Chapter 4

Machine Language Coding and the Debug Program
Instruction Encoding

80x86’s instruction set is hybrid length

- Multiple instruction sizes, but all have byte wide lengths—
  - 1 to 6 bytes in length for 8088/8086
  - Up to 17 bytes for 80386, 80486, and Pentium
- Advantages of hybrid length
  - Allows for many addressing modes
  - Allows full size (32-bit) immediate data and addresses
- Disadvantage of variable length
  - Requires more complicated decoding hardware—speed of decoding is critical in modern uP
Instruction Encoding

Information encoded in an instruction

- What operation?
- What operands?
- Byte, word or double-word?
- Operands in register or memory?
- How the address is to be generated, if mem?
First Byte of an Instruction

- **Opcode field (6-bits)—**specifies the operation to be performed by the instruction
  - Move immediate to registers/memory = 1100011
  - Move memory to accumulator = 1010000
  - Move segment register to register/memory = 10001100

- **REG (3-bit)—**selects a first operand as a register
  - Move immediate to register = 1011(w)(reg)—only requires one register which is the destination
    - Accumulator register= 000
    - Count register = 001
    - Data Register = 010
  - **W (1-bit)—**data size word/byte for all registers
    - Byte = 0, Word =1
  - **D (1-bit)—**register direction: tells whether the register which is selected by the REG field in the second byte is the source or destination
    - Add register to register = 000000(d)(w)
    - D=0 → source operand, D=1 → destination operand
Example of One-Byte Instruction

One Byte Example:

- Encode the instruction in machine code

\[ \text{INC CX} \]

Solution:
- Use "INC register" instruction format—special short form for 16-bit register
  \[ 01000 \text{ (REG)} \]
- CX is destination register
  \[ CX = 001 \]
- Machine code is
  \[ 01000 (001) = 01000001 = 41H \rightarrow \text{one byte instruction} \]
  \[ \text{INC CX} = 41H \]
Second Byte of an Instruction

Byte 2 information:

- **MOD (2-bit mode field)**—specifies the type of the second operand
  - Memory mode: 00, 01, 10—Register to memory move operation
    - 00 = no immediate displacement (register used for addressing)
    - 01 = 8-bit displacement (imm8) follows (8-bit offset address)
    - 10 = 16-bit displacement (imm16) follows (16-bit offset address)
  - Register mode: 11—register to register move operation
    - 11 = register specified as the second operand

---

<table>
<thead>
<tr>
<th>CODE</th>
<th>EXPLANATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Memory Mode, no displacement follows*</td>
</tr>
<tr>
<td>01</td>
<td>Memory Mode, 8-bit displacement follows</td>
</tr>
<tr>
<td>10</td>
<td>Memory Mode, 16-bit displacement follows</td>
</tr>
<tr>
<td>11</td>
<td>Register Mode (no displacement)</td>
</tr>
</tbody>
</table>

*Except when R/M = 110, then 16-bit displacement follows*
Second Byte of an Instruction (Cont.)

Byte 2 information (continued):
- REG (3-bit register field)—selects the register for a first operand, which may be the source or destination
  - Accumulator register = 000
  - Count register = 001
  - Data Register = 010
  - Move register/memory to/from register
    - Byte 1 = 100010(d)(w)
    - Byte 2 = (mod) (reg) (r/m)

- Affected by byte 1 information:
  - W (1-bit)—data size word/byte for all registers
    - Byte = 0
    - Word = 1
The 80386, 80486, and Pentium Processors

Prof. Yan Luo, UMass Lowell

Second Byte of an Instruction (Cont.)

R/M (3-bit register/memory field)—specifies the second operand as a register or a storage location in memory

- Dependent on MOD field
  - Mod = 11 R/M selects a register
    - R/M = 000 Accumulator register
    - R/M = 001 = Count register
    - R/M = 010 = Data Register
  - Move register/memory to/from register
    - Byte 1 = 100010(d)(w)
    - Byte 2 = (mod) (reg) (r/m)

- Affected by byte 1 information:
  - W (1-bit)—data size word/byte for all registers
    - Byte = 0
    - Word = 1
  - D (1-bit)—register direction for first operand in byte 2 (reg)
    - D = 0 \(\rightarrow\) source operand
    - D = 1 \(\rightarrow\) destination operand
MOD = 00, 01, or 10 selects an addressing mode for the second operand that is a storage location in memory, which may be the source or destination

- Dependent on MOD field
- Mod = 00 R/M
  - R/M = 100 → effective address computed as 
    
    \[ EA = (SI) \]

- R/M= 000 = → effective address computed as 
  
  \[ EA = (BX)+(SI) \]

- R/M = 110 = → effective address is coded in the instruction as a direct address
  
  \[ EA = \text{direct address} = \text{imm8 or imm16} \]
Example: MOV

- Move register/memory to/from register
  - Byte 1 = 100010(d)(w)
  - Byte 2 = (mod) (reg) (r/m)
- Affect of byte 1 information:
  - W (1-bit)—data size word/byte for all registers
    - Byte = 0
    - Word = 1
  - D (1-bit)—register direction for first operand (REG) in byte 2
    - D = 0 → source operand
    - D = 1 → destination operand
Example: INC CL

Two byte example using R/M field for a register:
- Encode the instruction in machine code
  
  \[
  \text{INC CL}
  \]

