

ref.x86asm.net

coder32

edition

| pf | OF | po | so | o | proc | st | m | r | l | mnemonic | op1 | op2 | op3 | op4 | iext | tested f | modif f | def f | undef f | f values | description, notes | | | | | | |
|----|----|----|----|-----|------|----|---|---|---|------------------|----------|----------|-----|-----|------|----------|----------|----------|-----------------|----------|--|----------------------------|--|--|--|--|---|
| | | 00 | r | | | | | | L | ADD | r/m8 | r8 | | | | | o..szapc | o..szapc | | | Add | | | | | | |
| | | 01 | r | | | | | | L | ADD | r/m16/32 | r16/32 | | | | | o..szapc | o..szapc | | | Add | | | | | | |
| | | 02 | r | | | | | | | ADD | r8 | r/m8 | | | | | o..szapc | o..szapc | | | Add | | | | | | |
| | | 03 | r | | | | | | | ADD | r16/32 | r/m16/32 | | | | | o..szapc | o..szapc | | | Add | | | | | | |
| | | 04 | | | | | | | | ADD | AL | imm8 | | | | | o..szapc | o..szapc | | | Add | | | | | | |
| | | 05 | | | | | | | | ADD | eAX | imm16/32 | | | | | o..szapc | o..szapc | | | Add | | | | | | |
| | | 06 | | | | | | | | PUSH | ES | | | | | | | | | | Push Word, Doubleword or Quadword Onto the Stack | | | | | | |
| | | 07 | | | | | | | | POP | ES | | | | | | | | | | | Pop a Value from the Stack | | | | | |
| | | 08 | r | | | | | | L | OR | r/m8 | r8 | | | | | o..szapc | o..szapc |a..o.....c | | Logical Inclusive OR | | | | | | |
| | | 09 | r | | | | | | L | OR | r/m16/32 | r16/32 | | | | | o..szapc | o..szapc |a..o.....c | | Logical Inclusive OR | | | | | | |
| | | 0A | r | | | | | | | OR | r8 | r/m8 | | | | | o..szapc | o..szapc |a..o.....c | | Logical Inclusive OR | | | | | | |
| | | 0B | r | | | | | | | OR | r16/32 | r/m16/32 | | | | | o..szapc | o..szapc |a..o.....c | | Logical Inclusive OR | | | | | | |
| | | 0C | | | | | | | | OR | AL | imm8 | | | | | o..szapc | o..szapc |a..o.....c | | Logical Inclusive OR | | | | | | |
| | | 0D | | | | | | | | OR | eAX | imm16/32 | | | | | o..szapc | o..szapc |a..o.....c | | Logical Inclusive OR | | | | | | |
| | | 0E | | | | | | | | PUSH | CS | | | | | | | | | | Push Word, Doubleword or Quadword Onto the Stack | | | | | | |
| | | 0F | | 02+ | | | | | | <i>two-byte</i> | | | | | | | | | | | | | | | | | |
| | | 10 | r | | | | | | L | ADC | r/m8 | r8 | | | |c | o..szapc | o..szapc | | | Add with Carry | | | | | | |
| | | 11 | r | | | | | | L | ADC | r/m16/32 | r16/32 | | | |c | o..szapc | o..szapc | | | Add with Carry | | | | | | |
| | | 12 | r | | | | | | | ADC | r8 | r/m8 | | | |c | o..szapc | o..szapc | | | Add with Carry | | | | | | |
| | | 13 | r | | | | | | | ADC | r16/32 | r/m16/32 | | | |c | o..szapc | o..szapc | | | Add with Carry | | | | | | |
| | | 14 | | | | | | | | ADC | AL | imm8 | | | |c | o..szapc | o..szapc | | | Add with Carry | | | | | | |
| | | 15 | | | | | | | | ADC | eAX | imm16/32 | | | |c | o..szapc | o..szapc | | | Add with Carry | | | | | | |
| | | 16 | | | | | | | | PUSH | SS | | | | | | | | | | Push Word, Doubleword or Quadword Onto the Stack | | | | | | |
| | | 17 | | | | | | | | POP | SS | | | | | | | | | | Pop a Value from the Stack | | | | | | |
| | | 18 | r | | | | | | L | SBB | r/m8 | r8 | | | |c | o..szapc | o..szapc | | | Integer Subtraction with Borrow | | | | | | |
| | | 19 | r | | | | | | L | SBB | r/m16/32 | r16/32 | | | |c | o..szapc | o..szapc | | | Integer Subtraction with Borrow | | | | | | |
| | | 1A | r | | | | | | | SBB | r8 | r/m8 | | | |c | o..szapc | o..szapc | | | Integer Subtraction with Borrow | | | | | | |
| | | 1B | r | | | | | | | SBB | r16/32 | r/m16/32 | | | |c | o..szapc | o..szapc | | | Integer Subtraction with Borrow | | | | | | |
| | | 1C | | | | | | | | SBB | AL | imm8 | | | |c | o..szapc | o..szapc | | | Integer Subtraction with Borrow | | | | | | |
| | | 1D | | | | | | | | SBB | eAX | imm16/32 | | | |c | o..szapc | o..szapc | | | Integer Subtraction with Borrow | | | | | | |
| | | 1E | | | | | | | | PUSH | DS | | | | | | | | | | Push Word, Doubleword or Quadword Onto the Stack | | | | | | |
| | | 1F | | | | | | | | POP | DS | | | | | | | | | | Pop a Value from the Stack | | | | | | |
| | | 20 | r | | | | | | L | AND | r/m8 | r8 | | | | | o..szapc | o..szapc |a..o.....c | | Logical AND | | | | | | |
| | | 21 | r | | | | | | L | AND | r/m16/32 | r16/32 | | | | | o..szapc | o..szapc |a..o.....c | | Logical AND | | | | | | |
| | | 22 | r | | | | | | | AND | r8 | r/m8 | | | | | o..szapc | o..szapc |a..o.....c | | Logical AND | | | | | | |
| | | 23 | r | | | | | | | AND | r16/32 | r/m16/32 | | | | | o..szapc | o..szapc |a..o.....c | | Logical AND | | | | | | |
| | | 24 | | | | | | | | AND | AL | imm8 | | | | | o..szapc | o..szapc |a..o.....c | | Logical AND | | | | | | |
