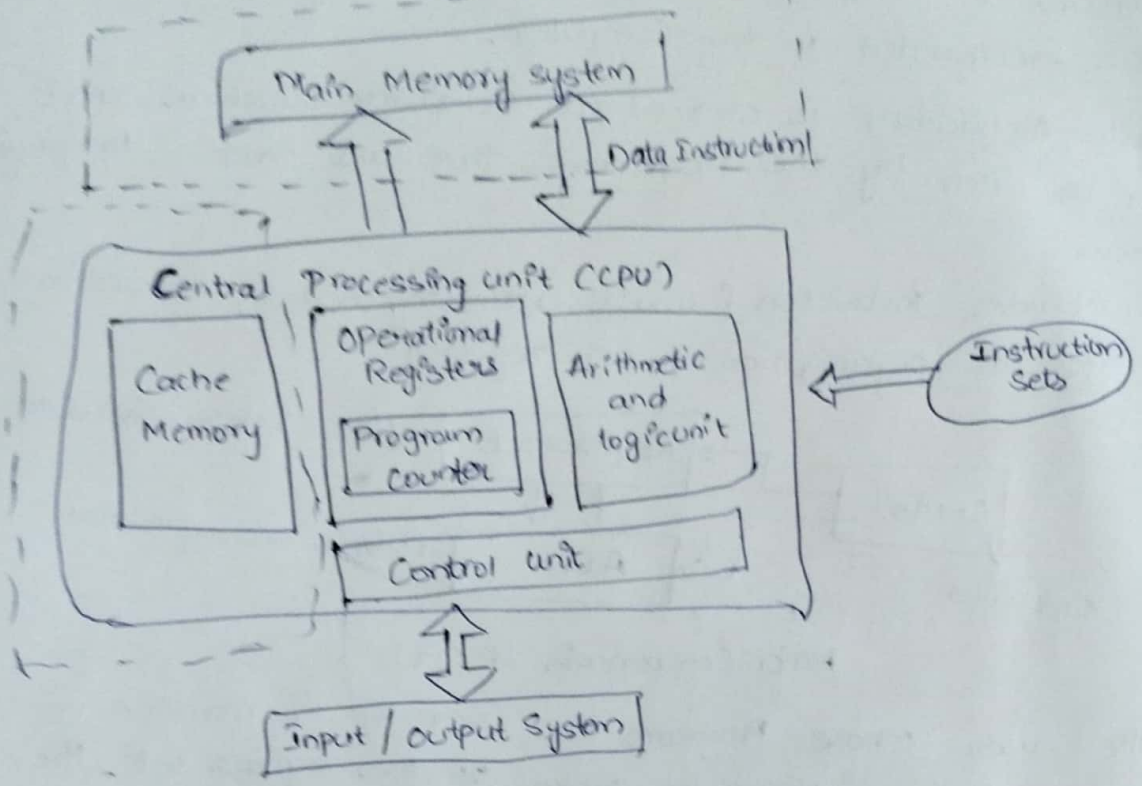
* Introduction to CPU *



* CPU (Processor) :- Central Processing unit.
→ CPU Executes Instruction and Controls the operation of other sub systems in Computer System.

→ CPU starts the Execution of a program with :
→ Fetching of Instructions one at a time,
→ Decoding the instruction and
→ Performing the Operations Specified.

→ The Part of Computer that performs the bulk of data processing operations is Called CPU.
→ CPU is made of 3 major parts.
Register Set stores intermediate data used during the execution of Instructions
→ ALU performs the required microoperations for executing the Instructions
→ Control unit Supervises the transfer of Information among the registers and, Instructs the ALU as to which operation to be performed.

→ CPU Performs a variety of functions dictated by type of instructions that are incorporated in the Computer.

→ Computer Architecture is defined as Computer structure and behavior as seen by the programmer that uses machine language instructions

→ This includes Instruction formats, addressing modes, instruction set and general organization of CPU register.

Major Components of CPU

→ The user whose Programs the Computer in machine or assembly language must be aware of the register set, the memory structure, the type of data supported by instructions and the function that each instruction performs.

