Assembly

Chapter 1(Basic features of PC hardware)

This chapter provides an explanation of the basic hardware (bits,bytes,registers,memory,processor,and data bus).

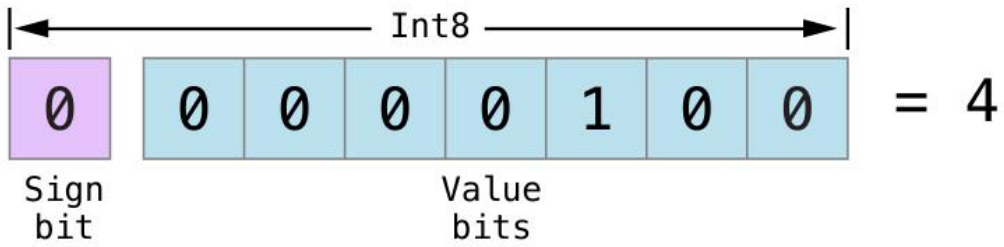
1-What is Bit and Bytes?

-Bit is the basic block of computer storage , and it has two values(0-> off) and (1 -> on).

-Bytes =>Each byte consists of eight bits for data and one bit for parity.

2-what is the rule of parity?

-the number of bits that are 'on' in each byte must always be odd.



3-What is the data item or data field?

-a program can treat group of one or more related bytes as unit of data to define a particular value.the procesor supports certain data sizes that are natural to it:

(Word-> 2byte)(DoupleWord -> 4 byte)(Quadword->8-byte)(paragraph -> 16-byte)(kilobyte ->640K => 640*1024 byte)(MegaByte=>2^{20})

4-Binary Arithmetic

Position	7	6	5	4	3	2	1	0
Exponent	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Value	128	64	32	16	8	4	2	1
Binary number	1	0	1	1	0	1	0	1

5-Hexadecimal arithmetic

Number	0	1	2	3	4	5	6	7
Binary	0000	0001	0010	0011	0100	0101	0110	0111
Hexadecimal	0	1	2	3	4	5	6	7

Number	8	9	10	11	12	13	14	15
Binary	1000	1001	1010	1011	1100	1101	1110	1111
Hexadecimal	8	9	A	B	C	D	E	F

6-What is the ASCII CODE?

-is a standard for storing and interchanging the information between the computers ,Data on PC may be classed as numeric (binary data used for arithmetic) or as alphanumeric(character and descriptive data).

7-What is the PC Components?

-The main component of the PC is its system board ,it contains(the processor/Co_processors /main memory/connectors/Expansion slots for optional)

-the slots and components provide access to such components as ROM ,RAM,Hard Disk,CD-ROM drives,video units,keyboard,mouse,parallel devices and serial devices.

8-What is the processor?

-The brain of the PC is a processor (also known as central processing unit),based on the Intel 8086 family.
-Perform all executing of instructions and processing of data.
-there are a basic components in the the processor(capacity of memory,registers ,and data bus)

9-breif description of various Intel processors:

A-Intel 8088 processor.

B-Intel 8086 processor.
C-Intel 80386 processor.
D-Intel 80286 processor.
E-Intel 80486 Processor.

9-What is Execution unit and Bus interface unit:-

-The processor is partitioned into two logical units:an execution unit ,and bus interface unit.

