Chapter 6

Real-Mode 80386DX Microprocessor Programming 2
Part 1
Introduction

6.2 Flag control Instructions
6.3 Compare and Set Instructions
6.4 Jump Instructions
6.5 Subroutines and Subroutine-Handling Instructions
6.6 The Loop and Loop-Handling Instructions
6.7 Strings and Sting-Handling Instructions
Flag Control Instructions - Loading, Storing, and Modifying Flags

- **LAHF/SAHF** → Load/save control flags
- **CLC/STC/CMC** → Modify carry
  - **CLI/STI** → Modify interrupt flag
- **Modifying the carry flag** — **CLC/STC/CMC**
  - Used to initialize the carry flag
  - Clear carry flag
    - **CLC ; 0 → (CF)**
  - Set carry flag
    - **STC ; 1 → (CF)**
  - Complement carry flag
    - **CMC ; (CF*) → (CF)**
- **Modifying the interrupt flag** — **CLI/STI**
  - Used to turn on/off external hardware interrupts
  - Clear interrupt flag
    - **CLI ; 0 → (IF)** Disable interrupts
  - Set interrupt flag
    - **STI ; 1 → (IF)** Enable interrupts

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Meaning</th>
<th>Operation</th>
<th>Flags affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAHF</td>
<td>Load AH from flags</td>
<td>(AH) ← (Flags)</td>
<td>None</td>
</tr>
<tr>
<td>SAHF</td>
<td>Store AH into flags</td>
<td>(Flags) ← (AH)</td>
<td>SF,ZF,AF,PF,CF</td>
</tr>
<tr>
<td>CLC</td>
<td>Clear carry flag</td>
<td>(CF) ← 0</td>
<td>CF</td>
</tr>
<tr>
<td>STC</td>
<td>Set carry flag</td>
<td>(CF) ← 1</td>
<td>CF</td>
</tr>
<tr>
<td>CMC</td>
<td>Complement carry flag</td>
<td>(CF) ← (CF*)</td>
<td>CF</td>
</tr>
<tr>
<td>CLI</td>
<td>Clear interrupt flag</td>
<td>(IF) ← 0</td>
<td>IF</td>
</tr>
<tr>
<td>STI</td>
<td>Set interrupt flag</td>
<td>(IF) ← 1</td>
<td>IF</td>
</tr>
</tbody>
</table>
Flag Control Instructions - Example

- Debug flag notation
  - CF $\rightarrow$ CY = 1, NC = 0
- Example—Execution of carry flag modification instructions
  CY=1
  CLC ;Clear carry flag
  STC ;Set carry flag
  CMC ;Complement carry flag
Loading and Saving the Flag Register

- All loads and stores of flags take place through the AH register
  - Format of the flags in the AH register
    - B0 = CF
    - B2 = PF
    - B4 = AF
    - B6 = ZF
    - B7 = SF
  - Load the AH register with the content of the flags registers
    LAHF
    (Flags) \(\rightarrow\) (AH)
    Flags unchanged
  - Store the content of AH in the flags register—SAHF
    SAHF
    (AH) \(\rightarrow\) (Flags)
    SF,ZF,AF,PF,CF \(\rightarrow\) updated
Loading and Saving the Flag Register

- Application—saving a copy of the flags and initializing with new values
  
  LAHF ;Load of flags into AH
  MOV [MEM1],AH ;Save old flags at address MEM1
  MOV AH,[MEM2] ;Read new flags from MEM2 into AH
  SAHF ;Store new flags in flags register

- Diagram:

```
    7  0
AH | SF | ZF | AF | PF | CF |
```

- SF = Sign flag
  - ZF = Zero flag
  - AF = Auxiliary
  - PF = Parity flag
  - CF = Carry flag
  - = Undefined (do not use)
Flag Control Instructions - Example

- Example—Execution of the flags save and initialization sequence
- Other flag notation:
  - Flag = 1/0
  - SF = NG/PL
  - ZF = ZR/NZ
  - AF = AC/NA
  - PF = PE/PO
Compare Instructions

