Complete 8086 instruction set

Quick reference:

AAA
AAD
AAM
AAS
ADC
ADD
AND
CALL
CBW
CLC
CLD
CMPSB
CMPSW
CWD
DAA
DAS
DEC
DIV
HLT
IDIV
IMUL
IN
JAE
JB
JBE
JC
JCXZ
JE
JG
JGE
JL
JLE
JMP
JNBE
JNC
JNE
JNG
JNGE
JNL
JNLE
JNO
JNP
JNS
JNZ
JPO
JS
JZ
LAHF
LDS
LEA
LES
LODSB
LODSW
LOOP
LOOPE
MOV
MOVSB
MOVSW
MUL
NEG
NOP
NOT
OR
OUT
POP
POPA
RCR
REP
REPE
REPNE
REPNZ
REPZ
RET
RETF
ROL
ROR
SAHF
SCASB
SCASW
SHL
SHR
STC
STD
STI
STOSB
STOSW
SUB
TEST

CLI
CMC
CMP
INC
INT
INTO
IRET
JA
JNA
JNAE
JNB
JO
JP
JPE
LOOPNE
LOOPNZ
LOOPZ
POPF
PUSH
PUSHA
PUSHF
RCL
SAL
SAR
SBB
XCHG
XLATB
XOR

Operand types:

REG: AX, BX, CX, DX, AH, AL, BL, BH, CH, CL, DH, DL, DI, SI, BP, SP.

SREG: DS, ES, SS, and only as second operand: CS.

memory: [BX], [BX+SI+7], variable, etc...(see **Memory Access**).

immediate: 5, -24, 3Fh, 10001101b, etc...

Notes:

- When two operands are required for an instruction they are separated by comma. For example:

  ```
  REG, memory
  ```

- When there are two operands, both operands must have the same size (except shift and rotate instructions). For example:

  ```
  AL, DL
  DX, AX
  m1 DB ?
  AL, m1
  m2 DW ?
  AX, m2
  ```

- Some instructions allow several operand combinations. For example:

  ```
  memory, immediate
  REG, immediate

  memory, REG
  REG, SREG
  ```

- Some examples contain macros, so it is advisable to use **Shift + F8** hot key to *Step Over* (to make macro code execute at maximum speed set **step delay** to zero), otherwise emulator will step through each instruction of a macro. Here is an example that uses PRINTN macro:
-
-

-
  ```
  include 'emu8086.inc'
  ```

- ORG 100h
- MOV AL, 1
- MOV BL, 2
- PRINTN 'Hello World!'   ; macro.
- MOV CL, 3
- PRINTN 'Welcome!'   ; macro.

RET

These marks are used to show the state of the flags:

1 - instruction sets this flag to **1**.
0 - instruction sets this flag to **0**.
r - flag value depends on result of the instruction.
? - flag value is undefined (maybe **1** or **0**).

Some instructions generate exactly the same machine code, so disassembler may have a problem decoding to your original code. This is especially important for Conditional Jump instructions (see "Program Flow Control" in Tutorials for more information).

Instructions in alphabetical order:

Instruction	Operands	Description
AAA	No operands	ASCII Adjust after Addition. Corrects result in AH and AL after addition when working with BCD values. It works according to the following Algorithm: if low nibble of AL > 9 or AF = 1 then: • AL = AL + 6 • AH = AH + 1 • AF = 1 • CF = 1 else • AF = 0 • CF = 0

		in both cases: clear the high nibble of AL. Example: MOV AX, 15 ; AH = 00, AL = 0Fh AAA ; AH = 01, AL = 05 RET C Z S O P A r ? ? ? ? r
AAD	No operands	ASCII Adjust before Division. Prepares two BCD values for division. Algorithm: - AL = (AH * 10) + AL - AH = 0 Example: MOV AX, 0105h ; AH = 01, AL = 05 AAD ; AH = 00, AL = 0Fh (15) RET C Z S O P A ? r r ? r ?
AAM	No operands	ASCII Adjust after Multiplication. Corrects the result of multiplication of two BCD values. Algorithm: - AH = AL / 10 - AL = remainder Example: MOV AL, 15 ; AL = 0Fh AAM ; AH = 01, AL = 05 RET C Z S O P A ? r r ? r ?
AAS	No operands	ASCII Adjust after Subtraction. Corrects result in AH and AL after subtraction when working with BCD values. Algorithm: if low nibble of AL > 9 or AF = 1 then:

<table>
<tr><td></td><td></td><td>• AL = AL - 6
• AH = AH - 1
• AF = 1
  • CF = 1
else

• AF = 0
• CF = 0

in both cases:
clear the high nibble of AL.