| | | 25 | | | | | | | | AND | eAX | imm16/32 | | | | | o..szapc | o..szapc |a..o.....c | | Logical AND | | | | | | |
| | | 26 | | | | | | | | ES | ES | | | | | | | | | | ES segment override prefix | | | | | | |
| | | 26 | | P4+ | | | | | | <i>undefined</i> | | | | | | | | | | | | | | | | | (use with any branch instruction is reserved) |
| | | 27 | | | | | | | | DAA | AL | | | | |a.c | o..szapc | ...szapc | o..... | | Decimal Adjust AL after Addition | | | | | | |
| | | 28 | r | | | | | | L | SUB | r/m8 | r8 | | | | | o..szapc | o..szapc | | | Subtract | | | | | | |
| | | 29 | r | | | | | | L | SUB | r/m16/32 | r16/32 | | | | | o..szapc | o..szapc | | | Subtract | | | | | | |
| | | 2A | r | | | | | | | SUB | r8 | r/m8 | | | | | o..szapc | o..szapc | | | Subtract | | | | | | |
| | | 2B | r | | | | | | | SUB | r16/32 | r/m16/32 | | | | | o..szapc | o..szapc | | | Subtract | | | | | | |
| | | 2C | | | | | | | | SUB | AL | imm8 | | | | | o..szapc | o..szapc | | | Subtract | | | | | | |
| | | 2D | | | | | | | | SUB | eAX | imm16/32 | | | | | o..szapc | o..szapc | | | Subtract | | | | | | |
| | | 2E | | | | | | | | CS | CS | | | | | | | | | | CS segment override prefix | | | | | | |

| Op | OF | po | so | o | proc | st | m | r1 | l | mnemonic | op1 | op2 | op3 | op4 | iext | tested f | modif f | def f | undef f | f values | Description, notes |
|----|----|----|----|-----|------|----|---|----------------|---|-------------------|----------------------|---------------|-----|-----|------|----------|-----------|----------|---------|----------|---|
| | | 6D | | 03+ | | | | f ¹ | | INS INSD | m16/32 m32 | DX DX | | | | .d..... | | | | | Input from Port to String |
| | | 6E | | 01+ | | | | f ¹ | | OUTS OUTSB | DX DX | m8 m8 | | | | .d..... | | | | | Output String to Port |
| | | 6F | | 01+ | | | | f ¹ | | OUTS OUTSW | DX DX | m16 m16 | | | | .d..... | | | | | Output String to Port |
| | | 6F | | 03+ | | | | f ¹ | | OUTS OUTSD | DX DX | m16/32 m32 | | | | .d..... | | | | | Output String to Port |
| | | 70 | | | | | | | | JO | rel8 | | | | | o..... | | | | | Jump short if overflow (OF=1) |
| | | 71 | | | | | | | | JNO | rel8 | | | | | o..... | | | | | Jump short if not overflow (OF=0) |
| | | 72 | | | | | | | | JB JNAE JC | rel8 rel8 rel8 | | | | |c | | | | | Jump short if below/not above or equal/carry (CF=1) |
| | | 73 | | | | | | | | JNB JAE JNC | rel8 rel8 rel8 | | | | |c | | | | | Jump short if not below/above or equal/not carry (CF=0) |
| | | 74 | | | | | | | | JZ JE | rel8 rel8 | | | | |z... | | | | | Jump short if zero/equal (ZF=0) |
| | | 75 | | | | | | | | JNZ JNE | rel8 rel8 | | | | |z... | | | | | Jump short if not zero/not equal (ZF=1) |
| | | 76 | | | | | | | | JBE JNA | rel8 rel8 | | | | |z.c | | | | | Jump short if below or equal/not above (CF=1 AND ZF=1) |
| | | 77 | | | | | | | | JNBE JA | rel8 rel8 | | | | |z.c | | | | | Jump short if not below or equal/above (CF=0 AND ZF=0) |
| | | 78 | | | | | | | | JS | rel8 | | | | | ...s... | | | | | Jump short if sign (SF=1) |
| | | 79 | | | | | | | | JNS | rel8 | | | | | ...s... | | | | | Jump short if not sign (SF=0) |
| | | 7A | | | | | | | | JP JPE | rel8 rel8 | | | | |p. | | | | | Jump short if parity/parity even (PF=1) |
| | | 7B | | | | | | | | JNP JPO | rel8 rel8 | | | | |p. | | | | | Jump short if not parity/parity odd |
| | | 7C | | | | | | | | JL JNGE | rel8 rel8 | | | | | o..s... | | | | | Jump short if less/not greater (SF!=OF) |
| | | 7D | | | | | | | | JNL JGE | rel8 rel8 | | | | | o..s... | | | | | Jump short if not less/greater or equal (SF=OF) |
| | | 7E | | | | | | | | JLE JNG | rel8 rel8 | | | | | o..sz... | | | | | Jump short if less or equal/not greater ((ZF=1) OR (SF!=OF)) |
| | | 7F | | | | | | | | JNLE JG | rel8 rel8 | | | | | o..sz... | | | | | Jump short if not less nor equal/greater ((ZF=0) AND (SF=OF)) |
| | | 80 | 0 | | | | | | L | ADD | r/m8 | imm8 | | | | o..szapc | o..szapc | | | | Add |
| | | 80 | 1 | | | | | | L | OR | r/m8 | imm8 | | | | o..szapc | o..sz.p.c |a.. | o.....c | | Logical Inclusive OR |
| | | 80 | 2 | | | | | | L | ADC | r/m8 | imm8 | | | |c | o..szapc | o..szapc | | | Add with Carry |
| | | 80 | 3 | | | | | | L | SBB | r/m8 | imm8 | | | |c | o..szapc | o..szapc | | | Integer Subtraction with Borrow |
| | | 80 | 4 | | | | | | L | AND | r/m8 | imm8 | | | | o..szapc | o..sz.p.c |a.. | o.....c | | Logical AND |
| | | 80 | 5 | | | | | | L | SUB | r/m8 | imm8 | | | | o..szapc | o..szapc | | | | Subtract |
| | | 80 | 6 | | | | | | L | XOR | r/m8 | imm8 | | | | o..szapc | o..sz.p.c |a.. | o.....c | | Logical Exclusive OR |