• Used to compare two values of data and update the state of the flags to reflect their relationship
• General format: CMP D,S
• Operation: Compares the content of the source to the destination
  \[(D) - (S) \rightarrow \text{Flags updated to reflect relationship}\]
  • Source and destination contents unchanged
  • Allowed operand variations:
    • Values in two registers
    • Values in a memory location and a register
    • Immediate source operand and a value in a register or memory
  • Allows SW to perform conditional control flow—typically testing of a flag by jump instruction
    • ZF = 1 \rightarrow D = S = Equal
    • ZF = 0, CF = 1 \rightarrow D < S = Unequal, less than
    • ZF = 0, CF = 0 \rightarrow D > S = Unequal, greater than

---

The 80386, 80486, and Pentium Processors, Triebel
Prof. Yan Luo, UMass Lowell
Compare Instructions- Example

Example—Initialization of internal registers with immediate data and compare. Example:

```
MOV AX,1234H    ;Initialize AX
MOV BX,ABCDH    ;Initialize BX
CMP AX,BX       ;Compare AX-BX
```

- Data registers AX and BX initialized from immediate data
  
  \[
  \text{IMM16} \rightarrow (AX) = 1234H \rightarrow + \text{integer} \\
  \text{IMM16} \rightarrow (BX) = ABCDH \rightarrow - \text{integer}
  \]

- Compare computation performed as:
  
  \[
  (AX) = 0001001000110100_2 \quad (BX) = 1010101111001101_2 \\
  (AX) - (BX) = 0001001000110100_2 - 1010101111001101_2
  \]

  
  \[
  \begin{align*}
  ZF = 0 & = \text{NZ} \\
  SF = 0 & = \text{PL} \quad ; \text{treats as signed numbers} \\
  CF = 1 & = \text{CY} \\
  AF = 1 & = \text{AC} \\
  OF = 0 & = \text{NV} \\
  PF = 0 & = \text{PO}
  \end{align*}
  \]

<table>
<thead>
<tr>
<th>Instruction</th>
<th>ZF</th>
<th>SF</th>
<th>CF</th>
<th>AF</th>
<th>OF</th>
<th>PF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial state</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MOV AX,1234H</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MOV BX,0ABCDH</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CMP AX,BX</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Compare Instructions- Listing and Debug Execution

C:\DOS>DEBUG A:EX66.EXE
-U O D
0F50:0000 1E     PUSH DS
0F50:0001 B80000 MOV AX, 0000
0F50:0004 50     PUSH AX
0F50:0005 B83412 MOV AX, 1234
0F50:0008 B8CDAB MOV BX, ABCD
0F50:000B 3BC3   CMP AX, BX
0F50:000D CB     RETF -G B
AX=1234 BX=ABCD CX=000E DX=0000 SP=003C BP=0000 SI=0000 DI=0000
DS=0F40 ES=0F40 SS=0F51 CS=0F50 IP=000B NV UP EI PL NZ NA PO NC
0F50:000B 3BC3   CMP AX, BX
-\T
AX=1234 BX=ABCD CX=000E DX=0000 SP=003C BP=0000 SI=0000 DI=0000
DS=0F40 ES=0F40 SS=0F51 CS=0F50 IP=000D NV UP EI PL NZ AC PC CY
0F50:000D CB     RETF -G
Program terminated normally
-Q
C:\DOS>

The 80386, 80486, and Pentium Processors, Triebel
Prof. Yan Luo, UMass Lowell 10
Byte Set on Condition Instruction

Byte set on condition instruction
- Used to test results in the flags, such as those of a compare operation, for a specific conditional relationships and then produce a logical result of True or False reflecting the result
- General format:
  SETcc  D
  cc = one of the supported conditional relationships
Byte Set on Condition Instruction

Operation: Flags tested for conditions defined by “cc” and the destination in a register or memory updated as follows

If cc test True:
\[ 11111111_2 = FFH \rightarrow D \]
If cc test False:
\[ 00000000_2 = 00H \rightarrow D \]

