Lecture 09: Flag Control, Compare and Jump Instructions

A group of instructions that directly affect the state of the flags

- LAHF Load AH from flags (AH) ← (Flags)
- SAHF Store AH into flags (Flags) ← (AH)
  Flags affected: SF, ZF, AF, PF, CF
- CLC Clear Carry Flag (CF) ← 0
- STC Set Carry Flag (CF) ← 1
- CLI Clear Interrupt Flag (IF) ← 0
- STI Set interrupt flag (IF) ← 1

SF	ZF		AF		PF		CF

Format of the AH register for the LAHF and SAHF instructions

Ex: Write an instruction sequence to save the current contents of the 8088's flags in the memory location pointed to by SI and then reload the flags with the contents of memory location pointed to by DI

```
LAHF
MOV [SI], AH
MOV AH, [DI]
SAHF
```

The instructions CLC, STC, and CMC are used to clear, set, and complement the carry flag.

Compare Instruction

CMP instruction is used to compare two 8-bit or 16-bit numbers. The results of comparison is reflected by changes in the status flags as a result of subtracting the source from the destination without saving result of subtraction.

Mnemonic	Meaning	Format	Operation	Flags Affected
CMP	Compare	CMP D,S	(D) – (S) is used in setting or resetting the flags	CF, AF, OF, PF, SF, ZF

Lecture 09: Flag Control, Compare and Jump Instructions

Destination	Source
Register	Register
Register	Memory
Memory	Register
Register	Immediate
Memory	Immediate
Accumulator	Immediate

Allowed operands for CMP instruction

EX: Describe what happens to the status flags as the sequence of instructions Is executed

```
MOV AX, 1234H
MOV BX, 0ABCDH
CMP AX, BX
```

The First two instructions makes (AX) = 0001001000110100B
(BX) = 1010101111001101B

The compare instruction performs
(AX)-(BX)= 0001001000110100B - 1010101111001101B = 0110011001100111B

The results of the subtraction is nonzero (ZF=0), positive (SF=0), overflow did not occur OF=0, Carry and auxiliary carry occurred therefore, (CF=1, and AF =1). Finally, the result has odd parity (PF=0).

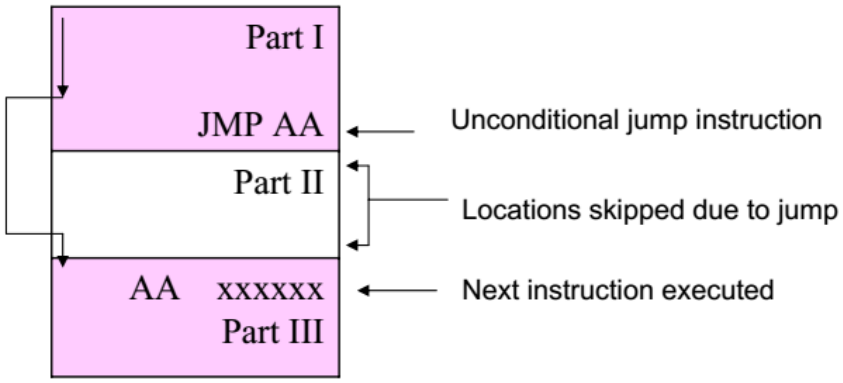
Jump Instructions

There are two types of jump, unconditional and conditional

In unconditional jump, as the instruction is executed, the jump always takes Place to change the execution sequence.

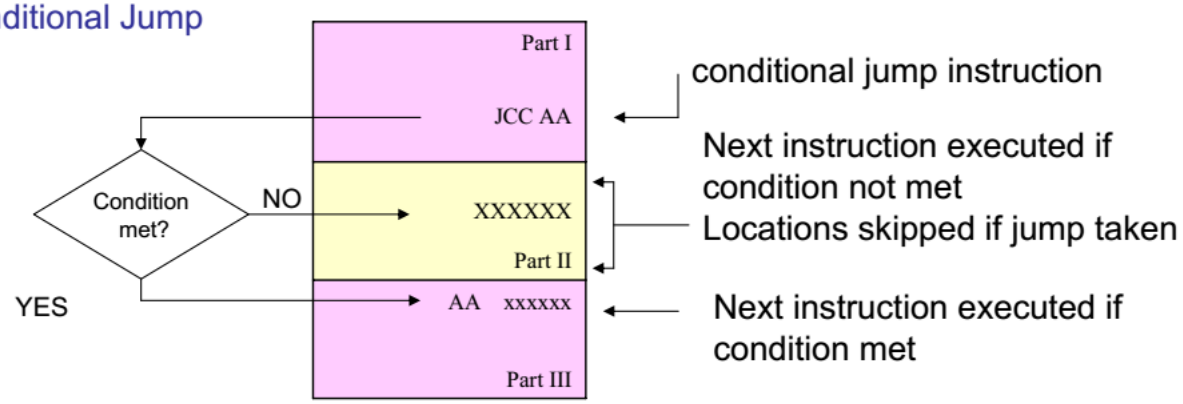
Lecture 09: Flag Control, Compare and Jump Instructions

Unconditional Jump



In conditional jump, status conditions at the time of jump instruction execution
Decide whether or not the jump will occur.

Conditional Jump



Unconditional Jump

Mnemonic	Meaning	Format	Operation	Flags Affected
JMP	Unconditional jump	JMP Operand	Jump is initiated to the address specified by the operand	None

Operand
Short-label
Near label
Far-label
Memptr16
Regptr16
Memptr32

Allowed operand for the jump instruction

JMP 1234H; IP will take the value 1234H

JMP BX ; IP will take the value in BX

JMP [BX]; IP will take the value in memory location pointed to by BX

JMP DWORD PTR [DI]; DS:DI points to two words in memory, the first word identifies the new IP and the next word identifies the new CS.

