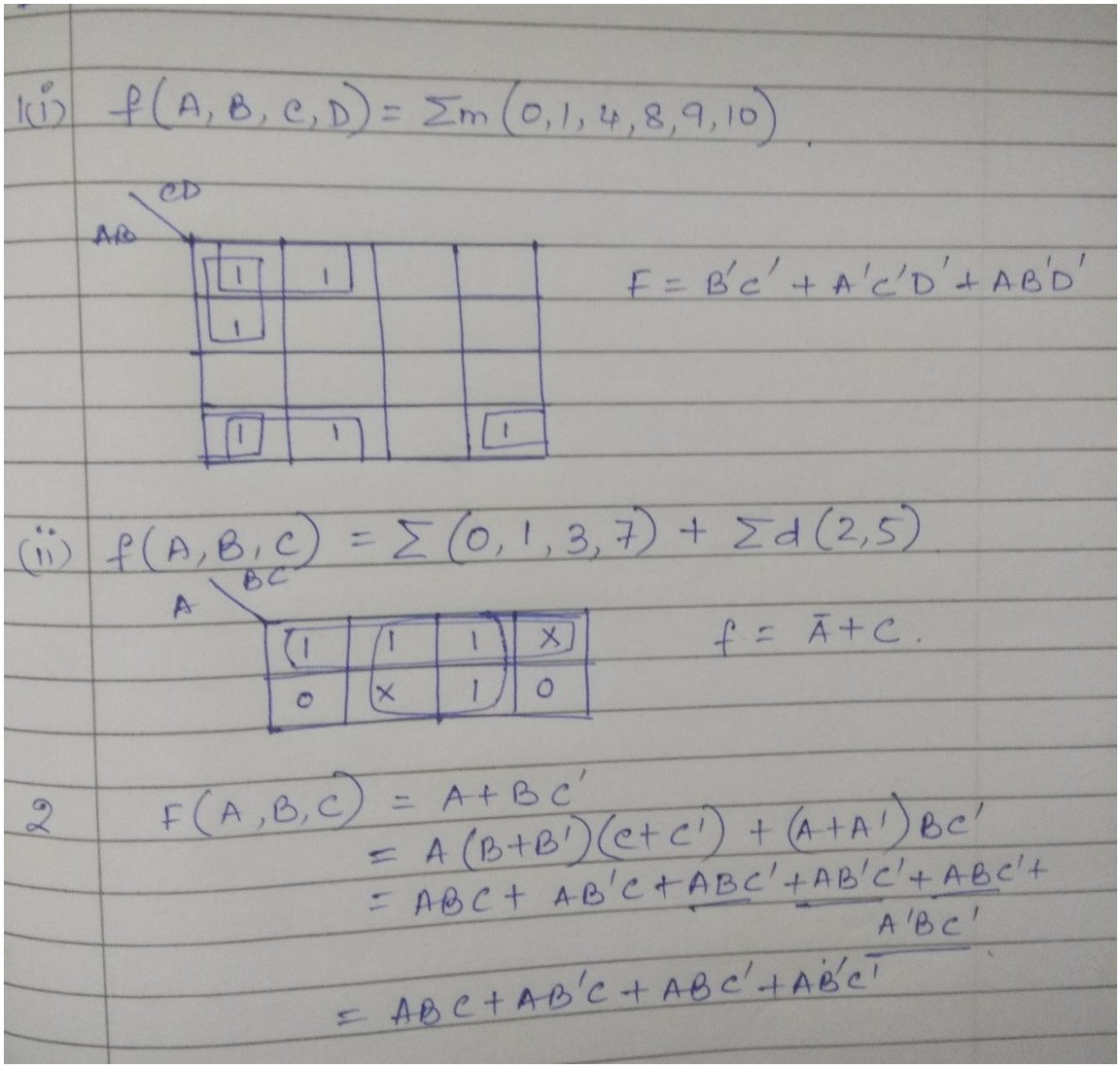
USN | | | | | | | | | | |

Internal Assessment Test 3 – Jan. 2020

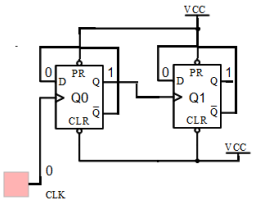
Sub:	Computer Organization					Sub Code:	18MCA15	Branch:	MCA
Date:	16/1/2020	Duration:	90 mins	Max Marks:	50	Sem / Sec:	I		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1	Reduce the following functions using K-map i. f(A,B,C,D)=∑m(0,1,4,8,9,10) ii. f(A,B,C)=∑(0,1,3,7) + d(2,5)	[10]	CO3	L1
	OR			
2	Represent the Boolean expression as SOP: F(A,B,C) = A+BC'	[10]	CO1	L1
3	Explain the different types of counters with circuit diagrams.	[10]	CO2	L1
	OR			
4	Explain with a circuit diagram the working of a 4-bit parallel subtractor.	[10]	CO1	L4
5	Why bus arbitration is required? Illustrate any two approaches of bus arbitration	[10]	CO2	L4
	OR			
6	What are condition codes? Explain various condition code flags.	[10]	CO4	L1
7	Write a note on Assembly Directives	[10]	CO4	L4
	OR			
8	Evaluate the expression Z = A*B+C*D using one-address, two-address and three-address instruction format.	[10]	CO2	L4