| | | 80 | 7 | | | | | | | CMP | r/m8 | imm8 | | | | o..szapc | o..szapc | | | | Compare Two Operands |
| | | 81 | 0 | | | | | | L | ADD | r/m16/32 | imm16/32 | | | | o..szapc | o..szapc | | | | Add |
| | | 81 | 1 | | | | | | L | OR | r/m16/32 | imm16/32 | | | | o..szapc | o..sz.p.c |a.. | o.....c | | Logical Inclusive OR |
| | | 81 | 2 | | | | | | L | ADC | r/m16/32 | imm16/32 | | | |c | o..szapc | o..szapc | | | Add with Carry |

| pf | OF | po | so | o | proc | st | m | r1 | l | mnemonic | op1 | op2 | op3 | op4 | iext | tested f | modif f | def f | undef f | f values | description, notes | | | | | | | |
|----|----|------|----|-----|------|----------------|---|----|-----------------|-------------|----------|----------|-----|-----|------|----------|----------|----------|----------|----------|--------------------|--|--|--|--|--|--|--------------|
| | | 81 | 3 | | | | | | L | SBB | r/m16/32 | imm16/32 | | | |c | o..szapc | o..szapc | | | | Integer Subtraction with Borrow | | | | | | |
| | | 81 | 4 | | | | | | L | AND | r/m16/32 | imm16/32 | | | | | o..szapc | o..sz.pc |a.. | o.....c | | Logical AND | | | | | | |
| | | 81 | 5 | | | | | | L | SUB | r/m16/32 | imm16/32 | | | | | o..szapc | o..szapc | | | | Subtract | | | | | | |
| | | 81 | 6 | | | | | | L | XOR | r/m16/32 | imm16/32 | | | | | o..szapc | o..sz.pc |a.. | o.....c | | Logical Exclusive OR | | | | | | |
| | | 81 | 7 | | | | | | CMP | r/m16/32 | imm16/32 | | | | | | o..szapc | o..szapc | | | | Compare Two Operands | | | | | | |
| | | 82 | 0 | | | | | | L | ADD | r/m8 | imm8 | | | | | o..szapc | o..szapc | | | | Add | | | | | | |
| | | 82 | 1 | | | | | | L | OR | r/m8 | imm8 | | | | | o..szapc | o..sz.pc |a.. | o.....c | | Logical Inclusive OR | | | | | | |
| | | 82 | 2 | | | | | | L | ADC | r/m8 | imm8 | | | |c | o..szapc | o..szapc | | | | Add with Carry | | | | | | |
| | | 82 | 3 | | | | | | L | SBB | r/m8 | imm8 | | | |c | o..szapc | o..szapc | | | | Integer Subtraction with Borrow | | | | | | |
| | | 82 | 4 | | | | | | L | AND | r/m8 | imm8 | | | | | o..szapc | o..sz.pc |a.. | o.....c | | Logical AND | | | | | | |
| | | 82 | 5 | | | | | | L | SUB | r/m8 | imm8 | | | | | o..szapc | o..szapc | | | | Subtract | | | | | | |
| | | 82 | 6 | | | | | | L | XOR | r/m8 | imm8 | | | | | o..szapc | o..sz.pc |a.. | o.....c | | Logical Exclusive OR | | | | | | |
| | | 82 | 7 | | | | | | CMP | r/m8 | imm8 | | | | | | o..szapc | o..szapc | | | | Compare Two Operands | | | | | | |
| | | 83 | 0 | | | | | | L | ADD | r/m16/32 | imm8 | | | | | o..szapc | o..szapc | | | | Add | | | | | | |
| | | 83 | 1 | 03+ | | | | | L | OR | r/m16/32 | imm8 | | | | | o..szapc | o..sz.pc |a.. | o.....c | | Logical Inclusive OR | | | | | | |
| | | 83 | 2 | | | | | | L | ADC | r/m16/32 | imm8 | | | |c | o..szapc | o..szapc | | | | Add with Carry | | | | | | |
| | | 83 | 3 | | | | | | L | SBB | r/m16/32 | imm8 | | | |c | o..szapc | o..szapc | | | | Integer Subtraction with Borrow | | | | | | |
| | | 83 | 4 | 03+ | | | | | L | AND | r/m16/32 | imm8 | | | | | o..szapc | o..sz.pc |a.. | o.....c | | Logical AND | | | | | | |
| | | 83 | 5 | | | | | | L | SUB | r/m16/32 | imm8 | | | | | o..szapc | o..szapc | | | | Subtract | | | | | | |
| | | 83 | 6 | 03+ | | | | | L | XOR | r/m16/32 | imm8 | | | | | o..szapc | o..sz.pc |a.. | o.....c | | Logical Exclusive OR | | | | | | |
| | | 83 | 7 | | | | | | CMP | r/m16/32 | imm8 | | | | | | o..szapc | o..szapc | | | | Compare Two Operands | | | | | | |
| | | 84 | r | | | | | | TEST | r/m8 | r8 | | | | | | o..szapc | o..sz.pc |a.. | o.....c | | Logical Compare | | | | | | |
| | | 85 | r | | | | | | TEST | r/m16/32 | r16/32 | | | | | | o..szapc | o..sz.pc |a.. | o.....c | | Logical Compare | | | | | | |
| | | 86 | r | | | | | | L | XCHG | r8 | r/m8 | | | | | | | | | | Exchange Register/Memory with Register | | | | | | |
| | | 87 | r | | | | | | L | XCHG | r16/32 | r/m16/32 | | | | | | | | | | Exchange Register/Memory with Register | | | | | | |
| | | 88 | r | | | | | | MOV | r/m8 | r8 | | | | | | | | | | | Move | | | | | | |
| | | 89 | r | | | | | | MOV | r/m16/32 | r16/32 | | | | | | | | | | | Move | | | | | | |
| | | 8A | r | | | | | | MOV | r8 | r/m8 | | | | | | | | | | | Move | | | | | | |
| | | 8B | r | | | | | | MOV | r16/32 | r/m16/32 | | | | | | | | | | | Move | | | | | | |
| | | 8C | r | | | | | | MOV | r/m16/32 | Sreg | | | | | | | | | | | Move | | | | | | |
| | | 8D | r | | | | | | LEA | r16/32 | m | | | | | | | | | | | Load Effective Address | | | | | | |
| | | 8E | r | | | | | | MOV | Sreg | r/m16/32 | | | | | | | | | | | Move | | | | | | |
| | | 8F | 0 | | | | | | POP | r/m16/32 | | | | | | | | | | | | Pop a Value from the Stack | | | | | | |
