| [pf](http://ref.x86asm.net/#column_pf) | [0F](http://ref.x86asm.net/#column_0F) | [po](http://ref.x86asm.net/#column_po) | [so](http://ref.x86asm.net/#column_so) | [o](http://ref.x86asm.net/#column_o) | [proc](http://ref.x86asm.net/#column_proc) | [st](http://ref.x86asm.net/#column_st) | [m](http://ref.x86asm.net/#column_m) | [rl](http://ref.x86asm.net/#column_rl) | [x](http://ref.x86asm.net/#column_x) | [mnemonic](http://ref.x86asm.net/#column_mnemonic) | [op1](http://ref.x86asm.net/#column_op) | [op2](http://ref.x86asm.net/#column_op) | [op3](http://ref.x86asm.net/#column_op) | [op4](http://ref.x86asm.net/#column_op) | [iext](http://ref.x86asm.net/#column_iext) | [tested f](http://ref.x86asm.net/#column_tested_f_modif_f_def_f_undef_f) | [modif f](http://ref.x86asm.net/#column_tested_f_modif_f_def_f_undef_f) | [def f](http://ref.x86asm.net/#column_tested_f_modif_f_def_f_undef_f) | [undef f](http://ref.x86asm.net/#column_tested_f_modif_f_def_f_undef_f) | [f values](http://ref.x86asm.net/#column_f_values) | [description, notes](http://ref.x86asm.net/#column_description_notes) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0F | 00 |  | 0 | 02+ |  | P |  |  | SLDT | **m16** | *LDTR* |  |  |  |  |  |  |  |  | Store Local Descriptor Table Register |
| SLDT | **r16/32** | *LDTR* |  |  |
|  | 0F | 00 |  | 1 | 02+ |  | P |  |  | STR | **m16** | *TR* |  |  |  |  |  |  |  |  | Store Task Register |
| STR | **r16/32** | *TR* |  |  |
|  | 0F | 00 |  | 2 | 02+ |  | P | 0 |  | LLDT | ***LDTR*** | r/m16 |  |  |  |  |  |  |  |  | Load Local Descriptor Table Register |
|  | 0F | 00 |  | 3 | 02+ |  | P | 0 |  | LTR | ***TR*** | r/m16 |  |  |  |  |  |  |  |  | Load Task Register |
|  | 0F | 00 |  | 4 | 02+ |  | P |  |  | VERR | r/m16 |  |  |  |  |  | ....z... | ....z... |  |  | Verify a Segment for Reading |
|  | 0F | 00 |  | 5 | 02+ |  | P |  |  | VERW | r/m16 |  |  |  |  |  | ....z... | ....z... |  |  | Verify a Segment for Writing |
|  | 0F | 01 |  | 0 | 02+ |  |  |  |  | SGDT | **m** | *GDTR* |  |  |  |  |  |  |  |  | Store Global Descriptor Table Register |
|  | 0F | 01 | C1 | 0 | P4++ | D[24](http://ref.x86asm.net/coder32.html#gen_note_VMX_vs_SVM) | P | 0 |  | VMCALL |  |  |  |  | vmx |  | o..szapc | o..szapc |  |  | Call to VM Monitor |
|  | 0F | 01 | C2 | 0 | P4++ | D[24](http://ref.x86asm.net/coder32.html#gen_note_VMX_vs_SVM) | P | 0 |  | VMLAUNCH |  |  |  |  | vmx |  | o..szapc | o..szapc |  |  | Launch Virtual Machine |
|  | 0F | 01 | C3 | 0 | P4++ | D[24](http://ref.x86asm.net/coder32.html#gen_note_VMX_vs_SVM) | P | 0 |  | VMRESUME |  |  |  |  | vmx |  | o..szapc | o..szapc |  |  | Resume Virtual Machine |
|  | 0F | 01 | C4 | 0 | P4++ | D[24](http://ref.x86asm.net/coder32.html#gen_note_VMX_vs_SVM) | P | 0 |  | VMXOFF |  |  |  |  | vmx |  | o..szapc | o..szapc |  |  | Leave VMX Operation |
|  | 0F | 01 |  | 1 | 02+ |  |  |  |  | SIDT | **m** | *IDTR* |  |  |  |  |  |  |  |  | Store Interrupt Descriptor Table Register |
|  | 0F | 01 | C8 | 1 | P4++ |  |  | 0 |  | MONITOR | *m8* | *ECX* | *EDX* |  | sse3 |  |  |  |  |  | Set Up Monitor Address |
|  | 0F | 01 | C9 | 1 | P4++ |  |  | 0 |  | MWAIT | *EAX* | *ECX* |  |  | sse3 |  |  |  |  |  | Monitor Wait |
|  | 0F | 01 |  | 2 | 02+ |  |  | 0 |  | LGDT | ***GDTR*** | m |  |  |  |  |  |  |  |  | Load Global Descriptor Table Register |
|  | 0F | 01 | D0 | 2 | C2++ |  |  |  |  | XGETBV | ***EDX*** | ***EAX*** | *ECX* | *XCR* |  |  |  |  |  |  | Get Value of Extended Control Register |
|  | 0F | 01 | D1 | 2 | C2++ |  |  | 0 |  | XSETBV | ***XCR*** | *ECX* | *EDX* | *EAX* |  |  |  |  |  |  | Set Extended Control Register |
|  | 0F | 01 |  | 3 | 02+ |  |  | 0 |  | LIDT | ***IDTR*** | m |  |  |  |  |  |  |  |  | Load Interrupt Descriptor Table Register |
|  | 0F | 01 |  | 4 | 02+ | D[13](http://ref.x86asm.net/coder32.html#gen_note_SMSW_0F01_4) |  |  |  | SMSW | **m16** | *MSW* |  |  |  |  |  |  |  |  | Store Machine Status Word |
| SMSW | **r16/32** | *MSW* |  |  |
|  | 0F | 01 |  | 6 | 02+ |  |  | 0 |  | LMSW | ***MSW*** | r/m16 |  |  |  |  |  |  |  |  | Load Machine Status Word |
|  | 0F | 01 |  | 7 | 04+ |  |  | 0 |  | INVLPG | m |  |  |  |  |  |  |  |  |  | Invalidate TLB Entry |
|  | 0F | 01 | F9 | 7 | C7+ |  |  | f[2](http://ref.x86asm.net/coder32.html#cr4_tsd) |  | RDTSCP | ***EAX*** | ***EDX*** | ***ECX*** | ... |  |  |  |  |  |  | Read Time-Stamp Counter and Processor ID |
|  | 0F | 02 |  | r | 02+ |  | P |  |  | LAR | **r16/32** | m16 |  |  |  |  | ....z... | ....z... |  |  | Load Access Rights Byte |
| LAR | **r16/32** | r16/32 |  |  |
|  | 0F | 03 |  | r | 02+ |  | P |  |  | LSL | **r16/32** | m16 |  |  |  |  | ....z... | ....z... |  |  | Load Segment Limit |
| LSL | **r16/32** | r16/32 |  |  |
|  | 0F | 06 |  |  | 02+ |  |  | 0 |  | CLTS | ***CR0*** |  |  |  |  |  |  |  |  |  | Clear Task-Switched Flag in CR0 |
|  | 0F | 08 |  |  | 04+ |  |  | 0 |  | INVD |  |  |  |  |  |  |  |  |  |  | Invalidate Internal Caches |
|  | 0F | 09 |  |  | 04+ |  |  | 0 |  | WBINVD |  |  |  |  |  |  |  |  |  |  | Write Back and Invalidate Cache |
|  | 0F | 0B |  |  | 02+ |  |  |  |  | UD2 |  |  |  |  |  |  |  |  |  |  | Undefined Instruction |
|  | 0F | 0D |  |  | PP+ | M[14](http://ref.x86asm.net/coder32.html#gen_note_NOP_0F0D) |  |  |  | NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | No Operation |
|  | 0F | 10 |  | r | P3+ |  |  |  |  | MOVUPS | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Move Unaligned Packed Single-FP Values |
| F3 | 0F | 10 |  | r | P3+ |  |  |  |  | MOVSS | **xmm** | xmm/m32 |  |  | sse1 |  |  |  |  |  | Move Scalar Single-FP Values |
| 66 | 0F | 10 |  | r | P4+ |  |  |  |  | MOVUPD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Move Unaligned Packed Double-FP Value |
| F2 | 0F | 10 |  | r | P4+ |  |  |  |  | MOVSD | **xmm** | xmm/m64 |  |  | sse2 |  |  |  |  |  | Move Scalar Double-FP Value |
|  | 0F | 11 |  | r | P3+ |  |  |  |  | MOVUPS | **xmm/m128** | xmm |  |  | sse1 |  |  |  |  |  | Move Unaligned Packed Single-FP Values |
| F3 | 0F | 11 |  | r | P3+ |  |  |  |  | MOVSS | **xmm/m32** | xmm |  |  | sse1 |  |  |  |  |  | Move Scalar Single-FP Values |
| 66 | 0F | 11 |  | r | P4+ |  |  |  |  | MOVUPD | **xmm/m128** | xmm |  |  | sse2 |  |  |  |  |  | Move Unaligned Packed Double-FP Values |
| F2 | 0F | 11 |  | r | P4+ |  |  |  |  | MOVSD | **xmm/m64** | xmm |  |  | sse2 |  |  |  |  |  | Move Scalar Double-FP Value |
|  | 0F | 12 |  | r | P3+ |  |  |  |  | MOVHLPS | **xmm** | xmm |  |  | sse1 |  |  |  |  |  | Move Packed Single-FP Values High to Low |
|  | 0F | 12 |  | r | P3+ |  |  |  |  | MOVLPS | **xmm** | m64 |  |  | sse1 |  |  |  |  |  | Move Low Packed Single-FP Values |
| 66 | 0F | 12 |  | r | P4+ |  |  |  |  | MOVLPD | **xmm** | m64 |  |  | sse2 |  |  |  |  |  | Move Low Packed Double-FP Value |
| F2 | 0F | 12 |  | r | P4++ |  |  |  |  | MOVDDUP | **xmm** | xmm/m64 |  |  | sse3 |  |  |  |  |  | Move One Double-FP and Duplicate |
| F3 | 0F | 12 |  | r | P4++ |  |  |  |  | MOVSLDUP | **xmm** | xmm/m64 |  |  | sse3 |  |  |  |  |  | Move Packed Single-FP Low and Duplicate |
|  | 0F | 13 |  | r | P3+ |  |  |  |  | MOVLPS | **m64** | xmm |  |  | sse1 |  |  |  |  |  | Move Low Packed Single-FP Values |
| 66 | 0F | 13 |  | r | P4+ |  |  |  |  | MOVLPD | **m64** | xmm |  |  | sse2 |  |  |  |  |  | Move Low Packed Double-FP Value |
|  | 0F | 14 |  | r | P3+ |  |  |  |  | UNPCKLPS | **xmm** | xmm/m64 |  |  | sse1 |  |  |  |  |  | Unpack and Interleave Low Packed Single-FP Values |
| 66 | 0F | 14 |  | r | P4+ |  |  |  |  | UNPCKLPD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Unpack and Interleave Low Packed Double-FP Values |
|  | 0F | 15 |  | r | P3+ |  |  |  |  | UNPCKHPS | **xmm** | xmm/m64 |  |  | sse1 |  |  |  |  |  | Unpack and Interleave High Packed Single-FP Values |
| 66 | 0F | 15 |  | r | P4+ |  |  |  |  | UNPCKHPD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Unpack and Interleave High Packed Double-FP Values |
|  | 0F | 16 |  | r | P3+ |  |  |  |  | MOVLHPS | **xmm** | xmm |  |  | sse1 |  |  |  |  |  | Move Packed Single-FP Values Low to High |
|  | 0F | 16 |  | r | P3+ |  |  |  |  | MOVHPS | **xmm** | m64 |  |  | sse1 |  |  |  |  |  | Move High Packed Single-FP Values |
| 66 | 0F | 16 |  | r | P4+ |  |  |  |  | MOVHPD | **xmm** | m64 |  |  | sse2 |  |  |  |  |  | Move High Packed Double-FP Value |
| F3 | 0F | 16 |  | r | P4++ |  |  |  |  | MOVSHDUP | **xmm** | xmm/m64 |  |  | sse3 |  |  |  |  |  | Move Packed Single-FP High and Duplicate |
|  | 0F | 17 |  | r | P3+ |  |  |  |  | MOVHPS | **m64** | xmm |  |  | sse1 |  |  |  |  |  | Move High Packed Single-FP Values |
| 66 | 0F | 17 |  | r | P4+ |  |  |  |  | MOVHPD | **m64** | xmm |  |  | sse2 |  |  |  |  |  | Move High Packed Double-FP Value |
|  | 0F | 18 |  | 0 | P3+ |  |  |  |  | PREFETCHNTA | m8 |  |  |  | sse1 |  |  |  |  |  | Prefetch Data Into Caches |
|  | 0F | 18 |  | 1 | P3+ |  |  |  |  | PREFETCHT0 | m8 |  |  |  | sse1 |  |  |  |  |  | Prefetch Data Into Caches |
|  | 0F | 18 |  | 2 | P3+ |  |  |  |  | PREFETCHT1 | m8 |  |  |  | sse1 |  |  |  |  |  | Prefetch Data Into Caches |
|  | 0F | 18 |  | 3 | P3+ |  |  |  |  | PREFETCHT2 | m8 |  |  |  | sse1 |  |  |  |  |  | Prefetch Data Into Caches |
|  | 0F | 18 |  | 4 | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 18 |  | 5 | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 18 |  | 6 | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 18 |  | 7 | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 19 |  |  | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 1A |  |  | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 1B |  |  | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 1C |  |  | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 1D |  |  | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 1E |  |  | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 1F |  | 0 | P4++ |  |  |  |  | NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | No Operation |
|  | 0F | 1F |  | 1 | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 1F |  | 2 | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 1F |  | 3 | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 1F |  | 4 | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 1F |  | 5 | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 1F |  | 6 | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 1F |  | 7 | PP+ | M[15](http://ref.x86asm.net/coder32.html#gen_note_hintable_nop_0F18_0F1F) |  |  |  | HINT\_NOP | r/m16/32 |  |  |  |  |  |  |  |  |  | Hintable NOP |
|  | 0F | 20 |  | r | 03+ |  |  | 0 |  | MOV | **r32** | CRn |  |  |  |  | o..szapc |  | o..szapc |  | Move to/from Control Registers |
|  | 0F | 20 |  | r | 03+ | U[16](http://ref.x86asm.net/coder32.html#gen_note_u_MOV_CR_DR_TR_0F20_0F21_0F22_0F23_0F24_0F26) |  | 0 |  | MOV | **r32** | CRn |  |  |  |  | o..szapc |  | o..szapc |  | Move to/from Control Registers |
|  | 0F | 21 |  | r | 03+ |  |  | 0 |  | MOV | **r32** | DRn |  |  |  |  | o..szapc |  | o..szapc |  | Move to/from Debug Registers |
|  | 0F | 21 |  | r | 03+ | U[16](http://ref.x86asm.net/coder32.html#gen_note_u_MOV_CR_DR_TR_0F20_0F21_0F22_0F23_0F24_0F26) |  | 0 |  | MOV | **r32** | DRn |  |  |  |  | o..szapc |  | o..szapc |  | Move to/from Debug Registers |
|  | 0F | 22 |  | r | 03+ |  |  | 0 |  | MOV | **CRn** | r32 |  |  |  |  | o..szapc |  | o..szapc |  | Move to/from Control Registers |
|  | 0F | 22 |  | r | 03+ | U[16](http://ref.x86asm.net/coder32.html#gen_note_u_MOV_CR_DR_TR_0F20_0F21_0F22_0F23_0F24_0F26) |  | 0 |  | MOV | **CRn** | r32 |  |  |  |  | o..szapc |  | o..szapc |  | Move to/from Control Registers |
|  | 0F | 23 |  | r | 03+ |  |  | 0 |  | MOV | **DRn** | r32 |  |  |  |  | o..szapc |  | o..szapc |  | Move to/from Debug Registers |
|  | 0F | 23 |  | r | 03+ | U[16](http://ref.x86asm.net/coder32.html#gen_note_u_MOV_CR_DR_TR_0F20_0F21_0F22_0F23_0F24_0F26) |  | 0 |  | MOV | **DRn** | r64 |  |  |  |  | o..szapc |  | o..szapc |  | Move to/from Debug Registers |
|  | 0F | 28 |  | r | P3+ |  |  |  |  | MOVAPS | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Move Aligned Packed Single-FP Values |
| 66 | 0F | 28 |  | r | P4+ |  |  |  |  | MOVAPD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Move Aligned Packed Double-FP Values |
|  | 0F | 29 |  | r | P3+ |  |  |  |  | MOVAPS | **xmm/m128** | xmm |  |  | sse1 |  |  |  |  |  | Move Aligned Packed Single-FP Values |
| 66 | 0F | 29 |  | r | P4+ |  |  |  |  | MOVAPD | **xmm/m128** | xmm |  |  | sse2 |  |  |  |  |  | Move Aligned Packed Double-FP Values |
|  | 0F | 2A |  | r | P3+ |  |  |  |  | CVTPI2PS | **xmm** | mm/m64 |  |  | sse1 |  |  |  |  |  | Convert Packed DW Integers to Single-FP Values |
| F3 | 0F | 2A |  | r | P3+ |  |  |  |  | CVTSI2SS | **xmm** | r/m32 |  |  | sse1 |  |  |  |  |  | Convert DW Integer to Scalar Single-FP Value |
| 66 | 0F | 2A |  | r | P4+ |  |  |  |  | CVTPI2PD | **xmm** | mm/m64 |  |  | sse2 |  |  |  |  |  | Convert Packed DW Integers to Double-FP Values |