Example:
MOV AX, 02FFh ; AH = 02, AL = 0FFh
AAS ; AH = 01, AL = 09
RET
<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td>r</td><td>?</td><td>?</td><td>?</td><td>?</td><td>r</td></tr></table></td></tr>
<tr><td>ADC</td><td>REG, memory
memory, REG
REG, REG
memory, immediate
REG, immediate</td><td>Add with Carry.

Algorithm:

operand1 = operand1 + operand2 + CF

Example:
STC ; set CF = 1
MOV AL, 5 ; AL = 5
ADC AL, 1 ; AL = 7
RET
<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td></tr></table></td></tr>
<tr><td>ADD</td><td>REG, memory
memory, REG
REG, REG
memory, immediate
REG, immediate</td><td>Add.

Algorithm:

operand1 = operand1 + operand2

Example:
MOV AL, 5 ; AL = 5
ADD AL, -3 ; AL = 2
RET
<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td></tr></table></td></tr>
<tr><td>AND</td><td>REG, memory</td><td>Logical AND between all bits of two operands. Result is stored in</td></tr>
</table>

	memory, REG REG, REG memory, immediate REG, immediate	operand1. These rules apply: 1 AND 1 = 1 1 AND 0 = 0 0 AND 1 = 0 0 AND 0 = 0 Example: MOV AL, 'a' ; AL = 01100001b AND AL, 11011111b ; AL = 01000001b ('A') RET C Z S O P 0 r r 0 r
CALL	procedure name label 4-byte address	Transfers control to procedure, return address is (IP) is pushed to stack. *4-byte address* may be entered in this form: 1234h:5678h, first value is a segment second value is an offset (this is a far call, so CS is also pushed to stack). Example: ORG 100h ; directive to make simple .com file. CALL p1 ADD AX, 1 RET ; return to OS. p1 PROC ; procedure declaration. MOV AX, 1234h RET ; return to caller. p1 ENDP C Z S O P A unchanged
CBW	No operands	Convert byte into word. Algorithm: if high bit of AL = 1 then: • AH = 255 (0FFh) else • AH = 0

		Example: MOV AX, 0 ; AH = 0, AL = 0 MOV AL, -5 ; AX = 000FBh (251) CBW ; AX = 0FFFBh (-5) RET C Z S O P A unchanged
CLC	No operands	Clear Carry flag. Algorithm: CF = 0 C 0
CLD	No operands	Clear Direction flag. SI and DI will be incremented by chain instructions: CMPSB, CMPSW, LODSB, LODSW, MOVSB, MOVSW, STOSB, STOSW. Algorithm: DF = 0 D 0
CLI	No operands	Clear Interrupt enable flag. This disables hardware interrupts. Algorithm: IF = 0 I 0
CMC	No operands	Complement Carry flag. Inverts value of CF. Algorithm: if CF = 1 then CF = 0 if CF = 0 then CF = 1 C r

		⬆
CMP	REG, memory memory, REG REG, REG memory, immediate REG, immediate	⬆ Compare. Algorithm: operand1 - operand2 result is not stored anywhere, flags are set (OF, SF, ZF, AF, PF, CF) according to result. Example: MOV AL, 5 MOV BL, 5 CMP AL, BL ; AL = 5, ZF = 1 (so equal!) RET C Z S O P A r r r r r r
CMPSB	No operands	Compare bytes: ES:[DI] from DS:[SI]. Algorithm: • DS:[SI] - ES:[DI] • set flags according to result: OF, SF, ZF, AF, PF, CF • if DF = 0 then ○ SI = SI + 1 ○ DI = DI + 1 else ⬆ ○ SI = SI - 1 ○ DI = DI - 1 Example: see **cmpsb.asm** in c:\emu8086\examples\. C Z S O P A r r r r r r
CMPSW	No operands	Compare words: ES:[DI] from DS:[SI]. Algorithm: • DS:[SI] - ES:[DI] • set flags according to result: OF, SF, ZF, AF, PF, CF • if DF = 0 then ○ SI = SI + 2 ○ DI = DI + 2