Unconditional Jump

- Short Jump: JMP short Label1 (8 bit)
- Near Jump: JMP near Label
  - This is the **default** jump: JMP Label
  - The displacement (16 bit) is added to the IP of the instruction following jump instruction.
  - The displacement can be in the range of –32,768 to 32,768.
  - The target address can be register indirect, or assigned by the label.
  - Register indirect JMP: the target address is the contents of two memory locations pointed at by the register.
  - Ex: JMP [SI] will replace the IP with the contents of the memory locations pointed by DS:SI and SI+1
  - Ex: JMP [BP + SI + 1000] in SS
- Far Jump: JMP far Label1
  - this is a jump out of the current segment.
  - EX: JMP DWORD PTR [DI]

Example. The PC Typewriter

- Write an 80x86 program to input keystrokes from the PC's keyboard and display the characters on the system monitor. Pressing any of the function keys F1-F10 should cause the program to end.
- Algorithm:
  1. Get the code for the key pressed
  2. If this code is ASCII, display the key pressed on the monitor and continue
  3. Quit when a non-ASCII key is pressed
- INT 16, BIOS service 0 – Read next keyboard character
  – Returns 0 in AL for non-ASCII characters or the character is simply stored in AL
- To display the character, we use INT 10, BIOS service 0E write character in teletype mode.
  AL should hold the character to be displayed.
- INT 20 for program termination

Example:

```
        MES DB 'type any letter, number, or punctuation key'
        DB 'any F1 to F10 to end the program"
        DB 0d,0a,0a,'$'
     MOV DX, OFFSET MES
        MOV AH,09h
        INT 21h ; to output the characters starting from the offset
AGAIN: MOV AH,0h
        INT 16h; to check the keyboard
        CMP AL,00h
        JZ QUIT ;check the value of the input data
        MOV AH, 0Eh
        INT 10h; echo the character to output
        JMP AGAIN
QUIT: INT 20h
```

Conditional Jump

The folowing flags are affected based on a general comparison instruction

Mnemonic	Description	Flag/Registers
JZ	Jump if ZERO	ZF=1
JE	Jump if EQUAL	ZF=1
JNZ	Jump if NOT ZERO	ZF=0
JNE	Jump if NOT EQUAL	ZF=0
JC	Jump if CARRY	CF=1
JNC	Jump if NO CARRY	CF=0
JCXZ	Jump if CX=0	CX=1
JECXZ	Jump if ECX=0	ECX=0
JP	Jump if PARITY EVEN	PF=1
JNP	Jump if PARITY ODD	PF=0

Jump Based on Unsigned Comparison

The following flags are based on unsigned numbers comparison

Mnemonic	Description	Flag/Registers
JA	Jump if above op 1>op2	CF=0 and ZF=0
JNBE	Jump if not below or equal op1 not <= op2	CF=0 and ZF=0
JAE	Jump if above or equal op1>=op2	CF=0
JNB	Jump if not below op1 not <op2	CF=0
JB	Jump if below op1<op2	CF=1
JNAE	Jump if not above nor equal op1<op2	CF=1
JBE	Jump if below or equal Op1<=op2	CX=1 or ZF=1
JNA	Jump if not above Op1<=op2	CF=1 or ZF=1

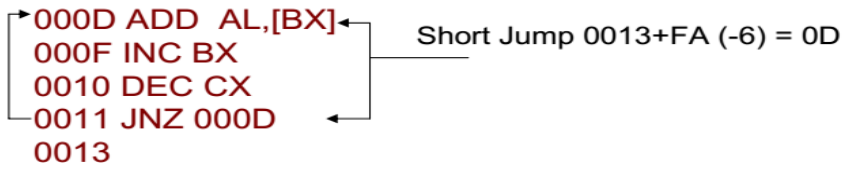
Lecture 09: Flag Control, Compare and Jump Instructions

The following flags are based on signed numbers comparison

Mnemonic	Description	Flag/Registers
JG	Jump if GREATER op1>op2	SF=OF and ZF=0
JNLE	Jump if NOT LESS THAN or equal op1>op2	SF=OF and ZF=0
JGE	Jump if GREATER THAN or equal op1>=op2	SF=OF
JNL	Jump if not LESS THAN op1>=op2	SF=OF
JL	Jump if LESS THAN op1<op2	SF<>OF
JNGE	Jump if not GREATER THAN nor equal op1<op2	SF<>OF
JLE	Jump if LESS THAN or equal Op1<=op2	ZF=1 or SF<>OF
JNG	Jump if not GREATER THAN op1<=op2 Op1<=op2	ZF=1 or SF<>OF
JS	Jump if SIGNED (OP1 NEGATIVE)	SF=1
JNS	Jump if NOT SIGNED (OP1 POSITIVE)	SF=0
JO	Jump if OVERFLOW	OF=1
JNO	Jump if NO OVERFLOW	OF=0

Short Jumps

- **Conditional Jump is a two byte instruction.**
- **In a jump backward the second byte is the 2's complement of the displacement value.**
- **To calculate the target the second byte is added to the IP of the instruction right after the jump.**
- **Ex:**



Example:

Write a program that adds 6 bytes of data and saves the result.
The data should be the following decimal numbers: 52, 21, 51, 31,11, 74

```
.model small
.stack 100h
.data
        Data_in DB 52, 21, 51, 31, 11, 74
         Sum DB ?
.code
main proc far
          MOV AX, @Data
          MOV DS, AX
          MOV CX, 06h
          MOV BX, offset Data_in
          MOV AL, 0
```

```
again: ADD AL, [BX]
          INC BX
          DEC CX
          JNZ Again
          MOV Sum, AL
          MOV AH, 4Ch
          INT 21H
Main endp
end main
```