| | | 90+r | | | | | | | XCHG | r16/32 | eAX | | | | | | | | | | | Exchange Register/Memory with Register | | | | | | |
| | | 90 | | | | | | | NOP | | | | | | | | | | | | | No Operation | | | | | | |
| F3 | | 90 | | | | D ³ | | | no mnemonic nop | | | | | | | | | | | | | | | | | | | No Operation |
| F3 | | 90 | | P4+ | | | | | PAUSE | | | | | | | | | | | | | Spin Loop Hint | | | | | | |
| | | 98 | | | | | | | CBW | AH | AL | | | | | | | | | | | Convert Byte to Word | | | | | | |
| | | 98 | | 03+ | | | | | CWDE | EAX | AX | | | | | | | | | | | Convert Word to Doubleword | | | | | | |
| | | 99 | | | | | | | CWD | DX | AX | | | | | | | | | | | Convert Word to Doubleword | | | | | | |
| | | 99 | | 03+ | | | | | CDQ | EDX | EAX | | | | | | | | | | | Convert Doubleword to Quadword | | | | | | |
| | | 9A | | | | | | | CALLF | ptr16:16/32 | | | | | | | | | | | | Call Procedure | | | | | | |
| | | 9C | | | | | | | PUSHF | | | | | | | | odiszapc | | | | | Push rFLAGS Register onto the Stack | | | | | | |
| | | 9C | | 03+ | | | | | PUSHF | | | | | | | | odiszapc | | | | | Push rFLAGS Register onto the Stack | | | | | | |
| | | 9C | | | | | | | PUSHFD | | | | | | | | | | | | | Push rFLAGS Register onto the Stack | | | | | | |
| | | 9D | | | | | | | POPF | | | | | | | | odiszapc | odiszapc | | | | Pop Stack into rFLAGS Register | | | | | | |
| | | 9D | | 03+ | | | | | POPF | | | | | | | | odiszapc | odiszapc | | | | Pop Stack into rFLAGS Register | | | | | | |
| | | 9D | | | | | | | POPF | | | | | | | | | | | | | Pop Stack into rFLAGS Register | | | | | | |
| | | 9D | | 03+ | | | | | POPF | | | | | | | | | | | | | Pop Stack into rFLAGS Register | | | | | | |

| pf | OF | po | so | o | proc | st | mr | l | l | mnemonic | op1 | op2 | op3 | op4 | iext | tested f | modif f | def f | undef f | f values | description, notes | |
|----|----|----|----|-----|------|----------------|----|---|---|----------|----------|----------|-----------|-----|--------|----------|----------|----------|----------|----------|--------------------|---|
| | | C0 | 7 | | | | | | | SAR | r/m8 | imm8 | | | | | o..szapc | o..sz.pc | o....a.. | | Shift | |
| | | C1 | 0 | | | | | | | ROL | r/m16/32 | imm8 | | | | | o..szapc | o..szapc | o..... | | Rotate | |
| | | C1 | 1 | | | | | | | ROR | r/m16/32 | imm8 | | | | | o..szapc | o..szapc | o..... | | Rotate | |
| | | C1 | 2 | | | | | | | RCL | r/m16/32 | imm8 | | | |c | o..szapc | o..szapc | o..... | | Rotate | |
| | | C1 | 3 | | | | | | | RCR | r/m16/32 | imm8 | | | |c | o..szapc | o..szapc | o..... | | Rotate | |
| | | C1 | 4 | | | | | | | SHL | r/m16/32 | imm8 | | | | | o..szapc | o..sz.pc | o....a.c | | Shift | |
| | | | | | | | | | | SAL | r/m16/32 | imm8 | | | | | | | | | | |
| | | C1 | 5 | | | | | | | SHR | r/m16/32 | imm8 | | | | | o..szapc | o..sz.pc | o....a.c | | Shift | |
| | | C1 | 6 | | | U ⁵ | | | | SAL | r/m16/32 | imm8 | | | | | o..szapc | o..sz.pc | o....a.c | | Shift | |
| | | | | | | | | | | SHL | r/m16/32 | imm8 | | | | | | | | | | |
| | | C1 | 7 | | | | | | | SAR | r/m16/32 | imm8 | | | | | o..szapc | o..sz.pc | o....a.. | | Shift | |
| | | C2 | | | | | | | | RETN | imm16 | | | | | | | | | | | Return from procedure |
| | | C3 | | | | | | | | RETN | | | | | | | | | | | | Return from procedure |
| | | C4 | r | | | | | | | LES | ES | r16/32 | m16:16/32 | | | | | | | | | Load Far Pointer |
| | | C5 | r | | | | | | | LDS | DS | r16/32 | m16:16/32 | | | | | | | | | Load Far Pointer |
| | | C6 | 0 | | | | | | | MOV | r/m8 | imm8 | | | | | | | | | | Move |
| | | C7 | 0 | | | | | | | MOV | r/m16/32 | imm16/32 | | | | | | | | | | Move |
| | | C8 | | 01+ | | | | | | ENTER | eBP | imm16 | imm8 | | | | | | | | | Make Stack Frame for Procedure Parameters |
| | | C9 | | 01+ | | | | | | LEAVE | eBP | | | | | | | | | | | High Level Procedure Exit |
| | | CA | | | | | | f | | RETF | imm16 | | | | | | | | | | | Return from procedure |
| | | CB | | | | | | f | | RETF | | | | | | | | | | | | Return from procedure |
| | | CC | | | | | | f | | INT | 3 | | | | | ..i..... | ..i..... | | ..i..... | | | Call to Interrupt Procedure |
| | | CD | | | | | | f | | INT | imm8 | | | | | ..i..... | ..i..... | | ..i..... | | | Call to Interrupt Procedure |
| | | CE | | | | | | f | | INTO | | | | | o..... | ..i..... | ..i..... | | ..i..... | | | Call to Interrupt Procedure |
| | | CF | | | | | | f | | IRET | | | | | | odiszapc | odiszapc | | | | | Interrupt Return |
| | | CF | | 03+ | | | | f | | IRETD | | | | | | odiszapc | odiszapc | | | | | Interrupt Return |
| | | D0 | 0 | | | | | | | ROL | r/m8 | 1 | | | | o..szapc | o..szapc | | | | | Rotate |
