Computer Structure and Language

The 8086/8088 Assembly Language

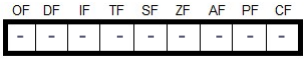
Hamid Sarbazi-Azad
Department of Computer Engineering
Sharif University of Technology (SUT)
Tehran, Iran

8086/88 has 7 types of instructions:

1. Data Transfer Instructions
2. Arithmetic Instructions
3. Bit Manipulation Instructions
4. String Instructions
5. Program Execution Transfer Instructions
6. Processor Control Instructions
7. Interrupt Instructions

The 8086/88's Control Transfer Instructions:

Unconditional Jump Instructions: JMP



- **Jump Direct within Segment**

11101001	IP_Low	IP_High

IP ← IP_High : IP_low;

Example 1: jmp in_segment_lable ≡ IP ← 0500h;
@in_segment_lable = 0500h

Machine code: 11101001 00000000 00000101 ≡ E90005h

- **Jump Direct within Segment Short**

11101011	IP-Inc8

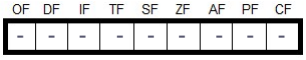
IP ← (IP) + IP_Inc8;

Example 1: jmp in_segment_lable ≡ IP ← 0500h;
@in_segment_lable = 0500h, (IP) = 04EFh

Machine code: 11101011 00010001 ≡ EB11h

The 8086/88's Control Transfer Instructions:

Unconditional Jump Instructions: JMP



- **Jump Indirect within Segment**

11111111	Md	100	R/M	Disp. Low-byte	Disp. High-byte (for 16-bit displacement)

IP ← (M_{EA});

Example: jmp near ptr [bx] ≡ IP ← ($M_{(bx)}$);

Machine code: 11111111 00 100 111 ≡ FF27h
Md R/M

- **Jump Direct Intersegment**

11101001	IP_Low	IP_High	CS_Low	CS_High

IP ← IP_High : IP_Low; CS ← CS_High : CS_Low;

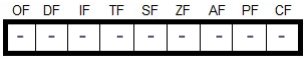
Example: jmp far ptr out_segment_lable ≡ IP ← 0500h; CS←?;
@out_segment_lable = 0500h in other segment

Machine code:

11101010 00000000 00000101 ???????? ???????? ≡ EA0005????h

The 8086/88's Control Transfer Instructions:

Unconditional Jump Instructions: JMP



- **Jump Indirect Intersegment**

IP ← (M_{EA}); CS ← (M_{EA+2});

Example:

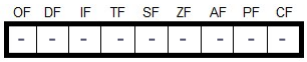
jmp far ptr [bx] ≡ IP ← ($M_{(bx)}$); CS ← ($M_{(bx)+2}$);

Machine code:

11111111 (Md: 00, 101, R/M: 111) ≡ FF2Fh

The 8086/88's Control Transfer Instructions:

Conditional Jump (all within segment short):



- **Jump on Equal/Zero: JE/JZ**

01110100 **IP-Inc8** if ZF=1 then IP ← (IP) + IP_Inc8;

Example 1: jz zero_lable ≡ IP ← 0500h;
@zero_lable = 0500h, (IP) = 04EFh, ZF=1

Machine code: 01110100 00010001 ≡ 7411h

- **Jump on Less/Not Greater or Equal: JL/JNGE**

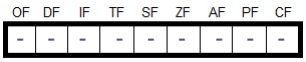
01111100 **IP-Inc8** if SF xor OF=1 then IP ← (IP) + IP_Inc8;

- **Jump on Less or Equal/Not Greater: JLE/JNG**

01111110 **IP-Inc8** if (SF xor OF) or ZF =1 then IP←(IP)+IP_Inc8;

The 8086/88's Control Transfer Instructions:

Conditional Jump (all within segment short):



- **Jump on Below/Not Above or Equal: JB/JNAE**

  01110010 **IP-Inc8** if CF=1 then IP ← (IP) + IP_Inc8;

- **Jump on Below or Equal/Not Above: JBE/JNA**

  01110110 **IP-Inc8** if CF or ZF=1 then IP ← (IP) + IP_Inc8;

- **Jump on Parity/Parity Even: JP/JPE**

  01111110 **IP-Inc8** if PF=1 then IP←(IP)+IP_Inc8;

- **Jump on Overflow: JO**

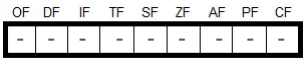
  01110000 **IP-Inc8** if OF=1 then IP←(IP)+IP_Inc8;

- **Jump on Sign: JS**

  01111000 **IP-Inc8** if SF=1 then IP←(IP)+IP_Inc8;

The 8086/88's Control Transfer Instructions:

Conditional Jump (all within segment short):



- **Jump on Not Equal/Not Zero: JNE/JNZ**

  01110101 **IP-Inc8** if ZF=0 then IP ← (IP) + IP_Inc8;

- **Jump on Not Less/Greater or Equal: JNL/JGE**

  01111101 **IP-Inc8** if SF xor OF=0 then IP ← (IP) + IP_Inc8;

- **Jump on Not Less or Equal/Greater: JNLE/JG**

  01111111 **IP-Inc8** if (SF xor OF) or ZF =0 then IP←(IP)+IP_Inc8;

- **Jump on Not Below/Above or Equal: JNB/JAE**

  01110011 **IP-Inc8** if CF=0 then IP←(IP)+IP_Inc8;

