**~~8088 (8086) instruction set~~**

|  |  |  |
| --- | --- | --- |
| **Instructions** | **Description** | **Notes** |
| **ADC** | Add with Carry | destination := destination + source + carry\_flag |
| **ADD** | Add without Carry | (1) r/m += r/imm; (2) r += m/imm; |
| **AND** | Logical AND | (1) r/m &= r/imm; (2) r &= m/imm; |
| **CALL** | call Subroutine | **~~push~~** eip; |
| **CLI** | Global Interrupt Disable | I ← 0 |
| **CMP** | Compare |  |
| **DEC** | Decrement by 1 |  |
| **DIV** | Unsigned divide | ~~DX:AX = DX:AX / r/m;~~ resulting DX = remainder |
| **IN** | Input from port | (1) AL = port[imm]; (2) AL = port[DX];  (3) AX = port[DX]; |
| **INC** | Increment by 1 |  |
| **INT** | Call to interrupt |  |
| **IRET** | Return from interrupt |  |
| **JE** | Branch if Equal | if (Z = 1) then PC ← PC + k + 1 |
| **JGE** | Branch if Same or Higher | if (C = 0) then PC ← PC + k + 1 |
| **JL** | Branch if Lower | if (C = 1) then PC ← PC + k + 1 |
| **JMP** | Jump |  |
| **JNE** | Branch if Not Equal | if (Z = 0) then PC ← PC + k + 1 |
| **MOV** | copies data from one location to another | (1) r/m = r; (2) r = r/m; |
| **MUL** | Multiply Unsigned | (1) DX:AX = AX \* r/m; (2) AX = AL \* r/m; |
| **NOT** | Negate the operand, logical NOT | r/m ^= -1; |
| **OR** | Logical OR | (1) r/m |= r/imm; (2) r |= m/imm; |
| **OUT** | Output to port | (1) port[imm] = AL; (2) port[DX] = AL;  (3) port[DX] = AX; |
| **POP** | Pop Register from Stack | r/m = \*SP++; |
| **PUSH** | Push Register on Stack | \*--SP = r/m; |
| **RET** | Subroutine Return | It will be translated to a RETN or a RETF |
| **RETF** | Return from far procedure |  |
| **RETN** | Return from near procedure |  |
| **ROL** | Rotate left |  |
| **ROR** | Rotate right |  |
| **STI** | Global Interrupt Enable | I ← 1 |
| **SUB** | Subtract without Carry |  |
| **XOR** | Exclusive OR | (1) r/m ^= r/imm; (2) r ^= m/imm; |

(LDS: Load pointer using DS)