B.TECH. EEC–402

FOURTH SEMESTER THEORY EXAMINATION 2009-10

COMPUTER ARCHITECTURE AND ORGANIZATION

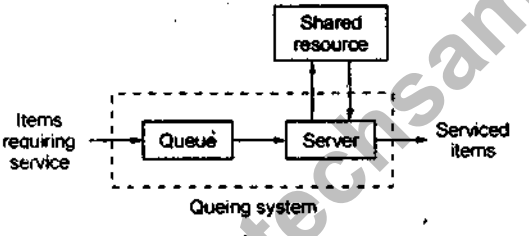
Time: 3 Hours *Total Marks: 100*

Note: *(i) Attempt all the questions.*
(ii) All questions carry equal marks.

1. Answer any two parts of the following: **(2 × 10 = 20)**

(a) What is M/M/1 queuing model of computer system? Explain.

Ans. The queuing model M/M/1 is the single-queue, single-server model.



The tests are activated or arrive at random times and are queued in memory until they can be processed (serviced) by the CPU on a first come first-served basis. The actual arrival and service rates vary randomly around the mean arrival rate (λ) and service rates (μ) respectively.

ρ denotes λ/μ and represents the mean utilization of the server. The M/M/1 case assumes a poisson arrival process.

$$\rho_f(t) = 1 - e^{-\lambda t}$$

In the same context, let $P_s(t)$ be the probability that the service required by a task is completed by the CPU in time t or less after its removal from the queue. Then the service process is characterized by

$$P_s(t) = 1 - e^{-\mu t}$$

The mean utilization of the server in an M/M/1 system

$$t_w = t_Q - \frac{1}{\mu} = \frac{1}{\mu(\mu - \lambda)}, \text{ where } t_Q = \frac{1}{\mu - \lambda}$$

and $\frac{1}{\mu}$ is the mean time if takes to service a task.

(b) Implement the following functions using PAL.

$W(A, B, C, D) = \Sigma(2, 12, 13)$

$X(A, B, C, D) = \Sigma(7, 8, 9, 10, 11, 12, 13, 14, 15)$

$Y(A, B, C, D) = \Sigma(0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15)$

$Z(A, B, C, D) = \Sigma(1, 2, 8, 12, 13)$

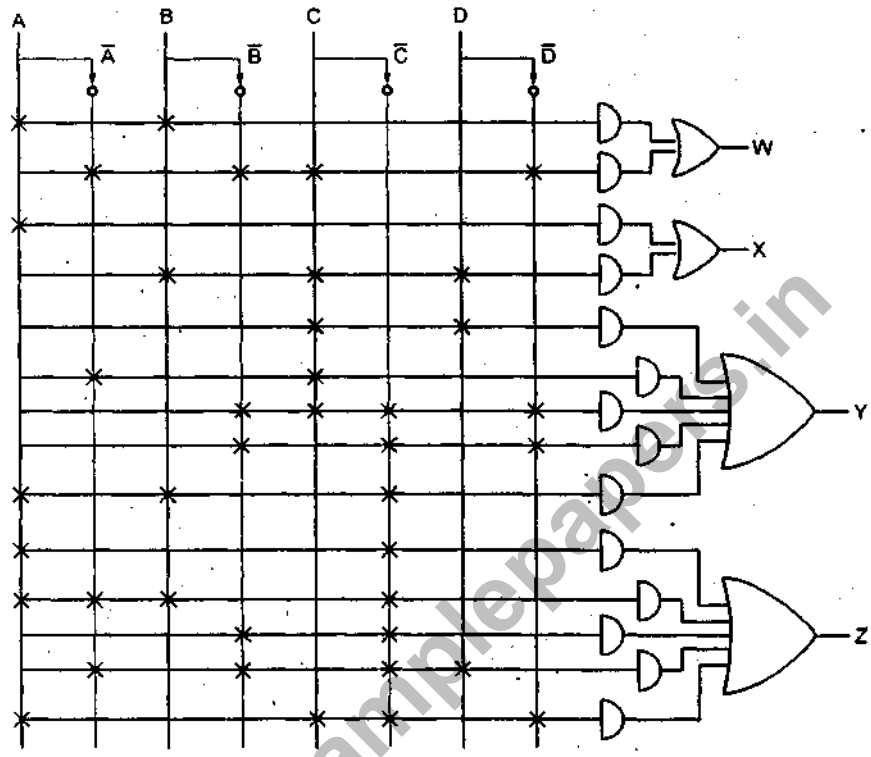
Ans. Solving using K-map, we get

$$W = AB\bar{C} + \bar{A}\bar{B}\bar{C}\bar{D}$$

$$X = A + BCD$$

$$Y = CD + \bar{A}C + \bar{B}C\bar{D} + \bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}$$

$$Z = A\bar{C}(\bar{D} + B) + \bar{A}\bar{B}(\bar{C}D + C\bar{D})$$



(c) List various design aspects in the processor level design.

