

B.Tech.

Fourth Semester Examination

Computer Architecture & Organization (CSE-210-F)

Note : Attempt only five questions. All questions carry equal marks.

Q. 1. Explain the followings :

20

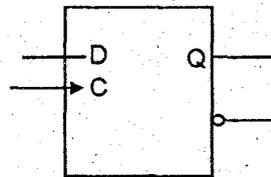
(i) D flip-flop

(ii) Edge-triggered flip-flop

(iii) Full Adder

(iv) Bidirection shift register.

Ans. (i) **D Flip-Flop** : The D (data) flip-flop is a slight modification of the SR flip-flop. An SR flip-flop is converted to a D flip-flop by inserting an inverter between S and R and assigning the symbol D to the single input. The D input is sampled during the occurrence of a clock transition from 0 to 1. If $D = 1$, the output of the flip-flop goes to the 1 state, but if $D = 0$, the output of the flip-flop goes to the 0 state.



(a) Graphic Symbol

D	Q (t + 1)
0	0
1	1

Clear to 0
Set to 1

(b) Characteristic Table

From the characteristic table we note that the next state $Q(t+1)$ is determined from the D input. The relationship can be expressed by a characteristic equation.

$$Q(t+1) = D$$

(ii) **Edge-Triggered Flip-Flop** : The most common type of flip-flop used to synchronize the state change during a clock pulse transition is the edge-triggered flip-flop. In this type of flip-flop, output transitions occur at a specific level of the clock pulse. Some edge-triggered flip-flops cause a transition on the rising edge of the clock-signal (positive-edge transition) and other cause a transition on the falling edge (negative-edge transition).

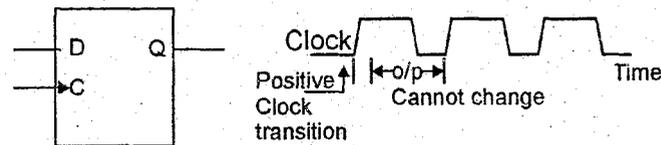


Fig. (a) Positive edge-triggered D flip-flop

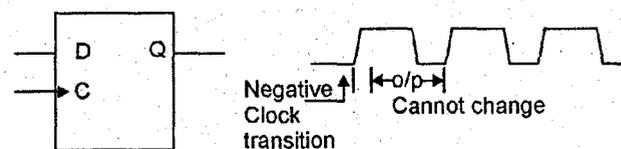


Fig.(b) Negative-edge triggered D flip-flop

In figure (a) the value in the D input is transferred to the Q output when the clock makes a positive transition. The output cannot change when the clock is in the 1 level, in the 0 level or in a transition from the 1 level to the 0 level. The effective +ve clock transition includes a minimum time called the set up time in which the D input must remain at a constant value before the transition, and a definite time called the hold time in which the D input must not change after the positive transition.

(iii) **Full Adder** : A full-adder is a combinational circuit that forms the arithmetic sum of three input bits. It consists of three inputs and two outputs. Two of the input variables, denoted by x and y, represent the two significant bits to be added. The third input, z, represents the carry from the previous lower significant position.

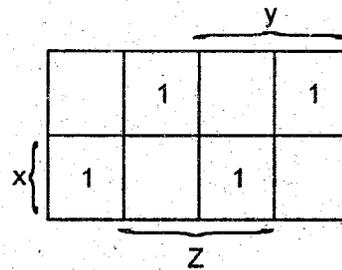
Truth Table for Full Adder :

Input			Output	
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Boolean expression for the full adder

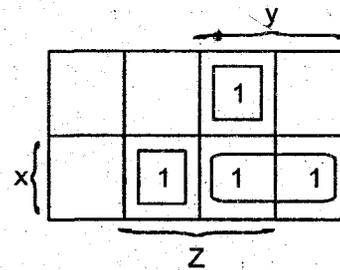
$$S = x \oplus y \oplus z$$

$$C = xy + (x \oplus y)z$$



$$S = x'y'z + x'yz' + xy'z' + xyz$$

$$= x \oplus y \oplus z$$



$$C = xy + xz + yz$$

$$= xy + (x'y + xy')z$$

(iv) **Bidirection Shift Register** : A register capable of shifting in one direction only is called a unidirectional shift register. A register that can shift in both directions is called a bidirectional shift register.

