1. Introduction to Assembly Language

Solved EXERCISE 1

Note:
Dear fellows I tried my best to solve this exercise questions if there's any mistake or doubt in any question correct it and also share with others and me.

EXERCISE 1 P#15

1. How the processor uses the address bus, the data bus, and the control bus to communicate with the system memory?

Address bus

The group of bits that the processor uses to inform the memory about which element to read or write is collectively known as address bus.

Data bus

Data bus is used to move the data from the memory to the processor in a read operation and from the processor to the memory in a write operation. *The third group consists of miscellaneous independent lines used for control purposes. For example, one line of the bus is used to inform the memory about whether to do the read operation or the write operation. These lines are collectively known as the *control bus*. *

Control bus

The third group consists of miscellaneous independent lines used for control purposes. For example, one line of the bus is used to inform the memory about whether to do the read operation or the write operation. These lines are collectively known as the *control bus*.

Control bus is special and relatively complex, because different lines comprising it behave differently. Some take information from the processor to a peripheral and some take information from the peripheral to the processor.

2. Which of the following are unidirectional and which are bidirectional?

a. **Address Bus**

The address bus is unidirectional. Address always travels from processor to memory.

b. **Data Bus**

Data bus is bidirectional. Data moves from both, processor to memory and memory to processor

c. **Control Bus**

Control bus is a bidirectional bus. It can carry information from processor to memory as well as from memory to processor

3. What are registers and what are the specific features of the accumulator, index registers, program counter, and program status word?

Accumulator

There is a central register in every processor called the accumulator. All mathematical and logical operations are performed on accumulator. The word size of a processor is defined by the width of its accumulator. A 32bit processor has an accumulator of 32 bits.

Index Register

Index register is used in such a situation to hold the address of the current array location. Now the value in the index register cannot be treated as data, but it is the address of data. In general whenever we need access to a memory location whose address is not known until runtime we need an index register. Without this register we would have needed to explicitly code each, iteration separately. In newer architectures the distinction between

accumulator and index registers has become vague. They have general registers, which are more versatile and can do both functions. They do have some specialized behaviors but basic operations can be done on all general registers.

Program counter

A register in the control unit of the CPU that is used to keep track of the address of the current or next instruction. Typically, the program counter is advanced to the next instruction, and then the current instruction is executed. Also known as a "sequence control register" and the "instruction pointer."

Everything must translate into a binary number for our dumb processor to understand it, be it an operand or an operation itself. Therefore the instructions themselves must be translated into numbers. For example to add numbers we understand the word "add." We translate this word into a number to make the processor understand it. This number is the actual instruction for the computer. All the objects, inheritance and encapsulation constructs in higher-level languages translate down to just a number in assembly language in the end. Addition, multiplication, shifting; all big programs are made using these simple building blocks. A number is at the bottom line since this is the only thing a computer can understand. (Page#4)

Program Status Word

This is a special register in every, architecture called the flags register or the program status word. Like the accumulator it is an 8, 16, or 32 bits register but unlike the accumulator it is meaningless as a unit, rather the individual bits carry different meanings. The bits of the accumulator work in parallel as a unit and each bit mean the same thing. The bit of the flags register work register work independently and individually, and combined its value is meaningless.

4. What is the size of the accumulator of a 64bit processor?

Size of the accumulator of a 64bit processor is 64-bits.

5. What is the difference between an instruction mnemonic and its opcode?

Opcode

Short for ***op****erational* ***code***, it is a number that determines the computer instruction to be executed. (machine language representation of an instruction)

6. How are instructions classified into groups?

INSTRUCTION GROUPS

Data Movement Instructions

These instructions are used to move data from one place to another. These places can be registers, memory, or even inside peripheral devices. Some examples are:

mov ax, bx

lad 1234

Arithmetic and Logic Instructions

Arithmetic instructions like addition, subtraction, multiplication, division and Logical instructions like logical and, logical or, logical xor, or complement are part of this group. Some examples are:

and ax, 1234

add bx, 0534

add bx, [1200]

The bracketed form is a complex variation meaning to add the data placed at address 1200.
In certain other cases we want the processor to first execute a separate block of code and then come back to resume processing where it left.
These are instructions that control the program execution and flow by playing with the instruction pointer and altering its normal behavior to point to the next instruction. Some examples are:
cmp ax, 0
jne 1234

Special Instructions

Another group called special instructions works like the special service commandos. They allow changing specific processor behaviors and are used to play with it. They are used rarely but are certainly used in any meaningful program. Some examples are:
cli
sti
Where cli clears the interrupt flag and sti sets it. Without delving deep into it, consider that the cli instruction instructs the processor to close its ears from the outside world and never listen to what is happening outside, possibly to do some very important task at hand, while sti restores normal behavior. Since these instructions change the processor behavior they are placed in the special instructions group.