- Solution:
  - Use “INC register/memory” instruction format—general form for 8-bit or 16-bit register/memory
  - Byte 1
    
    \[
    1111111(W) \quad \text{Cl}= \text{byte wide register} \rightarrow W = 0
    \]
    
    \[
    11111110 = \text{FEH}
    \]
**Example: INC CL**

- Two Byte example using R/M field for a register (continued):
  - Byte 2
    - (MOD) 000 (R/M)
    - Destination is register CL
      - MOD = 11
      - R/M = 001
        \[(11)000(001) = 11000001 = C1H\]
    - Machine code is
      \[(\text{Byte 1})(\text{Byte 2}) = 11111110 \; 11000001 = \text{FEC1H} \]
      \[\rightarrow\]
      \[\text{two byte instruction}\]
      \[\text{INC CL} = \text{FEC1H}\]
Example: MOV BL, AL

Two byte example using R/M field for a register:

- Encode the instruction in machine code

```
MOV BL, AL
```

Solution:
- Use “register/memory to/from register” instruction format—most general form of move instruction
- Byte 1

```
100010(D)(W)
```
- Assuming AL (source operand) is the register encoded in the REG field of byte 2 (1st register)
  - D = 0 = source
  - Both registers are byte wide
  - W = 0 = byte wide
- Byte 1 = \(100010(0)(0) = 10001000 = 88H\)
Two Byte Example (continued):

- Byte 2

\[(\text{MOD})(\text{REG})(\text{R/M})\]

- Both operands are registers
  - MOD = 11
  - 2nd register is destination register BL
    - R/M = 011
  - 1st register is source register AL
    - REG = 000

\[(11)000(011) = 11000011 = \text{C3H}\]

- Machine code is

\[(\text{Byte 1})(\text{Byte 2}) = 10000001 \ 11000111 = 88\text{C3H} \rightarrow \text{two byte instruction} \]

\[
\text{MOV BL, AL} = 88\text{C3H}
\]
Example: ADD AX, [SI]

- Two byte example using R/M field for memory:
  - Encode the instruction in machine code
    
    \[
    \text{ADD AX, [SI]} \]

- Solution:
  - Use "register/memory with register to either" instruction format
    - Most general form of add instruction
    - No displacement needed—register indirect addressing
  - Byte 1
    
    \[
    000000(D)(W) \]
    
    - AX (destination operand) is the register encoded in the REG field of byte 2 (1st register)
      - D = 1 = destination
    - Addition is of word wide data
      - W = 1 = word wide
    - Byte 1 = 00000011 = 03H
Example: ADD AX, [SI]

- Two Byte example using R/M field for memory (continued):
  - Byte 2
    (MOD)(REG)(R/M)
  - Second operand is in memory and pointed to by address is SI
    - MOD = 00 → [SI]
  - R/M specifies the addressing mode
    - R/M = 100 → [SI]
  - 1st register is destination register AX
    - REG = 000
      (00)000(100) = 00000100 = 04H
  - Machine code is
    (Byte 1)(Byte 2) = 00000011 00000100
    = 0304H → two byte instruction
    ADD AX, [SI] = 0304H

<table>
<thead>
<tr>
<th>EFFECTIVE ADDRESS CALCULATION</th>
<th>MOD = 00</th>
<th>MOD = 01</th>
<th>MOD = 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/M</td>
<td>MOD = 00</td>
<td>MOD = 01</td>
<td>MOD = 10</td>
</tr>
<tr>
<td>000 (BX) + (SI)</td>
<td>(BX) + (SI) + D8</td>
<td>(BX) + (SI) + D16</td>
<td>(BX) + (SI) + D16</td>
</tr>
<tr>
<td>001 (BX) + (DI)</td>
<td>(BX) + (DI) + D8</td>
<td>(BX) + (DI) + D16</td>
<td>(BX) + (DI) + D16</td>
</tr>
<tr>
<td>010 (BP) + (SI)</td>
<td>(BP) + (SI) + D8</td>
<td>(BP) + (SI) + D16</td>
<td>(BP) + (SI) + D16</td>
</tr>
<tr>
<td>011 (BP) + (DI)</td>
<td>(BP) + (DI) + D8</td>
<td>(BP) + (DI) + D16</td>
<td>(BP) + (DI) + D16</td>
</tr>
<tr>
<td>100 (SI)</td>
<td>(SI) + D8</td>
<td>(SI) + D16</td>
<td>(SI) + D16</td>
</tr>
<tr>
<td>101 (DI)</td>
<td>(DI) + D8</td>
<td>(DI) + D16</td>
<td>(DI) + D16</td>
</tr>
<tr>
<td>110 DIRECT ADDRESS (BX)</td>
<td>(BP) + D8</td>
<td>(BP) + D16</td>
<td>(BP) + D16</td>
</tr>
<tr>
<td>111 (BX)</td>
<td>(BX) + D8</td>
<td>(BX) + D16</td>
<td>(BX) + D16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REG</th>
<th>W = 0</th>
<th>W = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>AL</td>
<td>AX</td>
</tr>
<tr>
<td>001</td>
<td>CL</td>
<td>CX</td>
</tr>
<tr>
<td>010</td>
<td>DL</td>
<td>DX</td>
</tr>
<tr>
<td>011</td>
<td>BL</td>
<td>BX</td>
</tr>
<tr>
<td>100</td>
<td>AH</td>
<td>SP</td>
</tr>
<tr>
<td>101</td>
<td>CH</td>
<td>BP</td>
</tr>
<tr>
<td>110</td>
<td>DH</td>
<td>SI</td>
</tr>
<tr>
<td>111</td>
<td>BH</td>
<td>DI</td>
</tr>
</tbody>
</table>