9	With a neat diagram explain 16x8 memory organization.	[10]	CO2	L2
	OR			
10	Explain the concept of address mapping in virtual memory	[10]	CO4	L2



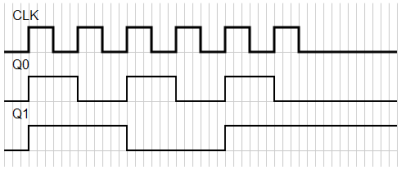
3. A counter is a digital sequential logic device that will go through a certain predefined states (for example counting up or down) based on the application of the input pulses. They are utilized in almost all computers and digital electronics systems
There are two main types of counters: Asynchronous and Synchronous counters.

Asynchronous Counters (Ripple Counter)
This type of counters has JK Flop-Flops arranged in a way that the output of one flip-flop feeds the clock of the following flip-flop

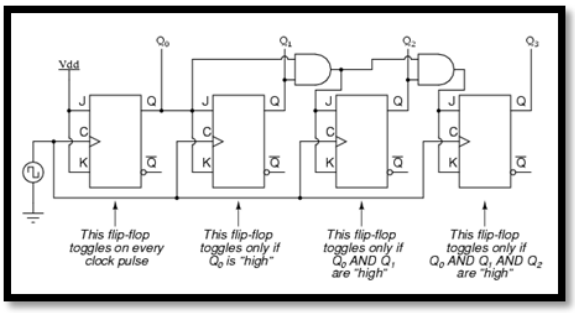


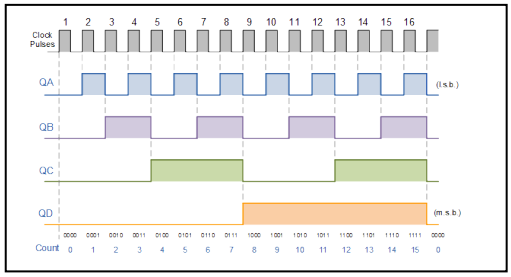
STATE TABLE

COUNT	Q1	Q0
3	1	1
2	1	0
1	0	1
0	0	0

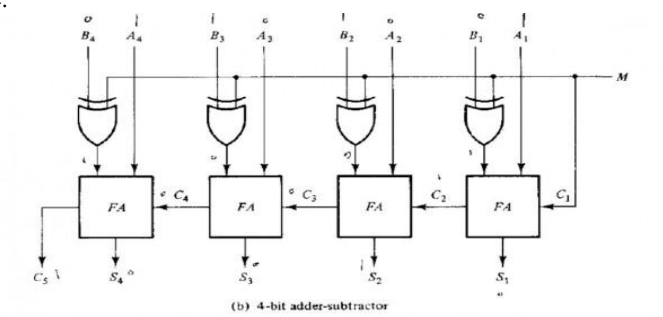


Synchronous Counter
This type of counters has each flip-flop clocked by the same clock source, thus eliminating the cumulative delay found in asynchronous counters





4.