Ans. The various aspects of the processor level design are

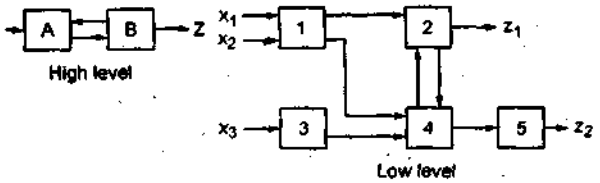
1. The total number of instructions that must be executed for a given task should be reduced.
2. The data types to be executed.
3. The various addressing modes to increase the execution speed.
4. CISC and RISC processors were the two categories to classify the processors.
5. Cache memory to provide faster access to instructions and data.
6. Pipelined processing to increase performance by allowing the processing of several instructions to be partially overlapped.
7. Super scalar processing to increase performance by allowing several instructions to be processed in parallel.

(d) What do you understand by design levels in the design of computer system?

Ans. The design of a complex system such as a computer is carried out. Three such levels are presently recognized

- The processor level, eg., CPU, memories I/O devices
- The register level eg., registers, counters, sequential circuits.
- The logic (gate) level e.g., logic gates, FFS.

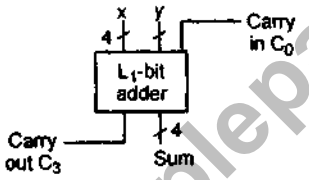
It is customary to refer to a design level as high or low, the more complex the components, the higher the level.



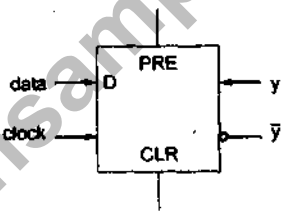
The design process might consist of the full three steps.

1. Specify the processor-level structure of the system
2. Specify the register level structure of each component type identified in step 1.
3. Specify the gate-level structure of each component type identified in step 2.

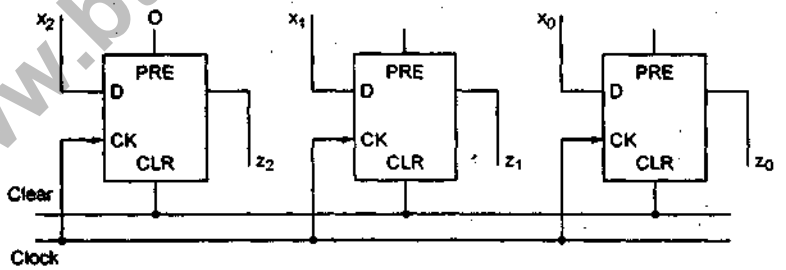
The gate level design is concerned with processing binary variables where possible values are restricted to the bits 0 and 1. The combinational circuits like adders are used.

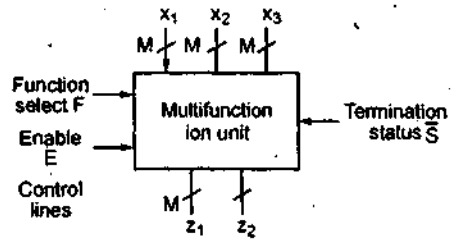


Flip-Flops



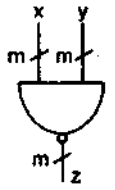
The registers level: At this level, the related information bits are grounded into ordered sets called words or vectors.



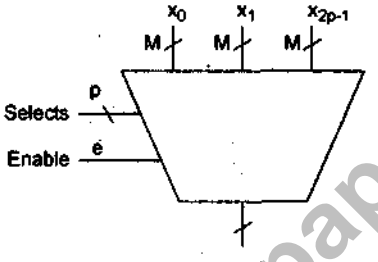


Word-based logical operations of this type are useful in some aspects of register level design.

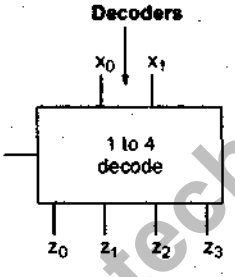
Two-input m-bit NAND word gate

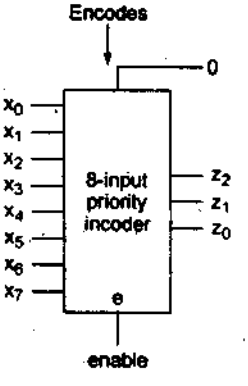


Multiplexers as function generators



Data out z





(e) Explain the design process at the register level.

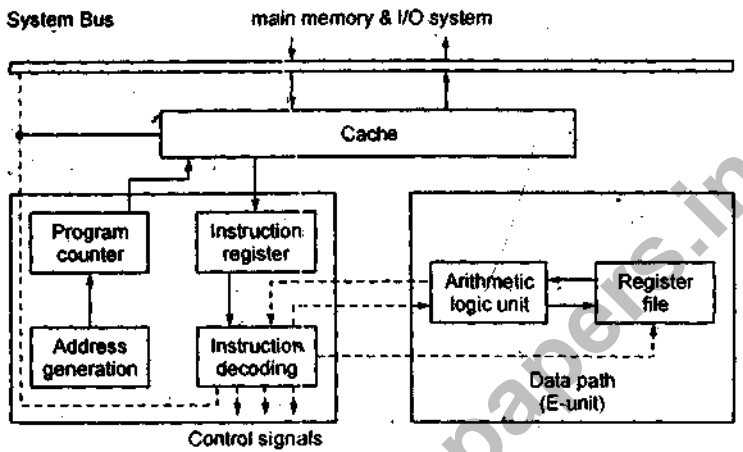
Ans. The design techniques for register-level system is as follows.