| | | D0 | 1 | | | | | | | ROR | r/m8 | 1 | | | | o..szapc | o..szapc | | | | | Rotate |
| | | D0 | 2 | | | | | | | RCL | r/m8 | 1 | | | |c | o..szapc | o..szapc | | | | Rotate |
| | | D0 | 3 | | | | | | | RCR | r/m8 | 1 | | | |c | o..szapc | o..szapc | | | | Rotate |
| | | D0 | 4 | | | | | | | SHL | r/m8 | 1 | | | | | o..szapc | o..sz.pc |a.. | | | Shift |
| | | | | | | | | | | SAL | r/m8 | 1 | | | | | | | | | | |
| | | D0 | 5 | | | | | | | SHR | r/m8 | 1 | | | | | o..szapc | o..sz.pc |a.. | | | Shift |
| | | D0 | 6 | | | U ⁵ | | | | SAL | r/m8 | 1 | | | | | o..szapc | o..sz.pc |a.. | | | Shift |
| | | | | | | | | | | SHL | r/m8 | 1 | | | | | | | | | | |
| | | D0 | 7 | | | | | | | SAR | r/m8 | 1 | | | | | o..szapc | o..sz.pc |a.. | | | Shift |
| | | D1 | 0 | | | | | | | ROL | r/m16/32 | 1 | | | | | o..szapc | o..szapc | | | | Rotate |
| | | D1 | 1 | | | | | | | ROR | r/m16/32 | 1 | | | | | o..szapc | o..szapc | | | | Rotate |
| | | D1 | 2 | | | | | | | RCL | r/m16/32 | 1 | | | |c | o..szapc | o..szapc | | | | Rotate |
| | | D1 | 3 | | | | | | | RCR | r/m16/32 | 1 | | | |c | o..szapc | o..szapc | | | | Rotate |
| | | D1 | 4 | | | | | | | SHL | r/m16/32 | 1 | | | | | o..szapc | o..sz.pc |a.. | | | Shift |
| | | | | | | | | | | SAL | r/m16/32 | 1 | | | | | | | | | | |
| | | D1 | 5 | | | | | | | SHR | r/m16/32 | 1 | | | | | o..szapc | o..sz.pc |a.. | | | Shift |
| | | D1 | 6 | | | U ⁵ | | | | SAL | r/m16/32 | 1 | | | | | o..szapc | o..sz.pc |a.. | | | Shift |
| | | | | | | | | | | SHL | r/m16/32 | 1 | | | | | | | | | | |
| | | D1 | 7 | | | | | | | SAR | r/m16/32 | 1 | | | | | o..szapc | o..sz.pc |a.. | | | Shift |
| | | D2 | 0 | | | | | | | ROL | r/m8 | CL | | | | | o..szapc | o..szapc | o..... | | | Rotate |
| | | D2 | 1 | | | | | | | ROR | r/m8 | CL | | | | | o..szapc | o..szapc | o..... | | | Rotate |

| pf | OF | po | so | o | proc | st | mr | l | mnemonic | op1 | op2 | op3 | op4 | iext | tested f | modif f | def f | undef f | f values | description, notes |
|----|----|----|-----|----------------|----------------|----|----------------|---|------------------|-----------------|-----------|------|-----|------|----------|----------|-----------|----------|----------|--|
| | | D2 | 2 | | | | | | RCL | r/m8 | CL | | | |c | o..szapc | o..szapc | o..... | | Rotate |
| | | D2 | 3 | | | | | | RCR | r/m8 | CL | | | |c | o..szapc | o..szapc | o..... | | Rotate |
| | | D2 | 4 | | | | | | SHL | r/m8 | CL | | | | | o..szapc | o..sz.p.c | o....a.c | | Shift |
| | | | | | | | | | SAL | r/m8 | CL | | | | | | | | | |
| | | D2 | 5 | | | | | | SHR | r/m8 | CL | | | | | o..szapc | o..sz.p.c | o....a.c | | Shift |
| | | D2 | 6 | | U ⁵ | | | | SAL | r/m8 | CL | | | | | o..szapc | o..sz.p.c | o....a.c | | Shift |
| | | | | | | | | | SHL | r/m8 | CL | | | | | | | | | |
| | | D2 | 7 | | | | | | SAR | r/m8 | CL | | | | | o..szapc | o..sz.p.c | o....a.. | | Shift |
| | | D3 | 0 | | | | | | ROL | r/m16/32 | CL | | | | | o..szapc | o..szapc | o..... | | Rotate |
| | | D3 | 1 | | | | | | ROR | r/m16/32 | CL | | | | | o..szapc | o..szapc | o..... | | Rotate |
| | | D3 | 2 | | | | | | RCL | r/m16/32 | CL | | | |c | o..szapc | o..szapc | o..... | | Rotate |
| | | D3 | 3 | | | | | | RCR | r/m16/32 | CL | | | |c | o..szapc | o..szapc | o..... | | Rotate |
| | | D3 | 4 | | | | | | SHL | r/m16/32 | CL | | | | | o..szapc | o..sz.p.c | o....a.c | | Shift |
| | | | | | | | | | SAL | r/m16/32 | CL | | | | | | | | | |
| | | D3 | 5 | | | | | | SHR | r/m16/32 | CL | | | | | o..szapc | o..sz.p.c | o....a.c | | Shift |
| | | D3 | 6 | | U ⁵ | | | | SAL | r/m16/32 | CL | | | | | o..szapc | o..sz.p.c | o....a.c | | Shift |
| | | | | | | | | | SHL | r/m16/32 | CL | | | | | | | | | |
| | | D3 | 7 | | | | | | SAR | r/m16/32 | CL | | | | | o..szapc | o..sz.p.c |a.. | | Shift |
| | | D4 | 0A | | | | | | AAM | AL | AH | | | | | o..szapc | ...sz.p. | o....a.c | | ASCII Adjust AX After Multiply |
| | | D4 | | | | | | | AMX | AL | AH | imm8 | | | | o..szapc | ...sz.p. | o....a.c | | Adjust AX After Multiply |
| | | D5 | 0A | | | | | | AAD | AL | AH | | | | | o..szapc | ...sz.p. | o....a.c | | ASCII Adjust AX Before Division |
| | | D5 | | | | | | | ADX | AL | AH | imm8 | | | | o..szapc | ...sz.p. | o....a.c | | Adjust AX Before Division |
| | | D6 | | 02+ | D ⁶ | | | | <i>undefined</i> | | | | | | | | | | | Undefined and Reserved; Does not Generate #UD |
| | | D6 | 02+ | U ⁷ | | | | | SALC | AL | | | | |c | | | | | Set AL If Carry |
| | | | | | | | | | SETALC | AL | | | | | | | | | | |
| | | D7 | | | | | | | XLAT | AL | m8 | | | | | | | | | Table Look-up Translation |
| | | | | | | | | | XLATB | AL | m8 | | | | | | | | | |