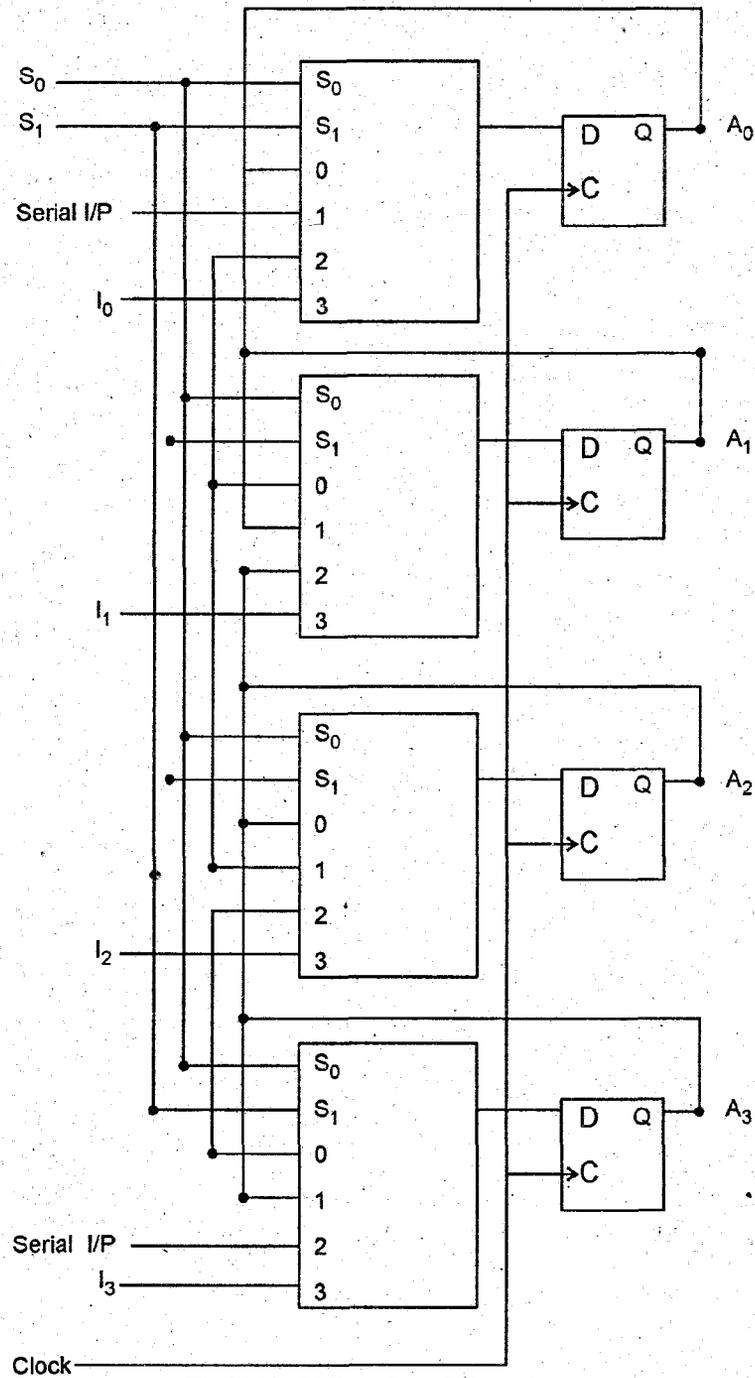


Fig. 4-Bit bidirection shift register

Function Table for Bidirection Shift Register :

S_1	S_0	Register Operation
0	0	No change
0	1	Shift right (down)
1	0	Shift left (up)
1	1	Parallel load

In a 4-bit bidirectional shift register, each stage consists of D flip-flop and a 4×1 multiplexer. The two selection input S_1 & S_0 select one of the multiplexer data inputs for the D flip-flop. The operation mode of bidirection shift register is given in above function table.