1. Define the desired behaviour by a set of sequence of register transfer operations. This constitutes an algorithm (AL) to be executed.
2. Analyse the AL to determine the types of components and the number of each type required for the datapath DP.
3. Make block diagram for DP using the components and make connections between the components. So that all data paths implied by AL are present and the given performance cost contraints are met.
4. Analyse AL and DP to identify the control signals needed.
5. Design a CU or DP that meets all the requirements of AL

   that the final design operates correctly meets all performance-cost goals.

(f) Given the internal structure of a CPU.

Ans.



The CPU is a synchronous sequential circuit whose clock period is the computer's basic unit of time.

PC-uses to hold the expected address of next instruction word.

The entire process of fetching, decoding and executing an instruction constitutes the CPU's **instruction cycle.**

Memories: Stores the programs and data required by the processors.

Input/Output devices: Act as data transducers to convert information from one physical representation to another.

2. Attempt any four parts: **(4 × 5 = 20)**

(a) Write a program to evaluate the arithmetic statement:

$$X = \frac{(A - B) + C * (D * E - F)}{G + H * K}$$

(i) Using an accumulator type computer with one address instruction.

(ii) Using a general register computer with three address instruction.

Ans. Three address Instructions

for G + H * K = D

MUL $R_1, H_1, K.$

ADD R_1, R_1, G

N = (A – B) + C * (D * E – F)

SUB R_1, A, B

ADD R_1, R_1, C

MUL R_2, D, E

SUB R_2, R_2, F

MUL R_2, R_2, C

ADD R_1, R_1, R_2

One address Instructions for D = G + H * K

LOAD H

MUL K

ADD G

STORE D

for N = (A – B) + C * (D * E – F)

LOAD A

SUB B

STORE T

LOAD D

MUL E

SUB F

MUL C

ADD T

STORE N

(b) What are the different instruction formats used in the computer architecture?

Ans. Data formats

In selecting the number representation the following factors should be taken into account.

1. **TYPES of NUMBERS** to be represented e.g., integers, real numbers, complex numbers
2. The **RANGE of VALUES** to be encountered
3. The **PRECISION** of the **NUMBER**
4. THE **COST** of **HARDWARE** required to store and process the numbers

Instruction formats

In selecting the instruction format(s) the following factors should be considered.

1. The addressability and addressing modes.
3. The ease of decoding.
4. Type of instruction field (fixed or variable)
5. The cost of hardware required to decode and execute instructions.

0-. 1-, or 3-addressable instruction formats

Instruction and data format

CA-IV-D and IF-6

Instruction formats

	OP-CODE	**Mode**	**Rn**		
1-Address	10 Bits	3 Bits	3 Bits		
	OP-CODE	**Mode**	**Rn**	**Mode**	**Rn**
2-ADDRESS	4 BITS	3 BITS	3 BITS	3 BITS	3 BITS
		SRC		DST	

3. BRANCH

OP-CODE	OFFSET
8 BITS	8 BITS

BRANCH ADDRESS = [Updates PC] + 2 × OFFSET

Data Format

16 BITS

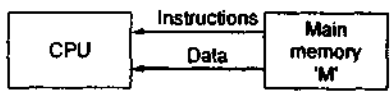
2's COMPLEMENT

8 BITS	8 BITS
CHARACTER	CHARACTER

(c) What are the major functions of a processor? Explain them with the help of a flow chart.

Ans. The primary function of the CPU and other instruction set processors is

1. To execute sequences of instructions (programs), stored in an external main memory.
2. External communication.

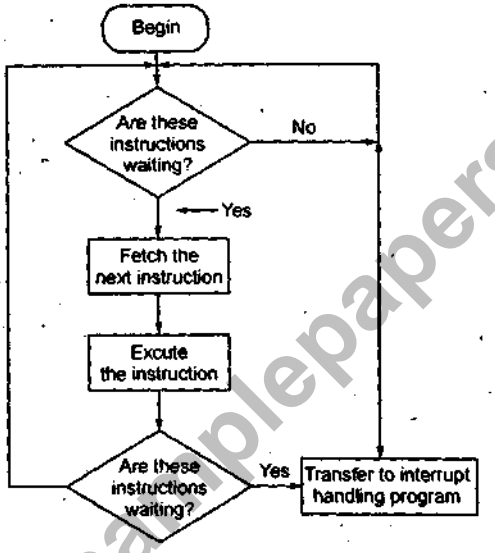


The CPU is significantly faster than the main memory i.e., it can read or write from CPU, register 5 to 10 times faster than it can read from or write to memory.