| | | E0 | | | | | | | LOOPNZ | eCX | rel8 | | | |z... | | | | | Decrement count; Jump short if count!=0 and ZF=0 |
| | | | | | | | | | LOOPNE | eCX | rel8 | | | | | | | | | |
| | | E1 | | | | | | | LOOPZ | eCX | rel8 | | | |z... | | | | | Decrement count; Jump short if count!=0 and ZF=1 |
| | | | | | | | | | LOOPE | eCX | rel8 | | | | | | | | | |
| | | E2 | | | | | | | LOOP | eCX | rel8 | | | | | | | | | Decrement count; Jump short if count!=0 |
| | | E3 | | | | | | | JCXZ | rel8 | CX | | | | | | | | | Jump short if eCX register is 0 |
| | | | | | | | | | JECXZ | rel8 | ECX | | | | | | | | | |
| | | E4 | | | | | f ¹ | | IN | AL | imm8 | | | | | | | | | Input from Port |
| | | E5 | | | | | f ¹ | | IN | eAX | imm8 | | | | | | | | | Input from Port |
| | | E6 | | | | | f ¹ | | OUT | imm8 | AL | | | | | | | | | Output to Port |
| | | E7 | | | | | f ¹ | | OUT | imm8 | eAX | | | | | | | | | Output to Port |
| | | E8 | | | | | | | CALL | rel16/32 | | | | | | | | | | Call Procedure |
| | | E9 | | | | | | | JMP | rel16/32 | | | | | | | | | | Jump |
| | | EA | | | | | | | JMPF | ptr16:16/32 | | | | | | | | | | Jump |
| | | EB | | | | | | | JMP | rel8 | | | | | | | | | | Jump |
| | | EC | | | | | f ¹ | | IN | AL | DX | | | | | | | | | Input from Port |
| | | ED | | | | | f ¹ | | IN | eAX | DX | | | | | | | | | Input from Port |
| | | EE | | | | | f ¹ | | OUT | DX | AL | | | | | | | | | Output to Port |
| | | EF | | | | | f ¹ | | OUT | DX | eAX | | | | | | | | | Output to Port |
| | | F0 | | | | | | | LOCK | | | | | | | | | | | Assert LOCK# Signal Prefix |

| pf | OF | po | so | proc | st | m | r1 | l | mnemonic | op1 | op2 | op3 | op4 | iext | tested f | modif f | def f | undef f | f values | description, notes |
|----|----|----|----|------|----------------|-----------------|----|---|-------------|--------------------|------------|----------|------|------|----------|----------|----------|----------|----------------------|---|
| | | F1 | | | D ⁶ | | | | undefined | | | | | | | | | | | Undefined and Reserved; Does not Generate #UD |
| | | F1 | | 03+ | U ⁸ | | | | INT1 | | | | | | | ..i..... | ..i..... | | ..i..... | Call to Interrupt Procedure |
| | | F2 | | | | | | | REPZ | eCX | | | | |z... | | | | | Repeat String Operation Prefix |
| | | F2 | | | | | | | REPNE | eCX | | | | | | | | | | Repeat String Operation Prefix |
| | | F2 | | | U | | | | REP | eCX | | | | | | | | | | Repeat String Operation Prefix |
| | | F2 | | P4+ | M | | | | no mnemonic | | | | sse2 | | | | | | | Scalar Double-precision Prefix |
| | | F3 | | | | | | | REPZ | eCX | | | | |z... | | | | | Repeat String Operation Prefix |
| | | F3 | | | | | | | REPE | eCX | | | | | | | | | | Repeat String Operation Prefix |
| | | F3 | | | | | | | REP | eCX | | | | | | | | | | Repeat String Operation Prefix |
| | | F3 | | P3+ | M | | | | no mnemonic | | | | sse | | | | | | | Scalar Single-precision Prefix |
| | | F4 | | | | | | | HLT | | | | | | | | | | | Halt |
| | | F5 | | | | | | | CMC | | | | | |c |c |c | | | Complement Carry Flag |
| | | F6 | | 0 | | | | | TEST | r/m8 | imm8 | | | | | o..szapc | o..sz.pc |a.. | o.....c | Logical Compare |
| | | F6 | | 1 | | U ⁹ | | | TEST | r/m8 | imm8 | | | | | o..szapc | o..sz.pc |a.. | o.....c | Logical Compare |
| | | F6 | | 2 | | | | | NOT | r/m8 | | | | | | | | | | One's Complement Negation |
| | | F6 | | 3 | | | | | NEG | r/m8 | | | | | | o..szapc | o..szapc | | | Two's Complement Negation |
| | | F6 | | 4 | | | | | MUL | AX | AL | r/m8 | | | | o..szapc | o.....c | ...szap. | | Unsigned Multiply |
| | | F6 | | 5 | | | | | IMUL | AX | AL | r/m8 | | | | o..szapc | o.....c | ...szap. | | Signed Multiply |
| | | F6 | | 6 | | | | | DIV | AL | AH | AX | r/m8 | | | o..szapc | | o..szapc | | Unsigned Divide |
| | | F6 | | 7 | | | | | IDIV | AL | AH | AX | r/m8 | | | o..szapc | | o..szapc | | Signed Divide |
| | | F7 | | 0 | | | | | TEST | r/m16/32 | imm16/32 | | | | | o..szapc | o..sz.pc |a.. | o.....c | Logical Compare |
| | | F7 | | 1 | | U ⁹ | | | TEST | r/m16/32 | imm16/32 | | | | | o..szapc | o..sz.pc |a.. | o.....c | Logical Compare |
| | | F7 | | 2 | | | | | NOT | r/m16/32 | | | | | | | | | | One's Complement Negation |
| | | F7 | | 3 | | | | | NEG | r/m16/32 | | | | | | o..szapc | o..szapc | | | Two's Complement Negation |
| | | F7 | | 4 | | | | | MUL | eDX | eAX | r/m16/32 | | | | o..szapc | o.....c | ...szap. | | Unsigned Multiply |
| | | F7 | | 5 | | | | | IMUL | eDX | eAX | r/m16/32 | | | | o..szapc | o.....c | ...szap. | | Signed Multiply |
| | | F7 | | 6 | | | | | DIV | eDX | eAX | r/m16/32 | | | | o..szapc | | o..szapc | | Unsigned Divide |