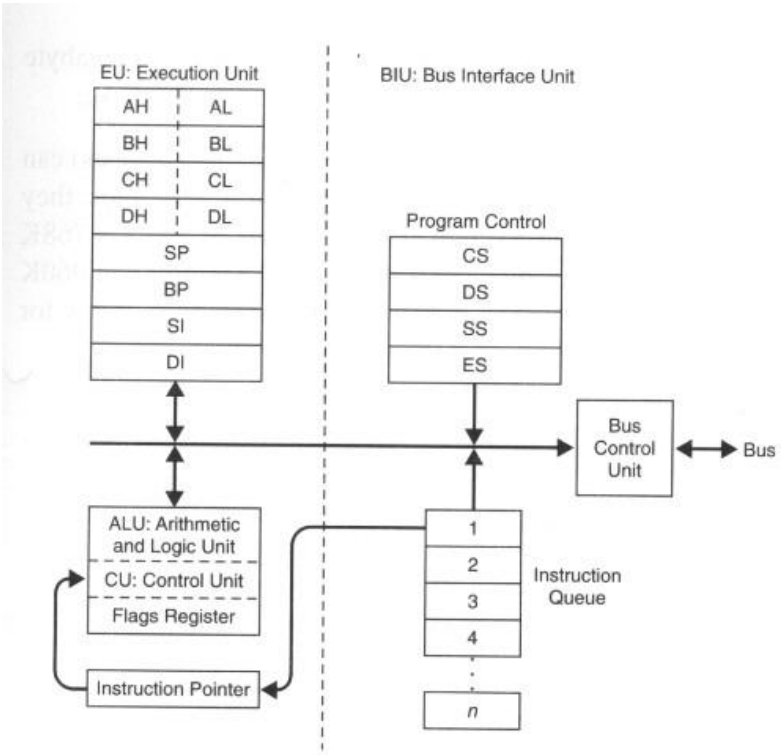


Figure 1-2 Execution Unit and Bus Interface Unit

9-What is the purpose of the execution unit and bus interface unit:

1-the role of EU is to execute instructions.

2-the role of BIU is to delivers instructions and data to the EU .
3-EU contains an arithmetic and logic unit (ALU),Control unit (CU),and number of registers
4-the important function of BIU is to manage (the bus control unit , segment registers,instruction queue).

10-BIU function:

-Control the buses that transfer data to the EU ,to memory ,and to external input/output devices.

-provide access to instructions (these instructions are in memory).
-BIU is enable to look ahead and prefetch instructions so that there is always a queue of instructions ready to execute.

11-Segment registers function:

-Control memory addressing.

12-Tell me what do you know about the internal memory:-

1-there are two types of memory on PC (Random access memory) and (Read only memory).

2-Bytes in memory are numbered ,beginning with 00

13-Physical memory map of an 8086 type PC:

14-ROM and RAM:

-ROM :consists of memory chips that can only be read,Instructions and data are permanently "burned into" the chips.

-RAM : is a temporary storage and "read/write memory" ,it is used for execution of programs .

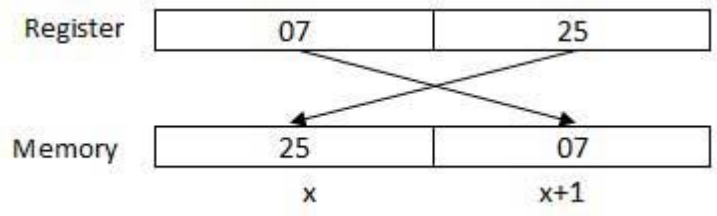
15-Basic Input/Output system (BOIS)

-Begins at 768k and handles input/output devices ,such as hard disk controller ,……,etc.

16-How the Processor store the data in both memory and registers ?

-The processor store the data in memory in reverse byte sequence :the low order byte in the low memory address ,and the high order byte in the high memory address.

-The processor store the data in registers as the data is entered.



17-Be careful about the difference between the address of a memory location and its content.

18-What is the deference between the Absolute address and segment: offset address

-absolute address is a value that directly references a specific location in memory.

-A segment:offset address combines the starting address of a segment with an offset value.

19-what is a segment?

-is a special areas defined in a program for containing the code ,the data ,and what is known as the stack.A segment begins on a paragraph boundary.

20-Paragraph boundary:

-The segment begins at a location evenly divisible by 16,or hex 10.

21-Code segment/data segment/stack segment

-Code segment : contains the machine instructions that are to execute,the code segment registers addresses the code segment.

-Data segment : contains the program’s defined data ,constraints,and work areas,the data segment register addresses the data segment.
-Stack segment : contains any data or addresses that the program need ,The stack segment register addresses the stack segment.

23-Segment boundary:

-A segment register is 16 bits in size and contains the starting address of a segment.

24-Segment registers properties:

-Segment registers of 8086/8088 are 16 bits in length and operate in real mode.

25-The six segment registers are CS,DS,SS,ES,FS,GS:-

1-CS register :- Contains the starting address of a program's code segment ,this starting address and the offset value in the Instruction pointer(IP) register =>indicates the address of an instruction to be fetched for execution.

2-DS register :- Contains the starting address of a program's data segment ,this address plus an offset value in an instruction .

3-SS register :- the system stores the starting address ,plus an offset value in the stack pointer (SP)register indicates the current word in the stack being addressed.

4-ES register :- used by some string operations to handle memory addressing

26-Pointer registers :-

-32-bit -> EIP,ESP,EBP

-16-bit -> IP,SP,BP

27-What is Instruction pointer?

-the 16-bit register that contains the offset address of the next instruction that is to execute.

-Associated with CS register as (CS:IP)

Attention : The (SP)stack pointer register and (BP)base pointer register are associated with the SS register and premit the system to access the data in the stack segment.