Q. 2. Briefly discuss the followings :

20

(i) Static & dynamic RAM & it's organization

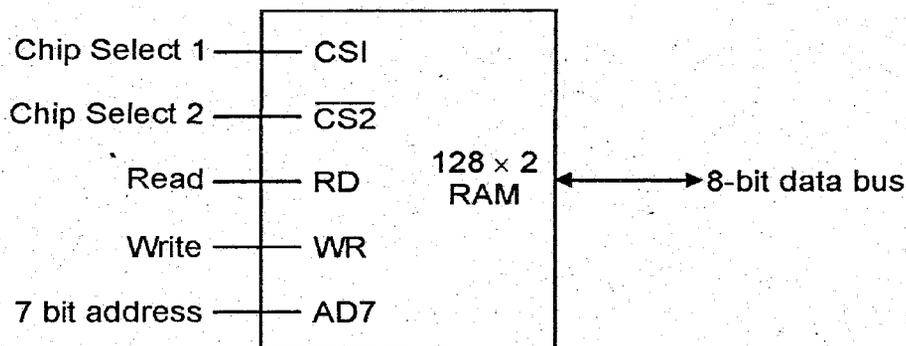
(ii) Amdahl's law

Ans. (i) Static & Dynamic RAM & it's Organization : The main memory is the central storage unit in a computer system. It is a relatively large and fast memory used to store programs & data during the computer operation. The principle technology used for the main memory is based on semiconductor integrated circuits. Integrated circuit RAM chips are available in two possible operating modes, static and dynamic. The static RAM consists essentially of internet flip-flops that store the binary information. The stored information remain valid as long as power is applied to the unit.

The dynamic RAM stores the binary information in the form of electric charges that are applied to capacitors. The capacitors are provided inside the chip by MOS transistors. The stored charge on the capacitors tend to discharge with time and capacitors must be periodically recharged by refreshing the dynamic memory. Refreshing is done by cycling through the words every few milliseconds to restore the decaying charge. The dynamic RAM offers reduced power consumption and larger storage capacity in a single memory chip. The static RAM is easier to use and has shorter read and write cycles.

RAM chips are available in a variety of sizes. If the memory needed for the computer is larger than the capacity of one chip, it is necessary to combine a number of chips to form the required memory size.

The block diagram of a RAM chip is shown in figure. The capacity of the memory is 128 words of eight bit (one byte) per word.



Block Diagram

CS1	CS2	RD	WR	Memory Function	State of data bus
0	0	×	×	Inhibit	High-impedance
0	1	×	×	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	×	Read	Output data from RAM
1	1	×	×	Inhibit	High-impedance

(ii) **Amdahl's Law** : This law is used for find out the speed up, S.

T = Estimated time taken to run a program.

f = Time taken by a uniprocessor to complete the task.

f is fraction of T.

1-f → Time taken by multiprocessors to complete the task.

1-f is also a fraction of T.