• Examples of conditional tests:
  SETE = set byte if equal \( \rightarrow ZF = 1 \)
  SETC = set byte if carry \( \rightarrow CF = 1 \)
  SETBE = set byte if below or equal \( \rightarrow CF = 1 + (\text{or}) ZF = 1 \)

• Example: SETA AL = set byte if above
  if \( CF = 0 \) • (and) \( ZF = 0 \)
  \( (AL) = FFH \)
  Otherwise,
  \( (AL) = 00H \)
Jump Instructions

- Jump operation alters the execution path of the instructions in the program—flow control
  - Unconditional Jump
    - Always takes place
    - No status requirements are imposed
  - Example
    - JMP AA instructions in part I executed
    - Control passed to next instruction identified by AA in Part III
    - Instructions in Part II skipped
Jump Instructions

- **Conditional jump**
  - May or may not take place
  - Status conditions must be satisfied
  - **Example**
    - Jcc AA instruction in Part 1 executed
    - Conditional relationship specified by cc is evaluated
    - If conditions met, jump takes place and control is passed to next instruction identified by AA in Part III
    - Otherwise, execution continues sequentially with first instruction in Part II
    - Condition cc specifies a relationship of status flags such as CF, PF, ZF, etc.
Unconditional Jump Instruction

- Unconditional jump instruction
  - Implements the unconditional jump operation needed by:
    - Branch program control flow structures
    - Loop program control flow structures
  - General format:
    JMP Operand
Types of Unconditional Jump Instruction

- Types of unconditional jumps
  - Intrasegment—branch to address is located in the current code segment
    - Only IP changes value
    - short-label
      - 8-bit signed displacement coded into the instruction
      - Immediate addressing
      - Range equal –126 to +129
      - New address computed as:
        \[(\text{Current IP}) + \text{short-label} \rightarrow \text{IP}\]
        Jump to address = (Current CS):(New IP)
  - near-label
    - 16-bit signed IP offset coded in the instruction
    - Example
      JMP 1234H
regptr16 Unconditional Jump Example

- regptr16
  - 16-bit value of IP specified as the content of a register
  - Register addressing
  - Operation: 
    \[(BX) \rightarrow (IP)\]
  - Jump to address = (Current (CS):(New IP))
- Example
  - 1342:0100 JMP BX
  - Prior to execution
    (IP) = 0100H
    (BX) = 0010H
  - After execution
    (IP) = 0010H
  - Address of next instruction
    (CS:IP) = 1342:0010
memptr16 Unconditional Jump Example

- memptr16
  - 16-bit value of IP specified as the content of a storage location in memory
- Memory addressing
- Example

```
C:\DOS>DEBUG
- A
1342:0100 JMP [BX]
1342:0102
-R BX
BX 0000
 :1000
-E 1000 00 02
-D 1000 1001
1342:1000 00 02
-R
AX=0000 BX=1000 CX=0000 DX=0000 SP=FFE8 BP=0000 SI=0000 DI=0000
DS=1342 ES=1342 SS=1342 CS=1342 IP=0100 NV UP EI PL NZ NA PO NC
1342:0100 FF27 JMP [BX]
- T
AX=0000 BX=1000 CX=0000 DX=0000 SP=FFE8 BP=0000 SI=0000 DI=0000
DS=1342 ES=1342 SS=1342 CS=1342 IP=0200 NV UP EI PL NZ NA PO NC
1342:0200 4D DEC BP
```

- Prior to execution
  - (IP) = 0100H
  - (DS) = 1342H
  - (BX) = 1000H
  - (DS:BX) = (1342H:1000H) = 0200H
- After execution
  - (IP) = 0200H
- Next instruction
  - (CS:IP) = 1342:0200H
Intersegment Unconditional Jump Operation (1)

- Intersegment—branch to address is located in another code segment
  - Both CS and IP change value
  - far-label
    - 32-bit immediate operand coded into the instruction
  - New address computed as:
    - $1^{st}$ 16 bits $\to$ (IP)
    - $2^{nd}$ 16 bits $\to$ (CS)

