II - ASSIGNMENT

(Start Writing From Here)

1) Explain about different types of Instructions set and Assembler directives?

A) Instruction set:

An instruction set is a collection of instructions that a processor can execute. It is the interface between the hardware and the software, defining the operations that a processor can perform.

1) Data transfer instructions:

All the instructions which perform data movement come under this category. The source data may be a register, memory location, port etc. the destination may be a register, memory location or port.

2) Arithmetic instructions:

Instructions of this group perform addition, subtraction, multiplication, division, increment, decrement, comparision, ASCII and decimal adjustment.

3) Logical instructions:

Instructions of these group perform logical AND, OR, XOR, NOT and TEST operations.

4) Branch instructions:

It is called program execution transfer instruction. Instructions of this group transfer program execution from the normal sequence of instructions to the specified destination or target

5) String instructions:

String is series of bytes or series of words stored in sequential memory locations. The 8086 provides some instructions which handle string operations such as string movement, comparision, scan, load and store.

Assembler directives

Assembler directives are the directions to the assembler which indicate how an operand or section of the program is to be processed. These are also called Pseudooperations which are not executable by the microprocessor.

Various Assembler directives are explained below:-

1) ASSUME:- The ASSUME directives is used to inform the assembler the name of the logical segment it should use for a specified segment.

2) DB:- Define byte. It is used to declare a byte variable or set aside one or more storage locations of type byte in memory.

3) DW:- Define word. It tells the assembler to define a variable of type word or to reserve storage locations of type word in memory.

4) DDC Define double word):- This directive is used to declare a variable of type double word or restore memory locations which can be accessed as type double word.

5) DQ (Define ~~ten byte~~ quadword):- It is used to tell the assembler to declare a variable 4 words in length or to reserve 4 of storage in words.

6) DT (Define ten bytes):- It is used to inform the assembler to define a variable which is 10 bytes in length or to reserve 10 bytes of storage in memory.

7) EQU- The Equate is used to give a name to some value or symbol. Everytime the assembler finds the given name in the program, it will replace the name with the value or symbol

we have equated with that name.

8) ORG (Originate) :- The ORG statement changes that starting offset address of the data.

9) PROC :- procedure : it is used to identify the start of a procedure or subroutine.

10) END :- End program. This directive indicates the assembler that this is the end of the Programm module.

11) ENDP :- End procedure. It indicates the end of the procedure (subroutine) to the assembler.

12) ENDS :- EndSegment : This directive is used with the name of the segment to indicate the end of that logical segment.

2) with an example, explain Division Algorithm.

A) Division of two fixed-point binary numbers in signed magnitude representation is performed with paper and pencil by a process of successive compare, shift and subtract operations. Binary division is much simpler than decimal division because here the quotient digits are either 0 and 1 and there is no need to estimate how many times the dividend or Partial remainder fits into the divisor.

=) Binary division is easier than decimal division because the quotient division is either 0 or 1. Also, there is no need to estimate how many times the dividend or partial remainders adjust to the divisor.

Example of Division Algorithm:

Divisor B = 10001, Dividend A = 011100000

```
                 11010
      10001 ) 011100000
              01110
              011100
              -10001
              -------
               01011 0
               -10001
               -------
                01010 0
                -10001
                -------
                 00101 0
                 -10001
                 -------
                  00011 0
                  -10001
                  -------
                   00010 (0)
```

The divisor is compared with the five most significant bits of the dividend. Since the 5 bit number is smaller than B. we again repeat the same process. Now the 6-bit number is greater than B, so we place a 1 for the quotient bit in the sixth position above the dividend. Now we shift the divisor once that the right and subtract it from the dividend. The difference is know as a partial remainder because the division could have stopped here to obtain a quotient of 1 and a remainder equal to the partial remainder. Comparing a partial remainder with the divisor continues the process. If the partial remainder is smaller than the divisor, the quotient bit is 0 and no subtraction is need. The divisor is shifted once to the right in anycase. Obviously the result gives both a quotient and a remainder.

Q) Discuss about priority interrupt.

A) Data transfer between CPU and I/O device is initiated by the

CPU. The CPU cannot start the transfer unless the device is ready to communicate with the CPU.

⇒ A priority interrupts is a system that establishes a priority over the various sources to determine which condition is to be serviced first when two or more request arrive simultaneously. The system may also determine which conditions are permitted to interrupt the computer while another interrupt is being serviced.

⇒ The disvantages of the software method is that if there are many interrupts, the time required to poll them can exceed the time available to service the I/O device. In this situation a hardware priority interrupt unit can be used to speed up the operation.

⇒ The hardware priority-interrupt unit functions as an overall manager in an interrupt system environment. It accepts interrupt requests from many sources, determines which of the incoming requests has the highest priority, and issues an interrupt request to the computer based on the determination.