- **Jump on Not Below or Equal/Above: JNBE/JA**

  01110111 **IP-Inc8** if CF or ZF=0 then IP←(IP)+IP_Inc8;

The 8086/88's Control Transfer Instructions:

Conditional Jump (all within segment short):

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

- **Jump on Not Parity/Parity Odd: JNP/JPO**

  01111011 **IP-Inc8** if PF=0 then IP ← (IP) + IP_Inc8;

- **Jump on Not Overflow: JNO**

  01110001 **IP-Inc8** if OF=0 then IP ← (IP) + IP_Inc8;

- **Jump on Not Sign: JNS**

  01111001 **IP-Inc8** if SF=0 then IP←(IP)+IP_Inc8;

- **Jump on CX Equal to Zero: JCXZ**

  11100011 **IP-Inc8** if (cx)=0 then IP←(IP)+IP_Inc8;

The 8086/88's Control Transfer Instructions:

Loop Instructions (all within segment short):

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

- **Loop (cx) Times: LOOP**

  11100010 **IP-Inc8** CX ← (CX)-1;
  if (CX)≠0 then IP ← (IP) + IP_Inc8;

- **Loop while Zero/Equal: LOOPZ/LOOPE**

  11100001 **IP-Inc8** CX ← (CX)-1;
  if (CX)≠0 and ZF=1 then IP ← (IP) + IP_Inc8;

- **Loop while Not Zero/Not Equal: LOOPNZ/LOOPNE**

  11100000 **IP-Inc8** CX ← (CX)-1;
  if (CX)≠0 and ZF=0 then IP ← (IP) + IP_Inc8;

The 8086/88's Control Transfer Instructions:

Call Instructions: CALL

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

- **Call Direct within Segment**

11101000	IP_Low	IP_High

SP ← (SP)-2; $M_{(SP)}$ ← (IP);
IP ← IP_High : IP_low;

Example: call intra_segment_procedure ≡ Push IP; IP ← 0500h;
@intra_segment_procedure = 0500h

Machine code: 11101000 00000000 00000101 ≡ E80005h

- **Call Indirect within Segment**

11111111	Md	010	R/M	Disp. Low-byte	Disp. High-byte (for 16-bit displacement)

SP ← (SP)-2;
$M_{(SP)}$ ← (IP);
IP ← (M_{EA});

The 8086/88's Control Transfer Instructions:

Call Instructions: CALL

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

- **Call Direct Inter-Segment**

10011010	IP_Low	IP_High	CS_Low	CS_High

SP ← (SP)-2; $M_{(SP)}$ ← (CS); SP ← (SP)-2; $M_{(SP)}$ ← (IP);
IP ← IP_High : IP_low; CS ← CS_High : CS_low;

Example: call inter_segment_procedure ≡ Push CS,IP; IP←0500h; CS←?;
@inter_segment_procedure = 0500h in other segment

Machine code:
10011010 00000000 00000101 ???????? ???????? ≡ 9A0005????h

- **Call Indirect Inter-Segment**

11111111	Md	010	R/M	Disp. Low-byte	Disp. High-byte (for 16-bit displacement)

SP ← (SP)-2; $M_{(SP)}$ ← (CS); SP ← (SP)-2; $M_{(SP)}$ ← (IP);
IP ← (M_{EA}); CS ← (M_{EA+2});

The 8086/88's Control Transfer Instructions:

Return from Procedure Instructions: RET

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

- **Return within Segment**

  11000011 IP ← $(M_{(SP)})$; SP ← (SP)+2;

- **Return within Segment adding Immediate to SP**

  11000010 **Data Low** **Data High** IP ← $(M_{(SP)})$; SP ← (SP)+Data+2;

- **Return Inter-Segment**

  11001011 IP ← $(M_{(SP)})$; SP ← (SP)+2;
  CS ← $(M_{(SP)})$; SP ← (SP)+2;

- **Return Inter-Segment adding Immediate to SP**

  11001010 **Data Low** **Data High** IP ← $(M_{(SP)})$; SP ← (SP)+2;
  CS ← $(M_{(SP)})$; SP ← (SP)+Data+2;

The 8086/88's Processor Control Instructions:

- **Clear Carry Flag: CLC**

  11111000 CF ← 0;

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	0

- **Complement Carry Flag: CMC**

  11110101 CF ← $\overline{CF}$;

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	x

- **Set Carry Flag: STC**

  11111001 CF ← 1;

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	1

- **Clear Direction Flag: CLD**

  11111100 DF ← 0;

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	0	-	-	-	-	-	-	-

- **Set Direction Flag: STD**

  11111101 DF ← 1;

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	1	-	-	-	-	-	-	-

- **Clear Interrupt Flag: CLI**

  11111010 IF ← 0;

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	0	-	-	-	-	-	-

The 8086/88's Processor Control Instructions:

- **Set Interrupt Flag: STI**

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	1	-	-	-	-	-	-

  11111011 IF ← 1;

- **Halt Processor: HLT**

  11110100 Force processor to sleep mode.

- **Wait for coprocessor: WAIT**

  10011011 Wait until Busy signal is active.

- **Escape to external device**

  11011 xxx | Md | yyy | R/M | Disp. Low-byte | Disp. High-byte (for 16-bit displacement)

  Current instruction is not executed by the processor and is passed.

- **Bus Lock Prefix: LOCK**

  11110000 Lock the bus until the end of execution of next instruction.

- **Segment Override Prefix**

  001 SR 110 Change the default segment register to SR for the next instruction.

End of Slides