| | | F7 | | 7 | | | | | IDIV | eDX | eAX | r/m16/32 | | | | o..szapc | | o..szapc | | Signed Divide |
| | | F8 | | | | | | | CLC | | | | | |c |c | |c | Clear Carry Flag | |
| | | F9 | | | | | | | STC | | | | | |c |c | |c | Set Carry Flag | |
| | | FA | | | | f ¹ | | | CLI | | | | | | ..i..... | ..i..... | | ..i..... | Clear Interrupt Flag | |
| | | FB | | | | f ¹ | | | STI | | | | | | ..i..... | ..i..... | | ..I..... | Set Interrupt Flag | |
| | | FC | | | | | | | CLD | | | | | | .d..... | .d..... | | .d..... | Clear Direction Flag | |
| | | FD | | | | | | | STC | | | | | | .d..... | .d..... | | .D..... | Set Direction Flag | |
| | | FE | | 0 | | | | | INC | r/m8 | | | | | | o..szap. | o..szap. | | | Increment by 1 |
| | | FE | | 1 | | | | | DEC | r/m8 | | | | | | o..szap. | o..szap. | | | Decrement by 1 |
| | | FF | | 0 | | | | | INC | r/m16/32 | | | | | | o..szap. | o..szap. | | | Increment by 1 |
| | | FF | | 1 | | | | | DEC | r/m16/32 | | | | | | o..szap. | o..szap. | | | Decrement by 1 |
| | | FF | | 2 | | | | | CALL | r/m16/32 | | | | | | | | | | Call Procedure |
| | | FF | | 3 | | D ¹⁰ | | | CALLF | r/m16:16/32 | | | | | | | | | | Call Procedure |
| | | FF | | 4 | | | | | JMP | r/m16/32 | | | | | | | | | | Jump |
| | | FF | | 5 | | D ¹⁰ | | | JMPF | r/m16:16/32 | | | | | | | | | | Jump |
| | | FF | | 6 | | | | | PUSH | r/m16/32 | | | | | | | | | | Pop a Value from the Stack |

| pf | OF | po | so | o | proc | st | mr | l | mnemonic | opl | op2 | op3 | op4 | ix | tested | f | modif | f | def | f | undef | f | f values | description, notes | |
|----|----|----|----|---|------|-----------------|----|---|----------|----------|--------|------|-----|----|--------|----------|----------|----------|-----|---|-------|---|----------|---|---|
| | 0F | 8D | | | 03+ | | | | JNL | r16/32 | | | | | | o..s.... | | | | | | | | Jump short if not less/greater or equal (SF=OF) | |
| | | | | | | | | | JGE | r16/32 | | | | | | | | | | | | | | | |
| | 0F | 8E | | | 03+ | | | | JLE | r16/32 | | | | | | o..sz... | | | | | | | | | Jump short if less or equal/not greater ((ZF=1) OR (SF!=OF)) |
| | | | | | | | | | JNG | r16/32 | | | | | | | | | | | | | | | |
| | 0F | 8F | | | 03+ | | | | JNLE | r16/32 | | | | | | o..sz... | | | | | | | | | Jump short if not less nor equal/greater ((ZF=0) AND (SF=OF)) |
| | | | | | | | | | JG | r16/32 | | | | | | | | | | | | | | | |
| | 0F | 90 | | 0 | 03+ | D ¹⁸ | | | SETO | r/m8 | | | | | | o..... | | | | | | | | Set Byte on Condition - overflow (OF=1) | |
| | 0F | 91 | | 0 | 03+ | D ¹⁸ | | | SETNO | r/m8 | | | | | | o..... | | | | | | | | Set Byte on Condition - not overflow (OF=0) | |
| | | | | | | | | | SETB | r/m8 | | | | | | | | | | | | | | | |
| | 0F | 92 | | 0 | 03+ | D ¹⁸ | | | SETNAE | r/m8 | | | | | |c | | | | | | | | Set Byte on Condition - below/not above or equal/carry (CF=1) | |
| | | | | | | | | | SETC | r/m8 | | | | | | | | | | | | | | | |
| | | | | | | | | | SETNB | r/m8 | | | | | | | | | | | | | | | |
| | 0F | 93 | | 0 | 03+ | D ¹⁸ | | | SETAE | r/m8 | | | | | |c | | | | | | | | Set Byte on Condition - not below/above or equal/not carry (CF=0) | |
| | | | | | | | | | SETNC | r/m8 | | | | | | | | | | | | | | | |
| | | | | | | | | | SETZ | r/m8 | | | | | |z... | | | | | | | | | Set Byte on Condition - zero/equal (ZF=0) |
| | | | | | | | | | SETE | r/m8 | | | | | | | | | | | | | | | |
| | 0F | 95 | | 0 | 03+ | D ¹⁸ | | | SETNZ | r/m8 | | | | | |z... | | | | | | | | Set Byte on Condition - not zero/not equal (ZF=1) | |
| | | | | | | | | | SETNE | r/m8 | | | | | | | | | | | | | | | |
| | | | | | | | | | SETBE | r/m8 | | | | | |z..c | | | | | | | | Set Byte on Condition - below or equal/not above (CF=1 AND ZF=1) | |
| | | | | | | | | | SETNA | r/m8 | | | | | | | | | | | | | | | |
| | | | | | | | | | 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 | | | SETS | r/m8 | | | | | | ...s.... | | | | | | | | Set Byte on Condition - sign (SF=1) | |
| | 0F | 99 | | 0 | 03+ | D ¹⁸ | | | SETNS | r/m8 | | | | | | ...s.... | | | | | | | | Set Byte on Condition - not sign (SF=0) | |
| | | | | | | | | | SETP | r/m8 | | | | | |p. | | | | | | | | Set Byte on Condition - parity/parity even (PF=1) | |
| | | | | | | | | | SETPE | r/m8 | | | | | | | | | | | | | | | |
| | 0F | 9B | | 0 | 03+ | D ¹⁸ | | | SETNP | r/m8 | | | | | |p. | | | | | | | | Set Byte on Condition - not parity/parity odd | |
| | | | | | | | | | SETPO | r/m8 | | | | | | | | | | | | | | | |
| | | | | | | | | | SETL | r/m8 | | | | | | o..s.... | | | | | | | | Set Byte on Condition - less/not greater (SF!=OF) | |
| | | | | | | | | | SETNGE | r/m8 | | | | | | | | | | | | | | | |