Let the number of multiprocessors be 'n' estimated time = T

$$\text{Calculated time} = ft + \frac{(1-f)T}{n}$$

Speed up,

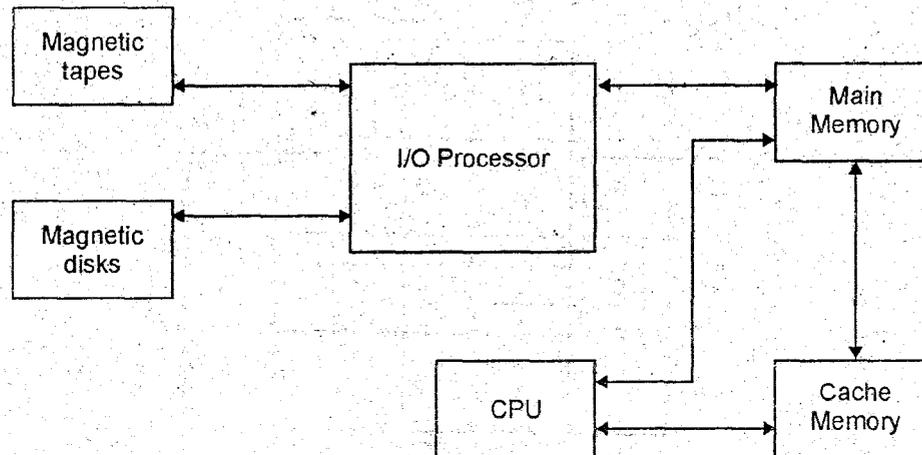
$$\begin{aligned}
 S &= \frac{\text{estimated time}}{\text{calculated time}} \\
 &= \frac{T}{fT + \frac{(1-f)T}{n}} \Rightarrow \frac{T}{T \left[f + \frac{(1-f)}{n} \right]} \\
 S &= \frac{1}{f + \frac{(1-f)}{n}} \Rightarrow \frac{1}{fn + (1-f)} \\
 &\Rightarrow \frac{n}{fn + 1 - f} \\
 &= \frac{n}{1 + f(n-1)} \\
 S &= \frac{n}{1 + f(n-1)}
 \end{aligned}$$

Q. 3. Why we need various types of memories computer system ? Explain the memory hierarchy.

Ans. The memory unit is an essential component in any digital computer, since it is needed for storing programs and data. A very small computer with a limited application may be able to fulfill its intended task without the need of additional storage capacity. Most general-purpose computer would run more efficiently if they were equipped with additional storage beyond the capacity of the main memory. Moreover, most computer users accumulate and continue to accumulate large amounts of data-processing software. Not all accumulated information is needed by the processor at the same time. Therefore, it is more economical to use low-cost storage devices to serve as a backup for storing the information that is not currently used by the CPU. The memory unit that communicates directly with the CPU is called the main memory. Devices that provide backup storage are called auxiliary memory. The

most common auxiliary memory devices used in computer systems are magnetic disks and tapes. They are used for storing system programs, large data files and other backup information only programs and data currently needed by the processor reside in main memory. All other information is stored in auxiliary memory and transferred to main memory when needed.

The total memory capacity of a computer can be visualized as being a hierarchy components. The memory hierarchy system consists of all storage devices employed in a computer system from the slow but high-capacity auxiliary memory to a relatively faster main memory, to an even smaller and faster cache memory accessible to the high-speed processing logic.



Memory hierarchy in a computer system

Figure illustrates the components in a typical memory hierarchy. At the bottom of the hierarchy are the relatively slow magnetic tapes used to store removable files. Next are the magnetic disks used as backup storage. The main memory occupies a central position by being able to communicate directly with the CPU and with auxiliary memory devices through an I/O processor. When programs not residing in main memory are needed by the CPU, they are brought in from auxiliary memory. Programs not currently needed in main memory are transferred into auxiliary memory to provide space for currently used programs and data.

A special very high-speed memory called a cache is sometimes used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate. The cache memory is employed in computer systems to compensate for the speed differential between main memory access time and processor logic. A technique used to compensate for the mismatch in operating speed is to employ an extremely fast, small cache between the CPU and main memory whose access time is close to processor logic clock cycle time.

Q. 4. Simplify the followings expressions in :

(i) SOP

(ii) POS

(a) $X'Z + Y'Z + YZ' + XY$

(b) $AC' + B'D + A'CD + ABCD$

Ans. (i) SOP : SOP means sum-of-product. In SOP, the product terms are AND terms and the sum denotes the ORing of these terms.

(ii) POS : POS means product-of-sum. It is convenient to obtain the algebraic expression for the function in a product-of-sums form. The sums are OR terms and the product denotes the ANDing of these terms. With a minor modification, a product-of-sums form can be obtained from a map.