3. User and supervisor modes: A user or application program handles a specific application.

A supervisor program manages various routine aspects of the computer system on behalf of its users.

5. Performs fetch operation and execute operation.



(d) Register A holds the 8-bit binary 11011001. Determine the B operand and the logtic micro operation to be performed in order to change the value in A to:

(i) 01101101

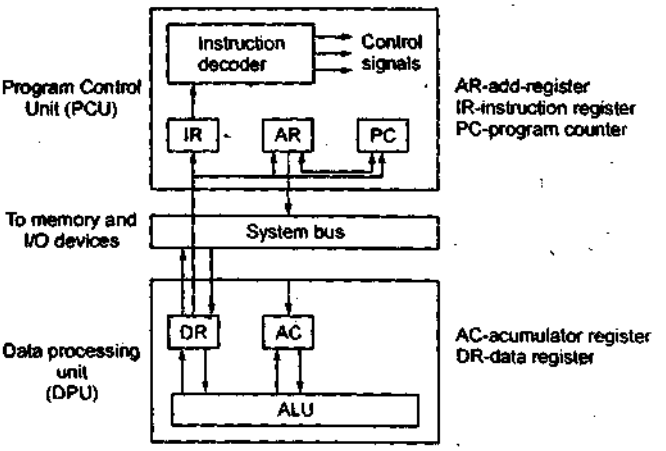
(ii) 11111101

Ans. (i) ADD of 00100100 to 1101100 yields 01101101 in the accumulator

(ii) OR operation of 11011001 with 00000100 yields 11111101

(e) Draw and explain the architecture of an accumulator based CPU.

Ans.



In accumulator based CPU, the accumulator played a central role. It is used to store an input or output operand (result) in the execution of many instructions.

Here, the instructions are fetched by the PCU, whose main register is the PC. They are executed in the data processing unit CPU, which contains an n-bit ALU and two data register AC and DR.

Most instructions perform operations of the form

$$X_1 = f(X_1, X_2)$$

where X_1 and X_2 denote a CPU register AC, DR or PC or an external memory address.

The instruction involves instruction of

(a) data transfer

(b) data processing

(c) Program control.

(f) A computer has 32-bit instructions and 12-bit addresses. If there are 250 two address instructions, how many one-address instructions can be formulated?

Ans. To read a data the address and data lines should be in bytes.

∴ 12 address lines 2^4 = 4 bits are required = 2 bytes

for two address instruction

$2 \times 8 + 2 = 18$ bytes instruction.

for one address instructions

$1 \times 8 + 2 = 10$ bytes, one-add. Instruction can be formed.

3. Attempt any two parts: (2 × 10 = 20)

(a) Explain the Booths algorithm for Multiplication. Perform (+15) × (–13) using Booth algorithm. Assume 5-bit registers that hold signed numbers.

Ans. Booth's Algorithm (Multiplication)

It gives a procedure for multiplying binary integers in signed - 2's complement representation. It operates on the fact that strings of 0's in the multiplier require no addition but just shifting and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as $2^{k+1} - 2^m$.

It requires examination of the multiplier bits and shifting of the partial product. Prior to the shifting the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to follow in rules.

(i) The multiplicand is subtracted from the partial product upon encountering the first least significant '1' in string of 1's in the multiplier.

(ii) The multiplicand is added to the partial product upon encountering the first '0' in a string of 0's in the multiplier.

(iii) The partial product does not change when the multiplier bit is identical to the previous multiplier bit.

Multiplication of (+15) and (–13) is also shown

(+15) 0 1 1 1 1

(–13) → 2's comp. 1 0 0 1 1

0	1	1	1	1	(+15)
0	+1	+1	–1	–1	(–13)

1	1	1	1	1	1	0	0	0	1
0	0	0	0	0	1	1	1	1	·
0	0	0	0	1	1	1	1	·	·
0	0	0	1	1	1	1	·		
0	0	0	0	0	0	·			
1	0	0	1	1	0	0	0	0	1	1

↓ ↓

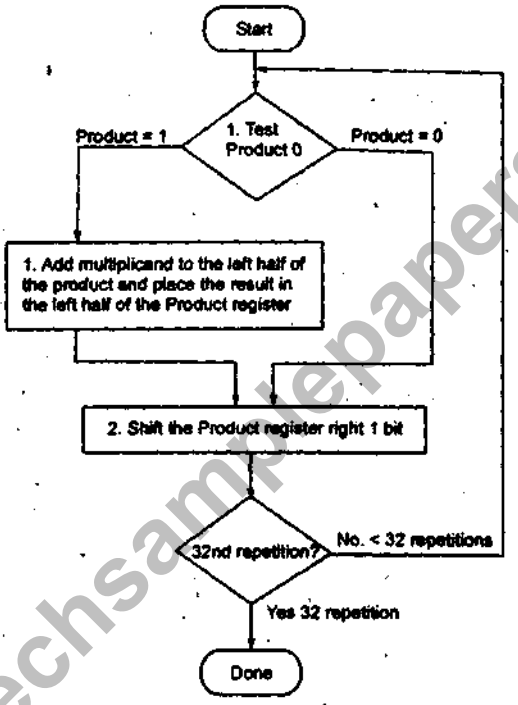
Signed bit multipled value

1→ (–)ve

Multiplied value → 195] –195

(b) Draw and explain the flow chart for floating point divisioa.