		⬆ else ○ SI = SI - 2 ○ DI = DI - 2 Example: see **cmpsw.asm** in c:\emu8086\examples\. C Z S O P A r r r r r r
CWD	No operands	Convert Word to Double word. Algorithm: if high bit of AX = 1 then: • DX = 65535 (0FFFFh) else • DX = 0 ⬆ Example: MOV DX, 0 ; DX = 0 MOV AX, 0 ; AX = 0 MOV AX, -5 ; DX AX = 00000h:0FFFBh CWD ; DX AX = 0FFFFh:0FFFBh RET C Z S O P A unchanged

DAA	No operands	Decimal adjust After Addition.
		Corrects the result of addition of two packed BCD values.
		
		Algorithm:
		
		if low nibble of AL > 9 or AF = 1 then:
		• AL = AL + 6
		• AF = 1
		
		if AL > 9Fh or CF = 1 then:
		• AL = AL + 60h
		• CF = 1
		
		Example:
		MOV AL, 0Fh ; AL = 0Fh (15)

<table>
<tr><td></td><td></td><td>DAA ; AL = 15h
RET
<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td></tr></table></td></tr>
<tr><td>DAS</td><td>No operands</td><td>Decimal adjust After Subtraction.
Corrects the result of subtraction of two packed BCD values.

Algorithm:

if low nibble of AL &gt; 9 or AF = 1 then:
<ul><li>AL = AL - 6</li><li>AF = 1</li></ul>if AL &gt; 9Fh or CF = 1 then:
<ul><li>AL = AL - 60h</li><li>CF = 1</li></ul>
Example:
MOV AL, 0FFh ; AL = 0FFh (-1)
DAS ; AL = 99h, CF = 1
RET
<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td></tr></table></td></tr>
<tr><td>DEC</td><td>REG
memory</td><td>Decrement.

Algorithm:

operand = operand - 1

Example:
MOV AL, 255 ; AL = 0FFh (255 or -1)
DEC AL ; AL = 0FEh (254 or -2)
RET
<table><tr><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td></tr></table>CF - unchanged!</td></tr>
<tr><td>DIV</td><td>REG
memory</td><td>Unsigned divide.

Algorithm:

when operand is a <b>byte</b>:
AL = AX / operand
AH = remainder (modulus)
when operand is a <b>word</b>:
AX = (DX AX) / operand</td></tr>
</table>

		DX = remainder (modulus) Example: MOV AX, 203 ; AX = 00CBh MOV BL, 4 DIV BL ; AL = 50 (32h), AH = 3 RET C Z S O P A ? ? ? ? ? ?
HLT	No operands	Halt the System. Example: MOV AX, 5 HLT C Z S O P A unchanged
IDIV	REG memory	Signed divide. Algorithm: when operand is a **byte**: AL = AX / operand AH = remainder (modulus) when operand is a **word**: AX = (DX AX) / operand DX = remainder (modulus) Example: MOV AX, -203 ; AX = 0FF35h MOV BL, 4 IDIV BL ; AL = -50 (0CEh), AH = -3 (0FDh) RET C Z S O P A ? ? ? ? ? ?
IMUL	REG memory	Signed multiply. Algorithm: when operand is a **byte**: AX = AL * operand. when operand is a **word**: (DX AX) = AX * operand. Example: MOV AL, -2 MOV BL, -4 IMUL BL ; AX = 8 RET C Z S O P A r ? ? r ? ?

<table>
<tr><td></td><td></td><td>CF=OF=0 when result fits into operand of IMUL.</td></tr>
<tr><td>IN</td><td>AL, im.byte
AL, DX
AX, im.byte
AX, DX</td><td>Input from port into <b>AL</b> or <b>AX</b>.
Second operand is a port number. If required to access port number over 255 - <b>DX</b> register should be used.
Example:
IN AX, 4 ; get status of traffic lights.
IN AL, 7 ; get status of stepper-motor.
<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td colspan="6">unchanged</td></tr></table></td></tr>
<tr><td>INC</td><td>REG
memory</td><td>Increment.

Algorithm:

operand = operand + 1

Example:
MOV AL, 4
INC AL ; AL = 5
RET
<table><tr><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td></tr></table>CF - unchanged!</td></tr>
<tr><td>INT</td><td>immediate byte</td><td>Interrupt numbered by immediate byte (0..255).