We wish to simplify the following Boolean function in both sum-of-products form and product-of-sums form :

		C			
	AB	00	01	11	10
CD	00	1	1	0	1
	01	0	1	0	0
	11	0	0	0	0
	10	1	1	0	1

$$F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$$

The 1's marked in the map, represent the minterms that produce a 1 for the function. The square marked with 0's represent the minterms not included in F & therefore denote the complement of F. Combining the square with 1's gives the simplified function in sum-of-products form

$$F = B' D' + B' C' + A' C' D$$

If the squares marked with 0's are combined, we obtain the simplified complemented function :

$$F' = AB + CD + BD'$$

Taking the complement of F' , we obtain the simplified function in product-of-sums form :

$$F = (A' + B')(C' + D')(B' + D)$$

Q. 5. Explain the instruction cycle flowchart for a con and explain all the memory reference instructions.

Ans. **Instruction Cycle** : A program residing in the memory unit of the computer consists of a sequence of instructions. The program is executed in the computer by going through a cycle for each instruction. Each instruction cycle in turn is subdivided into a sequence of subcycles or phases. In the basic computer each instruction cycle consists of the following phases :

- (i) Fetch an instruction from memory.
- (ii) Decode the instruction.
- (iii) Read the effective address from memory if the instruction has an indirect address.
- (iv) Execute the instruction.

Upon the completion of step 4, the control goes back to step 1 to fetch, decode and execute the next instruction. This process continues indefinitely unless a HALT instruction is encountered.

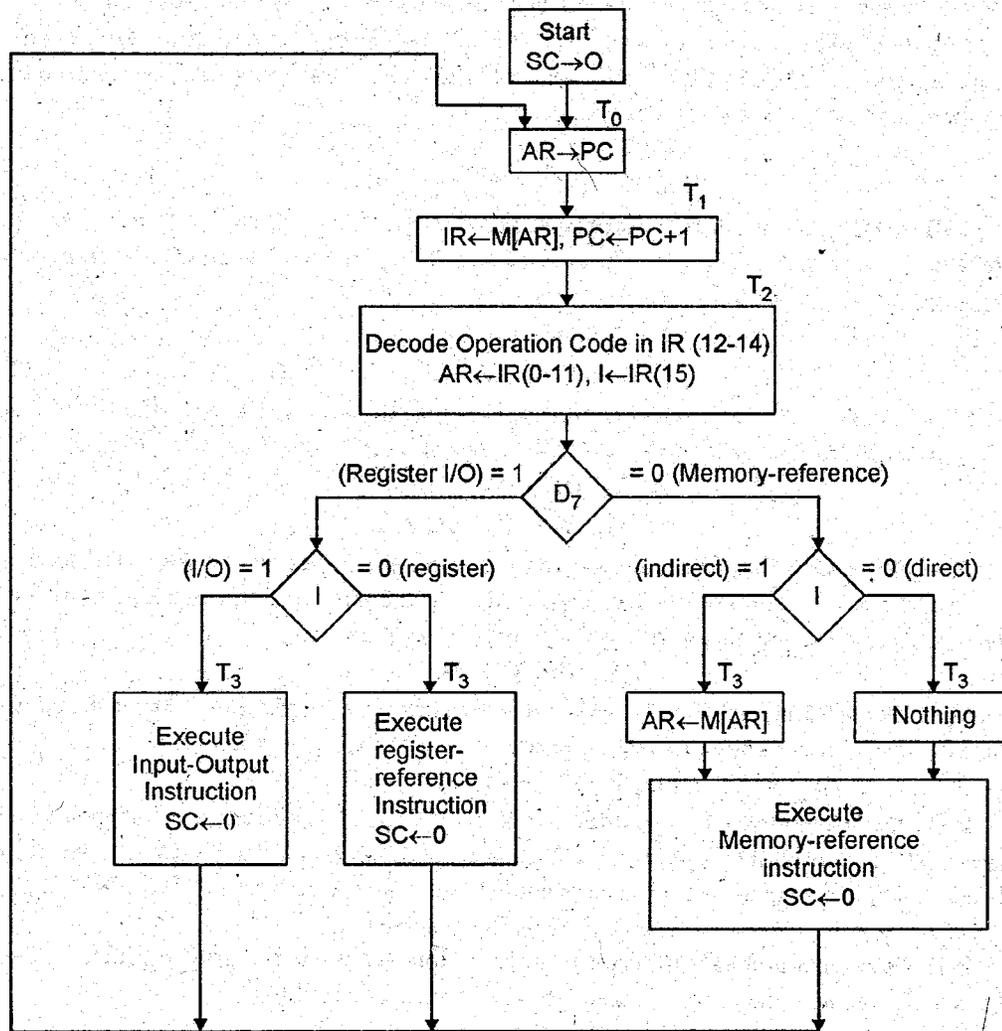
Fetch and Decode : Initially, the program counter PC is loaded with the address of the first instruction in the program. The sequence counter SC is cleared to 0, providing a decoded timing signal T_0 . After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T_0, T_1, T_2 and so on. The microoperations for the fetch and decode phases can be specified by the following register transfer statements