(b) 4-bit adder-subtractor

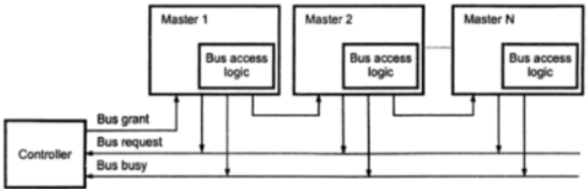
For M=1

5. The device that is allowed to initiate data transfers on the bus at any given time is called the bus master. In a computer system there may be more than one bus master such as processor, DMA controller etc.

- They share the system bus. When current master relinquishes control of the bus, another bus master can acquire the control of the bus.
- Bus arbitration is the process by which the next device to become the bus master is selected and bus mastership is transferred to it. The selection of bus master is usually done on the priority basis.
    - In centralized bus arbitration, a single bus arbiter performs the required arbitration. The bus arbiter may be the processor or a separate controller connected to the bus.
    - There are three different arbitration schemes that use the centralized bus arbitration approach. There schemes are:
      a. Daisy chaining
      b. Polling method
      c. Independent request

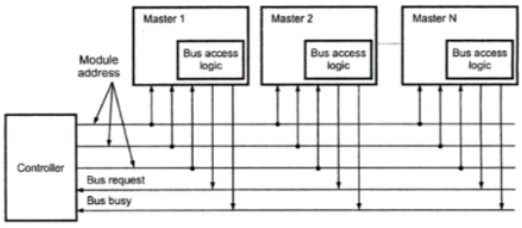
a) Daisy chaining

- The system connections for Daisy chaining method are shown in fig below.



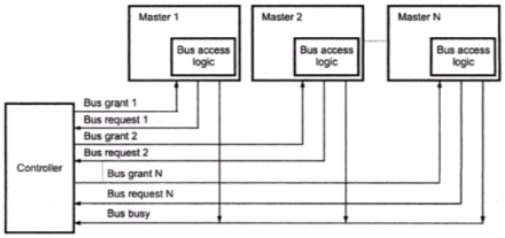
- It is simple and cheaper method. All masters make use of the same line for bus request.
- In response to the bus request the controller sends a bus grant if the bus is free.
- The bus grant signal serially propagates through each master until it encounters the first one that is requesting access to the bus. This master blocks the propagation of the bus grant signal, activities the busy line and gains control of the bus.
- Therefore any other requesting module will not receive the grant signal and hence cannot get the bus access.

b) Polling method



- The system connections for polling method are shown in figure above.
- In this the controller is used to generate the addresses for the master. Number of address line required depends on the number of master connected in the system.
- For example, if there are 8 masters connected in the system, at least three address lines are required.
- In response to the bus request controller generates a sequence of master address. When the requesting master recognizes its address, it activated the busy line ad begins to use the bus.

c) Independent request



- The figure below shows the system connections for the independent request scheme.
- In this scheme each master has a separate pair of bus request and bus grant lines and each pair has a priority assigned to it.

The built in priority decoder within the controller selects the highest priority request and asserts the corresponding bus grant signal

6. **Condition Code**: a group of bits indicating the condition of something inside a computer, often used to decide which instructions the computer will subsequently execute. A **status register**, **flag register**, or **condition code register** (**CCR**) is a collection of status flag bits for a processor. The status register is a hardware register that contains information about the state of the processor. Individual bits are implicitly or explicitly read and/or written by the machine code instructions executing on the processor. The status register lets an instruction take action contingent on the outcome of a previous instruction.