| F2 | 0F | 2A |  | r | P4+ |  |  |  |  | CVTSI2SD | **xmm** | r/m32 |  |  | sse2 |  |  |  |  |  | Convert DW Integer to Scalar Double-FP Value |
|  | 0F | 2B |  | r | P3+ |  |  |  |  | MOVNTPS | **m128** | xmm |  |  | sse1 |  |  |  |  |  | Store Packed Single-FP Values Using Non-Temporal Hint |
| 66 | 0F | 2B |  | r | P4+ |  |  |  |  | MOVNTPD | **m128** | xmm |  |  | sse2 |  |  |  |  |  | Store Packed Double-FP Values Using Non-Temporal Hint |
|  | 0F | 2C |  | r | P3+ |  |  |  |  | CVTTPS2PI | **mm** | xmm/m64 |  |  | sse1 |  |  |  |  |  | Convert with Trunc. Packed Single-FP Values to DW Integers |
| F3 | 0F | 2C |  | r | P3+ |  |  |  |  | CVTTSS2SI | **r32** | xmm/m32 |  |  | sse1 |  |  |  |  |  | Convert with Trunc. Scalar Single-FP Value to DW Integer |
| 66 | 0F | 2C |  | r | P4+ |  |  |  |  | CVTTPD2PI | **mm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Convert with Trunc. Packed Double-FP Values to DW Integers |
| F2 | 0F | 2C |  | r | P4+ |  |  |  |  | CVTTSD2SI | **r32** | xmm/m64 |  |  | sse2 |  |  |  |  |  | Conv. with Trunc. Scalar Double-FP Value to Signed DW Int |
|  | 0F | 2D |  | r | P3+ |  |  |  |  | CVTPS2PI | **mm** | xmm/m64 |  |  | sse1 |  |  |  |  |  | Convert Packed Single-FP Values to DW Integers |
| F3 | 0F | 2D |  | r | P3+ |  |  |  |  | CVTSS2SI | **r32** | xmm/m32 |  |  | sse1 |  |  |  |  |  | Convert Scalar Single-FP Value to DW Integer |
| 66 | 0F | 2D |  | r | P4+ |  |  |  |  | CVTPD2PI | **mm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Convert Packed Double-FP Values to DW Integers |
| F2 | 0F | 2D |  | r | P4+ |  |  |  |  | CVTSD2SI | **r32** | xmm/m64 |  |  | sse2 |  |  |  |  |  | Convert Scalar Double-FP Value to DW Integer |
|  | 0F | 2E |  | r | P3+ |  |  |  |  | UCOMISS | xmm | xmm/m32 |  |  | sse1 |  | ....z.pc | ....z.pc |  |  | Unordered Compare Scalar Single-FP Values and Set EFLAGS |
| 66 | 0F | 2E |  | r | P4+ |  |  |  |  | UCOMISD | xmm | xmm/m64 |  |  | sse2 |  | ....z.pc | ....z.pc |  |  | Unordered Compare Scalar Double-FP Values and Set EFLAGS |
|  | 0F | 2F |  | r | P3+ |  |  |  |  | COMISS | xmm | xmm/m32 |  |  | sse1 |  | ....z.pc | ....z.pc |  |  | Compare Scalar Ordered Single-FP Values and Set EFLAGS |
| 66 | 0F | 2F |  | r | P4+ |  |  |  |  | COMISD | xmm | xmm/m64 |  |  | sse2 |  | ....z.pc | ....z.pc |  |  | Compare Scalar Ordered Double-FP Values and Set EFLAGS |
|  | 0F | 30 |  |  | P1+ |  |  | 0 |  | WRMSR | ***MSR*** | *ECX* | *EAX* | *EDX* |  |  |  |  |  |  | Write to Model Specific Register |
|  | 0F | 31 |  |  | P1+ |  |  | f[2](http://ref.x86asm.net/coder32.html#cr4_tsd) |  | RDTSC | ***EAX*** | ***EDX*** | *IA32\_TIM…* |  |  |  |  |  |  |  | Read Time-Stamp Counter |
|  | 0F | 32 |  |  | P1+ |  |  | 0 |  | RDMSR | ***EAX*** | ***EDX*** | *ECX* | *MSR* |  |  |  |  |  |  | Read from Model Specific Register |
|  | 0F | 33 |  |  | PX+ |  |  | f[3](http://ref.x86asm.net/coder32.html#cr4_pce) |  | RDPMC | ***EAX*** | ***EDX*** | *PMC* |  |  |  |  |  |  |  | Read Performance-Monitoring Counters |
|  | 0F | 34 |  |  | P2+ |  | P |  |  | SYSENTER | ***SS*** | ***ESP*** | *IA32\_SYS…* | ... |  |  | ..i..... | ..i..... |  | ..i..... | Fast System Call |
|  | 0F | 35 |  |  | P2+ |  | P | 0 |  | SYSEXIT | ***SS*** | ***eSP*** | *IA32\_SYS…* | ... |  |  |  |  |  |  | Fast Return from Fast System Call |
|  | 0F | 37 |  |  | C2++ | D[17](http://ref.x86asm.net/coder32.html#gen_note_GETSEC_0F37) |  |  |  | GETSEC | *EAX* |  |  |  | smx |  |  |  |  |  | GETSEC Leaf Functions |
|  | 0F | 38 | 00 | r | C2+ |  |  |  |  | PSHUFB | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Packed Shuffle Bytes |
| 66 | 0F | 38 | 00 | r | C2+ |  |  |  |  | PSHUFB | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Packed Shuffle Bytes |
|  | 0F | 38 | 01 | r | C2+ |  |  |  |  | PHADDW | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Packed Horizontal Add |
| 66 | 0F | 38 | 01 | r | C2+ |  |  |  |  | PHADDW | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Packed Horizontal Add |
|  | 0F | 38 | 02 | r | C2+ |  |  |  |  | PHADDD | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Packed Horizontal Add |
| 66 | 0F | 38 | 02 | r | C2+ |  |  |  |  | PHADDD | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Packed Horizontal Add |
|  | 0F | 38 | 03 | r | C2+ |  |  |  |  | PHADDSW | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Packed Horizontal Add and Saturate |
| 66 | 0F | 38 | 03 | r | C2+ |  |  |  |  | PHADDSW | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Packed Horizontal Add and Saturate |
|  | 0F | 38 | 04 | r | C2+ |  |  |  |  | PMADDUBSW | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Multiply and Add Packed Signed and Unsigned Bytes |
| 66 | 0F | 38 | 04 | r | C2+ |  |  |  |  | PMADDUBSW | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Multiply and Add Packed Signed and Unsigned Bytes |
|  | 0F | 38 | 05 | r | C2+ |  |  |  |  | PHSUBW | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Packed Horizontal Subtract |
| 66 | 0F | 38 | 05 | r | C2+ |  |  |  |  | PHSUBW | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Packed Horizontal Subtract |
|  | 0F | 38 | 06 | r | C2+ |  |  |  |  | PHSUBD | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Packed Horizontal Subtract |
| 66 | 0F | 38 | 06 | r | C2+ |  |  |  |  | PHSUBD | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Packed Horizontal Subtract |
|  | 0F | 38 | 07 | r | C2+ |  |  |  |  | PHSUBSW | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Packed Horizontal Subtract and Saturate |
| 66 | 0F | 38 | 07 | r | C2+ |  |  |  |  | PHSUBSW | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Packed Horizontal Subtract and Saturate |
|  | 0F | 38 | 08 | r | C2+ |  |  |  |  | PSIGNB | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Packed SIGN |
| 66 | 0F | 38 | 08 | r | C2+ |  |  |  |  | PSIGNB | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Packed SIGN |
|  | 0F | 38 | 09 | r | C2+ |  |  |  |  | PSIGNW | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Packed SIGN |
| 66 | 0F | 38 | 09 | r | C2+ |  |  |  |  | PSIGNW | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Packed SIGN |
|  | 0F | 38 | 0A | r | C2+ |  |  |  |  | PSIGND | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Packed SIGN |
| 66 | 0F | 38 | 0A | r | C2+ |  |  |  |  | PSIGND | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Packed SIGN |
|  | 0F | 38 | 0B | r | C2+ |  |  |  |  | PMULHRSW | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Packed Multiply High with Round and Scale |
| 66 | 0F | 38 | 0B | r | C2+ |  |  |  |  | PMULHRSW | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Packed Multiply High with Round and Scale |
| 66 | 0F | 38 | 10 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PBLENDVB | **xmm** | xmm/m128 | *XMM0* |  | sse41 |  |  |  |  |  | Variable Blend Packed Bytes |
| 66 | 0F | 38 | 14 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | BLENDVPS | **xmm** | xmm/m128 | *XMM0* |  | sse41 |  |  |  |  |  | Variable Blend Packed Single-FP Values |
| 66 | 0F | 38 | 15 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | BLENDVPD | **xmm** | xmm/m128 | *XMM0* |  | sse41 |  |  |  |  |  | Variable Blend Packed Double-FP Values |
| 66 | 0F | 38 | 17 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PTEST | xmm | xmm/m128 |  |  | sse41 |  | o..szapc | o..szapc |  | o..s.ap. | Logical Compare |
|  | 0F | 38 | 1C | r | C2+ |  |  |  |  | PABSB | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Packed Absolute Value |
| 66 | 0F | 38 | 1C | r | C2+ |  |  |  |  | PABSB | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Packed Absolute Value |
|  | 0F | 38 | 1D | r | C2+ |  |  |  |  | PABSW | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Packed Absolute Value |
| 66 | 0F | 38 | 1D | r | C2+ |  |  |  |  | PABSW | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Packed Absolute Value |
|  | 0F | 38 | 1E | r | C2+ |  |  |  |  | PABSD | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Packed Absolute Value |
| 66 | 0F | 38 | 1E | r | C2+ |  |  |  |  | PABSD | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Packed Absolute Value |
| 66 | 0F | 38 | 20 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMOVSXBW | **xmm** | m64 |  |  | sse41 |  |  |  |  |  | Packed Move with Sign Extend |
| PMOVSXBW | **xmm** | xmm |  |  |
| 66 | 0F | 38 | 21 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMOVSXBD | **xmm** | m32 |  |  | sse41 |  |  |  |  |  | Packed Move with Sign Extend |
| PMOVSXBD | **xmm** | xmm |  |  |
| 66 | 0F | 38 | 22 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMOVSXBQ | **xmm** | m16 |  |  | sse41 |  |  |  |  |  | Packed Move with Sign Extend |
| PMOVSXBQ | **xmm** | xmm |  |  |
| 66 | 0F | 38 | 23 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMOVSXWD | **xmm** | m64 |  |  | sse41 |  |  |  |  |  | Packed Move with Sign Extend |
| PMOVSXWD | **xmm** | xmm |  |  |
| 66 | 0F | 38 | 24 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMOVSXWQ | **xmm** | m32 |  |  | sse41 |  |  |  |  |  | Packed Move with Sign Extend |
| PMOVSXWQ | **xmm** | xmm |  |  |
| 66 | 0F | 38 | 25 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMOVSXDQ | **xmm** | m64 |  |  | sse41 |  |  |  |  |  | Packed Move with Sign Extend |
| PMOVSXDQ | **xmm** | xmm |  |  |
| 66 | 0F | 38 | 28 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMULDQ | **xmm** | xmm/m128 |  |  | sse41 |  |  |  |  |  | Multiply Packed Signed Dword Integers |
| 66 | 0F | 38 | 29 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PCMPEQQ | **xmm** | xmm/m128 |  |  | sse41 |  |  |  |  |  | Compare Packed Qword Data for Equal |
| 66 | 0F | 38 | 2A | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | MOVNTDQA | **xmm** | m128 |  |  | sse41 |  |  |  |  |  | Load Double Quadword Non-Temporal Aligned Hint |
| 66 | 0F | 38 | 2B | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PACKUSDW | **xmm** | xmm/m128 |  |  | sse41 |  |  |  |  |  | Pack with Unsigned Saturation |
| 66 | 0F | 38 | 30 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMOVZXBW | **xmm** | m64 |  |  | sse41 |  |  |  |  |  | Packed Move with Zero Extend |
| PMOVZXBW | **xmm** | xmm |  |  |
| 66 | 0F | 38 | 31 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMOVZXBD | **xmm** | m32 |  |  | sse41 |  |  |  |  |  | Packed Move with Zero Extend |
| PMOVZXBD | **xmm** | xmm |  |  |
| 66 | 0F | 38 | 32 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMOVZXBQ | **xmm** | m16 |  |  | sse41 |  |  |  |  |  | Packed Move with Zero Extend |
| PMOVZXBQ | **xmm** | xmm |  |  |
| 66 | 0F | 38 | 33 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMOVZXWD | **xmm** | m64 |  |  | sse41 |  |  |  |  |  | Packed Move with Zero Extend |
| PMOVZXWD | **xmm** | xmm |  |  |
| 66 | 0F | 38 | 34 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMOVZXWQ | **xmm** | m32 |  |  | sse41 |  |  |  |  |  | Packed Move with Zero Extend |
| PMOVZXWQ | **xmm** | xmm |  |  |
| 66 | 0F | 38 | 35 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMOVZXDQ | **xmm** | m64 |  |  | sse41 |  |  |  |  |  | Packed Move with Zero Extend |
| PMOVZXDQ | **xmm** | xmm |  |  |
| 66 | 0F | 38 | 37 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PCMPGTQ | **xmm** | xmm/m128 |  |  | sse42 |  |  |  |  |  | Compare Packed Qword Data for Greater Than |
| 66 | 0F | 38 | 38 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMINSB | **xmm** | xmm/m128 |  |  | sse41 |  |  |  |  |  | Minimum of Packed Signed Byte Integers |
| 66 | 0F | 38 | 39 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMINSD | **xmm** | xmm/m128 |  |  | sse41 |  |  |  |  |  | Minimum of Packed Signed Dword Integers |
| 66 | 0F | 38 | 3A | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMINUW | **xmm** | xmm/m128 |  |  | sse41 |  |  |  |  |  | Minimum of Packed Unsigned Word Integers |
| 66 | 0F | 38 | 3B | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMINUD | **xmm** | xmm/m128 |  |  | sse41 |  |  |  |  |  | Minimum of Packed Unsigned Dword Integers |
| 66 | 0F | 38 | 3C | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMAXSB | **xmm** | xmm/m128 |  |  | sse41 |  |  |  |  |  | Maximum of Packed Signed Byte Integers |
| 66 | 0F | 38 | 3D | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMAXSD | **xmm** | xmm/m128 |  |  | sse41 |  |  |  |  |  | Maximum of Packed Signed Dword Integers |
| 66 | 0F | 38 | 3E | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMAXUW | **xmm** | xmm/m128 |  |  | sse41 |  |  |  |  |  | Maximum of Packed Unsigned Word Integers |
| 66 | 0F | 38 | 3F | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMAXUD | **xmm** | xmm/m128 |  |  | sse41 |  |  |  |  |  | Maximum of Packed Unsigned Dword Integers |
| 66 | 0F | 38 | 40 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PMULLD | **xmm** | xmm/m128 |  |  | sse41 |  |  |  |  |  | Multiply Packed Signed Dword Integers and Store Low Result |
| 66 | 0F | 38 | 41 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PHMINPOSUW | **xmm** | xmm/m128 |  |  | sse41 |  |  |  |  |  | Packed Horizontal Word Minimum |