4) Explain about RAM, ROM

A) Random Access Memory is used to store the programs and data being used by the CPU in real time. The data on the RAM can be read, written and erased any no. of times.

It is a voltaile memory. It is also called as Main memory or primary memory.

Types of RAM:

1) Static RAM:- Static RAM or SRAM stores a bit of data using the state of a six-digit transistor memory cell.

2) Dynamic RAM:- DRAM stores a bit using the pair of transistors and capacitors which constitute a DRAM memory cell.

ROM (Read only Memory):

It is a type of memory where the data has been prerecorded. Data stored in ROM is retained even after the computer is turned off.

It is called as a Secondary memory. It stores a program called BIOS - This program checks the status of all the devices attached to the computer.

Types of ROMs:

programmable ROMs: It is type of ROM where the data is written after the memory chip has been created.

EPROMs:- Erasable PROM is a type of ROM where the data is non-volatile memory chip can be erased by exposing to UV light.

EEPROMs: Electrically Erasable PROM is a type of ROM where memory chip can be erased using field electron emission.

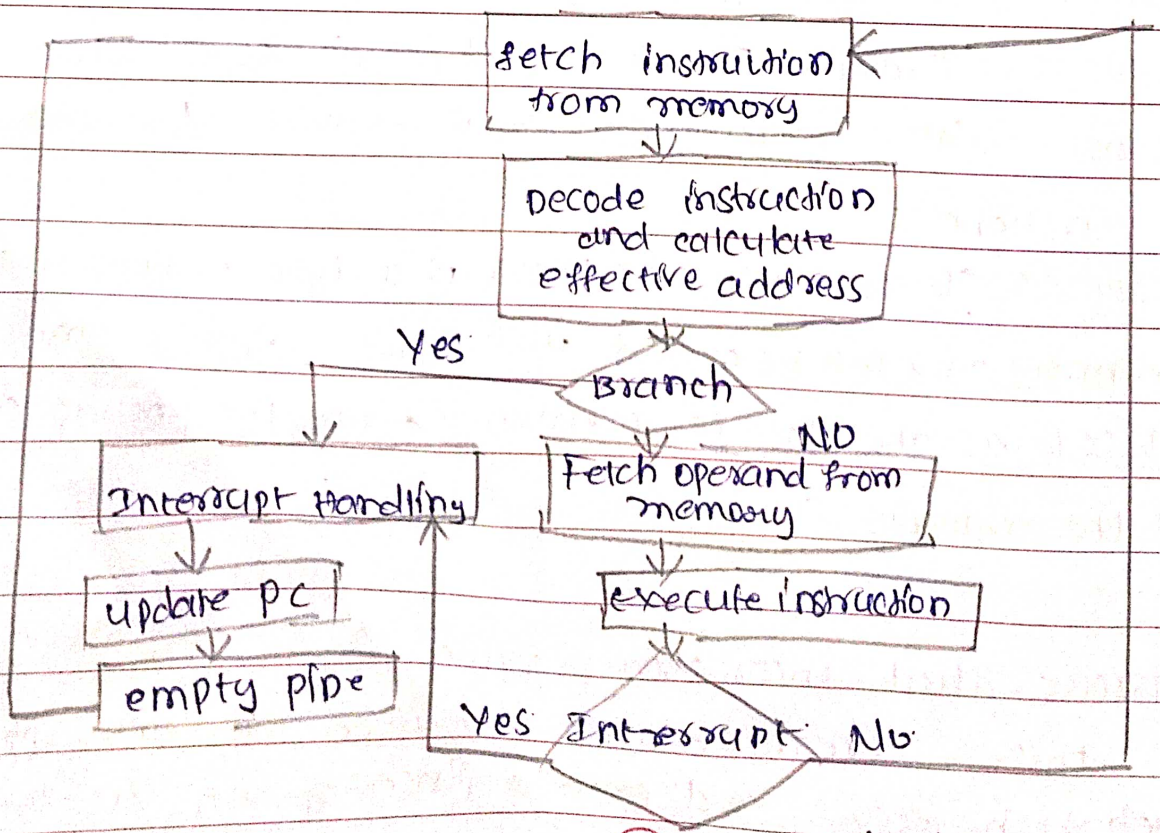
Mask ROMs: the data is written during the manufacturing of the memory chip.

5) Discuss about instruction pipeline.

A) In this a stream can be executed by overlapping ~~each~~ fetch, decode and execute phases of an instruction

This type of technique is used to increase the throughput of the computer system. The pipeline will be more effecient if the instruction cycle is divided into segments of equal duration. An instruction pipeline reads instruction from the memory while previous instructions are being executed in other segments of the pipeline. In the most general case computer needs to process each instruction in following sequence in steps.

1) Fetch the instruction from memory.
2) Decode instruction
3) Calculate the address
4) Fetch operands from memory.
5) execute the instruction.
6) Stores the result



Flowchart of instruction pipeline