Ans.



(c) Draw a space-time diagram for a six-segment pipeline showing the time it takes to process eight tasks. A non pipeline system takers 50 ns to process a task. The same task can be processed in a six-segment pipeline with a clock cycle of 10 ns. Determine the speed-up ratio of the pipeline for 100 tasks. What is the maximum speed up that can be achieved?

Ans. A space-time diagram for an m-stage pipeline has the form of an $m \times n$ grid, where 'n' is the no. of clock cycles to complete the processing of some sequence of N instructions of interest.

S_1	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	–	–	–	–	–	–	–
S_2	–	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	–	–	–	–	–	–
S_3	–	–	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	–	–	–	–	–
S_4	–	–	–	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	–	–	–	–
S_5	–	–	–	–	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	–	–	–
S_6	–	–	–	–	–	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	–	–
S_7	–	–	–	–	–	–	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	–
S_8	–	–	–	–	–	–	–	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Time (clock cycles)

$$\text{Speed up} = \frac{T(1)}{T(m)} = \frac{10ns}{50ns} = 0.2$$

S(m)

m-stages of pipeline

$$S(m) = m \times E(m) = 100 \times \frac{56}{128} = 43.75$$

(maximum speed up)

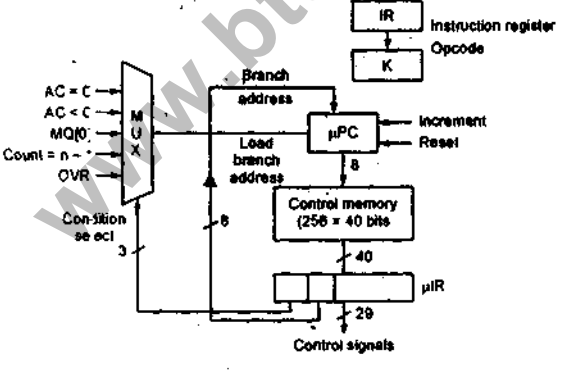
($m = 100$ given)

4. Attempt any two parts: (2 × 10 = 20)

(a) Draw and explain microinstruction sequencing organization.

Ans. A microprogram sequencer is a general purpose building block for microprogrammed control units. It contains a microprogram counter as well as the logic needed for conditional branching and transferring control between microprograms. A control unit can be constructed from three components a RAM or ROM is used as the control memory, a microinstruction register and a microprogram sequencer.

The microinstruction register can be implemented as a two-stage pipeline to allow micro-instruction fetching and execution to be overlapped.



(b) Explain the sequence of operations needed to perform following CPU functions.

(i) Storing a word in the memory

(ii) Register transfers

Ans. Storing a word into memory

The procedure of writing a word into memory location is similar to that for reading one from memory. The only difference is that the data word to be written is first loaded into the MDR, the write command is issued.

An example assumes that the data word to be stored in the memory is in register R1 and that the memory address is in register R2. The memory write operation requires the following sequence:

1. Mar ← [R2]
2. MDR ← [R1]
3. Write
4. Wait for MFC

In this case step 1 and step 2 are independent and so they can be carried out in any order. In fact, step 1 and 2 can be carried out simultaneously, if this is allowed by the architecture, that is, if these two data transfer (memory address and data) do not use the same data path.

In case of both memory read and memory write operation, the total time duration depends on wait for the MFC signal, which depends on the speed of the memory module.

These is a scope to improve the performance of the CPU, if CPU is allowed to perform some other operation while waiting for MFC signal. During the period, CPU can perform some other instructions which do not require the use of MAR and MDR.

(ii) Register Transfer Operation:

Register transfer operations enable data transfer between various blocks connected to the common bus of CPU. We have several registers inside CPU and it is needed to transfer information from one register to another. As for example during memory write operation data from appropriate register must be moved to MDR. Since the input output lines of all the register are connected to the common internal bus, we need appropriate input output gating. The input and output gates for register R_i are controlled by the signal R_{in} and R_{out} respectively.

Thus, when $R_{i\ in}$ set to 1 the data available in the common bus is loaded into R_i. Similarly when, R_{iout} is set to 1, the contents of the register R_i are placed on the bus. To transfer data from one register to other register, we need to generate the appropriate register gating signal.

For example, to transfer the contents of register R1 to register R2, the following actions are needed:

• Enable the output gate of register R1 by setting R1 out to 1.

This places the contents of R1 on the CPU bus.

• Enable the input gate of register R2 by setting R2 out to.

This loads data from the CPU bus into the register R2.

Performing the arithmetic or logic operation:

Generally ALU is used inside CPU to perform arithmetic and logic operation. ALU is a combinational logic circuit which does not have any internal storage.

