ADC - Add With Carry
Usage: ADC dest,src
Modifies flags: AF CF OF SF PF ZF

Sums two binary operands placing the result in the destination.
If CF is set, a 1 is added to the destination.

ADD - Arithmetic Addition
Usage: ADD dest,src
Modifies flags: AF CF OF PF SF ZF

Adds "src" to "dest" and replacing the original contents of "dest".
Both operands are binary.

AND - Logical And
Usage: AND dest,src
Modifies flags: CF OF PF SF ZF (AF undefined)

Performs a logical AND of the two operands replacing the destination with the result.

CALL - Procedure Call
Usage: CALL destination
Modifies flags: None

Pushes Instruction Pointer (and Code Segment for far calls) onto stack & loads Instruction Pointer with the address of proc-name. Code continues with execution at CS:IP.

CLC - Clear Carry
Usage: CLC
Modifies flags: CF

Clears the Carry Flag.

CMP - Compare
Usage: CMP dest,src
Modifies flags: AF CF OF PF SF ZF

Subtracts source from destination and updates the flags but does not save result. Flags can subsequently be checked for conditions.

DEC - Decrement
Usage: DEC dest
Modifies flags: AF OF PF SF ZF

Unsigned binary subtraction of one from the destination.

INC - Increment
Usage: INC dest
Modifies flags: AF OF PF SF ZF

Adds one to destination unsigned binary operand.

IN - Input Byte or Word From Port
Usage: IN accum,port
Modifies flags: None

A byte or word is read from "port" and placed in AL or AX. If the port number is in the range of 0-255 it can be specified as an immediate, otherwise the port number must be specified in DX.
INT - Interrupt
Usage: INT num
Modifies flags: TF IF

Initiates software interrupt by pushing the flags, clearing the TF & IF Flags, pushing CS followed by IP and loading CS:IP with the value found in the interrupt vector table. Execution then begins at the location in new CS:IP

IRET/IRETD - Interrupt Return
Usage: IRET
Modifies flags: AF CF DF IF PF SF TF ZF

Returns control to point of interruption by popping IP, CS, and the Flags from stack. Continues execution at old CS:IP location. CPU exception interrupts will return to the instruction that cause the exception because the CS:IP placed on the stack during the interrupt is the address of the offending instruction.

Jxx - Jump Instructions Table

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Meaning</th>
<th>Jump Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>JA</td>
<td>Jump if Above</td>
<td>CF=0 and ZF=0</td>
</tr>
<tr>
<td>JAE</td>
<td>Jump if Above or Equal</td>
<td>CF=0</td>
</tr>
<tr>
<td>JB</td>
<td>Jump if Below</td>
<td>CF=1</td>
</tr>
<tr>
<td>JBE</td>
<td>Jump if Below or Equal</td>
<td>CF=1 or ZF=1</td>
</tr>
<tr>
<td>JC</td>
<td>Jump if Carry</td>
<td>CF=1</td>
</tr>
<tr>
<td>JE</td>
<td>Jump if Equal</td>
<td>ZF=1</td>
</tr>
<tr>
<td>JG</td>
<td>Jump if Greater (signed)</td>
<td>ZF=0 and SF=OF</td>
</tr>
<tr>
<td>JGE</td>
<td>Jump if Greater or Equal (signed)</td>
<td>SF=OF</td>
</tr>
<tr>
<td>JL</td>
<td>Jump if Less (signed)</td>
<td>SF != OF</td>
</tr>
<tr>
<td>JLE</td>
<td>Jump if Less or Equal (signed)</td>
<td>ZF=1 or SF != OF</td>
</tr>
<tr>
<td>JMP</td>
<td>Unconditional Jump</td>
<td>unconditional</td>
</tr>
<tr>
<td>JNA</td>
<td>Jump if Not Above</td>
<td>CF=1 or ZF=1</td>
</tr>
<tr>
<td>JNAE</td>
<td>Jump if Not Above or Equal</td>
<td>CF=1</td>
</tr>
<tr>
<td>JNB</td>
<td>Jump if Not Below</td>
<td>CF=0</td>
</tr>
<tr>
<td>JNBE</td>
<td>Jump if Not Below or Equal</td>
<td>CF=0 and ZF=0</td>
</tr>
<tr>
<td>JNC</td>
<td>Jump if Not Carry</td>
<td>CF=0</td>
</tr>
<tr>
<td>JNE</td>
<td>Jump if Not Equal</td>
<td>ZF=0</td>
</tr>
<tr>
<td>JNG</td>
<td>Jump if Not Greater (signed)</td>
<td>ZF=1 or SF != OF</td>
</tr>
<tr>
<td>JNGE</td>
<td>Jump if Not Greater or Equal (signed)</td>
<td>SF != OF</td>
</tr>
<tr>
<td>JNL</td>
<td>Jump if Not Less (signed)</td>
<td>SF=OF</td>
</tr>
<tr>
<td>JNLE</td>
<td>Jump if Not Less or Equal (signed)</td>
<td>ZF=0 and SF=OF</td>
</tr>
<tr>
<td>JNS</td>
<td>Jump if Not Signed (signed)</td>
<td>SF=0</td>
</tr>
<tr>
<td>JNZ</td>
<td>Jump if Not Zero</td>
<td>ZF=0</td>
</tr>
<tr>
<td>JS</td>
<td>Jump if Signed (signed)</td>
<td>SF=1</td>
</tr>
<tr>
<td>JZ</td>
<td>Jump if Zero</td>
<td>ZF=1</td>
</tr>
</tbody>
</table>