Algorithm:

Push to stack:<ul><li>flags register</li><li>CS</li><li>IP</li></ul><ul><li>IF = 0</li><li>Transfer control to interrupt procedure</li></ul>
Example:
MOV AH, 0Eh ; teletype.
MOV AL, 'A'
INT 10h ; BIOS interrupt.
RET
<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td><td>I</td></tr><tr><td colspan="6">unchanged</td><td>0</td></tr></table></td></tr>
<tr><td>INTO</td><td>No operands</td><td>Interrupt 4 if Overflow flag is 1.

Algorithm:</td></tr>
</table>

		if OF = 1 then INT 4 Example: ; -5 - 127 = -132 (not in -128..127) ; the result of SUB is wrong (124), ; so OF = 1 is set: MOV AL, -5 SUB AL, 127 ; AL = 7Ch (124) INTO ; process error. RET
IRET	No operands	Interrupt Return. Algorithm: Pop from stack: ○ IP ○ CS ○ flags register C Z S O P A popped

JA	label	Short Jump if first operand is Above second operand (as set by CMP instruction). Unsigned.
		
		Algorithm:
		
		if (CF = 0) and (ZF = 0) then jump
		Example:
		   include 'emu8086.inc'
		
		   ORG 100h
		   MOV AL, 250
		   CMP AL, 5
		   JA label1
		   PRINT 'AL is not above 5'
		   JMP exit
		label1:
		   PRINT 'AL is above 5'
		exit:
		   RET
		C Z S O P A
		unchanged

JAE	label	Short Jump if first operand is Above or Equal to second operand (as set by CMP instruction). Unsigned. Algorithm: if CF = 0 then jump Example:

		⬆ include 'emu8086.inc' ORG 100h MOV AL, 5 CMP AL, 5 JAE label1 PRINT 'AL is not above or equal to 5' JMP exit label1: PRINT 'AL is above or equal to 5' exit: RET C Z S O P A unchanged
JB	label	⬆ Short Jump if first operand is Below second operand (as set by CMP instruction). Unsigned. Algorithm: if CF = 1 then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 1 CMP AL, 5 JB label1 PRINT 'AL is not below 5' JMP exit label1: PRINT 'AL is below 5' exit: RET C Z S O P A unchanged
JBE	label	Short Jump if first operand is Below or Equal to second operand (as set by CMP instruction). Unsigned. Algorithm: if CF = 1 or ZF = 1 then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 5 CMP AL, 5 JBE label1 PRINT 'AL is not below or equal to 5' JMP exit label1: PRINT 'AL is below or equal to 5' exit:

		RET C Z S O P A unchanged
JC	label	Short Jump if Carry flag is set to 1. Algorithm: if CF = 1 then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 255 ADD AL, 1 JC label1 PRINT 'no carry.' JMP exit label1: PRINT 'has carry.' exit: RET C Z S O P A unchanged
JCXZ	label	Short Jump if CX register is 0. Algorithm: if CX = 0 then jump Example: include 'emu8086.inc' ORG 100h MOV CX, 0 JCXZ label1 PRINT 'CX is not zero.' JMP exit label1: PRINT 'CX is zero.' exit: RET C Z S O P A unchanged
JE	label	Short Jump if first operand is Equal to second operand (as set by CMP instruction). Signed/Unsigned. Algorithm: if ZF = 1 then jump Example:

		include 'emu8086.inc' ORG 100h MOV AL, 5 CMP AL, 5 JE label1 PRINT 'AL is not equal to 5.' JMP exit label1: PRINT 'AL is equal to 5.' exit: RET C Z S O P A unchanged
JG	label	Short Jump if first operand is Greater then second operand (as set by CMP instruction). Signed. Algorithm: if (ZF = 0) and (SF = OF) then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 5 CMP AL, -5 JG label1 PRINT 'AL is not greater -5.' JMP exit label1: PRINT 'AL is greater -5.' exit: RET C Z S O P A unchanged
JGE	label	Short Jump if first operand is Greater or Equal to second operand (as set by CMP instruction). Signed. Algorithm: if SF = OF then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 2 CMP AL, -5 JGE label1 PRINT 'AL < -5' JMP exit label1: PRINT 'AL >= -5' exit:

