8086 INSTRUCTION REFERENCE

`ADD dest, src` — **dest = dest + src**

Flags: ZF = 1 iff result = 0 SF = 1 iff dest < 0
CF = 1 iff result generates carry out of most signif. bit
OF = 1 iff result creates signed overflow

`CMP dest, src` — **calc dest – src**

NB: Does **not** modify dest, modifies FLAGS only!
Flags: ZF = 1 iff result = 0 SF = 1 iff result < 0
CF = 1 iff result requires borrow into most signif. bit
OF = 1 iff result creates signed overflow

`DIV src` — **unsigned integer divide**

if 8-bit source: AL = AX / src (division)
AH = AX mod src (remainder)
if 16-bit source: AX = DX:AX / src (division)
DX = AX mod src (remainder)
Addressing modes: src may be reg/mem8 or reg/mem16
NB: src may **not** be immediate!
Flags: the flags are undefined following DIV

`DEC dest` — **`dest = dest - 1`**
`INC dest` — **`dest = dest + 1`**

Addressing modes: dest may be reg/mem8 or reg/mem16
Flags: ZF = 1 iff result = 0 SF = 1 iff result < 0
CF = 1 iff result generates carry out of or borrow in to most signif. bit
OF = 1 iff result creates signed overflow

`Jcc target` — **conditional jump**

IP:= IP + offset_to_target (if condition **true**)
Addressing modes: target uses short relative addressing
Flags: no effect
The possible forms are summarized below and to the right:

JA	Jump Above CF = 0 and ZF = 0 (**JNBE**)
JAE	Jump Above or Equal CF = 0 (**JNB**)
JB	Jump Below CF = 1 (**JNAE**)
JBE	Jump Below or Equal CF = 1 or ZF = 1 (**JNA**)
JC	Jump Carry CF = 1
JE	Jump Equal ZF = 1
JG	Jump Greater Than SF = OF and ZF = 0 (**JNLE**)
JGE	Jump Greater than or Equal SF = OF (**JNL**)
JL	Jump Less Than SF != OF (**JNGE**)
JLE	Jump Less than or Equal SF != OF or ZF = 1 (**JNG**)
JNC	Jump No Carry CF = 0
JNE	Jump Not Equal ZF = 0
JNO	Jump No Overflow OF = 0
JNS	Jump No Sign SF = 0
JNZ	Jump Not Zero ZF = 0
JO	Jump Overflow OF = 1
JS	Jump Sign SF = 1
JZ	Jump Zero ZF = 1

`JMP target` — **unconditional jump**

IP = IP + offset_to_target
Addressing modes: target uses relative addressing
Flags: no effect

`MOV dest, src` — **dest = src**

Flags: no effect

`MUL src` — **unsigned integer multiplication**

if 8-bit source: AX := AL * src
if 16-bit source: DX:AX := AX * src
Addressing modes: src may be reg/mem8 or reg/mem16
NB: src may not be immediate!
Flags: CF = 0 iff AH = 0 (for 8-bit src)
or DX = 0 (for 16-bit src)
OF = CF
ZF and SF are undefined following MUL

`NEG dest` — **dest = 0 – dest**

Addressing modes: dest may be 8 or 16 reg/mem
Flags: CF = 0 iff result = 0 SF and ZF reflect result
OF = 1 iff value could not be negated correctly

`OUT [DX], AL` — **port[DX] = AL**

Addressing modes: only those shown (indirect DX)
Flags: no effect

`POP dest` — **pop from stack**

dest := mem[SP]
SP := SP + 2
Addressing modes: dest may be 16 bit reg/mem **only**
Flags: no effect

`PUSH src` — **push onto stack**

SP := SP - 2
mem[SP] := src
Addressing modes: src may be 16 bit reg/mem **only**
Flags: no effect

`SHL dest,CL` — **`dest = dest shifted_left_by CL`**
`SHR dest,CL` — **`dest = dest shifted_right_by CL`**

Addressing modes: dest may be reg/mem8 or reg/mem16
Flags:CF = value of bit shifted out of reg
OF = 1 iff result and original value have different signs
ZF and SF reflect the result

`SUB dest, src` — **dest := dest - src**

Flags: ZF = 1 iff result = 0 SF = 1 iff result < 0
CF = 1 iff result generates borrow into most-signif bit
OF = 1 iff result creates signed overflow