28-What is the stack pointer register?

-The 16-bit SP register provides an offset value ,and associated with (SS)

Stack segment (SS:SP), refers to the current value being processed in the stack.

29-What is the base pointer ?

-The 16-bit BP facilities referencing parameters ,which are data and addresses

that a program passes via the stack.

-the program combines the address in SS register with BP.

30-General purpose register:-

1-are the 32-bit EAX ,EBX ,ECX ,EDX

2-The rightmost 16-bit portions are AX,BX,CX,DX.
(AL,BL,CL,DL)=> the leftmost eight bits.

31-AX register :-

-The primary accumulator , is used for operations involving input/output and most arithmetic (multiply/divide/and translate instructions assume the use of AX.

32-BX register :-

-BX is known as the base register since it is the only general purpose register that can be used as an index to extend addressing, used for computations.

32-CX register :

-Is known as the count register

-It may contain a value to control the number of times a loop is repeated or a value to shift bits.

33-DX registers :-

-DX is known as the data register, some input/output operations ,and

multiply ,divide operations that involve large values .

34-What is the Index registers ?

-The index registers are 32-bi ESI and EDI; the rightmost 16-bit portions are SI

and DI.

-are available for indexed addressing and for some use in addition and subtraction.

35-SI registers :

-The 16-bit source index registers

38-What is the Flags registers ?

-The 32-bit Eflag contains bits that indicate the status of various activities,

The rightmost 16-bit portion of EFLAGS is the flag registers (16-bit).

39-What is the common flags bits?

-OF->Over Flow ….Indicates overflow of a high-order (leftmost) bit following arithmetic.

-DF (Direction flow) Determines left or right direction for moving or comparing string data.
-IF (interrupt flow) Indicates that all external interrupts ,such as keyboard entry , are to be processed or ignored.
-TF (trap flow) permits operation of the processor in single step mode.
-SF(Sign Flow) contains the resulting sign of an arithmetic operation (0=negative and 1= positive)
-ZF Indicates the result of an arithmetic or comparison operation (0=nonzero and 1=zero result)
-AF (auxiliary carry)contains a carry out if bit 3 into bit 4 in an arithmetic operation ,for specialized arithmetic.
-PF (parity) Indicates the number of 1-bits that result from an operation .
-CF (carry) it contains carries from a high-order bit following at arithmetic operation .

Chapter 2

Instruction Addressing and execution

1-What is the main functions of the operating system?

- File Management
- Input/Output
- Program loading
- Memory management
- Interrupt handling

2- What is happening when turning on the computer?

1-The processor enter a reset state.

2-Clear all memory locations to zero

3-Perform a parity check of memory

4-set the CS register to segment address FFFF[0]H that the BOIS routine begins,and to offset zero.

5-BIOS contains a set of routines in ROM to provide device support.

3-What is the two data areas in the BIOS?

1-Interrupt Vector table : begins in low memory at location 0 and contains 256 (4byte) addresses in the form segment:offset .BIOS and operating system uses these addresses for interrupts that occur.

2-BIOS data area :- begins at location 40[0] , largely concerned with status of attached devices.

4-What is the difference between .COM and .EXE?

-Both .COM and .EXE are types of executable programs

-.COM Programs consists of one segment that contains code ,data ,and the stack .
-.COM is useful as small utility program or as a resident program .
-.EXE program consists of seperate code ,data ,and stack segments.

5-What does the system program loader do when you request the system to load an .EXE program from desk into memory for execution?

1-Accesses the .EXE program from desk.

2-Constructs a 256-byte (100H) program segment prefix (PSP) on a paragraph boundary in available internal memory.
3-Stores the program in memory immediately following the PSP.
4-Loads the address of the PSP in the DS and ES registers.

5-Loads the address of the code segment in the CS register and sets the IP register to the offset of the first instruction.
6-Loads the address of the stack in the SS registers and sets the SP register to the size of the stack.
7-Transfers control to the program for execution ,beginning with the first instruction in the code segment.

6-The Stack has a three main uses:

1-Save the return addresses which the subroutine later uses for returning.

2-The program that calls the subroutine may also pass data by placing it in he stack,where the subroutine accesses it.

3-Store the can save the present content of the registers on the stack,make the calculations ,and then restore the data from the stack to the registers.

7-.COM -> the program loader automaticly s=defines the stack.

8-.EXE -> you must explicitly define a stack .

9-Storing data in the stack is different from other segments ,it begins storing data at the highest location in the segment and stores downword through memory.