		RET C Z S O P A unchanged
JL	label	Short Jump if first operand is Less then second operand (as set by CMP instruction). Signed. Algorithm: if SF <> OF then jump Example: include 'emu8086.inc' ORG 100h MOV AL, -2 CMP AL, 5 JL label1 PRINT 'AL >= 5.' JMP exit label1: PRINT 'AL < 5.' exit: RET C Z S O P A unchanged
JLE	label	Short Jump if first operand is Less or Equal to second operand (as set by CMP instruction). Signed. Algorithm: if SF <> OF or ZF = 1 then jump Example: include 'emu8086.inc' ORG 100h MOV AL, -2 CMP AL, 5 JLE label1 PRINT 'AL > 5.' JMP exit label1: PRINT 'AL <= 5.' exit: RET C Z S O P A unchanged
JMP	label 4-byte address	Unconditional Jump. Transfers control to another part of the program. *4-byte address* may be entered in this form: 1234h:5678h, first value is a segment second value is an offset.

		Algorithm: always jump Example: include 'emu8086.inc' ORG 100h MOV AL, 5 JMP label1 ; jump over 2 lines! PRINT 'Not Jumped!' MOV AL, 0 label1: PRINT 'Got Here!' RET C Z S O P A unchanged
JNA	label	Short Jump if first operand is Not Above second operand (as set by CMP instruction). Unsigned. Algorithm: if CF = 1 or ZF = 1 then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 2 CMP AL, 5 JNA label1 PRINT 'AL is above 5.' JMP exit label1: PRINT 'AL is not above 5.' exit: RET C Z S O P A unchanged
JNAE	label	Short Jump if first operand is Not Above and Not Equal to second operand (as set by CMP instruction). Unsigned. Algorithm: if CF = 1 then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 2 CMP AL, 5 JNAE label1 PRINT 'AL >= 5.' JMP exit

		label1: PRINT 'AL < 5.' exit: RET C Z S O P A unchanged
JNB	label	Short Jump if first operand is Not Below second operand (as set by CMP instruction). Unsigned. Algorithm: if CF = 0 then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 7 CMP AL, 5 JNB label1 PRINT 'AL < 5.' JMP exit label1: PRINT 'AL >= 5.' exit: RET C Z S O P A unchanged
JNBE	label	Short Jump if first operand is Not Below and Not Equal to second operand (as set by CMP instruction). Unsigned. Algorithm: if (CF = 0) and (ZF = 0) then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 7 CMP AL, 5 JNBE label1 PRINT 'AL <= 5.' JMP exit label1: PRINT 'AL > 5.' exit: RET C Z S O P A unchanged
JNC	label	Short Jump if Carry flag is set to 0.

		Algorithm: if CF = 0 then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 2 ADD AL, 3 JNC label1 PRINT 'has carry.' JMP exit label1: PRINT 'no carry.' exit: RET C Z S O P A unchanged
JNE	label	Short Jump if first operand is Not Equal to second operand (as set by CMP instruction). Signed/Unsigned. Algorithm: if ZF = 0 then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 2 CMP AL, 3 JNE label1 PRINT 'AL = 3.' JMP exit label1: PRINT 'Al <> 3.' exit: RET C Z S O P A unchanged
JNG	label	Short Jump if first operand is Not Greater then second operand (as set by CMP instruction). Signed. Algorithm: if (ZF = 1) and (SF <> OF) then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 2 CMP AL, 3 JNG label1

		PRINT 'AL > 3.' JMP exit label1: PRINT 'Al <= 3.' exit: RET C Z S O P A unchanged
JNGE	label	Short Jump if first operand is Not Greater and Not Equal to second operand (as set by CMP instruction). Signed. Algorithm: if SF <> OF then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 2 CMP AL, 3 JNGE label1 PRINT 'AL >= 3.' JMP exit label1: PRINT 'Al < 3.' exit: RET C Z S O P A unchanged
JNL	label	Short Jump if first operand is Not Less then second operand (as set by CMP instruction). Signed. Algorithm: if SF = OF then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 2 CMP AL, -3 JNL label1 PRINT 'AL < -3.' JMP exit label1: PRINT 'Al >= -3.' exit: RET C Z S O P A unchanged

<table>
<tr>
<td>JNLE</td>
<td>label</td>
<td>
Short Jump if first operand is Not Less and Not Equal to second operand (as set by CMP instruction). Signed.

