LECTURE # 03

PREPARED BY: TAHMINA KHAN

INTRODUCTION TO ASSEMBLY PROGRAMMING

- The CPU can work only in binary, very high speeds.
  - It is tedious & slow for humans to deal with 0s & 1s in order to program the computer.
- A program of 0s & 1s is called machine language.
  - Early computer programmers actually coded programs in machine language.
- Eventually, Assembly languages were developed, which provided *mnemonics* for machine code.
  - Mnemonic is typically used in computer science and engineering literature to refer to codes & abbreviations that are relatively easy to remember.

INTRODUCTION TO ASSEMBLY PROGRAMMING

- Assembly language is referred to as a *low-level* language because it deals directly with the internal structure of the CPU.
  - Assembly language programs must be translated into machine code by a program called an *assembler.*
  - To program in Assembly language, programmers must know the number of registers and their size.
    - As well as other details of the CPU.

INTRODUCTION TO ASSEMBLY PROGRAMMING

- Today there are many different programming languages, such as C/C++, BASIC, C#, etc.
  - Called *high-level* languages because the programmer does not have to be concerned with internal CPU details.
- High-level languages are translated into machine code by a program called a *compiler.*
  - To write a program in C, one must use a *C compiler* to translate the program into machine language.

INTRODUCTION TO ASSEMBLY PROGRAMMING

- An Assembly language program consists of a series of lines of Assembly language instructions.
- An Assembly language instruction consists of a mnemonic, optionally followed by one or two *operands*.
  - Operands are the data items being manipulated.
  - Mnemonics are commands to the CPU, telling it what to do with those items.
- Two widely used instructions are *move* & *add*.

MOV INSTRUCTION

- The MOV instruction copies data from one location to another, using this format:

```
MOV  destination,source ;copy source operand to destination
```

- This instruction tells the CPU to move (in reality, copy) the source operand to the destination operand.
  - For example, the instruction "**MOV DX,CX**" copies the contents of register CX to register DX.
  - After this instruction is executed, register DX will have the same value as register CX.

MOV INSTRUCTION

- This program first loads CL with value 55H, then moves this value around to various registers inside the CPU.

```
MOV  CL,55H ;move 55H into register CL
MOV  DL,CL ;copy the contents of CL into DL (now DL=CL=55H)
MOV  AH,DL ;copy the contents of DL into AH (now AH=DL=55H)
MOV  AL,AH ;copy the contents of AH into AL (now AL=AH=55H)
MOV  BH,CL ;copy the contents of CL into BH (now BH=CL=55H)
MOV  CH,BH ;copy the contents of BH into CH (now CH=BH=55H)
```

MOV INSTRUCTION

- The use of 16-bit registers is shown here:

```
MOV  CX,468FH  ;move 468FH into CX (now CH=46,CL=8F)
MOV  AX,CX     ;copy contents of CX to AX (now AX=CX=468FH)
MOV  DX,AX     ;copy contents of AX to DX (now DX=AX=468FH)
MOV  BX,DX     ;copy contents of DX to BX (now BX=DX=468FH)
MOV  DI,BX     ;now DI=BX=468FH
MOV  SI,DI     ;now SI=DI=468FH
MOV  DS,SI     ;now DS=SI=468FH
MOV  BP,DI     ;now BP=DI=468FH
```

MOV INSTRUCTION

- In the 8086 CPU, data can be moved among all the registers, as long as the source and destination registers match in size (*Except* the flag register.)
  - There is no such instruction as "`MOV FR,AX`".
- Code such as "`MOV AL,DX`" will cause an error.
  - One cannot move the contents of a 16-bit register into an 8-bit register.

Table 1-4: Registers of the 8088/86/286 by Category

Category	Bits	Register Names
General	16	AX, BX, CX, DX
	8	AH, AL, BH, BL, CH, CL, DH, DL

MOV INSTRUCTION

- Using the MOV instruction, data can be moved directly into nonsegment registers only.
  - The following demonstrates legal & illegal instructions.

```
MOV   AX,58FCH   ;move 58FCH into AX   (LEGAL)
MOV   DX,6678H   ;move 6678H into DX   (LEGAL)
MOV   SI,924BH   ;move 924B  into SI   (LEGAL)
MOV   BP,2459H   ;move 2459H into BP   (LEGAL)
MOV   DS,2341H   ;move 2341H into DS   (ILLEGAL)
MOV   CX,8876H   ;move 8876H into CX   (LEGAL)
MOV   CS,3F47H   ;move 3F47H into CS   (ILLEGAL)
MOV   BH,99H     ;move 99H into BH     (LEGAL)
```

MOV INSTRUCTION

- Values *cannot* be loaded directly into any segment register (CS, DS, ES, or SS).
  - To load a value into a segment register, load it to a non-segment register, then move it to the segment register.

```
MOV  AX,2345H  ;load 2345H into AX
MOV  DS,AX     ;then load the value of AX into DS

MOV  DI,1400H  ;load 1400H into DI
MOV  ES,DI     ;then move it into ES, now ES=DI=1400
```

MOV INSTRUCTION

- If a value less than FFH is moved into a 16-bit register, the rest of the bits are assumed to be zeros.
  - For example, in "**MOV BX,5**" the result will be BX = 0005.
    - BH = 00 and BL = 05.
- Moving a value that is too large into a register will cause an error.

```
MOV  BL,7F2H      ;ILLEGAL: 7F2H is larger than 8 bits
MOV  AX,2FE456H   ;ILLEGAL: the value is larger than AX
```

ADD INSTRUCTION

- The ADD instruction has the following format:

```
ADD  destination,source  ;ADD the source operand to the destination
```

- ADD tells the CPU to add the source & destination operands and put the result in the destination.
  - To add two numbers such as 25H and 34H, each can be moved to a register, then added together:

```
MOV  AL,25H   ;move 25 into AL
MOV  BL,34H   ;move 34 into BL
ADD  AL,BL    ;AL = AL + BL
```

  - Executing the program above results in: AL = 59H (25H + 34H = 59H) and BL = 34H.
    - The contents of BL do not change.

ADD INSTRUCTION

- *Is it necessary to move both data items into registers before adding them together?*
  - No, it is not necessary.

```
MOV DH,25H  ;load one operand into DH
ADD DH,34H  ;add the second operand to DH
```

  - In the case above, while one register contained one value, the second value followed the instruction as an operand.
    - This is called an immediate operand.

ADD INSTRUCTION

- An 8-bit register can hold numbers up to FFH.
  - For numbers larger than FFH (255 decimal), a 16-bit register such as AX, BX, CX, or DX must be used.
- The following program can add 34EH & 6A5H:

```
MOV AX,34EH   ;move 34EH into AX
MOV DX,6A5H   ;move 6A5H into DX
ADD  DX,AX    ;add AX to DX: DX = DX + AX
```

  - Running the program gives DX = 9F3H.
    - (34E + 6A5 = 9F3) and AX = 34E.

ADD INSTRUCTION

- Any 16-bit nonsegment registers could have been used to perform the action above:

```
MOV  CX,34EH  ;load 34EH into CX
ADD CX,6A5H ;add 6A5H to CX (now CX=9F3H)
```

  - The general-purpose registers are typically used in arithmetic operations
    - Register AX is sometimes referred to as the *accumulator.*