7. A combination of 8bits is called a byte. What is the name for 4bits and for 16bits?

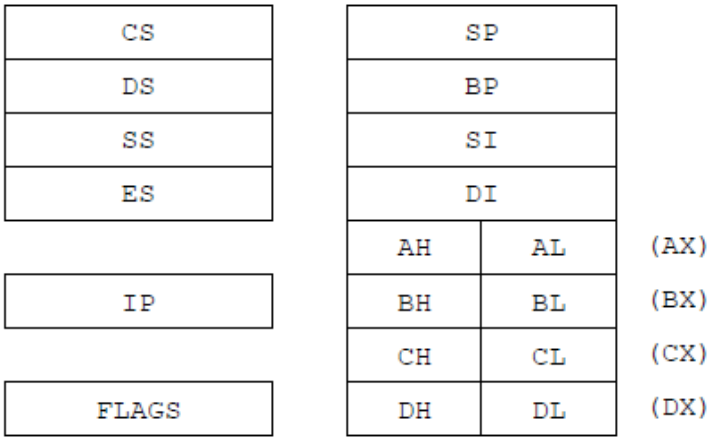
A number in base 16 is called a hex number and can be represented by 4 bits. The collection of 4 bits is called a nibble. One hexadecimal digit takes 4 bits so 4 hexadecimal digits make one word or two bytes.
4 bits = 1 nibble
8 bits =1 byte = 2 nibbles
16 bits = 2 bytes = 4 nibbles

8. What is the maximum memory 8088 can access?

8088 processor can access 1MB of memory.

9. List down the 14 registers of the 8088 architecture and briefly describe their uses.

1. **CS**
2. **DS**
3. **SS**
4. **ES**
5. **IP**
6. **SP**
7. **BP**
8. **SI**
9. **DI**
10. **AH AL (AX)**



11. **BH BL (BX)**
12. **CH CL (CX)**
13. **DH DL (DX)**
14. **FLAGS**

General Registers (AX, BX, CX, and DX)

The registers AX, BX, CX, and DX behave as general purpose registers in Intel architecture and do some specific functions in addition to it. X in their names stand for extended meaning 16bit registers. For example AX means we are referring to the extended 16bit "A" register. Its upper and lower byte are separately accessible as AH (A high byte) and AL (A low byte). All general-purpose registers can be accessed as one 16bit register or as two 8bit registers. The two registers AH and AL are part of the big whole AX. Any change in AH or AL is reflected in AX as well. AX is a composite or extended register formed by gluing together the two parts AH and AL.

The A of AX stands for Accumulator. Even though all general-purpose registers can act as accumulator in most instructions there are some specific variations, which can only work on AX, which is why it is named the accumulator. The B of BX stands for Base because of its role in memory addressing as discussed in the next chapter. The C of CX stands for Counter as there are certain instructions that work with an automatic count in the CX register. The D of DX stands for Destination as it acts as the destination in I/O operations. The A, B, C, and D are in letter sequence as well as depict some special functionality of the register.

Index Registers (SI and DI)

These are the index registers of the Intel architecture, which hold address of data and used in memory access. Being an open and flexible architecture, Intel allows many mathematical and logical operations on these registers as well like the general registers. The source and destination are named because of their implied functionality as the source or the destination in a special class of instructions called the string instructions. However their use is not at all restricted to string instructions. SI and DI are 16bit and cannot be used as 8bit register pairs like AX, BX, CX, and DX.

Instruction Pointer (IP)

This is the special register containing the address of the next instruction to be executed. No mathematics or memory access can be done through this register. It is out of our direct control and is automatically used. Playing with it is dangerous and needs special care. Program control instructions change the IP register.

Stack Pointer (SP)

It is a memory pointer and is used indirectly by a set of instructions. This register will be explored in the discussion of the system stack.

Base Pointer (BP)

It is also a memory pointer containing the address in a special area of memory called the stack and will be explored alongside SP in the discussion of the stack.

Flags Register

The flags register as previously discussed is not meaningful as a unit rather it is bit wise significant and accordingly each bit is named separately.

The bits not named are unused. The Intel FLAGS register has its bits organized as follows:
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
O D I T S Z A P C
The individual flags are explained in the following table.

Segment Registers (CS, DS, SS, and ES)
The code segment register, data segment register, stack segment register, and the extra segment register are special registers related to the Intel segmented memory model.

10. What flags are defined in the 8088 FLAGS register? Describe the function of the zero flag, the carry flag, the sign flag, and the overflow flag.