| 66 | 0F | 38 | 80 | r | C2++ | D[24](http://ref.x86asm.net/coder32.html#gen_note_VMX_vs_SVM) | P | 0 |  | INVEPT | r32 | m128 |  |  | vmx |  | o..szapc | o..szapc |  |  | Invalidate Translations Derived from EPT |
| 66 | 0F | 38 | 81 | r | C2++ | D[24](http://ref.x86asm.net/coder32.html#gen_note_VMX_vs_SVM) | P | 0 |  | INVVPID | r32 | m128 |  |  | vmx |  | o..szapc | o..szapc |  |  | Invalidate Translations Based on VPID |
|  | 0F | 38 | F0 | r | C2++ |  |  |  |  | MOVBE | **r16/32** | m16/32 |  |  |  |  |  |  |  |  | Move Data After Swapping Bytes |
| F2 | 0F | 38 | F0 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | CRC32 | **r32** | r/m8 |  |  | sse42 |  |  |  |  |  | Accumulate CRC32 Value |
|  | 0F | 38 | F1 | r | C2++ |  |  |  |  | MOVBE | **m16/32** | r16/32 |  |  |  |  |  |  |  |  | Move Data After Swapping Bytes |
| F2 | 0F | 38 | F1 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | CRC32 | **r32** | r/m16/32 |  |  | sse42 |  |  |  |  |  | Accumulate CRC32 Value |
| 66 | 0F | 3A | 08 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | ROUNDPS | **xmm** | xmm/m128 | imm8 |  | sse41 |  |  |  |  |  | Round Packed Single-FP Values |
| 66 | 0F | 3A | 09 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | ROUNDPD | **xmm** | xmm/m128 | imm8 |  | sse41 |  |  |  |  |  | Round Packed Double-FP Values |
| 66 | 0F | 3A | 0A | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | ROUNDSS | **xmm** | xmm/m32 | imm8 |  | sse41 |  |  |  |  |  | Round Scalar Single-FP Values |
| 66 | 0F | 3A | 0B | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | ROUNDSD | **xmm** | xmm/m64 | imm8 |  | sse41 |  |  |  |  |  | Round Scalar Double-FP Values |
| 66 | 0F | 3A | 0C | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | BLENDPS | **xmm** | xmm/m128 | imm8 |  | sse41 |  |  |  |  |  | Blend Packed Single-FP Values |
| 66 | 0F | 3A | 0D | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | BLENDPD | **xmm** | xmm/m128 | imm8 |  | sse41 |  |  |  |  |  | Blend Packed Double-FP Values |
| 66 | 0F | 3A | 0E | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PBLENDW | **xmm** | xmm/m128 | imm8 |  | sse41 |  |  |  |  |  | Blend Packed Words |
|  | 0F | 3A | 0F | r | C2+ |  |  |  |  | PALIGNR | **mm** | mm/m64 |  |  | ssse3 |  |  |  |  |  | Packed Align Right |
| 66 | 0F | 3A | 0F | r | C2+ |  |  |  |  | PALIGNR | **xmm** | xmm/m128 |  |  | ssse3 |  |  |  |  |  | Packed Align Right |
| 66 | 0F | 3A | 14 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PEXTRB | **m8** | xmm | imm8 |  | sse41 |  |  |  |  |  | Extract Byte |
| PEXTRB | **r32** | xmm | imm8 |  |
| 66 | 0F | 3A | 15 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PEXTRW | **m16** | xmm | imm8 |  | sse41 |  |  |  |  |  | Extract Word |
| PEXTRW | **r32** | xmm | imm8 |  |
| 66 | 0F | 3A | 16 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PEXTRD | **r/m32** | xmm | imm8 |  | sse41 |  |  |  |  |  | Extract Dword/Qword |
| PEXTRQ | **r/m64** | xmm | imm8 |  |
| 66 | 0F | 3A | 17 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | EXTRACTPS | **r/m32** | xmm | imm8 |  | sse41 |  |  |  |  |  | Extract Packed Single-FP Value |
| 66 | 0F | 3A | 20 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PINSRB | **xmm** | m8 | imm8 |  | sse41 |  |  |  |  |  | Insert Byte |
| PINSRB | **xmm** | r32 | imm8 |  |
| 66 | 0F | 3A | 21 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | INSERTPS | **xmm** | xmm | imm8 |  | sse41 |  |  |  |  |  | Insert Packed Single-FP Value |
| INSERTPS | **xmm** | m32 | imm8 |  |
| 66 | 0F | 3A | 22 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PINSRD | **xmm** | r/m32 | imm8 |  | sse41 |  |  |  |  |  | Insert Dword/Qword |
| PINSRQ | **xmm** | r/m64 | imm8 |  |
| 66 | 0F | 3A | 40 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | DPPS | **xmm** | xmm/m128 |  |  | sse41 |  |  |  |  |  | Dot Product of Packed Single-FP Values |
| 66 | 0F | 3A | 41 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | DPPD | **xmm** | xmm/m128 |  |  | sse41 |  |  |  |  |  | Dot Product of Packed Double-FP Values |
| 66 | 0F | 3A | 42 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | MPSADBW | **xmm** | xmm/m128 | imm8 |  | sse41 |  |  |  |  |  | Compute Multiple Packed Sums of Absolute Difference |
| 66 | 0F | 3A | 60 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PCMPESTRM | ***XMM0*** | xmm | xmm/m128 | ... | sse42 |  | o..szapc | o..szapc |  | .....ap. | Packed Compare Explicit Length Strings, Return Mask |
| 66 | 0F | 3A | 61 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PCMPESTRI | ***ECX*** | xmm | xmm/m128 | ... | sse42 |  | o..szapc | o..szapc |  | .....ap. | Packed Compare Explicit Length Strings, Return Index |
| 66 | 0F | 3A | 62 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PCMPISTRM | ***XMM0*** | xmm | xmm/m128 | imm8 | sse42 |  | o..szapc | o..szapc |  | .....ap. | Packed Compare Implicit Length Strings, Return Mask |
| 66 | 0F | 3A | 63 | r | C2++ | D[25](http://ref.x86asm.net/coder32.html#gen_note_SSE4_amd) |  |  |  | PCMPISTRI | ***ECX*** | xmm | xmm/m128 | imm8 | sse42 |  | o..szapc | o..szapc |  | .....ap. | Packed Compare Implicit Length Strings, Return Index |
|  | 0F | 40 |  | r | PP+ |  |  |  |  | CMOVO | **r16/32** | r/m16/32 |  |  |  | o....... |  |  |  |  | Conditional Move - overflow (OF=1) |
|  | 0F | 41 |  | r | PP+ |  |  |  |  | CMOVNO | **r16/32** | r/m16/32 |  |  |  | o....... |  |  |  |  | Conditional Move - not overflow (OF=0) |
|  | 0F | 42 |  | r | PP+ |  |  |  |  | CMOVB | **r16/32** | r/m16/32 |  |  |  | .......c |  |  |  |  | Conditional Move - below/not above or equal/carry (CF=1) |
| CMOVNAE | **r16/32** | r/m16/32 |  |  |
| CMOVC | **r16/32** | r/m16/32 |  |  |
|  | 0F | 43 |  | r | PP+ |  |  |  |  | CMOVNB | **r16/32** | r/m16/32 |  |  |  | .......c |  |  |  |  | Conditional Move - not below/above or equal/not carry (CF=0) |
| CMOVAE | **r16/32** | r/m16/32 |  |  |
| CMOVNC | **r16/32** | r/m16/32 |  |  |
|  | 0F | 44 |  | r | PP+ |  |  |  |  | CMOVZ | **r16/32** | r/m16/32 |  |  |  | ....z... |  |  |  |  | Conditional Move - zero/equal (ZF=0) |
| CMOVE | **r16/32** | r/m16/32 |  |  |
|  | 0F | 45 |  | r | PP+ |  |  |  |  | CMOVNZ | **r16/32** | r/m16/32 |  |  |  | ....z... |  |  |  |  | Conditional Move - not zero/not equal (ZF=1) |
| CMOVNE | **r16/32** | r/m16/32 |  |  |
|  | 0F | 46 |  | r | PP+ |  |  |  |  | CMOVBE | **r16/32** | r/m16/32 |  |  |  | ....z..c |  |  |  |  | Conditional Move - below or equal/not above (CF=1 AND ZF=1) |
| CMOVNA | **r16/32** | r/m16/32 |  |  |
|  | 0F | 47 |  | r | PP+ |  |  |  |  | CMOVNBE | **r16/32** | r/m16/32 |  |  |  | ....z..c |  |  |  |  | Conditional Move - not below or equal/above (CF=0 AND ZF=0) |
| CMOVA | **r16/32** | r/m16/32 |  |  |
|  | 0F | 48 |  | r | PP+ |  |  |  |  | CMOVS | **r16/32** | r/m16/32 |  |  |  | ...s.... |  |  |  |  | Conditional Move - sign (SF=1) |
|  | 0F | 49 |  | r | PP+ |  |  |  |  | CMOVNS | **r16/32** | r/m16/32 |  |  |  | ...s.... |  |  |  |  | Conditional Move - not sign (SF=0) |
|  | 0F | 4A |  | r | PP+ |  |  |  |  | CMOVP | **r16/32** | r/m16/32 |  |  |  | ......p. |  |  |  |  | Conditional Move - parity/parity even (PF=1) |
| CMOVPE | **r16/32** | r/m16/32 |  |  |
|  | 0F | 4B |  | r | PP+ |  |  |  |  | CMOVNP | **r16/32** | r/m16/32 |  |  |  | ......p. |  |  |  |  | Conditional Move - not parity/parity odd |
| CMOVPO | **r16/32** | r/m16/32 |  |  |
|  | 0F | 4C |  | r | PP+ |  |  |  |  | CMOVL | **r16/32** | r/m16/32 |  |  |  | o..s.... |  |  |  |  | Conditional Move - less/not greater (SF!=OF) |
| CMOVNGE | **r16/32** | r/m16/32 |  |  |
|  | 0F | 4D |  | r | PP+ |  |  |  |  | CMOVNL | **r16/32** | r/m16/32 |  |  |  | o..s.... |  |  |  |  | Conditional Move - not less/greater or equal (SF=OF) |
| CMOVGE | **r16/32** | r/m16/32 |  |  |
|  | 0F | 4E |  | r | PP+ |  |  |  |  | CMOVLE | **r16/32** | r/m16/32 |  |  |  | o..sz... |  |  |  |  | Conditional Move - less or equal/not greater ((ZF=1) OR (SF!=OF)) |
| CMOVNG | **r16/32** | r/m16/32 |  |  |
|  | 0F | 4F |  | r | PP+ |  |  |  |  | CMOVNLE | **r16/32** | r/m16/32 |  |  |  | o..sz... |  |  |  |  | Conditional Move - not less nor equal/greater ((ZF=0) AND (SF=OF)) |
| CMOVG | **r16/32** | r/m16/32 |  |  |
|  | 0F | 50 |  | r | P3+ |  |  |  |  | MOVMSKPS | **r32** | xmm |  |  | sse1 |  |  |  |  |  | Extract Packed Single-FP Sign Mask |
| 66 | 0F | 50 |  | r | P4+ |  |  |  |  | MOVMSKPD | **r32** | xmm |  |  | sse2 |  |  |  |  |  | Extract Packed Double-FP Sign Mask |
|  | 0F | 51 |  | r | P3+ |  |  |  |  | SQRTPS | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Compute Square Roots of Packed Single-FP Values |
| F3 | 0F | 51 |  | r | P3+ |  |  |  |  | SQRTSS | **xmm** | xmm/m32 |  |  | sse1 |  |  |  |  |  | Compute Square Root of Scalar Single-FP Value |
| 66 | 0F | 51 |  | r | P4+ |  |  |  |  | SQRTPD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Compute Square Roots of Packed Double-FP Values |
| F2 | 0F | 51 |  | r | P4+ |  |  |  |  | SQRTSD | **xmm** | xmm/m64 |  |  | sse2 |  |  |  |  |  | Compute Square Root of Scalar Double-FP Value |
|  | 0F | 52 |  | r | P3+ |  |  |  |  | RSQRTPS | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Compute Recipr. of Square Roots of Packed Single-FP Values |
| F3 | 0F | 52 |  | r | P3+ |  |  |  |  | RSQRTSS | **xmm** | xmm/m32 |  |  | sse1 |  |  |  |  |  | Compute Recipr. of Square Root of Scalar Single-FP Value |
|  | 0F | 53 |  | r | P3+ |  |  |  |  | RCPPS | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Compute Reciprocals of Packed Single-FP Values |
| F3 | 0F | 53 |  | r | P3+ |  |  |  |  | RCPSS | **xmm** | xmm/m32 |  |  | sse1 |  |  |  |  |  | Compute Reciprocal of Scalar Single-FP Values |
|  | 0F | 54 |  | r | P3+ |  |  |  |  | ANDPS | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Bitwise Logical AND of Packed Single-FP Values |
| 66 | 0F | 54 |  | r | P4+ |  |  |  |  | ANDPD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Bitwise Logical AND of Packed Double-FP Values |
|  | 0F | 55 |  | r | P3+ |  |  |  |  | ANDNPS | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Bitwise Logical AND NOT of Packed Single-FP Values |
| 66 | 0F | 55 |  | r | P4+ |  |  |  |  | ANDNPD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Bitwise Logical AND NOT of Packed Double-FP Values |
|  | 0F | 56 |  | r | P3+ |  |  |  |  | ORPS | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Bitwise Logical OR of Single-FP Values |
| 66 | 0F | 56 |  | r | P4+ |  |  |  |  | ORPD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Bitwise Logical OR of Double-FP Values |
|  | 0F | 57 |  | r | P3+ |  |  |  |  | XORPS | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Bitwise Logical XOR for Single-FP Values |
| 66 | 0F | 57 |  | r | P4+ |  |  |  |  | XORPD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Bitwise Logical XOR for Double-FP Values |
|  | 0F | 58 |  | r | P3+ |  |  |  |  | ADDPS | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Add Packed Single-FP Values |
| F3 | 0F | 58 |  | r | P3+ |  |  |  |  | ADDSS | **xmm** | xmm/m32 |  |  | sse1 |  |  |  |  |  | Add Scalar Single-FP Values |
| 66 | 0F | 58 |  | r | P4+ |  |  |  |  | ADDPD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Add Packed Double-FP Values |
| F2 | 0F | 58 |  | r | P4+ |  |  |  |  | ADDSD | **xmm** | xmm/m64 |  |  | sse2 |  |  |  |  |  | Add Scalar Double-FP Values |
|  | 0F | 59 |  | r | P3+ |  |  |  |  | MULPS | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Multiply Packed Single-FP Values |
| F3 | 0F | 59 |  | r | P3+ |  |  |  |  | MULSS | **xmm** | xmm/m32 |  |  | sse1 |  |  |  |  |  | Multiply Scalar Single-FP Value |
| 66 | 0F | 59 |  | r | P4+ |  |  |  |  | MULPD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Multiply Packed Double-FP Values |
| F2 | 0F | 59 |  | r | P4+ |  |  |  |  | MULSD | **xmm** | xmm/m64 |  |  | sse2 |  |  |  |  |  | Multiply Scalar Double-FP Values |
|  | 0F | 5A |  | r | P4+ |  |  |  |  | CVTPS2PD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Convert Packed Single-FP Values to Double-FP Values |
| 66 | 0F | 5A |  | r | P4+ |  |  |  |  | CVTPD2PS | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Convert Packed Double-FP Values to Single-FP Values |
| F3 | 0F | 5A |  | r | P4+ |  |  |  |  | CVTSS2SD | **xmm** | xmm/m32 |  |  | sse2 |  |  |  |  |  | Convert Scalar Single-FP Value to Scalar Double-FP Value |
| F2 | 0F | 5A |  | r | P4+ |  |  |  |  | CVTSD2SS | **xmm** | xmm/m64 |  |  | sse2 |  |  |  |  |  | Convert Scalar Double-FP Value to Scalar Single-FP Value |
|  | 0F | 5B |  | r | P4+ |  |  |  |  | CVTDQ2PS | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Convert Packed DW Integers to Single-FP Values |
| 66 | 0F | 5B |  | r | P4+ |  |  |  |  | CVTPS2DQ | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Convert Packed Single-FP Values to DW Integers |
| F3 | 0F | 5B |  | r | P4+ |  |  |  |  | CVTTPS2DQ | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Convert with Trunc. Packed Single-FP Values to DW Integers |