Flag	Name	Description
Z	Zero flag	Indicates that the result of an arithmetic or logical operation (or, sometimes, a load) was zero.
C	Carry flag	Enables numbers larger than a single word to be added/subtracted by carrying a binary digit from a less significant word to the least significant bit of a more significant word as needed. It is also used to extend bit shifts and rotates in a similar manner on many processors (sometimes done via a dedicated **X** flag).
S / **N**	Sign flag Negative flag	Indicates that the result of a mathematical operation is negative. In some processors,[1] the N and S flags are distinct with different meanings and usage: One indicates whether the last result was negative whereas the other indicates whether a subtraction or addition has taken place.
V / **O** / **W**	Overflow flag	Indicates that the signed result of an operation is too large to fit in the register width using two's complement representation.
P	Parity flag	Indicates whether the number of set bits of the last result is odd or even.
I	Interrupt flag	On some processors, this bit indicates whether interrupts are enabled or masked.[2] If the processor has multiple interrupt priority levels, such as the PDP-11, several bits may be used to indicate the priority of the current thread, allowing it to be interrupted only by hardware set to a higher priority. On other architectures, a bit may indicate that an interrupt is currently active, and that the current thread is part of an interrupt handler.

7. Assembly language consists of two types of statements viz.

Executable statements- These are the statements to be executed by the processor. It consists of the entire instruction set of 8086.

Assembler directives- These are the statements that direct the assembler to do something. As the name says, it directs the assembler to do a task.

The specialty of these statements is that they are effective only during the assembly of a program but they do not generate any code that is machine executable.

The assembler directives can be divided into two categories namely the general purpose directives and the special directives.

They are classified into the following categories based on the function performed by them- Simplified segment directives Data allocation directives Segment directives Macros related directives Code label directives Scope directives Listing control directives Miscellaneous directives

.CODE- This assembler directive indicates the beginning of the code segment. Its format is as follows: .CODE [name] The name in this format is optional.

.DATA- This directive indicates the beginning of the data segment.

.MODEL- This directive is used for selecting a standard memory model for the assembly language program. Each memory model has various limitations depending on the maximum space available for code and data.

.STACK- This directive is used for defining the stack. Its format is as follows: .STACK [size]

Define Byte [DB]- This directive defines the byte type variable.

Define Word [DW]- The DW directive defines items that are one word (two bytes) in length.

Define Double word [DD]- It defines the data items that are a double word (four bytes) in length.

Define Quad word [DQ]- This directive is used to tell the assembler to declare variable 4 words in length or to reserve 4 words of storage in memory.

Define Ten bytes [DT]- It is used to define the data items that are 10 bytes long.

ASSUME- The directive is used for telling the assembler the name of the logical segment which should be used.

END- This is placed at the end of a source and it acts as the last statement of a program. This is because the END directive terminates the entire program.

ALIGN- This directive will tell the assembler to align the next instruction on an address which corresponds to the given value.

LABEL- This directive assigns name to the current value of the location counter.

INCLUDE- This directive is used to tell the assembler to insert a block of source code from the named file into the current source module. This shortens the source code. Its format is: INCLUDE path: file name

8. 1-address Instructions:

LOAD	A	AC = M[A]
ADD	B	AC = AC + M[B]
STORE	T	M[T] = AC
LOAD	C	AC = M[C]
ADD	D	AC = AC + M[D]
MUL	T	AC = AC * M[T]
STORE	X	M[X] = AC

2-address Instructions:

MOV	R1, A	R1 = M[A]
ADD	R1, B	R1 = R1 + M[B]
MOV	R2, C	R2 = C
ADD	R2, D	R2 = R2 + D
MUL	R1, R2	R1 = R1 * R2
MOV	X, R1	M[X] = R1

3-address Instructions:

ADD	R1, A, B	R1 = M[A] + M[B]
ADD	R2, C, D	R2 = M[C] + M[D]
MUL	X, R1, R2	M[X] = R1 * R2

9. **Memory Connection to CPU**

RAM and ROM chips are connected to a CPU through the data and address buses. The low-order lines in the address bus select the byte within the chips and other lines in the address bus select a particular chip through its chip select inputs. The connection of memory chips to the CPU is shown in Fig. 12-4. This configuration gives a memory capacity of 512 bytes of RAM and 512 bytes of ROM. It implements the memory map of Table 12-1. Each RAM receives the seven low-order bits of the address bus to select one of 128 possible bytes. The particular RAM chip selected is determined from lines 8 and 9 in the address bus. This is done through a 2 × 4 decoder whose outputs go to the CS1 inputs in each RAM chip. Thus, when address lines 8 and 9 are equal to 00, the first RAM chip is selected. When 01, the second RAM chip is selected, and so on. The RD and WR outputs from the microprocessor are applied to the inputs of each RAM chip.

The selection between RAM and ROM is achieved through bus line 10. The RAMs are selected when the bit in this line is 0, and the ROM when the bit is 1. The other chip select input in the ROM is connected to the RD control line for the ROM chip to be enabled only during a read operation. Address bus lines 1 to 9 are applied to the input address of ROM without going through the decoder. This assigns addresses 0 to 511 to RAM and 512 to 1023 to ROM. The data bus of the ROM has only an output capability, whereas the data bus connected to the RAMs can transfer information in both directions.

The example just shown gives an indication of the interconnection complexity that can exist between memory chips and the CPU. The more chips that are connected, the more external decoders are required for selection among the chips. The designer must establish a memory map that assigns addresses to the various chips from which the required connections are determined.

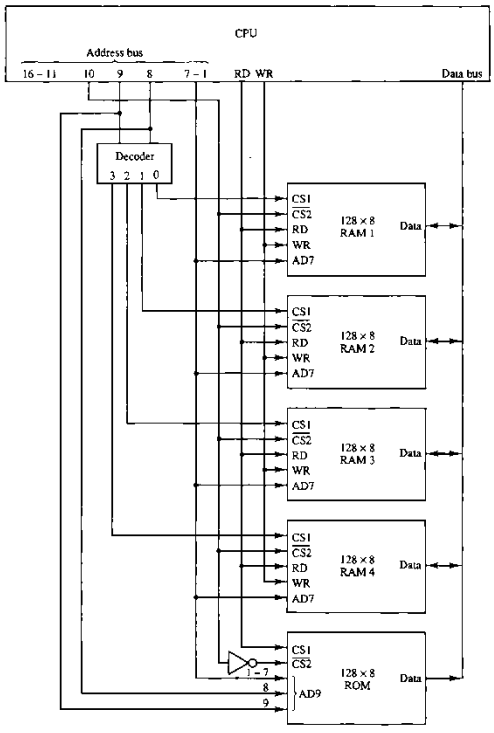
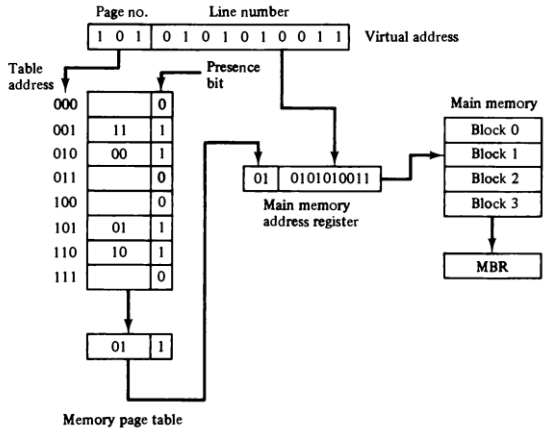


Figure 12-4 Memory connection to the CPU.

10.


Page no.
Line number
1 0 1
0 1 0 1 0 1 0 0 1 1
Virtual address
Table address
Presence bit
000
001 11 1
010 00 1
011 0
100 0
101 01 1
110 10 1
111 0
01 1
Memory page table
01 0101010011
Main memory address register
Main memory
Block 0
Block 1
Block 2
Block 3
MBR