$T_0 : AR \leftarrow PC$

$T_1 : IR \leftarrow M[AR], PC \leftarrow PC + 1$

$T_2 : D_0, \dots, D_7 \leftarrow \text{Decode IR}(12-14), AR \leftarrow IR(0-11), I \leftarrow IR(15)$

Only AR is connected to the address inputs of memory, it is necessary to transfer the address from PC to AR during the clock transition associated with.



Memory-Reference Instructions

Symbol	Operation Decoder	Symbolic Description
AND	D ₀	$AC \leftarrow AC \wedge m[AR]$
ADD	D ₁	$AC \leftarrow AC + M[AR], E \leftarrow C_{out}$
LDA	D ₂	$AC \leftarrow M[AR]$
STA	D ₃	$M[AR] \leftarrow AC$
BUN	D ₄	$PC \leftarrow AR$
BSA	D ₅	$M[AR] \leftarrow PC, PC \leftarrow AR + 1$
ISZ	D ₆	$M[AR] \leftarrow M[AR] + 1, \text{ If } M[AR] + 1 = 0 \text{ then } PC \leftarrow PC + 1$

Table gives the list of seven memory reference instructions. The decoded output D_i for $i = 0, 1, 2, 3, 4, 5, \& 6$ from the operation decoder that belongs to each instruction is included in the table.

AND to AC: This is an instruction that performs the AND logic operation on pairs of bits in AC and the memory word specified by the effective address. The result of the operation is transferred to AC. The microoperation that execute this instruction are :

$$D_0 T_4 : DR \leftarrow M[AR]$$

$$D_0 T_5 : AC \leftarrow AC \wedge DR, SC \leftarrow 0$$

ADD to AC: This instruction address the content of the memory word specified by the effective address, to the value of AC. The sum is transferred into AC and the output carry C_{out} is transferred to the E (extended accumulator) flip-flop.

$$D_1 T_4 : DR \leftarrow M[AR]$$

$$D_1 T_5 : AC \leftarrow AC + DR$$

LDA: Load to AC: This instruction transfers the memory word specified by the effective address to AC. The microoperations needed to execute this instruction are

$$D_2 T_4 : DR \leftarrow M[AR]$$

$$D_2 T_5 : AC \leftarrow DR, SC \leftarrow 0$$

STA: Store AC: This instruction stores the content of AC into the memory word specified by the effective address. Since the output of AC is applied to the bus and the data input of memory is connected to the bus. We can execute this instruction with one microoperation :

$$D_3 T_4 : M[AR] \leftarrow AC, SC \leftarrow 0$$

BUN: Branch Unconditionally: This instruction transfers the program to the instruction specified by the effective address. The instruction is executed with one microoperation :

$$D_4 T_4 : PC \leftarrow AR, SC \leftarrow 0$$

BSA: Branch and Save Return Address: This instruction is useful for branching to a portion of the program called a subroutine or procedure. When executed, the BSA instruction stores the address of the next instruction in sequence instruction a memory location specified by the effective address.

$$M[AR] \leftarrow PC, PC \leftarrow AR + 1$$

ISZ: Increment and Skip if Zero: This instruction increment the word specified by the effective address and if the increment value is equal to 0, PC is incremented by 1.