Zero flag
The Zero flag is set if the last mathematical or logical instruction has produced a zero in its destination.

Carry flag
When two 16bit numbers are added the answer can be 17 bits long or when two 8bit numbers are added the answer can be 9 bits long. This extra bit that won't fit in the target register is placed in the carry flag where it can be used and tested.

Sign flag
A signed number is represented in its two's complement form in the computer. The most significant bit (MSB) of a negative number in this representation is 1 and for a positive number it is zero. The sign bit of the last mathematical or logical operation's destination is copied into the sign flag.

Overflow flag
The overflow flag is set during signed arithmetic, e.g. addition or subtraction, when the sign of the destination changes unexpectedly. The actual process sets the overflow flag whenever the carry into the MSB is different from the carry out of the MSB.

11. Give the value of the zero flag, the carry flag, the sign flag, and the overflow flag after each of the following instructions if AX is initialized with 0x1254 and BX is initialized with 0x0FFF.
a. add ax, 0xEDAB
b. add ax, bx
c. add bx, 0xF001

- Carry Flag=1
- Zero Flag=1
- Sign Flag=0
- Overflow Flag=0

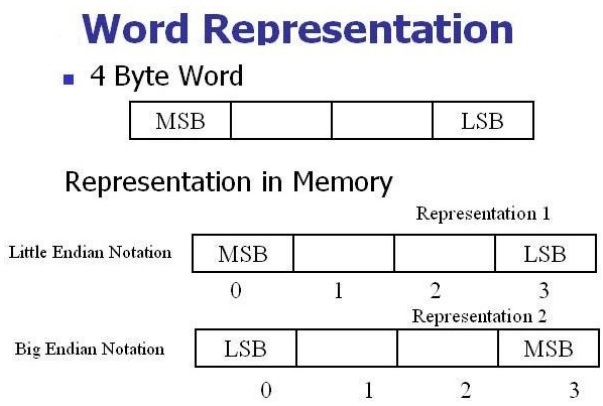
12. What is the difference between little endian and big endian formats? Which format is used by the Intel 8088 microprocessor?

Little endian	Big endian
• It can be least significant, more significant, more significant, and most significant called the little-endian order and is used by Intel. • The little-endian have the argument that this scheme places the less significant value at a lesser address and more significant value at a higher address.	• 32bit numbers either the order can be most significant; less significant, lesser significant, and least significant called the big-endian order used by Motorola. • The big-endian have the argument that it is more natural to read and comprehend

- Little-endian order is used by Intel 8088 microprocessor.

13. For each of the following words identify the byte that is stored at lower memory address and the byte that is stored at higher memory address in a little endian computer.

a. 1234
b. ABFC
c. B100
d. B800



a. 1234
1 and 2 are stored at lower memory address and 3,4 stored at higher memory address.
b. ABFC
A and B are stored at lower memory address and F, C are stored at higher memory address.
c. B100
B and 1 are stored at lower memory and 0,0 are stored at higher memory address.
d. B800
B and 8 are stored at lower memory address and 0,0 are stored at higher memory address.

(MSB) (LSB)

14. What are the contents of memory locations 200, 201, 202, and 203 if the word 1234 is stored at offset 200 and the word 5678 is stored at offset 202?

In Little Endean, least significant byte is stored first at lower addresses; most significant byte is stored after it. Like for example the number 0x1234 is stored at memory address 0x123 then it will appear like as below:

Address	Contents
…	
0x123	34
0x124	12
…	

Do try it for the given memory addresses with given numbers.

15. What is the offset at which the first executable instruction of a COM file must be placed?

The very first instruction in our assembly programs i.e. [org 0x0100] tell that its .com file shall be loaded at an offset address 100 in the segment. This is done to maintain backward compatibility.

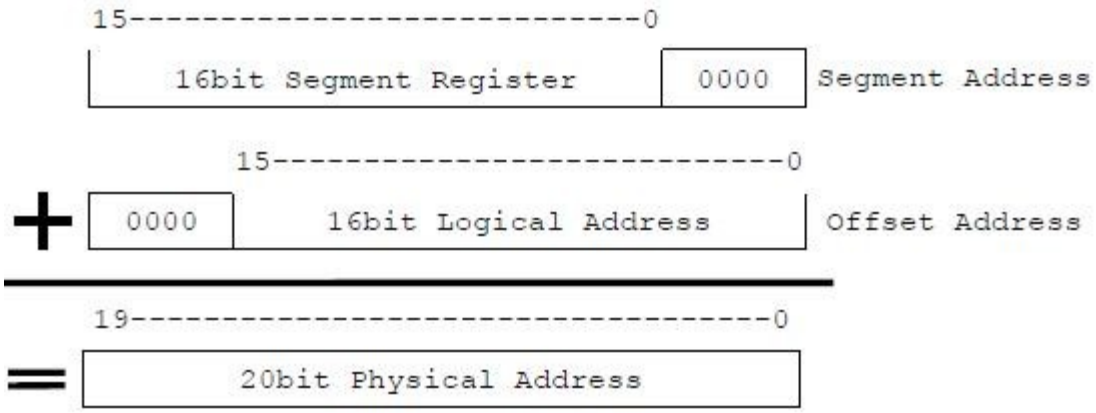
16. Why was segmentation originally introduced in 8088 architecture?