Algorithm:

```
        if (SF = OF) and (ZF = 0) then jump
```

Example:

```
   include 'emu8086.inc'

   ORG 100h
   MOV AL, 2
   CMP AL, -3
   JNLE label1
   PRINT 'AL <= -3.'
   JMP exit
label1:
   PRINT 'Al > -3.'
exit:
   RET
```

<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td colspan="6">unchanged</td></tr></table>
</td>
</tr>
<tr>
<td>JNO</td>
<td>label</td>
<td>
Short Jump if Not Overflow.

Algorithm:

```
        if OF = 0 then jump
```

Example:

```
    ; -5 - 2 = -7 (inside -128..127)
    ; the result of SUB is correct,
    ; so OF = 0:

include 'emu8086.inc'

ORG 100h
  MOV AL, -5
  SUB AL, 2   ; AL = 0F9h (-7)
JNO  label1
  PRINT 'overflow!'
JMP exit
label1:
  PRINT 'no overflow.'
exit:
  RET
```

<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td colspan="6">unchanged</td></tr></table>
</td>
</tr>
<tr>
<td>JNP</td>
<td>label</td>
<td>
Short Jump if No Parity (odd). Only 8 low bits of result are checked. Set by CMP, SUB, ADD, TEST, AND, OR, XOR instructions.

Algorithm:

```
        if PF = 0 then jump
```

Example:
</td>
</tr>
</table>

		include 'emu8086.inc' ORG 100h MOV AL, 00000111b ; AL = 7 OR AL, 0 ; just set flags. JNP label1 PRINT 'parity even.' JMP exit label1: PRINT 'parity odd.' exit: RET C Z S O P A unchanged
JNS	label	Short Jump if Not Signed (if positive). Set by CMP, SUB, ADD, TEST, AND, OR, XOR instructions. Algorithm: if SF = 0 then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 00000111b ; AL = 7 OR AL, 0 ; just set flags. JNS label1 PRINT 'signed.' JMP exit label1: PRINT 'not signed.' exit: RET C Z S O P A unchanged
JNZ	label	Short Jump if Not Zero (not equal). Set by CMP, SUB, ADD, TEST, AND, OR, XOR instructions. Algorithm: if ZF = 0 then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 00000111b ; AL = 7 OR AL, 0 ; just set flags. JNZ label1 PRINT 'zero.' JMP exit label1: PRINT 'not zero.' exit:

		RET C Z S O P A unchanged
JO	label	Short Jump if Overflow. Algorithm: if OF = 1 then jump Example: ; -5 - 127 = -132 (not in -128..127) ; the result of SUB is wrong (124), ; so OF = 1 is set: include 'emu8086.inc' org 100h MOV AL, -5 SUB AL, 127 ; AL = 7Ch (124) JO label1 PRINT 'no overflow.' JMP exit label1: PRINT 'overflow!' exit: RET C Z S O P A unchanged
JP	label	Short Jump if Parity (even). Only 8 low bits of result are checked. Set by CMP, SUB, ADD, TEST, AND, OR, XOR instructions. Algorithm: if PF = 1 then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 00000101b ; AL = 5 OR AL, 0 ; just set flags. JP label1 PRINT 'parity odd.' JMP exit label1: PRINT 'parity even.' exit: RET C Z S O P A unchanged

JPE	label	Short Jump if Parity Even. Only 8 low bits of result are checked. Set by CMP, SUB, ADD, TEST, AND, OR, XOR instructions. Algorithm: if PF = 1 then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 00000101b ; AL = 5 OR AL, 0 ; just set flags. JPE label1 PRINT 'parity odd.' JMP exit label1: PRINT 'parity even.' exit: RET C Z S O P A unchanged
JPO	label	Short Jump if Parity Odd. Only 8 low bits of result are checked. Set by CMP, SUB, ADD, TEST, AND, OR, XOR instructions. Algorithm: if PF = 0 then jump Example: include 'emu8086.inc' ORG 100h MOV AL, 00000111b ; AL = 7 OR AL, 0 ; just set flags. JPO label1 PRINT 'parity even.' JMP exit label1: PRINT 'parity odd.' exit: RET C Z S O P A unchanged
JS	label	Short Jump if Signed (if negative). Set by CMP, SUB, ADD, TEST, AND, OR, XOR instructions. Algorithm: if SF = 1 then jump Example: include 'emu8086.inc' ORG 100h