Therefore, to perform any arithmetic or logic operation (say binary operation) both the input should be made available at the two inputs of the ALU simultaneously. Once both the inputs are available then appropriate signal is generated to perform the required operation.

We may have to use temporary storage (register) to carry out the operation in ALU.

The sequence of operations that have to carried out to perform one ALU operation depends on the organization of the CPU. Consider an organization in which one of the operand of ALU is stored in some temporary register Y and other operand is directly taken from CPU internal bus. The result of the ALU operation is stored in another temporary register Z.

Therefore, the sequence of operations to add the contents of register R1 to register R2 and store the result in register R3 should be as follows:

1. R1 out Y_{in}
2. R2out, Add, Z_{in}
3. Z_{out}, $R3_{in}$

In step 2 of this sequence the contents of register R2 are gated to the bus, hence to input–B of the ALU which is directly connected to the bus. The contents of register Y are always available at input A of ALU. The function performed by the ALU depends on the signal applied to the ALU control lines. In this example, the Add control line of ALU is set to 1, which indicate the addition operation and the output of ALU is the sum of the two numbers at input A and B. The sum is loaded into register Z, since the input gate is enabled (Z_{in}). In step 3, the contents of register Z are transferred to the destination register R3.

(c) What is microprogramming and microprogrammed control unit?

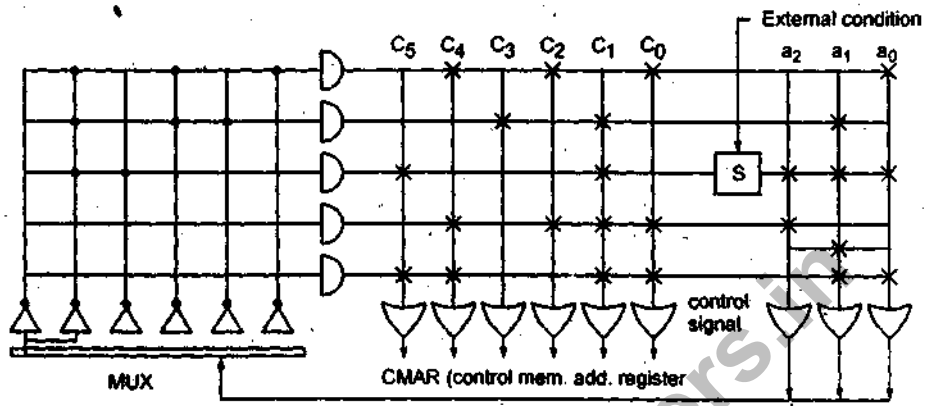
Ans. Microprogramming is a method of control-unit design in which the control signal selection and sequencing information is stored in a ROM or RAM called a 'control memory [CM]. The control signals to be activated at any time are specified by a microinstruction, which is fetched from CM. Each microinstruction also explicitly or implicitly specifies the next microinstruction to be used, thereby providing the necessary information for micro-operation sequencing. A set of related, micro-instructions forms a microprogram.

Microprograms can be changed relatively easily by changing the contents of CM, hence microprogramming yields control units that are more flexible than their hardwired counter parts.

Microprogramming continues to be used in pentium and 680 *XO*.

The control unit organization of a microinstruction has two parts:

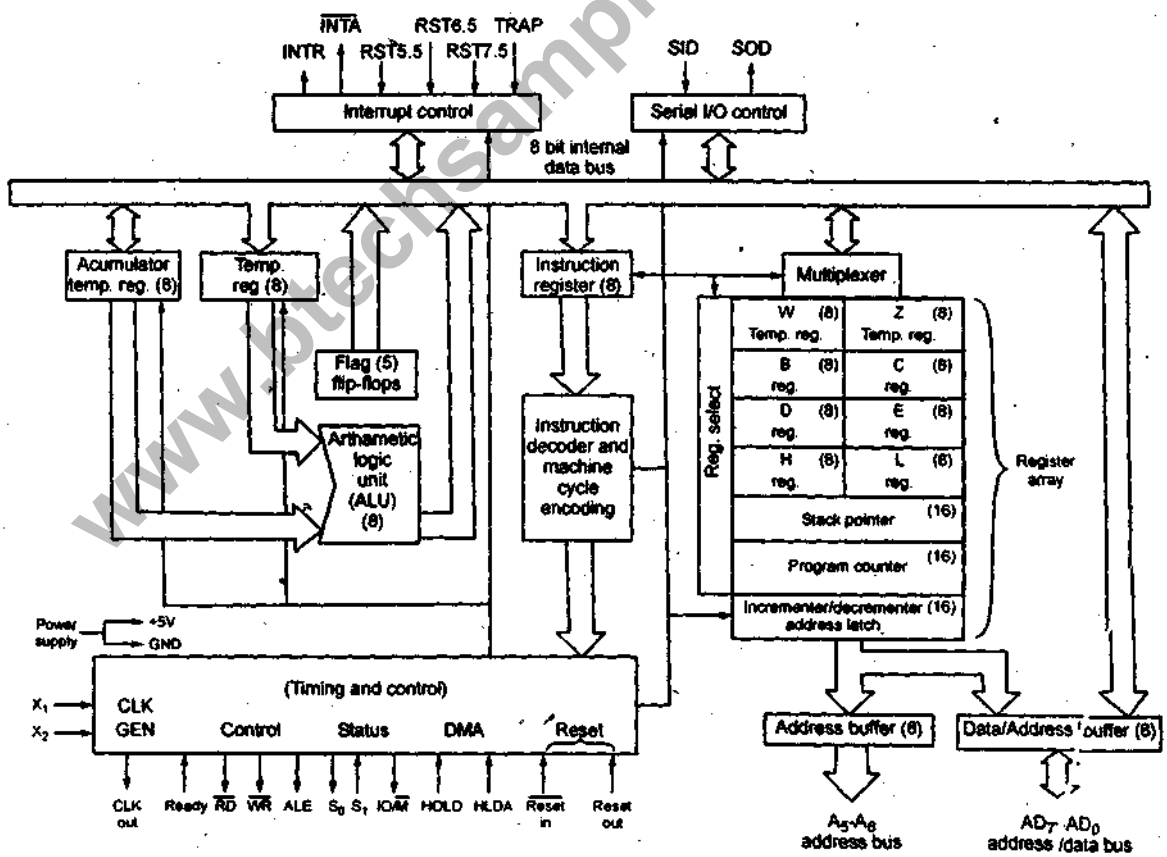
1. A set of control fields that specify the control signals to be activated.
2. The address fields that contains the address in CM of the next microinstruction to be executed.