Four windows of 64K
Code window
Data window
Stack window
The segmented memory model allows multiple functional windows into the main memory, a code window, a data window etc. The processor sees code from the code window and data from the data window. The size of one window is restricted to 64K. 8085 software fits in just one such window. It sees code, data, and stack from this one window, so downward compatibility is attained. **(Try to find better answer)**

17. Why a segment start cannot start from the physical address 55555.

For any segment base address, segment first physical address will have 0 in the least significant position in hexadecimal format.
Let say, our Segment base =0x1234, and we calculate segment first physical address as 0x12340 <- 0x12340 + 0x00000 (Segment First Address)
Thus all segments starting physical address has 0 at its least significant position. In case of 55555 as segment first physical address there in no 0 at least significant positions so this cannot be a segment starting physical address.

18. Calculate the physical memory address generated by the following segment offset pairs.



a. 1DDD:0436

1DDD0	segment
+00436	offset
=1E206	physical memory address

Hexadecimal	Decimal
A	10
B	11
C	12
D	13
E	14
F	15

b. 1234:7920

12340
+07920

=19C60

c. 74F0: 2123

74F00
+02123

=77023

d. 0000:6727

00000
+06727

=06727

e. FFFF: 4336

FFFF0
+04336

=104326

f. 1080:0100

10800
+00100

=10900

g. AB01: FFFF

AB010
+0FFFF

=BB00F

19. What are the first and the last physical memory addresses accessible using the following segment values?

First 0x0000 and last 0xFFFF

First physical address	Last physical address
a. 1000	
0x10000	0x10000
+0x00000	+0x0FFFF
=0x10000	**=0x1FFFF**
b. 0FFF	
0x0FFF0	0x0FFF0
+0x00000	+0x0FFFF
=0x0FFF0	**=0x1FFEF**
c. 1002	
0x10020	0x10020
+0x00000	+0x0FFFF
=0x10020	**=0x2001F**
d. 0001	
0x00010	0x00010
+0x00000	+0x0FFFF
=0x00010	**=0x1000F**
e. E000	
0xE0000	0xE0000
+0x00000	+0x0FFFF
=0xE0000	**=0xEFFFF**

Note:

- That we have added Base & Offset addresses to calculate Physical addresses. Moreover also notice that we placed a Zero in Base & Offset address to produce a 20-bit address.
- I have used hexadecimal representation as shown by 0x used in the beginning of a number.

20. Write instructions that perform the following operations.

a. Copy BL into CL

mov CL, BL

b. Copy DX into AX

mov AX, DX Moreover DX can not be copied into AL because of size mis-match.

c. Store 0x12 into AL

add AL, 0x12

d. Store 0x1234 into AX

add AX,0x1234
e. Store 0xFFFF into AX
add AX, 0xFFFF

21. Write a program in assembly language that calculates the square of six by adding six to the accumulator six times.

[Move 6 into ax register in first instruction and then keep adding 6 in next 5 instructions. You will get square of 6.]

```
; Calculate square of six by adding six to accumulator six times
;[org 0x0100]

        mov ax,6        ; load first number in ax
        add ax,6        ; accumulate sum
        add ax,6
        add ax,6
        add ax,6
        add ax,6

        mov ax, 0x4c00   ; terminate program
        int 0x21
```

```
 1    ; calculates square of six by adding six to accumulator six times
 2
 3                          _                   [org 0x0100]
 4
 5 00000000 B80600                               mov ax,6
 6 00000003 050600                               add ax,6
 7 00000006 B80600                               mov ax,6
 8 00000009 B80600                               mov ax,6
 9 0000000C B80600                               mov ax,6
10 0000000F B80600                               mov ax,6
11 00000012 B80600                               mov ax,6
12
13 00000015 B8004C                               mov ax, 0x4c00
14 00000018 CD21                                 int 0x21
```

Q:What is the instruction of this COPY AX INTO MEMORY LOCATION WITH OFFSET 0FFF IN THE CURRENT DATA SEGMENT?

The instruction that copy ax into memory location with offset 0FFF in the current data segment is given below:

Mov [DS:0FFF], AX