$D_6 T_4 : DR \leftarrow M[AR]$

$D_6 T_5 : DR \leftarrow DR + 1$

$D_6 T_7 : M[AR] \leftarrow DR$

If

$(DR = 0)$

Then

$(PC \leftarrow PC + 1), SC \leftarrow 0$

Q. 6. (a) What are the basic difference between an instruction, a call subroutine instruction program interrupt ?

Ans. Instruction : A computer instruction is a binary code that specifies a sequence of microoperations for the computer. Instruction codes together with data are stored in memory. The computer reads each instruction from memory and places it in a control register.

An instruction code is a group of bits that instruct, the computer to perform a specific operation. It is usually divided into parts, each having its own particular interpretation. The most basic part of an instruction code is its operation part.

Subroutine : A set of common instructions that can be used in a program many times is called a subroutine. Each time that a subroutine is used in the main part of the program, a branch is executed to the beginning of the subroutine. After the subroutine has been executed, a branch is made back to the main program.

The procedure for branching to a subroutine and returning to the main program is referred to as a subroutine linkage. The BSA instruction performs an operation commonly called subroutine call. When a subroutine is called, the main program must transfer the data it wishes the subroutine to work with.

The running time of input and output programs is made up primarily of the time spent by the computer in waiting for the external device to set its flag. The waiting loop that checks the flag keeps the computer occupied with a task that wastes a large amount of time. This waiting time can be eliminated if the interrupt facility is used to notify the computer when a flag is set. The advantage of using the interrupt is that the information transfer is initiated upon request from the external device.

Q. 6. (b) Give five examples of external interrupt examples of internal interrupt.

Ans. External Interrupts : External interrupt come from input-output (I/O) devices, from a timing device, from a circuit monitoring the power supply or from any other external source. Examples that cause external interrupts are I/O device requesting transfer of data, I/O device finished transfer of data, elapsed time of an event, or power failure. Timeout interrupt may result from a program that is in an endless loop and thus exceeded its time allocation. Power failure interrupt may have as its service routine a program that transfers the complete state of the CPU into a non-destructive memory in the few milliseconds before power ceases.

Internal Interrupts : Internal interrupts arise from illegal or erroneous use of an instruction or data. Internal interrupts are also called traps. Examples of interrupt caused by internal error conditions are register overflow, attempt to divide by zero, an invalid operation code, stack overflow and protection violation.

Q. 7. Why does DMA have priority over CPU who request a memory transfer ? Explain DMA details.

Ans. Direct Memory Access (DMA) : The transfer of data between a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU. Removing the CPU from the path

and letting the peripheral device manage buses directly would improve the speed of transfer. This transfer technique is called direct memory access (DMA). During DMA transfer, the CPU is idle and has no control of the memory buses. A DMA controller takes over the buses to manage the transfer directly between the input/output device and memory.