5. Attempt any two parts: **(2 × 10 = 20)**

***(a)* Draw the functional block diagram of microprocessor 8085 and explain it in brief.**

Ans. The functional block diagram or architecture of 8085 Microprocessor is very important as it gives the complete details about a Microprocessor.



Address Bus:

- The address bus is a group of 16 lines generally identified as A_0 to A_{15}.
- The address bus is unidirectional: bits flow in one direction-from the MPU to peripheral devices.
- The MPU uses the address bus to perform the first function: identifying a peripheral or a memory location.

Data Bus:

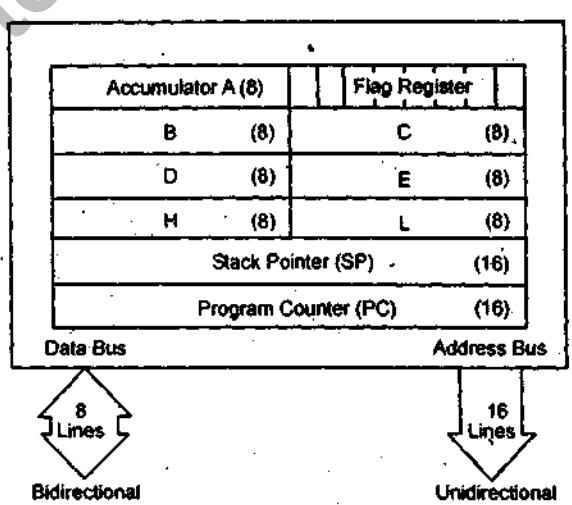
- The data bus is a group of eight lines used for data flow.
- These lines are bi-directional-data flow in both directions between the MPU and memory and peripheral devices.
- The MPU uses the data bus to perform the second function: transferring binary information.
- The eight data lines enable the MPU to manipulate 8-bit data ranging from 00 to FF (28 = 256) numbers).
- The largest number that can appear on the data bus is 11111111.
- The control bus carries synchronization signals and providing timing signals.
- The MPU generates specific control signals for every operation it performs. These signals are used to identify a device type with which the MPU wants to communicate.

Registers of 8085:

- The 8085 have six general-purpose registers to store 8-bit data during program execution.
- These registers are identified as B, C, D, E, H, and L.
- They can be combined as register pairs-BC, DE, and HL-to perform some 16-bit operations.

Accumulator (A):

- The accumulator is an 8-bit register that is part of the arithmetic/logic unit (ALU).
- This register is used to store 8-bit data and to perform arithmetic and logical operations.
- The result of an operation is stored in the accumulator.



Flags:

- The ALU includes five flip-flops that are set or reset according to the result of an operation.
- The microprocessor uses the flags for testing the data conditions.
- They are Zero (Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags. The most commonly used flags are Sign, Zero and Carry.

The bit position for the flags register is,

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

1. Sign Flag (S):

After execution of any arithmetic and logical operation, if D7 of the result is 1, the sign flag is set. Otherwise it is reset.

D7 is reserved for indicating the sign; the remaining is the magnitude of number.

If D7 is 1, the number will be viewed as negative number. If D7 is 0, the number will be viewed as positive number.

2. Zero Flags (z):

If the result of arithmetic and logical operation is zero, then zero flags is set otherwise it is reset.

3. Auxiliary Carry Flag (AC):

If D3 generates any carry when doing any arithmetic and logical operation, this flag is set. Otherwise it is reset.