|  | 0F | 5C |  | r | P3+ |  |  |  |  | SUBPS | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Subtract Packed Single-FP Values |
| F3 | 0F | 5C |  | r | P3+ |  |  |  |  | SUBSS | **xmm** | xmm/m32 |  |  | sse1 |  |  |  |  |  | Subtract Scalar Single-FP Values |
| 66 | 0F | 5C |  | r | P4+ |  |  |  |  | SUBPD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Subtract Packed Double-FP Values |
| F2 | 0F | 5C |  | r | P4+ |  |  |  |  | SUBSD | **xmm** | xmm/m64 |  |  | sse2 |  |  |  |  |  | Subtract Scalar Double-FP Values |
|  | 0F | 5D |  | r | P3+ |  |  |  |  | MINPS | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Return Minimum Packed Single-FP Values |
| F3 | 0F | 5D |  | r | P3+ |  |  |  |  | MINSS | **xmm** | xmm/m32 |  |  | sse1 |  |  |  |  |  | Return Minimum Scalar Single-FP Value |
| 66 | 0F | 5D |  | r | P4+ |  |  |  |  | MINPD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Return Minimum Packed Double-FP Values |
| F2 | 0F | 5D |  | r | P4+ |  |  |  |  | MINSD | **xmm** | xmm/m64 |  |  | sse2 |  |  |  |  |  | Return Minimum Scalar Double-FP Value |
|  | 0F | 5E |  | r | P3+ |  |  |  |  | DIVPS | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Divide Packed Single-FP Values |
| F3 | 0F | 5E |  | r | P3+ |  |  |  |  | DIVSS | **xmm** | xmm/m32 |  |  | sse1 |  |  |  |  |  | Divide Scalar Single-FP Values |
| 66 | 0F | 5E |  | r | P4+ |  |  |  |  | DIVPD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Divide Packed Double-FP Values |
| F2 | 0F | 5E |  | r | P4+ |  |  |  |  | DIVSD | **xmm** | xmm/m64 |  |  | sse2 |  |  |  |  |  | Divide Scalar Double-FP Values |
|  | 0F | 5F |  | r | P3+ |  |  |  |  | MAXPS | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Return Maximum Packed Single-FP Values |
| F3 | 0F | 5F |  | r | P3+ |  |  |  |  | MAXSS | **xmm** | xmm/m32 |  |  | sse1 |  |  |  |  |  | Return Maximum Scalar Single-FP Value |
| 66 | 0F | 5F |  | r | P4+ |  |  |  |  | MAXPD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Return Maximum Packed Double-FP Values |
| F2 | 0F | 5F |  | r | P4+ |  |  |  |  | MAXSD | **xmm** | xmm/m64 |  |  | sse2 |  |  |  |  |  | Return Maximum Scalar Double-FP Value |
|  | 0F | 60 |  | r | PX+ |  |  |  |  | PUNPCKLBW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Unpack Low Data |
| 66 | 0F | 60 |  | r | P4+ |  |  |  |  | PUNPCKLBW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Unpack Low Data |
|  | 0F | 61 |  | r | PX+ |  |  |  |  | PUNPCKLWD | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Unpack Low Data |
| 66 | 0F | 61 |  | r | P4+ |  |  |  |  | PUNPCKLWD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Unpack Low Data |
|  | 0F | 62 |  | r | PX+ |  |  |  |  | PUNPCKLDQ | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Unpack Low Data |
| 66 | 0F | 62 |  | r | P4+ |  |  |  |  | PUNPCKLDQ | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Unpack Low Data |
|  | 0F | 63 |  | r | PX+ |  |  |  |  | PACKSSWB | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Pack with Signed Saturation |
| 66 | 0F | 63 |  | r | P4+ |  |  |  |  | PACKSSWB | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Pack with Signed Saturation |
|  | 0F | 64 |  | r | PX+ |  |  |  |  | PCMPGTB | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Compare Packed Signed Integers for Greater Than |
| 66 | 0F | 64 |  | r | P4+ |  |  |  |  | PCMPGTB | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Compare Packed Signed Integers for Greater Than |
|  | 0F | 65 |  | r | PX+ |  |  |  |  | PCMPGTW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Compare Packed Signed Integers for Greater Than |
| 66 | 0F | 65 |  | r | P4+ |  |  |  |  | PCMPGTW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Compare Packed Signed Integers for Greater Than |
|  | 0F | 66 |  | r | PX+ |  |  |  |  | PCMPGTD | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Compare Packed Signed Integers for Greater Than |
| 66 | 0F | 66 |  | r | P4+ |  |  |  |  | PCMPGTD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Compare Packed Signed Integers for Greater Than |
|  | 0F | 67 |  | r | PX+ |  |  |  |  | PACKUSWB | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Pack with Unsigned Saturation |
| 66 | 0F | 67 |  | r | P4+ |  |  |  |  | PACKUSWB | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Pack with Unsigned Saturation |
|  | 0F | 68 |  | r | PX+ |  |  |  |  | PUNPCKHBW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Unpack High Data |
| 66 | 0F | 68 |  | r | P4+ |  |  |  |  | PUNPCKHBW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Unpack High Data |
|  | 0F | 69 |  | r | PX+ |  |  |  |  | PUNPCKHWD | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Unpack High Data |
| 66 | 0F | 69 |  | r | P4+ |  |  |  |  | PUNPCKHWD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Unpack High Data |
|  | 0F | 6A |  | r | PX+ |  |  |  |  | PUNPCKHDQ | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Unpack High Data |
| 66 | 0F | 6A |  | r | P4+ |  |  |  |  | PUNPCKHDQ | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Unpack High Data |
|  | 0F | 6B |  | r | PX+ |  |  |  |  | PACKSSDW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Pack with Signed Saturation |
| 66 | 0F | 6B |  | r | P4+ |  |  |  |  | PACKSSDW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Pack with Signed Saturation |
| 66 | 0F | 6C |  | r | P4+ |  |  |  |  | PUNPCKLQDQ | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Unpack Low Data |
| 66 | 0F | 6D |  | r | P4+ |  |  |  |  | PUNPCKHQDQ | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Unpack High Data |
|  | 0F | 6E |  | r | PX+ |  |  |  |  | MOVD | **mm** | r/m32 |  |  | mmx |  |  |  |  |  | Move Doubleword |
| 66 | 0F | 6E |  | r | P4+ |  |  |  |  | MOVD | **xmm** | r/m32 |  |  | sse2 |  |  |  |  |  | Move Doubleword |
|  | 0F | 6F |  | r | PX+ |  |  |  |  | MOVQ | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Move Quadword |
| 66 | 0F | 6F |  | r | P4+ |  |  |  |  | MOVDQA | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Move Aligned Double Quadword |
| F3 | 0F | 6F |  | r | P4+ |  |  |  |  | MOVDQU | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Move Unaligned Double Quadword |
|  | 0F | 70 |  | r | P3+ |  |  |  |  | PSHUFW | **mm** | mm/m64 | imm8 |  | sse1 |  |  |  |  |  | Shuffle Packed Words |
| F2 | 0F | 70 |  | r | P4+ |  |  |  |  | PSHUFLW | **xmm** | xmm/m128 | imm8 |  | sse2 |  |  |  |  |  | Shuffle Packed Low Words |
| F3 | 0F | 70 |  | r | P4+ |  |  |  |  | PSHUFHW | **xmm** | xmm/m128 | imm8 |  | sse2 |  |  |  |  |  | Shuffle Packed High Words |
| 66 | 0F | 70 |  | r | P4+ |  |  |  |  | PSHUFD | **xmm** | xmm/m128 | imm8 |  | sse2 |  |  |  |  |  | Shuffle Packed Doublewords |
|  | 0F | 71 |  | 2 | PX+ |  |  |  |  | PSRLW | **mm** | imm8 |  |  | mmx |  |  |  |  |  | Shift Packed Data Right Logical |
| 66 | 0F | 71 |  | 2 | P4+ |  |  |  |  | PSRLW | **xmm** | imm8 |  |  | sse2 |  |  |  |  |  | Shift Packed Data Right Logical |
|  | 0F | 71 |  | 4 | PX+ |  |  |  |  | PSRAW | **mm** | imm8 |  |  | mmx |  |  |  |  |  | Shift Packed Data Right Arithmetic |
| 66 | 0F | 71 |  | 4 | P4+ |  |  |  |  | PSRAW | **xmm** | imm8 |  |  | sse2 |  |  |  |  |  | Shift Packed Data Right Arithmetic |
|  | 0F | 71 |  | 6 | PX+ |  |  |  |  | PSLLW | **mm** | imm8 |  |  | mmx |  |  |  |  |  | Shift Packed Data Left Logical |
| 66 | 0F | 71 |  | 6 | P4+ |  |  |  |  | PSLLW | **xmm** | imm8 |  |  | sse2 |  |  |  |  |  | Shift Packed Data Left Logical |
|  | 0F | 72 |  | 2 | PX+ |  |  |  |  | PSRLD | **mm** | imm8 |  |  | mmx |  |  |  |  |  | Shift Double Quadword Right Logical |
| 66 | 0F | 72 |  | 2 | P4+ |  |  |  |  | PSRLD | **xmm** | imm8 |  |  | sse2 |  |  |  |  |  | Shift Double Quadword Right Logical |
|  | 0F | 72 |  | 4 | PX+ |  |  |  |  | PSRAD | **mm** | imm8 |  |  | mmx |  |  |  |  |  | Shift Packed Data Right Arithmetic |
| 66 | 0F | 72 |  | 4 | P4+ |  |  |  |  | PSRAD | **xmm** | imm8 |  |  | sse2 |  |  |  |  |  | Shift Packed Data Right Arithmetic |
|  | 0F | 72 |  | 6 | PX+ |  |  |  |  | PSLLD | **mm** | imm8 |  |  | mmx |  |  |  |  |  | Shift Packed Data Left Logical |
| 66 | 0F | 72 |  | 6 | P4+ |  |  |  |  | PSLLD | **xmm** | imm8 |  |  | sse2 |  |  |  |  |  | Shift Packed Data Left Logical |
|  | 0F | 73 |  | 2 | PX+ |  |  |  |  | PSRLQ | **mm** | imm8 |  |  | mmx |  |  |  |  |  | Shift Packed Data Right Logical |
| 66 | 0F | 73 |  | 2 | P4+ |  |  |  |  | PSRLQ | **xmm** | imm8 |  |  | sse2 |  |  |  |  |  | Shift Packed Data Right Logical |
| 66 | 0F | 73 |  | 3 | P4+ |  |  |  |  | PSRLDQ | **xmm** | imm8 |  |  | sse2 |  |  |  |  |  | Shift Double Quadword Right Logical |
|  | 0F | 73 |  | 6 | PX+ |  |  |  |  | PSLLQ | **mm** | imm8 |  |  | mmx |  |  |  |  |  | Shift Packed Data Left Logical |
| 66 | 0F | 73 |  | 6 | P4+ |  |  |  |  | PSLLQ | **xmm** | imm8 |  |  | sse2 |  |  |  |  |  | Shift Packed Data Left Logical |
| 66 | 0F | 73 |  | 7 | P4+ |  |  |  |  | PSLLDQ | **xmm** | imm8 |  |  | sse2 |  |  |  |  |  | Shift Double Quadword Left Logical |
|  | 0F | 74 |  | r | PX+ |  |  |  |  | PCMPEQB | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Compare Packed Data for Equal |
| 66 | 0F | 74 |  | r | P4+ |  |  |  |  | PCMPEQB | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Compare Packed Data for Equal |
|  | 0F | 75 |  | r | PX+ |  |  |  |  | PCMPEQW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Compare Packed Data for Equal |
| 66 | 0F | 75 |  | r | P4+ |  |  |  |  | PCMPEQW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Compare Packed Data for Equal |
|  | 0F | 76 |  | r | PX+ |  |  |  |  | PCMPEQD | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Compare Packed Data for Equal |
| 66 | 0F | 76 |  | r | P4+ |  |  |  |  | PCMPEQD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Compare Packed Data for Equal |
|  | 0F | 77 |  |  | PX+ |  |  |  |  | EMMS |  |  |  |  | mmx |  |  |  |  |  | Empty MMX Technology State |
|  | 0F | 78 |  | r | P4++ | D[24](http://ref.x86asm.net/coder32.html#gen_note_VMX_vs_SVM) | P | 0 |  | VMREAD | **r/m32** | r32 |  |  | vmx |  | o..szapc | o..szapc |  |  | Read Field from Virtual-Machine Control Structure |
|  | 0F | 79 |  | r | P4++ | D[24](http://ref.x86asm.net/coder32.html#gen_note_VMX_vs_SVM) | P | 0 |  | VMWRITE | r32 | r/m32 |  |  | vmx |  | o..szapc | o..szapc |  |  | Write Field to Virtual-Machine Control Structure |
| 66 | 0F | 7C |  | r | P4++ |  |  |  |  | HADDPD | **xmm** | xmm/m128 |  |  | sse3 |  |  |  |  |  | Packed Double-FP Horizontal Add |
| F2 | 0F | 7C |  | r | P4++ |  |  |  |  | HADDPS | **xmm** | xmm/m128 |  |  | sse3 |  |  |  |  |  | Packed Single-FP Horizontal Add |
| 66 | 0F | 7D |  | r | P4++ |  |  |  |  | HSUBPD | **xmm** | xmm/m128 |  |  | sse3 |  |  |  |  |  | Packed Double-FP Horizontal Subtract |
| F2 | 0F | 7D |  | r | P4++ |  |  |  |  | HSUBPS | **xmm** | xmm/m128 |  |  | sse3 |  |  |  |  |  | Packed Single-FP Horizontal Subtract |
|  | 0F | 7E |  | r | PX+ |  |  |  |  | MOVD | **r/m32** | mm |  |  | mmx |  |  |  |  |  | Move Doubleword |
| 66 | 0F | 7E |  | r | P4+ |  |  |  |  | MOVD | **r/m32** | xmm |  |  | sse2 |  |  |  |  |  | Move Doubleword |
| F3 | 0F | 7E |  | r | P4+ |  |  |  |  | MOVQ | **xmm** | xmm/m64 |  |  | sse2 |  |  |  |  |  | Move Quadword |
|  | 0F | 7F |  | r | PX+ |  |  |  |  | MOVQ | **mm/m64** | mm |  |  | mmx |  |  |  |  |  | Move Quadword |
| 66 | 0F | 7F |  | r | P4+ |  |  |  |  | MOVDQA | **xmm/m128** | xmm |  |  | sse2 |  |  |  |  |  | Move Aligned Double Quadword |
| F3 | 0F | 7F |  | r | P4+ |  |  |  |  | MOVDQU | **xmm/m128** | xmm |  |  | sse2 |  |  |  |  |  | Move Unaligned Double Quadword |
|  | 0F | 80 |  |  | 03+ |  |  |  |  | JO | rel16/32 |  |  |  |  | o....... |  |  |  |  | Jump short if overflow (OF=1) |
|  | 0F | 81 |  |  | 03+ |  |  |  |  | JNO | rel16/32 |  |  |  |  | o....... |  |  |  |  | Jump short if not overflow (OF=0) |
|  | 0F | 82 |  |  | 03+ |  |  |  |  | JB | rel16/32 |  |  |  |  | .......c |  |  |  |  | Jump short if below/not above or equal/carry (CF=1) |
| JNAE | rel16/32 |  |  |  |
| JC | rel16/32 |  |  |  |
|  | 0F | 83 |  |  | 03+ |  |  |  |  | JNB | rel16/32 |  |  |  |  | .......c |  |  |  |  | Jump short if not below/above or equal/not carry (CF=0) |
| JAE | rel16/32 |  |  |  |
| JNC | rel16/32 |  |  |  |
|  | 0F | 84 |  |  | 03+ |  |  |  |  | JZ | rel16/32 |  |  |  |  | ....z... |  |  |  |  | Jump short if zero/equal (ZF=0) |
| JE | rel16/32 |  |  |  |
|  | 0F | 85 |  |  | 03+ |  |  |  |  | JNZ | rel16/32 |  |  |  |  | ....z... |  |  |  |  | Jump short if not zero/not equal (ZF=1) |
| JNE | rel16/32 |  |  |  |
|  | 0F | 86 |  |  | 03+ |  |  |  |  | JBE | rel16/32 |  |  |  |  | ....z..c |  |  |  |  | Jump short if below or equal/not above (CF=1 AND ZF=1) |
| JNA | rel16/32 |  |  |  |
|  | 0F | 87 |  |  | 03+ |  |  |  |  | JNBE | rel16/32 |  |  |  |  | ....z..c |  |  |  |  | Jump short if not below or equal/above (CF=0 AND ZF=0) |
| JA | rel16/32 |  |  |  |
|  | 0F | 88 |  |  | 03+ |  |  |  |  | JS | rel16/32 |  |  |  |  | ...s.... |  |  |  |  | Jump short if sign (SF=1) |