MOV - Move Byte or Word
Usage: MOV dest,src
Modifies flags: None

Copies byte or word from the source operand to the destination operand. If the destination is SS interrupts are disabled except on early buggy 808x CPUs. Some CPUs disable interrupts if the destination is any of the segment registers

NEG - Two's Complement Negation
Usage: NEG dest
Modifies flags: AF CF OF PF SF ZF

Subtracts "dest" from 0 and saves the 2s complement of "dest" back into "dest".

NOT - One's Compliment Negation (Logical NOT)
Usage: NOT dest
NOT (continued...)
  Modifies flags: None

Inverts the bits of the "dest" operand forming the 1s complement.

OR - Inclusive Logical OR
Usage: OR dest,src
Modifies flags: CF OF PF SF ZF (AF undefined)

Logical inclusive OR of the two operands returning the result in the destination. Any bit set in either operand will be set in the destination.

OUT - Output Data to Port
Usage: OUT port,accum
Modifies flags: None

Transfers byte in AL or word in AX to the specified hardware port address.

POP - Pop Word off Stack
Usage: POP dest
Modifies flags: None

Transfers word at the current stack top (SS:SP) to the destination then increments SP by two to point to the new stack top. CS is not a valid destination.

PUSH - Push Word onto Stack
Usage: PUSH src
Modifies flags: None

Decrements SP by the size of the operand (two or four, byte values are sign extended) and transfers one word from source to the stack top (SS:SP).

RET - Return From Procedure
Usage: RET nBytes
Modifies flags: None

Transfers control from a procedure back to the instruction address saved on the stack. "n bytes" is an optional number of bytes to release. Far returns (RETF) pop the IP followed by the CS; near returns (RETN) pop only the IP register.

SAL/SHL - Shift Arithmetic Left / Shift Logical Left
Usage: SAL dest,count            SHL dest,count
Modifies flags: CF OF PF SF ZF (AF undefined)

Shifts the destination left by "count" bits with zeroes shifted in on right. The Carry Flag contains the last bit shifted out.

SAR - Shift Arithmetic Right
Usage: SAR dest,count
Modifies flags: CF OF PF SF ZF (AF undefined)

Shifts the destination right by "count" bits with the current sign bit replicated in the leftmost bit. The Carry Flag contains the last bit shifted out.

XOR - Exclusive OR
Usage: XOR dest,src
Modifies flags: CF OF PF SF ZF (AF undefined)

Performs a bitwise exclusive OR of the operands and returns result in "dest".