4. Parity Flag (P):

If the result of arithmetic and logical operation contains even number of 1's then this flag will be set an if it is odd number of 1's it will be reset.

5. Carry Flag (CY):

If any arithmetic and logical operation result any carry then carry flag is set otherwise it is reset.

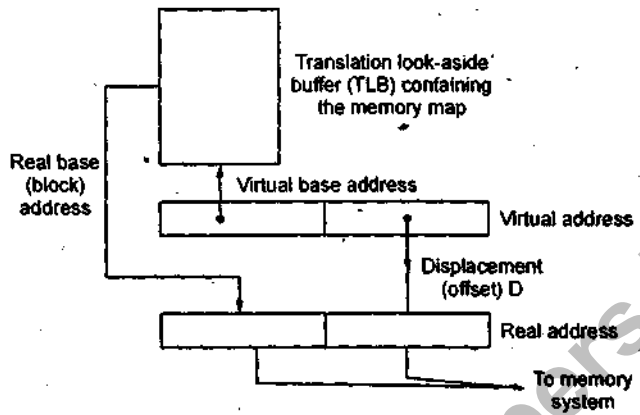
(*b*) Draw and explain the virtual memory organization.

Ans. Virtual memory is the term applied when the main and secondary memories appear to a user program like a single, large and directly addressable memory.

Reasons for using virtual memory.

1. To free user programs from the need to carry out storage allocation and to permit efficient sharing of available memory space among different users.
2. To make programs independent of the configuration and capacity of the physical memory present for their execution.

3. To achieve the very low access time and cost per bit.



A memory system is addressed by a set of logical or virtual addresses derived from identifiers or implicits specified in an object program.

A set of physical or real address R identifies the fixed physical store locations in each memory unit. An efficient and flexible mechanism to implement address mapping of the form f: $V \rightarrow R$ is the key to successful design of multilevel memory.

(c) What are the various Mapping Techniques used in the cache organization?

Ans. Cache Mapping Techniques: There is a critical trade off in cache performance that has led to the creation of the various cache mapping techniques. In order for the cache to have good performance you want to maximize both of the following:

- **Hit Ratio:** You want to increase as much as possible the likelihood of the cache containing the memory addresses that the processor wants. Otherwise, you lose much of the benefit of caching because there will be too many misses.

- **Search Speed:** You want to be able to determine as quickly as possible if you have scored a hit in the cache. Otherwise, you lose a small amount of time on every access, hit or miss, while you search the cache.

Now let's look at the three cache types.

- **Direct Mapped Cache:** The direct mapped cache is the simplest form of cache and the easiest to check for a hit. Since there is only one possible place that any memory location can be cached, there is nothing to search; the line either contains the memory information we are looking for, or it doesn't.

Unfortunately, the direct mapped cache also has the worst performance, because again there is only one place that any address can be stored. Let's look again at our 512 KB level 2 cache and 64 MB of system memory. As you recall this cache has 16,384 lines (assuming 32-byte cache lines) and so each one is shared by 4,096 memory addresses. In the absolute worst case, imaging that the processor needs 2 different addresses (call them X and Y) that both map to the same cache line, in alternating sequence (X, Y, X, Y). This could happen in a small loop if you were unlucky. The processor will load X from memory and store it in cache. Then it will look in the cache for Y, but Y uses the same cache line as X, so it won't be there. So Y is loaded from memory, and stored in the cache for future use. But then the processor requests X, and looks in the cache only to find

Y. This conflict repeats over and over. The net result is that the hit ratio here is 0%. This is a worst case scenario, but in general the performance is worst for this type of mapping.

• **Fully Associative Cache:** The fully associative cache has the best hit ratio because any line in the cache can hold any address that needs to be cached. This means the problem seen in the direct mapped cache disappears, because there is no dedicated single line that an address must use.

However, this cache suffers from problems involving searching the cache. If a given address can be stored in any of 16,384 lines, how do you know where it is? Even with specialized hardware to do the searching, a performance penalty is incurred. And this penalty occurs for all accesses to memory, whether a cache hit occurs or not, because it is part of searching the cache to determine a hit. In addition, more logic must be added to determine which of the various lines to use when a new entry must be added (usually some form of a "least recently used" algorithm is employed to decide which cache line to use next). All this overhead adds cost, complexity and execution time.

• **N-Way Set Associative Cache:** The set associative cache is a good compromise between the direct mapped and set associative caches. Let's consider the 4-way set associative cache. Here, each address can be cached in any of 4 places. This means that in the example described in the direct mapped cache description above, where we accessed alternately two addresses that map to the same cache line, they would now map to the same cache set instead. This set has 4 lines in it, so one could hold X and another could hold Y. This raises the hit ratio from 0% to near 100%! Again an extreme example, of course. As for searching, since the set only has 4 lines to examine this is not very complicated to deal with, although it does have to do this small search, and it also requires additional circuitry to decide which cache line to use when saving a fresh from memory. Again, some form of LRU (least recently used) algorithm is typically used.