|  | 0F | 89 |  |  | 03+ |  |  |  |  | JNS | rel16/32 |  |  |  |  | ...s.... |  |  |  |  | Jump short if not sign (SF=0) |
|  | 0F | 8A |  |  | 03+ |  |  |  |  | JP | rel16/32 |  |  |  |  | ......p. |  |  |  |  | Jump short if parity/parity even (PF=1) |
| JPE | rel16/32 |  |  |  |
|  | 0F | 8B |  |  | 03+ |  |  |  |  | JNP | rel16/32 |  |  |  |  | ......p. |  |  |  |  | Jump short if not parity/parity odd |
| JPO | rel16/32 |  |  |  |
|  | 0F | 8C |  |  | 03+ |  |  |  |  | JL | rel16/32 |  |  |  |  | o..s.... |  |  |  |  | Jump short if less/not greater (SF!=OF) |
| JNGE | rel16/32 |  |  |  |
|  | 0F | 8D |  |  | 03+ |  |  |  |  | JNL | rel16/32 |  |  |  |  | o..s.... |  |  |  |  | Jump short if not less/greater or equal (SF=OF) |
| JGE | rel16/32 |  |  |  |
|  | 0F | 8E |  |  | 03+ |  |  |  |  | JLE | rel16/32 |  |  |  |  | o..sz... |  |  |  |  | Jump short if less or equal/not greater ((ZF=1) OR (SF!=OF)) |
| JNG | rel16/32 |  |  |  |
|  | 0F | 8F |  |  | 03+ |  |  |  |  | JNLE | rel16/32 |  |  |  |  | o..sz... |  |  |  |  | Jump short if not less nor equal/greater ((ZF=0) AND (SF=OF)) |
| JG | rel16/32 |  |  |  |
|  | 0F | 90 |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETO | **r/m8** |  |  |  |  | o....... |  |  |  |  | Set Byte on Condition - overflow (OF=1) |
|  | 0F | 91 |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETNO | **r/m8** |  |  |  |  | o....... |  |  |  |  | Set Byte on Condition - not overflow (OF=0) |
|  | 0F | 92 |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETB | **r/m8** |  |  |  |  | .......c |  |  |  |  | Set Byte on Condition - below/not above or equal/carry (CF=1) |
| SETNAE | **r/m8** |  |  |  |
| SETC | **r/m8** |  |  |  |
|  | 0F | 93 |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETNB | **r/m8** |  |  |  |  | .......c |  |  |  |  | Set Byte on Condition - not below/above or equal/not carry (CF=0) |
| SETAE | **r/m8** |  |  |  |
| SETNC | **r/m8** |  |  |  |
|  | 0F | 94 |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETZ | **r/m8** |  |  |  |  | ....z... |  |  |  |  | Set Byte on Condition - zero/equal (ZF=0) |
| SETE | **r/m8** |  |  |  |
|  | 0F | 95 |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETNZ | **r/m8** |  |  |  |  | ....z... |  |  |  |  | Set Byte on Condition - not zero/not equal (ZF=1) |
| SETNE | **r/m8** |  |  |  |
|  | 0F | 96 |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETBE | **r/m8** |  |  |  |  | ....z..c |  |  |  |  | Set Byte on Condition - below or equal/not above (CF=1 AND ZF=1) |
| SETNA | **r/m8** |  |  |  |
|  | 0F | 97 |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETNBE | **r/m8** |  |  |  |  | ....z..c |  |  |  |  | Set Byte on Condition - not below or equal/above (CF=0 AND ZF=0) |
| SETA | **r/m8** |  |  |  |
|  | 0F | 98 |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETS | **r/m8** |  |  |  |  | ...s.... |  |  |  |  | Set Byte on Condition - sign (SF=1) |
|  | 0F | 99 |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETNS | **r/m8** |  |  |  |  | ...s.... |  |  |  |  | Set Byte on Condition - not sign (SF=0) |
|  | 0F | 9A |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETP | **r/m8** |  |  |  |  | ......p. |  |  |  |  | Set Byte on Condition - parity/parity even (PF=1) |
| SETPE | **r/m8** |  |  |  |
|  | 0F | 9B |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETNP | **r/m8** |  |  |  |  | ......p. |  |  |  |  | Set Byte on Condition - not parity/parity odd |
| SETPO | **r/m8** |  |  |  |
|  | 0F | 9C |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETL | **r/m8** |  |  |  |  | o..s.... |  |  |  |  | Set Byte on Condition - less/not greater (SF!=OF) |
| SETNGE | **r/m8** |  |  |  |
|  | 0F | 9D |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETNL | **r/m8** |  |  |  |  | o..s.... |  |  |  |  | Set Byte on Condition - not less/greater or equal (SF=OF) |
| SETGE | **r/m8** |  |  |  |
|  | 0F | 9E |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETLE | **r/m8** |  |  |  |  | o..sz... |  |  |  |  | Set Byte on Condition - less or equal/not greater ((ZF=1) OR (SF!=OF)) |
| SETNG | **r/m8** |  |  |  |
|  | 0F | 9F |  | 0 | 03+ | D[18](http://ref.x86asm.net/coder32.html#gen_note_SETcc_0F90-0F9F) |  |  |  | SETNLE | **r/m8** |  |  |  |  | o..sz... |  |  |  |  | Set Byte on Condition - not less nor equal/greater ((ZF=0) AND (SF=OF)) |
| SETG | **r/m8** |  |  |  |
|  | 0F | A0 |  |  | 03+ |  |  |  |  | PUSH | FS |  |  |  |  |  |  |  |  |  | Push Word, Doubleword or Quadword Onto the Stack |
|  | 0F | A1 |  |  | 03+ |  |  |  |  | POP | **FS** |  |  |  |  |  |  |  |  |  | Pop a Value from the Stack |
|  | 0F | A2 |  |  | 04++ |  |  |  |  | CPUID | ***IA32\_BIOS\_…*** | ***EAX*** | ***ECX*** | ... |  |  |  |  |  |  | CPU Identification |
|  | 0F | A3 |  | r | 03+ |  |  |  |  | BT | r/m16/32 | r16/32 |  |  |  |  | o..szapc | .......c | o..szap. |  | Bit Test |
|  | 0F | A4 |  | r | 03+ |  |  |  |  | SHLD | **r/m16/32** | r16/32 | imm8 |  |  |  | o..szapc | o..sz.pc | o....a.c |  | Double Precision Shift Left |
|  | 0F | A5 |  | r | 03+ |  |  |  |  | SHLD | **r/m16/32** | r16/32 | CL |  |  |  | o..szapc | o..sz.pc | o....a.c |  | Double Precision Shift Left |
|  | 0F | A8 |  |  | 03+ |  |  |  |  | PUSH | GS |  |  |  |  |  |  |  |  |  | Push Word, Doubleword or Quadword Onto the Stack |
|  | 0F | A9 |  |  | 03+ |  |  |  |  | POP | **GS** |  |  |  |  |  |  |  |  |  | Pop a Value from the Stack |
|  | 0F | AA |  |  | 03++ |  | S |  |  | RSM | ***Flags*** |  |  |  |  |  |  |  |  |  | Resume from System Management Mode |
|  | 0F | AB |  | r | 03+ |  |  |  | L | BTS | **r/m16/32** | r16/32 |  |  |  |  | o..szapc | .......c | o..szap. |  | Bit Test and Set |
|  | 0F | AC |  | r | 03+ |  |  |  |  | SHRD | **r/m16/32** | r16/32 | imm8 |  |  |  | o..szapc | o..sz.pc | o....a.c |  | Double Precision Shift Right |
|  | 0F | AD |  | r | 03+ |  |  |  |  | SHRD | **r/m16/32** | r16/32 | CL |  |  |  | o..szapc | o..sz.pc | o....a.c |  | Double Precision Shift Right |
|  | 0F | AE |  | 0 | P2++ |  |  |  |  | FXSAVE | **m512** | *ST* | *ST1* | ... |  |  |  |  |  |  | Save x87 FPU, MMX, XMM, and MXCSR State |
|  | 0F | AE |  | 1 | P2++ |  |  |  |  | FXRSTOR | ***ST*** | ***ST1*** | ***ST2*** | ... |  |  |  |  |  |  | Restore x87 FPU, MMX, XMM, and MXCSR State |
|  | 0F | AE |  | 2 | P3+ |  |  |  |  | LDMXCSR | m32 |  |  |  | sse1 |  |  |  |  |  | Load MXCSR Register |
|  | 0F | AE |  | 3 | P3+ |  |  |  |  | STMXCSR | **m32** |  |  |  | sse1 |  |  |  |  |  | Store MXCSR Register State |
|  | 0F | AE |  | 4 | C2++ |  |  |  |  | XSAVE | **m** | *EDX* | *EAX* | ... |  |  |  |  |  |  | Save Processor Extended States |
|  | 0F | AE |  | 5 | P4+ |  |  |  |  | LFENCE |  |  |  |  | sse2 |  |  |  |  |  | Load Fence |
|  | 0F | AE |  | 5 | C2++ |  |  |  |  | XRSTOR | ***ST*** | ***ST1*** | ***ST2*** | ... |  |  |  |  |  |  | Restore Processor Extended States |
|  | 0F | AE |  | 6 | P4+ |  |  |  |  | MFENCE |  |  |  |  | sse2 |  |  |  |  |  | Memory Fence |
|  | 0F | AE |  | 7 | P3+ |  |  |  |  | SFENCE |  |  |  |  | sse1 |  |  |  |  |  | Store Fence |
|  | 0F | AE |  | 7 | P4+ |  |  |  |  | CLFLUSH | m8 |  |  |  | sse2 |  |  |  |  |  | Flush Cache Line |
|  | 0F | AF |  | r | 03+ |  |  |  |  | IMUL | **r16/32** | r/m16/32 |  |  |  |  | o..szapc | o......c | ...szap. |  | Signed Multiply |
|  | 0F | B0 |  | r | 04+ | D[19](http://ref.x86asm.net/coder32.html#gen_note_CMPXCHG_0FB0_0FB1) |  |  | L | CMPXCHG | **r/m8** | ***AL*** | r8 |  |  |  | o..szapc | o..szapc |  |  | Compare and Exchange |
|  | 0F | B1 |  | r | 04+ | D[19](http://ref.x86asm.net/coder32.html#gen_note_CMPXCHG_0FB0_0FB1) |  |  | L | CMPXCHG | **r/m16/32** | ***eAX*** | r16/32 |  |  |  | o..szapc | o..szapc |  |  | Compare and Exchange |
|  | 0F | B2 |  | r | 03+ |  |  |  |  | LSS | ***SS*** | **r16/32** | m16:16/32 |  |  |  |  |  |  |  | Load Far Pointer |
|  | 0F | B3 |  | r | 03+ |  |  |  | L | BTR | **r/m16/32** | r16/32 |  |  |  |  | o..szapc | .......c | o..szap. |  | Bit Test and Reset |
|  | 0F | B4 |  | r | 03+ |  |  |  |  | LFS | ***FS*** | **r16/32** | m16:16/32 |  |  |  |  |  |  |  | Load Far Pointer |
|  | 0F | B5 |  | r | 03+ |  |  |  |  | LGS | ***GS*** | **r16/32** | m16:16/32 |  |  |  |  |  |  |  | Load Far Pointer |
|  | 0F | B6 |  | r | 03+ |  |  |  |  | MOVZX | **r16/32** | r/m8 |  |  |  |  |  |  |  |  | Move with Zero-Extend |
|  | 0F | B7 |  | r | 03+ |  |  |  |  | MOVZX | **r16/32** | r/m16 |  |  |  |  |  |  |  |  | Move with Zero-Extend |
| F3 | 0F | B8 |  | r | C2++ |  |  |  |  | POPCNT | **r16/32** | r/m16/32 |  |  |  |  | o..szapc |  |  | o..s.apc | Bit Population Count |
|  | 0F | B9 |  | r | 02+ | M[20](http://ref.x86asm.net/coder32.html#gen_note_sug_UD_0FB9) |  |  |  | *UD* | r | r/m |  |  |  |  |  |  |  |  | Undefined Instruction |
|  | 0F | BA |  | 4 | 03+ |  |  |  |  | BT | r/m16/32 | imm8 |  |  |  |  | o..szapc | .......c | o..szap. |  | Bit Test |
|  | 0F | BA |  | 5 | 03+ |  |  |  | L | BTS | **r/m16/32** | imm8 |  |  |  |  | o..szapc | .......c | o..szap. |  | Bit Test and Set |
|  | 0F | BA |  | 6 | 03+ |  |  |  | L | BTR | **r/m16/32** | imm8 |  |  |  |  | o..szapc | .......c | o..szap. |  | Bit Test and Reset |
|  | 0F | BA |  | 7 | 03+ |  |  |  | L | BTC | **r/m16/32** | imm8 |  |  |  |  | o..szapc | .......c | o..szap. |  | Bit Test and Complement |
|  | 0F | BB |  | r | 03+ |  |  |  | L | BTC | **r/m16/32** | r16/32 |  |  |  |  | o..szapc | .......c | o..szap. |  | Bit Test and Complement |
|  | 0F | BC |  | r | 03+ |  |  |  |  | BSF | **r16/32** | r/m16/32 |  |  |  |  | o..szapc | ....z... | o..s.apc |  | Bit Scan Forward |
|  | 0F | BD |  | r | 03+ |  |  |  |  | BSR | **r16/32** | r/m16/32 |  |  |  |  | o..szapc | ....z... | o..s.apc |  | Bit Scan Reverse |
|  | 0F | BE |  | r | 03+ |  |  |  |  | MOVSX | **r16/32** | r/m8 |  |  |  |  |  |  |  |  | Move with Sign-Extension |
|  | 0F | BF |  | r | 03+ |  |  |  |  | MOVSX | **r16/32** | r/m16 |  |  |  |  |  |  |  |  | Move with Sign-Extension |
|  | 0F | C0 |  | r | 04+ |  |  |  | L | XADD | **r/m8** | **r8** |  |  |  |  | o..szapc | o..szapc |  |  | Exchange and Add |
|  | 0F | C1 |  | r | 04+ |  |  |  | L | XADD | **r/m16/32** | **r16/32** |  |  |  |  | o..szapc | o..szapc |  |  | Exchange and Add |
|  | 0F | C2 |  | r | P3+ |  |  |  |  | CMPPS | **xmm** | xmm/m128 | imm8 |  | sse1 |  |  |  |  |  | Compare Packed Single-FP Values |
| F3 | 0F | C2 |  | r | P3+ |  |  |  |  | CMPSS | **xmm** | xmm/m32 | imm8 |  | sse1 |  |  |  |  |  | Compare Scalar Single-FP Values |
| 66 | 0F | C2 |  | r | P4+ |  |  |  |  | CMPPD | **xmm** | xmm/m128 | imm8 |  | sse2 |  |  |  |  |  | Compare Packed Double-FP Values |
| F2 | 0F | C2 |  | r | P4+ |  |  |  |  | CMPSD | **xmm** | xmm/m64 | imm8 |  | sse2 |  |  |  |  |  | Compare Scalar Double-FP Values |
|  | 0F | C3 |  | r | P4+ |  |  |  |  | MOVNTI | **m32** | r32 |  |  | sse2 |  |  |  |  |  | Store Doubleword Using Non-Temporal Hint |
|  | 0F | C4 |  | r | P3+ |  |  |  |  | PINSRW | **mm** | r32 | imm8 |  | sse1 |  |  |  |  |  | Insert Word |
| PINSRW | **mm** | m16 | imm8 |  |
| 66 | 0F | C4 |  | r | P3+ |  |  |  |  | PINSRW | **xmm** | r32 | imm8 |  | sse1 |  |  |  |  |  | Insert Word |
| PINSRW | **xmm** | m16 | imm8 |  |
|  | 0F | C5 |  | r | P3+ |  |  |  |  | PEXTRW | **r32** | mm | imm8 |  | sse1 |  |  |  |  |  | Extract Word |
| 66 | 0F | C5 |  | r | P3+ |  |  |  |  | PEXTRW | **r32** | xmm | imm8 |  | sse1 |  |  |  |  |  | Extract Word |
|  | 0F | C6 |  | r | P3+ |  |  |  |  | SHUFPS | **xmm** | xmm/m128 | imm8 |  | sse1 |  |  |  |  |  | Shuffle Packed Single-FP Values |
| 66 | 0F | C6 |  | r | P4+ |  |  |  |  | SHUFPD | **xmm** | xmm/m128 | imm8 |  | sse2 |  |  |  |  |  | Shuffle Packed Double-FP Values |
|  | 0F | C7 |  | 1 | P1+ | D[21](http://ref.x86asm.net/coder32.html#gen_note_CMPXCHG8B_CMPXCHG16B_0FC7_1) |  |  | L | CMPXCHG8B | **m64** | ***EAX*** | ***EDX*** | ... |  |  | ....z... | ....z... |  |  | Compare and Exchange Bytes |
|  | 0F | C7 |  | 6 | P4++ | D[24](http://ref.x86asm.net/coder32.html#gen_note_VMX_vs_SVM) | P | 0 |  | VMPTRLD | m64 |  |  |  | vmx |  | o..szapc | o..szapc |  |  | Load Pointer to Virtual-Machine Control Structure |
| 66 | 0F | C7 |  | 6 | P4++ | D[24](http://ref.x86asm.net/coder32.html#gen_note_VMX_vs_SVM) | P | 0 |  | VMCLEAR | **m64** |  |  |  | vmx |  | o..szapc | o..szapc |  |  | Clear Virtual-Machine Control Structure |
| F3 | 0F | C7 |  | 6 | P4++ | D[24](http://ref.x86asm.net/coder32.html#gen_note_VMX_vs_SVM) | P | 0 |  | VMXON | m64 |  |  |  | vmx |  | o..szapc | o..szapc |  |  | Enter VMX Operation |
|  | 0F | C7 |  | 7 | P4++ | D[24](http://ref.x86asm.net/coder32.html#gen_note_VMX_vs_SVM) | P | 0 |  | VMPTRST | **m64** |  |  |  | vmx |  | o..szapc | o..szapc |  |  | Store Pointer to Virtual-Machine Control Structure |
|  | 0F | C8+r | |  | 04+ | D[22](http://ref.x86asm.net/coder32.html#gen_note_BSWAP_0FC8) |  |  |  | BSWAP | **r16/32** |  |  |  |  |  |  |  |  |  | Byte Swap |
| 66 | 0F | D0 |  | r | P4++ |  |  |  |  | ADDSUBPD | **xmm** | xmm/m128 |  |  | sse3 |  |  |  |  |  | Packed Double-FP Add/Subtract |