Jump to address = (New CS) : (New IP)
Intersegment Unconditional Jump Operation (2)

Intersegment—branch to address is located in another code segment

- `regptr32`
  - 32-bit value specified as the content of a register
  - Register addressing
- `memptr32`
  - 32-bit value specified in memory
  - Memory addressing

Example: `JMP DWORD PTR [SI]`

Operation:

\[(DS:SI) \rightarrow \text{new IP}\]
\[(DS:SI +2) \rightarrow \text{new CS}\]

Jump to address = (New CS) : (New IP)
### Conditional Jump Instruction

- **Condition jump instruction**
- **Implements the conditional jump operation**
- **General format:**
  
  \[
  \text{Jcc Operand}
  \]
- **cc = one of the supported conditional relationships**
- **Supports the same operand types as unconditional jump**

#### Jcc

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Meaning</th>
<th>Format</th>
<th>Operation</th>
<th>Flags Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCC</td>
<td>Conditional jump</td>
<td>Jcc Operand</td>
<td>If the specific condition cc is true, the jump to the address specified by the Operand is initiated; otherwise, the next instruction is executed</td>
<td>None</td>
</tr>
</tbody>
</table>

#### Condition Codes

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Meaning</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>JA</td>
<td>above</td>
<td>CF = 0 and ZF = 0</td>
</tr>
<tr>
<td>JAE</td>
<td>above or equal</td>
<td>CF = 0</td>
</tr>
<tr>
<td>JB</td>
<td>below</td>
<td>CF = 1</td>
</tr>
<tr>
<td>JBE</td>
<td>below or equal</td>
<td>CF = 1 or ZF = 1</td>
</tr>
<tr>
<td>JC</td>
<td>carry</td>
<td>CF = 1</td>
</tr>
<tr>
<td>JCXZ</td>
<td>CX register is zero</td>
<td>CX = 0000H</td>
</tr>
<tr>
<td>JE</td>
<td>equal</td>
<td>ZF = 1</td>
</tr>
<tr>
<td>JG</td>
<td>greater</td>
<td>ZF = 0 and SF = OF</td>
</tr>
<tr>
<td>JGE</td>
<td>greater or equal</td>
<td>SF = OF</td>
</tr>
<tr>
<td>JL</td>
<td>less</td>
<td>(SF xor OF) = 1</td>
</tr>
<tr>
<td>JLE</td>
<td>less or equal</td>
<td>((SF xor OF) or ZF) = 1</td>
</tr>
<tr>
<td>JNA</td>
<td>not above</td>
<td>CF = 1 or ZF = 1</td>
</tr>
<tr>
<td>JNAE</td>
<td>not above nor equal</td>
<td>CF = 1</td>
</tr>
<tr>
<td>JNB</td>
<td>not below</td>
<td>CF = 0</td>
</tr>
<tr>
<td>JNBE</td>
<td>not below nor equal</td>
<td>CF = 0 and ZF = 0</td>
</tr>
<tr>
<td>JNC</td>
<td>not carry</td>
<td>CF = 0</td>
</tr>
<tr>
<td>JNE</td>
<td>not equal</td>
<td>ZF = 0</td>
</tr>
<tr>
<td>JNG</td>
<td>not greater</td>
<td>((SF xor OF) or ZF) = 1</td>
</tr>
<tr>
<td>JNGE</td>
<td>not greater nor equal</td>
<td>(SF xor OF) = 1</td>
</tr>
<tr>
<td>JNL</td>
<td>not less</td>
<td>SF = OF</td>
</tr>
<tr>
<td>JNLE</td>
<td>not less nor equal</td>
<td>ZF = 0 and SF = OF</td>
</tr>
<tr>
<td>JNO</td>
<td>not overflow</td>
<td>OF = 0</td>
</tr>
<tr>
<td>JNP</td>
<td>not parity</td>
<td>PF = 0</td>
</tr>
<tr>
<td>JNS</td>
<td>not sign</td>
<td>SF = 0</td>
</tr>
<tr>
<td>JNZ</td>
<td>not zero</td>
<td>ZF = 0</td>
</tr>
<tr>
<td>JO</td>
<td>overflow</td>
<td>OF = 1</td>
</tr>
<tr>
<td>JP</td>
<td>parity</td>
<td>PF = 1</td>
</tr>
<tr>
<td>JPE</td>
<td>parity even</td>
<td>PF = 1</td>
</tr>
<tr>
<td>JPO</td>
<td>parity odd</td>
<td>PF = 0</td>
</tr>
<tr>
<td>JS</td>
<td>sign</td>
<td>SF = 1</td>
</tr>
<tr>
<td>JZ</td>
<td>zero</td>
<td>ZF = 1</td>
</tr>
</tbody>
</table>
### Conditional Jump Instruction