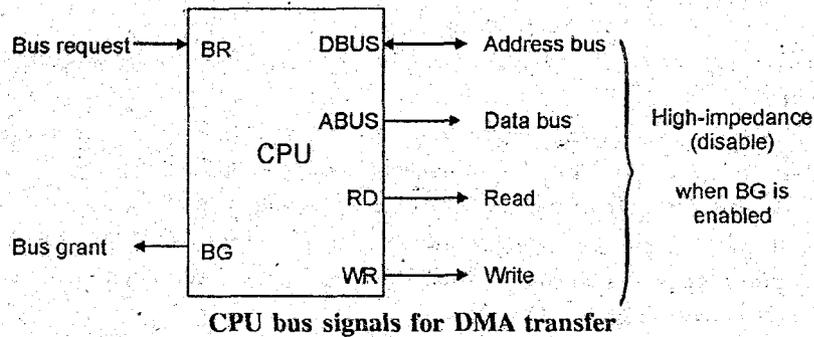


Figure shows two control signals in the CPU that facilitate the DMA transfer. The bus request (BR) input is used by the DMA controller to request the CPU to relinquish control of the buses. When this input is active, the CPU terminates the execution of the current instruction and places the address bus, the data bus and the read and write lines into a high-impedance state. The high-impedance state behaves like an open circuit, which means that the output is disconnected and does not have a logic significance. The CPU activates the bus grant (BG) output to inform the external DMA that the buses are in high-impedance state. The DMA that originated the bus request can now take control of the buses to conduct memory transfers without processor intervention when the DMA terminates the transfer; it disables the bus request line. The CPU disables the bus grant, takes control of the buses and returns to its normal operation.

When the DMA takes control of the bus system, it communicates directly with the memory. The transfer can be made in several ways. In DMA burst transfer, a block sequence consisting of a number of memory words is transferred in a continuous burst while the DMA controller is master of the memory buses. This mode of transfer is needed for fast devices such as magnetic disks, where data transmission cannot be stopped or slowed down until an entire block is transferred. An alternative technique called cycle stealing allows the DMA controller to transfer one data word at a time, after which it must return control of the buses to the CPU. The CPU merely delays its operation for one memory cycle to allow the direct memory I/O transfer to "steal" one memory cycle.

Q. 8. (a) What is cache coherence problem? Multiprocessor resolves it?

Ans. Cache Coherence: The primary advantage of cache is its ability to reduce the average access time in uniprocessors. When the processor finds a word in cache during a read operation, the main memory is not involved in the transfer. If the operation is to write, there are two commonly used procedures to update memory. In the write-through policy, both cache and main memory are updated with every write operation. In the write-back policy, only the cache is updated and the location is marked so that it can be copied later into main memory.

In a shared memory multiprocessor system, all the processors share a common memory. In addition, each processor may have a local memory, part or all of which may be a cache. The compelling

reason for having separate caches for each processor is to reduce the average access time in each processor. The same information may result in a number of copies in some caches and main memory. To ensure the ability of the system to execute memory operations correctly, the multiple copies must be kept identical. This requirement imposes a cache coherence problem. A memory scheme is coherent if the value returned on a load instruction is always the value given by the latest store instruction with the same address. Without a proper solution to the cache coherence problem, caching cannot be used in bus-oriented multiprocessors with two or more processors.

Cache coherence problems exist in multiprocessor with private caches because of the need to share writable data. Read-only data can safely be replicated without cache coherence enforcement mechanisms.

Q. 8. (b) Differentiate between RISC & CISC.

Ans. CISC: A computer with a large number of instructions is classified as a complex instruction set computer, abbreviated CISC.

Major characteristics of CISC architecture are :

- (i) A large number of instructions-typically from 100 to 250 instructions.
- (ii) Some instructions that perform specialized tasks and are used infrequently.
- (iii) A large variety of addressing modes-typically from 5 to 20 different modes.
- (iv) Variable-length instruction formats.
- (v) Instructions that manipulate operands in memory.

RISC: A number of computer designers recommended that computer use fewer instructions with simple constructs so they can be executed much faster within the CPU without having to use memory as often. This type of computer is classified as a reduced instruction set computer or RISC.

RISC characteristics:

- (i) Relatively few instruction.
- (ii) Relatively few addressing modes.
- (iii) Memory access limited to load and store instruction.
- (iv) All operations done within the registers of the CPU.
- (v) Fixed-length, easily decoded instruction format.
- (vi) Single-cycle instruction execution.
- (vii) Hardwired rather than microprogrammed control.