| F2 | 0F | D0 |  | r | P4++ |  |  |  |  | ADDSUBPS | **xmm** | xmm/m128 |  |  | sse3 |  |  |  |  |  | Packed Single-FP Add/Subtract |
|  | 0F | D1 |  | r | PX+ |  |  |  |  | PSRLW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Shift Packed Data Right Logical |
| 66 | 0F | D1 |  | r | P4+ |  |  |  |  | PSRLW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Shift Packed Data Right Logical |
|  | 0F | D2 |  | r | PX+ |  |  |  |  | PSRLD | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Shift Packed Data Right Logical |
| 66 | 0F | D2 |  | r | P4+ |  |  |  |  | PSRLD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Shift Packed Data Right Logical |
|  | 0F | D3 |  | r | PX+ |  |  |  |  | PSRLQ | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Shift Packed Data Right Logical |
| 66 | 0F | D3 |  | r | P4+ |  |  |  |  | PSRLQ | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Shift Packed Data Right Logical |
|  | 0F | D4 |  | r | PX+ |  |  |  |  | PADDQ | **mm** | mm/m64 |  |  | sse2 |  |  |  |  |  | Add Packed Quadword Integers |
| 66 | 0F | D4 |  | r | P4+ |  |  |  |  | PADDQ | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Add Packed Quadword Integers |
|  | 0F | D5 |  | r | PX+ |  |  |  |  | PMULLW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Multiply Packed Signed Integers and Store Low Result |
| 66 | 0F | D5 |  | r | P4+ |  |  |  |  | PMULLW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Multiply Packed Signed Integers and Store Low Result |
| 66 | 0F | D6 |  | r | P4+ |  |  |  |  | MOVQ | **xmm/m64** | xmm |  |  | sse2 |  |  |  |  |  | Move Quadword |
| F3 | 0F | D6 |  | r | P4+ |  |  |  |  | MOVQ2DQ | **xmm** | mm |  |  | sse2 |  |  |  |  |  | Move Quadword from MMX Technology to XMM Register |
| F2 | 0F | D6 |  | r | P4+ |  |  |  |  | MOVDQ2Q | **mm** | xmm |  |  | sse2 |  |  |  |  |  | Move Quadword from XMM to MMX Technology Register |
|  | 0F | D7 |  | r | P3+ |  |  |  |  | PMOVMSKB | **r32** | mm |  |  | sse1 |  |  |  |  |  | Move Byte Mask |
| 66 | 0F | D7 |  | r | P3+ |  |  |  |  | PMOVMSKB | **r32** | xmm |  |  | sse1 |  |  |  |  |  | Move Byte Mask |
|  | 0F | D8 |  | r | PX+ |  |  |  |  | PSUBUSB | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Subtract Packed Unsigned Integers with Unsigned Saturation |
| 66 | 0F | D8 |  | r | P4+ |  |  |  |  | PSUBUSB | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Subtract Packed Unsigned Integers with Unsigned Saturation |
|  | 0F | D9 |  | r | PX+ |  |  |  |  | PSUBUSW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Subtract Packed Unsigned Integers with Unsigned Saturation |
| 66 | 0F | D9 |  | r | PX+ |  |  |  |  | PSUBUSW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Subtract Packed Unsigned Integers with Unsigned Saturation |
|  | 0F | DA |  | r | P3+ |  |  |  |  | PMINUB | **mm** | mm/m64 |  |  | sse1 |  |  |  |  |  | Minimum of Packed Unsigned Byte Integers |
| 66 | 0F | DA |  | r | P3+ |  |  |  |  | PMINUB | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Minimum of Packed Unsigned Byte Integers |
|  | 0F | DB |  | r | PX+ |  |  |  |  | PAND | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Logical AND |
| 66 | 0F | DB |  | r | P4+ |  |  |  |  | PAND | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Logical AND |
|  | 0F | DC |  | r | PX+ |  |  |  |  | PADDUSB | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Add Packed Unsigned Integers with Unsigned Saturation |
| 66 | 0F | DC |  | r | P4+ |  |  |  |  | PADDUSB | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Add Packed Unsigned Integers with Unsigned Saturation |
|  | 0F | DD |  | r | PX+ |  |  |  |  | PADDUSW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Add Packed Unsigned Integers with Unsigned Saturation |
| 66 | 0F | DD |  | r | P4+ |  |  |  |  | PADDUSW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Add Packed Unsigned Integers with Unsigned Saturation |
|  | 0F | DE |  | r | P3+ |  |  |  |  | PMAXUB | **mm** | mm/m64 |  |  | sse1 |  |  |  |  |  | Maximum of Packed Unsigned Byte Integers |
| 66 | 0F | DE |  | r | P3+ |  |  |  |  | PMAXUB | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Maximum of Packed Unsigned Byte Integers |
|  | 0F | DF |  | r | PX+ |  |  |  |  | PANDN | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Logical AND NOT |
| 66 | 0F | DF |  | r | P4+ |  |  |  |  | PANDN | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Logical AND NOT |
|  | 0F | E0 |  | r | P3+ |  |  |  |  | PAVGB | **mm** | mm/m64 |  |  | sse1 |  |  |  |  |  | Average Packed Integers |
| 66 | 0F | E0 |  | r | P3+ |  |  |  |  | PAVGB | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Average Packed Integers |
|  | 0F | E1 |  | r | PX+ |  |  |  |  | PSRAW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Shift Packed Data Right Arithmetic |
| 66 | 0F | E1 |  | r | P4+ |  |  |  |  | PSRAW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Shift Packed Data Right Arithmetic |
|  | 0F | E2 |  | r | PX+ |  |  |  |  | PSRAD | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Shift Packed Data Right Arithmetic |
| 66 | 0F | E2 |  | r | P4+ |  |  |  |  | PSRAD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Shift Packed Data Right Arithmetic |
|  | 0F | E3 |  | r | P3+ |  |  |  |  | PAVGW | **mm** | mm/m64 |  |  | sse1 |  |  |  |  |  | Average Packed Integers |
| 66 | 0F | E3 |  | r | P3+ |  |  |  |  | PAVGW | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Average Packed Integers |
|  | 0F | E4 |  | r | P3+ |  |  |  |  | PMULHUW | **mm** | mm/m64 |  |  | sse1 |  |  |  |  |  | Multiply Packed Unsigned Integers and Store High Result |
| 66 | 0F | E4 |  | r | P3+ |  |  |  |  | PMULHUW | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Multiply Packed Unsigned Integers and Store High Result |
|  | 0F | E5 |  | r | PX+ |  |  |  |  | PMULHW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Multiply Packed Signed Integers and Store High Result |
| 66 | 0F | E5 |  | r | P4+ |  |  |  |  | PMULHW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Multiply Packed Signed Integers and Store High Result |
| F2 | 0F | E6 |  | r | P4+ |  |  |  |  | CVTPD2DQ | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Convert Packed Double-FP Values to DW Integers |
| 66 | 0F | E6 |  | r | P4+ |  |  |  |  | CVTTPD2DQ | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Convert with Trunc. Packed Double-FP Values to DW Integers |
| F3 | 0F | E6 |  | r | P4+ |  |  |  |  | CVTDQ2PD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Convert Packed DW Integers to Double-FP Values |
|  | 0F | E7 |  | r | P3+ |  |  |  |  | MOVNTQ | **m64** | mm |  |  | sse1 |  |  |  |  |  | Store of Quadword Using Non-Temporal Hint |
| 66 | 0F | E7 |  | r | P4+ |  |  |  |  | MOVNTDQ | **m128** | xmm |  |  | sse2 |  |  |  |  |  | Store Double Quadword Using Non-Temporal Hint |
|  | 0F | E8 |  | r | PX+ |  |  |  |  | PSUBSB | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Subtract Packed Signed Integers with Signed Saturation |
| 66 | 0F | E8 |  | r | P4+ |  |  |  |  | PSUBSB | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Subtract Packed Signed Integers with Signed Saturation |
|  | 0F | E9 |  | r | PX+ |  |  |  |  | PSUBSW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Subtract Packed Signed Integers with Signed Saturation |
| 66 | 0F | E9 |  | r | P4+ |  |  |  |  | PSUBSW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Subtract Packed Signed Integers with Signed Saturation |
|  | 0F | EA |  | r | P3+ |  |  |  |  | PMINSW | **mm** | mm/m64 |  |  | sse1 |  |  |  |  |  | Minimum of Packed Signed Word Integers |
| 66 | 0F | EA |  | r | P3+ |  |  |  |  | PMINSW | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Minimum of Packed Signed Word Integers |
|  | 0F | EB |  | r | PX+ |  |  |  |  | POR | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Bitwise Logical OR |
| 66 | 0F | EB |  | r | P4+ |  |  |  |  | POR | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Bitwise Logical OR |
|  | 0F | EC |  | r | PX+ |  |  |  |  | PADDSB | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Add Packed Signed Integers with Signed Saturation |
| 66 | 0F | EC |  | r | P4+ |  |  |  |  | PADDSB | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Add Packed Signed Integers with Signed Saturation |
|  | 0F | ED |  | r | PX+ |  |  |  |  | PADDSW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Add Packed Signed Integers with Signed Saturation |
| 66 | 0F | ED |  | r | P4+ |  |  |  |  | PADDSW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Add Packed Signed Integers with Signed Saturation |
|  | 0F | EE |  | r | P3+ |  |  |  |  | PMAXSW | **mm** | mm/m64 |  |  | sse1 |  |  |  |  |  | Maximum of Packed Signed Word Integers |
| 66 | 0F | EE |  | r | P3+ |  |  |  |  | PMAXSW | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Maximum of Packed Signed Word Integers |
|  | 0F | EF |  | r | PX+ |  |  |  |  | PXOR | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Logical Exclusive OR |
| 66 | 0F | EF |  | r | P4+ |  |  |  |  | PXOR | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Logical Exclusive OR |
| F2 | 0F | F0 |  | r | P4++ |  |  |  |  | LDDQU | **xmm** | m128 |  |  | sse3 |  |  |  |  |  | Load Unaligned Integer 128 Bits |
|  | 0F | F1 |  | r | PX+ |  |  |  |  | PSLLW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Shift Packed Data Left Logical |
| 66 | 0F | F1 |  | r | P4+ |  |  |  |  | PSLLW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Shift Packed Data Left Logical |
|  | 0F | F2 |  | r | PX+ |  |  |  |  | PSLLD | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Shift Packed Data Left Logical |
| 66 | 0F | F2 |  | r | P4+ |  |  |  |  | PSLLD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Shift Packed Data Left Logical |
|  | 0F | F3 |  | r | PX+ |  |  |  |  | PSLLQ | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Shift Packed Data Left Logical |
| 66 | 0F | F3 |  | r | P4+ |  |  |  |  | PSLLQ | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Shift Packed Data Left Logical |
|  | 0F | F4 |  | r | P4+ |  |  |  |  | PMULUDQ | **mm** | mm/m64 |  |  | sse2 |  |  |  |  |  | Multiply Packed Unsigned DW Integers |
| 66 | 0F | F4 |  | r | P4+ |  |  |  |  | PMULUDQ | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Multiply Packed Unsigned DW Integers |
|  | 0F | F5 |  | r | PX+ |  |  |  |  | PMADDWD | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Multiply and Add Packed Integers |
| 66 | 0F | F5 |  | r | P4+ |  |  |  |  | PMADDWD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Multiply and Add Packed Integers |
|  | 0F | F6 |  | r | P3+ |  |  |  |  | PSADBW | **mm** | mm/m64 |  |  | sse1 |  |  |  |  |  | Compute Sum of Absolute Differences |
| 66 | 0F | F6 |  | r | P3+ |  |  |  |  | PSADBW | **xmm** | xmm/m128 |  |  | sse1 |  |  |  |  |  | Compute Sum of Absolute Differences |
|  | 0F | F7 |  | r | P3+ | D[23](http://ref.x86asm.net/coder32.html#gen_note_MASKMOVQ_0FF7) |  |  |  | MASKMOVQ | ***m64*** | **mm** | mm |  | sse1 |  |  |  |  |  | Store Selected Bytes of Quadword |
| 66 | 0F | F7 |  | r | P4+ |  |  |  |  | MASKMOVDQU | ***m128*** | xmm | xmm |  | sse2 |  |  |  |  |  | Store Selected Bytes of Double Quadword |
|  | 0F | F8 |  | r | PX+ |  |  |  |  | PSUBB | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Subtract Packed Integers |
| 66 | 0F | F8 |  | r | P4+ |  |  |  |  | PSUBB | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Subtract Packed Integers |
|  | 0F | F9 |  | r | PX+ |  |  |  |  | PSUBW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Subtract Packed Integers |
| 66 | 0F | F9 |  | r | P4+ |  |  |  |  | PSUBW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Subtract Packed Integers |
|  | 0F | FA |  | r | PX+ |  |  |  |  | PSUBD | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Subtract Packed Integers |
| 66 | 0F | FA |  | r | P4+ |  |  |  |  | PSUBD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Subtract Packed Integers |
|  | 0F | FB |  | r | P4+ |  |  |  |  | PSUBQ | **mm** | mm/m64 |  |  | sse2 |  |  |  |  |  | Subtract Packed Quadword Integers |
| 66 | 0F | FB |  | r | P4+ |  |  |  |  | PSUBQ | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Subtract Packed Quadword Integers |
|  | 0F | FC |  | r | PX+ |  |  |  |  | PADDB | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Add Packed Integers |
| 66 | 0F | FC |  | r | P4+ |  |  |  |  | PADDB | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Add Packed Integers |
|  | 0F | FD |  | r | PX+ |  |  |  |  | PADDW | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Add Packed Integers |
| 66 | 0F | FD |  | r | P4+ |  |  |  |  | PADDW | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Add Packed Integers |
|  | 0F | FE |  | r | PX+ |  |  |  |  | PADDD | **mm** | mm/m64 |  |  | mmx |  |  |  |  |  | Add Packed Integers |
| 66 | 0F | FE |  | r | P4+ |  |  |  |  | PADDD | **xmm** | xmm/m128 |  |  | sse2 |  |  |  |  |  | Add Packed Integers |

