswell microarchitecture builds on that of the Sandy Bridge and Ivy Bridge microarchitectures, see Section E.2.2 and Section E.2.7. Additional enhancements in the front end include:

- The uop cache (or decoded ICache) is partitioned equally between two logical processors.
- The instruction decoders will alternate between each active logical processor. If one sibling logical processor is idle, the active logical processor will use the decoders continuously.
- The LSD in the micro-op queue (or IDQ) can detect small loops up to 56 micro-ops. The 56-entry micro-op queue is shared by two logical processors if Hyper-Threading Technology is active (Sandy Bridge microarchitecture provides duplicated 28-entry micro-op queue in each core).

E.1.2 The Out-of-Order Engine

The key components and significant improvements to the out-of-order engine are summarized below:

Renamer: The Renamer moves micro-ops from the micro-op queue to bind to the dispatch ports in the Scheduler with execution resources. Zero-idiom, one-idiom and zero-latency register move operations are performed by the Renamer to free up the Scheduler and execution core for improved performance.