| | 0F | 9D | | 0 | 03+ | D ¹⁸ | | | SETNL | r/m8 | | | | | | o..s.... | | | | | | | | Set Byte on Condition - not less/greater or equal (SF=OF) | |
| | | | | | | | | | SETGE | r/m8 | | | | | | | | | | | | | | | |
| | | | | | | | | | 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 | | | 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 | ... | | | | | | | | | | | | | | CPU Identification | |
| | 0F | A3 | | | 03+ | | | | BT | r/m16/32 | r16/32 | | | | | o..szapc |c | o..szap. | | | | | | Bit Test | |
| | 0F | A4 | | | 03+ | | | | SHLD | r/m16/32 | r16/32 | imm8 | | | | o..szapc | o..sz.pc | o....a.c | | | | | | Double Precision Shift Left | |
| | 0F | A5 | | | 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 | | | | | | | odiszapc | odiszapc | | | | | | | Resume from System Management Mode | |
| | 0F | AB | | | 03+ | | | L | BTS | r/m16/32 | r16/32 | | | | | o..szapc |c | o..szap. | | | | | | Bit Test and Set | |
| | 0F | AC | | | 03+ | | | | SHRD | r/m16/32 | r16/32 | imm8 | | | | o..szapc | o..sz.pc | o....a.c | | | | | | Double Precision Shift Right | |
| | 0F | AD | | | 03+ | | | | SHRD | r/m16/32 | r16/32 | CL | | | | o..szapc | o..sz.pc | o....a.c | | | | | | Double Precision Shift Right | |

General notes:

1. a. OPLOCK.LST, Revision 4.51, 15 Oct 1999 © Potemkin's Hackers Group 1994...1999
2. a. The microarchitecture of Intel and AMD CPUs, By Agner Fog, Copyright © 1996 - 2006.
3. a. Intel® 64 and IA-32 Architecture Software Developer's Manual Volume 2B: Instruction Set Reference, N-Z, PAUSE instruction
4. a. LAHF and SAHF are invalid on early steppings of EM64T architecture; that's why they need CPUID.80000001H:ECX.LAHF-SAHF[bit 0]
5. a. sandpile.org -- IA-32 architecture -- opcode groups
6. a. Intel® 64 and IA-32 Architecture Software Developer's Manual Volume 3: System Programming Guide, Interrupt and Exception Handling
7. a. sandpile.org -- IA-32 architecture -- one byte opcodes
b. AMD64 Architecture Programmer's Manual Volume 3, Table One-Bytes Opcodes
8. a. sandpile.org -- IA-32 architecture -- one byte opcodes
b. AMD64 Architecture Programmer's Manual Volume 3, Table One-Bytes Opcodes
c. 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."
9. a. sandpile.org -- IA-32 architecture -- opcode groups
b. Christian Ludloff wrote: "While the latest Intel manuals still omit this de-facto standard, the recent x86-64 manuals from AMD document it."
c. AMD64 Architecture Programmer's Manual Volume 3, Table One-Byte and Two-Byte Opcode ModRM Extensions
10. a. AMD64 architecture does not enable 64-bit offset: AMD64 Architecture Programmer's Manual Volume 3: If the operand-size is 32 or 64 bits, the operand is a 16-bit selector followed by a 32-bit offset.
11. a. sandpile.org -- IA-32 architecture -- two byte opcodes
b. www.x86.org - The LOADALL Instruction
12. a. On AMD64 architecture, SYSCALL is valid also in legacy mode
13. a. Intel® 64 and IA-32 Architecture Software Developer's Manual Volume 2B: Instruction Set Reference, N-Z, Two-byte Opcode Map
b. AMD architecture maps 3DNow! PREFETCH instructions here
14. a. AMD64 Architecture Programmer's Manual Volume 3, System Instruction Reference: If CPUID.80000001H:ECX.4, CR8 can be read and written in legacy mode using a LOCK prefix instead of a REX prefix to specify the additional opcode bit.
15. a. Christian Ludloff wrote: "For the MOVs from/to CRx/DRx/TRx, mod=00b/01b/10b is aliased to 11b."
b. 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.
16. a. On AMD64 architecture, SYSENTER is valid only in legacy mode.
17. a. On AMD64 architecture, SYSEXIT is not valid in long mode.
18. a. AMD64 Architecture Programmers Manual Volume 3: General-Purpose and System Instructions: The reg field in the ModR/M byte is unused.
19. a. AMD64 architecture does not enable 64-bit operands: AMD64 Architecture Programmers Manual Volume 3: General-Purpose and System Instructions: Executing LFS, LGS, or LSS with a 64-bit operand size only loads a 32-bit general purpose register and the specified segment register
20. a. Intel® 64 and IA-32 Architecture Software Developer's Manual Volume 2B: Instruction Set Reference, N-Z, Two-byte Opcode Map
b. sandpile.org -- IA-32 architecture -- two byte opcodes
21. a. On AMD64 architecture, BSF and BSR instructions act differently if the content of the source operand is 0
22. a. CMPXCHG16B is invalid on early steppings of AMD64 architecture
23. a. Use of operand-size prefix in 64-bit mode may result in implementation-dependent behaviour; on AMD64 architecture, this prefix acts as expected

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

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