Printing is not enabled. You can order a printed copy in the [store](http://ref.x86asm.net/store/), or get [access to benefits](http://ref.x86asm.net/#Why-to-Contribute---Benefits_desc), which include also printable HTML and PDF files.

**General notes:**

1. *90 NOP*
   1. 90 NOP is not really aliased to XCHG eAX, eAX instruction. This is important in 64-bit mode where the implicit zero-extension to RAX does not happen
2. *SAL*
   1. sandpile.org -- IA-32 architecture -- opcode groups
3. *D6 and F1 opcodes*
   1. Intel 64 and IA-32 Architecture Software Developer's Manual Volume 3: System Programming Guide, Interrupt and Exception Handling
4. *SALC*
   1. sandpile.org -- IA-32 architecture -- one byte opcodes
   2. AMD64 Architecture Programmer's Manual Volume 3, Table One-Bytes Opcodes
5. *FSTP1*
   1. Christian Ludloff wrote: While FSTP (D9 /3, mod < 11b), FSTP8 (DF /2, mod = 11b), and FSTP9 (DF /3, mod = 11b) do signal stack underflow, FSTP1 (D9 /3, mod = 11b) does not.
6. *FNENI and FNDISI*
   1. INTEL 80287 PROGRAMMER'S REFERENCE MANUAL 1987, Processor Control Instructions: The 8087 instructions FENI and FDISI perform no function in the 80287. If these opcodes are detected in an 80286/80287 instruction stream, the 80287 will perform no specific operation and no internal states will be affected.
7. *FNSETPM*
   1. INTEL 80387 PROGRAMMER'S REFERENCE MANUAL 1987, 6.1.2 Independent of CPU Addressing Modes: Unlike the 80287, the 80387 is not sensitive to the addressing and memory management of the CPU. The 80387 operates the same regardless of whether the 80386 CPU is operating in real-address mode, in protected mode, or in virtual 8086 mode.
8. *FFREEP*
   1. INTEL 80287 PROGRAMMER'S REFERENCE MANUAL 1987, Table A-2. Machine Instruction Decoding Guide: If the 80287 encounters one of these encodings (DF /1, mod = 11b) in the instruction stream, it will execute it as follows: FFREE ST(i) and pop stack
   2. Intel Architecture Optimization Reference Manual PIII, Table C-1 Pentium II and Pentium III Processors Instruction to Decoder Specification
   3. AMD Athlon Processor x86 Code Optimization Guide, Chapter 9, Use FFREEP Macro to Pop One Register from the FPU Stack
   4. sandpile.org -- IA-32 architecture -- ESC (FP) opcodes
9. *X87 aliases*
   1. sandpile.org -- IA-32 architecture -- ESC (FP) opcodes
10. *INT1, ICEBP*
    1. sandpile.org -- IA-32 architecture -- one byte opcodes
    2. AMD64 Architecture Programmer's Manual Volume 3, Table One-Bytes Opcodes
    3. Christian Ludloff wrote: Unlike INT 1 (CDh,01h), INT1 (F1h) doesn't perform the IOPL or DPL check and it can't be redirected via the TSS32.IRB.
11. *REP prefixes*
    1. Flags aren't updated until after the last iteration to make the operation faster
12. *TEST*
    1. sandpile.org -- IA-32 architecture -- opcode groups
    2. Christian Ludloff wrote: While the latest Intel manuals still omit this de-facto standard, the recent x86-64 manuals from AMD document it.
    3. AMD64 Architecture Programmer's Manual Volume 3, Table One-Byte and Two-Byte Opcode ModRM Extensions
13. *SMSW r32/64*
    1. Some processors support reading whole CR0 register, causing a security flaw.
14. *0F0D NOP*
    1. Intel 64 and IA-32 Architecture Software Developer's Manual Volume 2B: Instruction Set Reference, N-Z, Two-byte Opcode Map
    2. AMD architecture maps 3DNow! PREFETCH instructions here
15. *Hintable NOP*
    1. See U.S. Patent 5,701,442
    2. sandpile.org -- IA-32 architecture -- opcode groups
16. *MOV from/to CRn, DRn, TRn*
    1. Christian Ludloff wrote: For the MOVs from/to CRx/DRx/TRx, mod=00b/01b/10b is aliased to 11b.
    2. AMD64 Architecture Programmer's Manual Volume 3, System Instruction Reference: This instruction is always treated as a register-to-register instruction, regardless of the encoding of the MOD field in the MODR/M byte.
17. *GETSEC Leaf Functions*
    1. Intel 64 and IA-32 Architecture Software Developer's Manual Volume 2B: Instruction Set Reference, N-Z: The GETSEC instruction supports multiple leaf functions. Leaf functions are selected by the value in EAX at the time GETSEC is executed. The following leaf functions are available: CAPABILITIES, ENTERACCS, EXITAC, SENTER, SEXIT, PARAMETERS, SMCTRL, WAKEUP. GETSEC instruction operands are specific to selected leaf function.
18. *SETcc*
    1. AMD64 Architecture Programmers Manual Volume 3: General-Purpose and System Instructions: The reg field in the ModR/M byte is unused.
19. *CMPXCHG with memory operand*
    1. Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2A: Instruction Set Reference, A-M: This instruction can be used with a LOCK prefix …. To simplify the interface to the processor's bus, the destination operand receives a write cycle without regard to the result of the comparison.
    2. AMD64 Architecture Programmers Manual Volume 3: General-Purpose and System Instructions: CMPXCHG always does a read-modify-write on the memory operand.
20. *0FB9 UD*
    1. Intel 64 and IA-32 Architecture Software Developer's Manual Volume 2B: Instruction Set Reference, N-Z, Two-byte Opcode Map
    2. sandpile.org -- IA-32 architecture -- two byte opcodes
21. *CMPXCHG8B, CMPXCHG16B*
    1. Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2A: Instruction Set Reference, A-M: This instruction can be used with a LOCK prefix …. To simplify the interface to the processor's bus, the destination operand receives a write cycle without regard to the result of the comparison.
    2. AMD64 Architecture Programmers Manual Volume 3: General-Purpose and System Instructions: The CMPXCHG8B and CMPXCHG16B instructions always do a read-modify-write on the memory operand.
    3. CMPXCHG16B is invalid on early steppings of AMD64 architecture.
22. *BSWAP r16*
    1. Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2A: Instruction Set Reference, A-M: When the BSWAP instruction references a 16-bit register, the result is undefined.
    2. AMD64 Architecture Programmer's Manual Volume 3: General-Purpose and System Instructions: The result of applying the BSWAP instruction to a 16-bit register is undefined.
23. *MASKMOVQ*
    1. Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2A: Instruction Set Reference, A-M: This instruction causes a transition from x87 FPU to MMX technology state.
24. *Intel VMX*
    1. Intel VMX is not binary-compatible with AMD SVM
25. *Intel SSE4*
    1. AMD64 architecture does not support SSE4 instructions but PTEST as part of SSE5