10-What is the purpose of push and pop?

1- (modify the contents of the SP registers)

2-(and are used for storing data on the stack and retreiving it)

PUSH :- storing a value and decrementing SP by 2 to the next lower storage word.

POP :- returning the value from the stack and incrementing SP by 2 bytes to the next higher storage word.

11-the basic steps the processor takes in execuing an instruction is:

1-FETCH the next instruction to be executed from memory and palce it in the instruction queue.

2-DECODE (calculate addresses that reference memory,deliver data to the arithmetic logic unit ,increment the IP)

3-EXECUTE (perform the requested operation ,store the results in registers ,and set flags such as Zero or Carry where required)

12-Execute the instruction address from the CS address 5BE0H and IP offset 0023H :

Chapter 3

Examining Computer memory and Execution instructions

1-This chapter uses a DOS program named DEBUG to view memory ,to enter programs in memory ,and to trace thier execution.

2-What is the DEBUG commands ?

-to perform a number of instructions

A Assemble symbolic instructions into machine code.
D Display the contents of an area of memory in hex format
E Enter data into memory ,beginning at a specific location.
G Run the executable program in memory (G means 'go')
H Perfoem Hexadecimal arithmetic
N name a program
P proceed ,execute a set of of related instructions
Q quit the DEBUG
R Display the contentsof one or more registers in hex format
T Trace the execution of one instruction
U Unassemble machine code into sympolic code.

3-The rules of DEBUG ?

1-No difference between lowercase and uppercase

2-all number are in hexa

3-spaces are only used to seperate between parmeteres.

4- Segment : offset

4- the displayed screen for (D DS : 200) Or any other instuction consists of three parts :

1-theleft is the hex address of the leftmost displayed byte ,in segment : offset format

2-the wide area in the center is the hex representation of the displayed data .

3-the right is the ASCII representation of the diplayed data

Viewing memory locations Excercises

5-Excercise 1:Examining the bois data area(LOW memory unit) at locsstion 400H (segment address 40[0]H)

-checking the serial and parallel ports: the fisrt 16 byte

D 40:00

1-the first four bytes show serial ports.

2-the second bytes show parallel ports.

-Checking system equipment (410-411H)

D 40:10 <press enter>
فى الكتاب محتاج تقرا

-Checking the keyboard shoft status :

D 40:17
The result should be like this :

0040:0017 00 00….

-Checking the video status :

-the bois data area at location 4170H

- D 40:17
- 0040:0017 00 00……

6-Checking the video status :-

-the first video data area is 449H.

- D 40:49

-the first video conatians the current video mode

6-Exercise two: Examining ROM BIOS (High memory)

-Checking copyright notice and serial number:

-examine data in ROM BIOS in high memory

-at location FE000H

- D FE00:0

Checking the rom bios date:

-Begins at location FFFF5H

-D FFFF:5

7-Machine Language example 1 :Using immediate data

MACHINE INSTRUCTION :

B82301

SYMPOLIC CODE :

MOV AX,0123

RXPLANATION :

Move value 0123H to AX

Notes :

1-machine instructions may be one ,two ,or three bytes in length.

8-Using the INT instruction

-exits from the program and enters a bios routine ,perform the requested fuction and returns to your program.

-Getting the current Date and time:

-INT 21H and function code 2AH

```
A       100
MOV    AH,2A
INT     21
JMP    100
```

THE RESULT :
AL : Day of the week,where 0=Sunday
CX: Year (for example , 07D4H =2004)
DH:Month (01H through 0CH)
DL:Day of the month (01H through 1FH)

-The current time is INT 21H and fuction code 2CH

```
A       100
MOV     AH,2C
JMP      100
```

THE RESULT:
The operation delivers hours to CH (in 24 format,where 00=midnight)
CL :Minutes
DH:seconds

-Determining Installed Equipment :

-You checked **410H ,411H**

-INT 11H delivers the equipment data into AX

-Using INT for displaying :-

```
100     MOV    AH,09
102     MOV    DX,109
105     INT     21
107     JMP    100
109     DB     'YOUR NAME','$'<enter><enter>
```

-Using INT for keyboard input

```
100   MOV    AH,10
102   INT    16
104   JMP    100 <ENTER><ENTER>
```

Using the PTR operator :-

---????

Requirements for coding in assemply language

1-Some advantages to coding in assembly :

-Provides more control over handling particular hardware requirements.

-Generates smaller ,more compact executable modules.

-Result in faster execution.