- **Condition jump instruction**
  - Operation: Flags tested for conditions defined by `cc` and:
    - If `cc` test True:
      - IP, or IP and CS are updated with new value(s)—Jump is taken
    - If `cc` test False:
      - IP, or IP and CS are unchanged—continues with sequential execution
  - Examples of conditional tests:
    - `JC = jump on carry → CF = 1`
    - `JPE/JP = jump on parity even → PF = 1`
    - `JE/JZ = jump on equal → ZF = 1`
Branch Program Structures

Example—IF-THEN-ELSE using a flag condition

- One of the most widely used flow control program structure
- Implemented with CMP, JE, and JMP instructions
- Operation
  - AX compared to BX to update flags
  - JE tests for:
    \[ ZF = 1 \]
  - If \( (AX) \neq (BX) \); \( ZF = 0 \) → ELSE—next sequential instruction is executed
  - If \( (AX) = (BX) \); \( ZF = 1 \) → THEN—instruction pointed to by EQUAL executes
Branch Program Structures

Example—IF-THEN-ELSE using a register bit test

- Conditional test is made with JNZ instruction and taken if ZF =0
- Generation of test condition

\[(AL) = \text{x...x} \land 00000100 = 00000\text{x00}\]

if bit 2 = 1 ZF =0
if bit 2 = 0 ZF =1

Therefore, jump to BIT2_ONE only takes place if bit 2 of AL equals 1

- Same operation can be performed by shifting bit 2 to the CF and then testing with JC

\[CF = 1\]
Loop Program Structures

• Example—Loop program structure
  • Allows a part of a program to be conditionally repeated over an over
  • Employs post test—conditional test at end of sequence

• Important parameters
  • Initial count \(\rightarrow\) count register
  • Terminal count \(\rightarrow\) zero or other value

• Program flow of control:
  • Initialize count
    
    ```
    MOV CL,COUNT
    ```
  • Perform body of loop operation
    
    ```
    AGAIN: --- --- first of multiple instructions
    ```
  • Decrement count
    
    ```
    DEC CL
    ```
  • Conditional test for completion
    
    ```
    JNZ AGAIN
    ```

\[486,\text{ and Prentium Processors, Triebe}\]
\[\text{cf. Yan Luo, UMass Lowell}\]
Loop Program Structures

- Example—Another loop program structure
- Allows a part of a program to be conditionally repeated over an over
- Employs pre-test—at entry of loop
- Important parameters
  - Initial count \(\rightarrow\) count register
  - Terminal count \(\rightarrow\) zero or other value
- Program flow of control:
  - Initialize count
    MOV CL,COUNT
  - Pre-test
    AGAIN: JZ NEXT
  - Perform body of loop operation
    --- --- first of multiple instructions
  - Decrement count
    DEC CL
  - Unconditional return to start of loop
    JMP AGAIN
Block Move Program

```
MOV AX, DATASEGADDR
MOV DS, AX
MOV SI, BLK1ADDR
MOV DI, BLK2ADDR
MOV CX, N
MOV AH, [SI]
MOV [DI], AH
INC SI
INC DI
DEC CX
JNZ NXTPT
HLT
```

The 80386, 80486, and Pentium Processors, Triebel
‘of Yan Luo, UMass Lowell

27