**Notes for the Ring Level, used in case of *f* mark:**

1. rFlags.IOPL
2. CR4.TSD[bit 2]
3. CR4.PCE[bit 8]

Create a hypertext reference to this edition's opcode (append hexadecimal opcode at the end of the following line):

http://ref.x86asm.net/coder32.html#x

32-bit ModR/M Byte

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| r8(/r) | | | AL | CL | DL | BL | AH | CH | DH | BH |
| r16(/r) | | | AX | CX | DX | BX | SP | BP | SI | DI |
| r32(/r) | | | EAX | ECX | EDX | EBX | ESP | EBP | ESI | EDI |
| mm(/r) | | | MM0 | MM1 | MM2 | MM3 | MM4 | MM5 | MM6 | MM7 |
| xmm(/r) | | | XMM0 | XMM1 | XMM2 | XMM3 | XMM4 | XMM5 | XMM6 | XMM7 |
| sreg | | | ES | CS | SS | DS | FS | GS | res. | res. |
| eee | | | CR0 | invd | CR2 | CR3 | CR4 | invd | invd | invd |
| eee | | | DR0 | DR1 | DR2 | DR3 | DR4[1](http://ref.x86asm.net/coder32.html#modrm_dr4_dr5) | DR5[1](http://ref.x86asm.net/coder32.html#modrm_dr4_dr5) | DR6 | DR7 |
| (In decimal) /digit (Opcode) | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| (In binary) REG = | | | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| Effective Address | Mod | R/M | Value of ModR/M Byte (in Hex) | | | | | | | |
| [EAX] | 00 | 000 | 00 | 08 | 10 | 18 | 20 | 28 | 30 | 38 |
| [ECX] |  | 001 | 01 | 09 | 11 | 19 | 21 | 29 | 31 | 39 |
| [EDX] |  | 010 | 02 | 0A | 12 | 1A | 22 | 2A | 32 | 3A |
| [EBX] |  | 011 | 03 | 0B | 13 | 1B | 23 | 2B | 33 | 3B |
| [[*sib*](http://ref.x86asm.net/coder32.html#sib_byte_32)] |  | 100 | 04 | 0C | 14 | 1C | 24 | 2C | 34 | 3C |
| disp32 |  | 101 | 05 | 0D | 15 | 1D | 25 | 2D | 35 | 3D |
| [ESI] |  | 110 | 06 | 0E | 16 | 1E | 26 | 2E | 36 | 3E |
| [EDI] |  | 111 | 07 | 0F | 17 | 1F | 27 | 2F | 37 | 3F |
| [EAX]+disp8 | 01 | 000 | 40 | 48 | 50 | 58 | 60 | 68 | 70 | 78 |
| [ECX]+disp8 |  | 001 | 41 | 49 | 51 | 59 | 61 | 69 | 71 | 79 |
| [EDX]+disp8 |  | 010 | 42 | 4A | 52 | 5A | 62 | 6A | 72 | 7A |
| [EBX]+disp8 |  | 011 | 43 | 4B | 53 | 5B | 63 | 6B | 73 | 7B |
| [[*sib*](http://ref.x86asm.net/coder32.html#sib_byte_32)]+disp8 |  | 100 | 44 | 4C | 54 | 5C | 64 | 6C | 74 | 7C |
| [EBP]+disp8 |  | 101 | 45 | 4D | 55 | 5D | 65 | 6D | 75 | 7D |
| [ESI]+disp8 |  | 110 | 46 | 4E | 56 | 5E | 66 | 6E | 76 | 7E |
| [EDI]+disp8 |  | 111 | 47 | 4F | 57 | 5F | 67 | 6F | 77 | 7F |
| [EAX]+disp32 | 10 | 000 | 80 | 88 | 90 | 98 | A0 | A8 | B0 | B8 |
| [ECX]+disp32 |  | 001 | 81 | 89 | 91 | 99 | A1 | A9 | B1 | B9 |
| [EDX]+disp32 |  | 010 | 82 | 8A | 92 | 9A | A2 | AA | B2 | BA |
| [EBX]+disp32 |  | 011 | 83 | 8B | 93 | 9B | A3 | AB | B3 | BB |
| [[*sib*](http://ref.x86asm.net/coder32.html#sib_byte_32)]+disp32 |  | 100 | 84 | 8C | 94 | 9C | A4 | AC | B4 | BC |
| [EBP]+disp32 |  | 101 | 85 | 8D | 95 | 9D | A5 | AD | B5 | BD |
| [ESI]+disp32 |  | 110 | 86 | 8E | 96 | 9E | A6 | AE | B6 | BE |
| [EDI]+disp32 |  | 111 | 87 | 8F | 97 | 9F | A7 | AF | B7 | BF |
| AL/AX/EAX/ST0/MM0/XMM0 | 11 | 000 | C0 | C8 | D0 | D8 | E0 | E8 | F0 | F8 |
| CL/CX/ECX/ST1/MM1/XMM1 |  | 001 | C1 | C9 | D1 | D9 | E1 | E9 | F1 | F9 |
| DL/DX/EDX/ST2/MM2/XMM2 |  | 010 | C2 | CA | D2 | DA | E2 | EA | F2 | FA |
| BL/BX/EBX/ST3/MM3/XMM3 |  | 011 | C3 | CB | D3 | DB | E3 | EB | F3 | FB |
| AH/SP/ESP/ST4/MM4/XMM4 |  | 100 | C4 | CC | D4 | DC | E4 | EC | F4 | FC |
| CH/BP/EBP/ST5/MM5/XMM5 |  | 101 | C5 | CD | D5 | DD | E5 | ED | F5 | FD |
| DH/SI/ESI/ST6/MM6/XMM6 |  | 110 | C6 | CE | D6 | DE | E6 | EE | F6 | FE |
| BH/DI/EDI/ST7/MM7/XMM7 |  | 111 | C7 | CF | D7 | DF | E7 | EF | F7 | FF |

32-bit SIB Byte

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| r32 | | | EAX | ECX | EDX | EBX | ESP | →[1](http://ref.x86asm.net/coder32.html#sib32_base_101) | ESI | EDI |
| (In decimal) Base = | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| (In binary) Base = | | | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| Scaled Index | SS | Index | Value of SIB Byte (in Hexadecimal) | | | | | | | |
| [EAX] | 00 | 000 | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 |
| [ECX] |  | 001 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
| [EDX] |  | 010 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| [EBX] |  | 011 | 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F |
| *none* |  | 100 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| [EBP] |  | 101 | 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F |
| [ESI] |  | 110 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 |
| [EDI] |  | 111 | 38 | 39 | 3A | 3B | 3C | 3D | 3E | 3F |
| [EAX\*2] | 01 | 000 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| [ECX\*2] |  | 001 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F |
| [EDX\*2] |  | 010 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 |
| [EBX\*2] |  | 011 | 58 | 59 | 5A | 5B | 5C | 5D | 5E | 5F |
| *none* |  | 100 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 |
| [EBP\*2] |  | 101 | 68 | 69 | 6A | 6B | 6C | 6D | 6E | 6F |
| [ESI\*2] |  | 110 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
| [EDI\*2] |  | 111 | 78 | 79 | 7A | 7B | 7C | 7D | 7E | 7F |
| [EAX\*4] | 10 | 000 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 |
| [ECX\*4] |  | 001 | 88 | 89 | 8A | 8B | 8C | 8D | 8E | 8F |
| [EDX\*4] |  | 010 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 |
| [EBX\*4] |  | 011 | 98 | 99 | 9A | 9B | 9C | 9D | 9E | 9F |
| *none* |  | 100 | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 |
| [EBP\*4] |  | 101 | A8 | A9 | AA | AB | AC | AD | AE | AF |
| [ESI\*4] |  | 110 | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 |
| [EDI\*4] |  | 111 | B8 | B9 | BA | BB | BC | BD | BE | BF |
| [EAX\*8] | 11 | 000 | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 |
| [ECX\*8] |  | 001 | C8 | C9 | CA | CB | CC | CD | CE | CF |
| [EDX\*8] |  | 010 | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
| [EBX\*8] |  | 011 | D8 | D9 | DA | DB | DC | DD | DE | DF |
| *none* |  | 100 | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 |
| [EBP\*8] |  | 101 | E8 | E9 | EA | EB | EC | ED | EE | EF |
| [ESI\*8] |  | 110 | F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 |
| [EDI\*8] |  | 111 | F8 | F9 | FA | FB | FC | FD | FE | FF |

|  |  |
| --- | --- |
| **SIB Note 1** | |
| Mod bits | base |
| 00 | disp32 |
| 01 | EBP+disp8 |
| 10 | EBP+disp32 |

16-bit ModR/M Byte

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| r8(/r) | | | AL | CL | DL | BL | AH | CH | DH | BH |
| r16(/r) | | | AX | CX | DX | BX | SP | BP | SI | DI |
| r32(/r) | | | EAX | ECX | EDX | EBX | ESP | EBP | ESI | EDI |
| mm(/r) | | | MM0 | MM1 | MM2 | MM3 | MM4 | MM5 | MM6 | MM7 |
| xmm(/r) | | | XMM0 | XMM1 | XMM2 | XMM3 | XMM4 | XMM5 | XMM6 | XMM7 |
| sreg | | | ES | CS | SS | DS | FS | GS | res. | res. |
| eee | | | CR0 | invd | CR2 | CR3 | CR4 | invd | invd | invd |
| eee | | | DR0 | DR1 | DR2 | DR3 | DR4[1](http://ref.x86asm.net/coder32.html#modrm_dr4_dr5) | DR5[1](http://ref.x86asm.net/coder32.html#modrm_dr4_dr5) | DR6 | DR7 |
| (In decimal) /digit (Opcode) | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| (In binary) REG = | | | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| Effective Address | Mod | R/M | Value of ModR/M Byte (in Hex) | | | | | | | |
| [BX+SI] | 00 | 000 | 00 | 08 | 10 | 18 | 20 | 28 | 30 | 38 |
| [BX+DI] |  | 001 | 01 | 09 | 11 | 19 | 21 | 29 | 31 | 39 |
| [BP+SI] |  | 010 | 02 | 0A | 12 | 1A | 22 | 2A | 32 | 3A |
| [BP+DI] |  | 011 | 03 | 0B | 13 | 1B | 23 | 2B | 33 | 3B |
| [SI] |  | 100 | 04 | 0C | 14 | 1C | 24 | 2C | 34 | 3C |
| [DI] |  | 101 | 05 | 0D | 15 | 1D | 25 | 2D | 35 | 3D |
| disp16 |  | 110 | 06 | 0E | 16 | 1E | 26 | 2E | 36 | 3E |
| [BX] |  | 111 | 07 | 0F | 17 | 1F | 27 | 2F | 37 | 3F |
| [BX+SI]+disp8 | 01 | 000 | 40 | 48 | 50 | 58 | 60 | 68 | 70 | 78 |
| [BX+DI]+disp8 |  | 001 | 41 | 49 | 51 | 59 | 61 | 69 | 71 | 79 |
| [BP+SI]+disp8 |  | 010 | 42 | 4A | 52 | 5A | 62 | 6A | 72 | 7A |
| [BP+DI]+disp8 |  | 011 | 43 | 4B | 53 | 5B | 63 | 6B | 73 | 7B |
| [SI]+disp8 |  | 100 | 44 | 4C | 54 | 5C | 64 | 6C | 74 | 7C |
| [DI]+disp8 |  | 101 | 45 | 4D | 55 | 5D | 65 | 6D | 75 | 7D |
| [BP]+disp8 |  | 110 | 46 | 4E | 56 | 5E | 66 | 6E | 76 | 7E |
| [BX]+disp8 |  | 111 | 47 | 4F | 57 | 5F | 67 | 6F | 77 | 7F |
| [BX+SI]+disp16 | 10 | 000 | 80 | 88 | 90 | 98 | A0 | A8 | B0 | B8 |
| [BX+DI]+disp16 |  | 001 | 81 | 89 | 91 | 99 | A1 | A9 | B1 | B9 |
| [BP+SI]+disp16 |  | 010 | 82 | 8A | 92 | 9A | A2 | AA | B2 | BA |
| [BP+DI]+disp16 |  | 011 | 83 | 8B | 93 | 9B | A3 | AB | B3 | BB |
| [SI]+disp16 |  | 100 | 84 | 8C | 94 | 9C | A4 | AC | B4 | BC |
| [DI]+disp16 |  | 101 | 85 | 8D | 95 | 9D | A5 | AD | B5 | BD |
| [BP]+disp16 |  | 110 | 86 | 8E | 96 | 9E | A6 | AE | B6 | BE |
| [BX]+disp16 |  | 111 | 87 | 8F | 97 | 9F | A7 | AF | B7 | BF |
| AL/AX/EAX/ST0/MM0/XMM0 | 11 | 000 | C0 | C8 | D0 | D8 | E0 | E8 | F0 | F8 |
| CL/CX/ECX/ST1/MM1/XMM1 |  | 001 | C1 | C9 | D1 | D9 | E1 | E9 | F1 | F9 |
| DL/DX/EDX/ST2/MM2/XMM2 |  | 010 | C2 | CA | D2 | DA | E2 | EA | F2 | FA |
| BL/BX/EBX/ST3/MM3/XMM3 |  | 011 | C3 | CB | D3 | DB | E3 | EB | F3 | FB |
| AH/SP/ESP/ST4/MM4/XMM4 |  | 100 | C4 | CC | D4 | DC | E4 | EC | F4 | FC |
| CH/BP/EBP/ST5/MM5/XMM5 |  | 101 | C5 | CD | D5 | DD | E5 | ED | F5 | FD |
| DH/SI/ESI/ST6/MM6/XMM6 |  | 110 | C6 | CE | D6 | DE | E6 | EE | F6 | FE |
| BH/DI/EDI/ST7/MM7/XMM7 |  | 111 | C7 | CF | D7 | DF | E7 | EF | F7 | FF |

ModR/M Note 1: Debug Registers DR4 and DR5

References to debug registers DR4 and DR5 cause an undefined opcode (#UD) exception to be generated when CR4.DE[bit 3] (Debugging Extensions) set; when clear, processor aliases references to registers DR4 and DR5 to DR6 and DR7 for compatibility with software written to run on earlier IA-32 processors.

**Your Notes:**