2-Assemply language features :

-Certain characteristics of assemply language are first covered :

(Program comments,reserved words,identifiers,statements,and directives)

A-Comment :begins with a semicolon (;).

;calculate productivity

B-Reserved Words :

-Instruction ,such as MOV and ADD,which are operartions that the computer can execute.
-Directives , such as END or SEGMENT ,which you use to provide information that the computer can execute.
-operators such as FAR and SIZE ,which you use in expressions ;

-predefined sympols ,such as @DATA and @MODEL ,which return information to your program during the assemply.

C-Identifiers : there are two of identifiers are NAME and LABEL

1- Name refers to the address of a data item,such as COUNTER in

```
Counter DB 0
```

2-Label refers to the address of an instruction,procedure, or segment ,such as MAIN and B30: in the following :

```
MAIN PROC FAR
B30: ADD    BL,25
```

D-Statements :

The types of statements are:

1-Instructions ,such as MOV and ADD ,which the assembler translates to object code.

	Identifier	OPERATION	OPERAND
Directive :	COUNT	DB	1

2-Directives ,which tell the assembler to perform a specific action,such as a data item.

	IDENTIFIER	OPERATION	OPERAND(s)
INSTRUCTION :	L30:	MOV	AX,0

3-What is the PAGE and TITLE listing directives?

-Help to control the format of listing if an assemble of program,

PAGE define the maximun number of lines and the characters,like PAGE 60,132.

Title to define the name of the program

4-What is the SEGMENT directive ?

-the segment statement defines the start and the end of a segment ,the segment-name be present.

-In real mode ,a stack segment defines a storage ,a data segment defines a data items,and code segment provides for executable program.

```
SYNTAX :
SEGMENT-NAME      SEGMENT   {align}    {combine}    {'class'}
.....
SEGMENT-NAME      ENDS
```

-The maximum size of a segment in real mode is 64k.

-The align indicates the boundary on which the segment is to begin. (The typical requirement is PARA)(Paragraph boundary)

-**The combine** option indicates wether to combine segment with other segments ,combine types are COMMON,PUBLIC,STACK,AT.

-The CLASS option is used to group related segments when linking , ‘code’ for the code segment ,’data’ for the segment,’stack’ for the stack segment.

```
PAGE          60,132
TITLE         A04ASM1 SEGMENT FOR AN .EXE PROGRAM
;-------------------------------------------------------------------
STACK         SEGMENT PARA STACK ‘STACK’
                ….
STACK         ENDS
;-------------------------------------------------------------------
CODESEG    SEGMENT PARA ‘CODE’
               MAIN PROC FAR
               …
MAIN         ENDP                    ;End of procedure
CODESEG    ENDS                      ;End of segment
               END    MAIN            ;End of program
;-------------------------------------------------------------------
```

5-What is the ASSUME directive ?

-to tell the assembler the purpose of each segment in the program

- ASSUME SS:stackname , DS: datasegment , CS:codesegment, ….

6-How to end the program execution ?!

-Using the INT 21H as a request to end program execution with the fuction 4C00H.

7-There are many actions can be done by the INT 21H with different fuction code:

-Key board input -Screen handling -Disk I/O

-Printer Output -Ending a program execution

8-What is .STARTUP and .EXIT directives?

-.STARTUP generates the instructions to initialize the segment.

-.EXIT generates the INT 21H function 4CH instructions for exiting the program.

9-Defining types of data: [name] DN expression

-NAME : to reference the data item.

-Directive : to define the type the data item -> DB(define byte) ,DW(word),DD(doupleword),DF(farword),DQ(quadeword), and DT (tenbytes).

10-The expressions :

```
DATAX     DB     ?      ;uninitialized item
DATAY     DB     25     ;initialized item
DATAZ     DB     21,22,23,24,25,26,..
```

11-What is the simplified directives?

-Some shotcuts for defining segments .To use them, you have to initialize the memory model before defining any segment.the syntax is .MODEL memorymodel

and the memory model may be

Tiny,Small,Medium,Compact,Large,Huge,or Flat.

12-What is the requirements for these models?

MOEDL	NUM OF CODE SEGMENTS	NUM OD DATA SEGMENT
Small	1 <= 64k	1 <= 64k
Medium	Any number, Any size	1 <= 64K
Compact	1 <= 64K	Any number ,any size
Large	Any number ,any size	Any number ,any size
Huge	Any number , any size	Any number ,any size

-The .MODEL directive automatically generates the required ASSUME statement for all models.

13-After defining the .MODEL directive ?

.STACK [size] and .DATA and .CODE [segment-name]