3.1.1 Register Transfers

In Fig. 3.2, the data transfer between registers and common bus is shown by a line with arrow heads. But in actual practice each register has input and output gating and these gates are controlled by corresponding control signals. This is illustrated in Fig. 3.3. As shown in Fig. 3.3, control signals. $R_{i\ in}$ and $R_{i\ out}$ controls the input and output gating of register R_i. When $R_{i\ in}$ is set to 1, the data available on the common data bus is loaded into register R_i. Similarly, when $R_{i\ out}$ is set to 1, the contents of register R_i are placed on the common data bus. The signals $R_{i\ in}$ and $R_{i\ out}$ are commonly known as input enable and output enable signals of registers, respectively.

The input and output gates are nothing but the electronic switches which can be controlled by the control signals. When signal is 1, the switch is ON and when signal is 0, the switch is OFF.

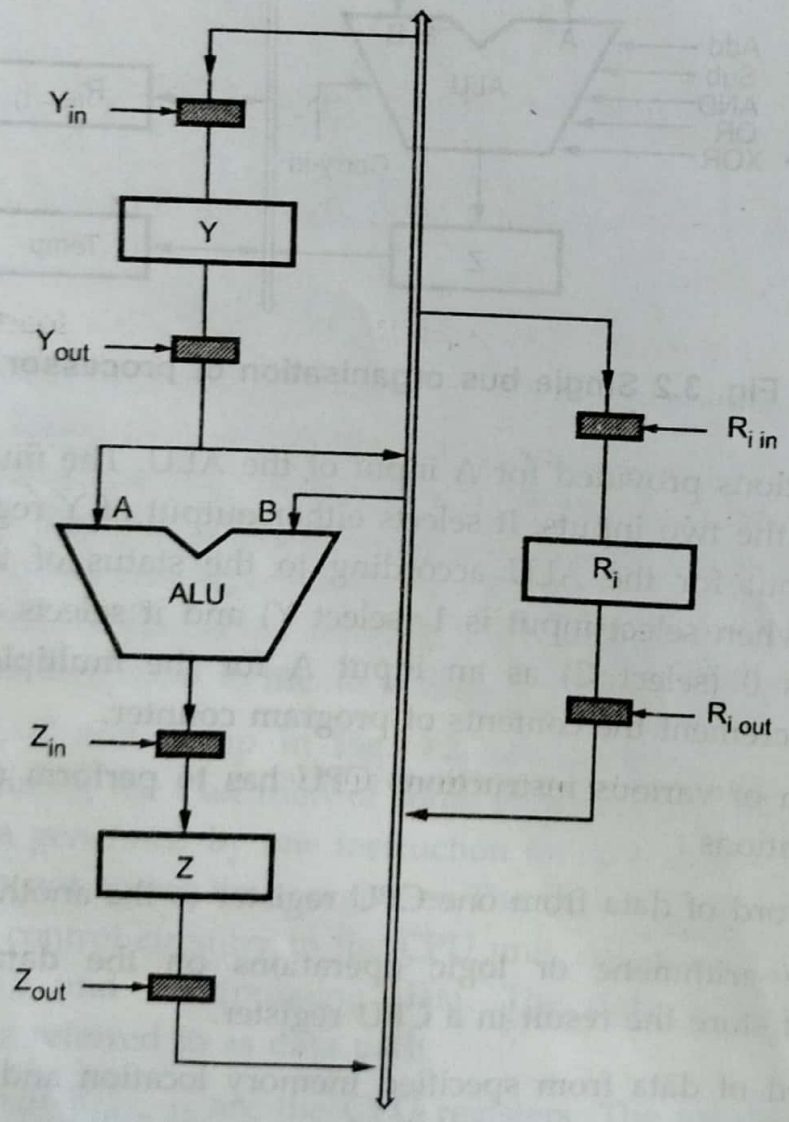


Fig. 3.3 Input and output gating for the register

The Fig. 3.4 shows the implementation of input and output gates of a 4-bit egister.

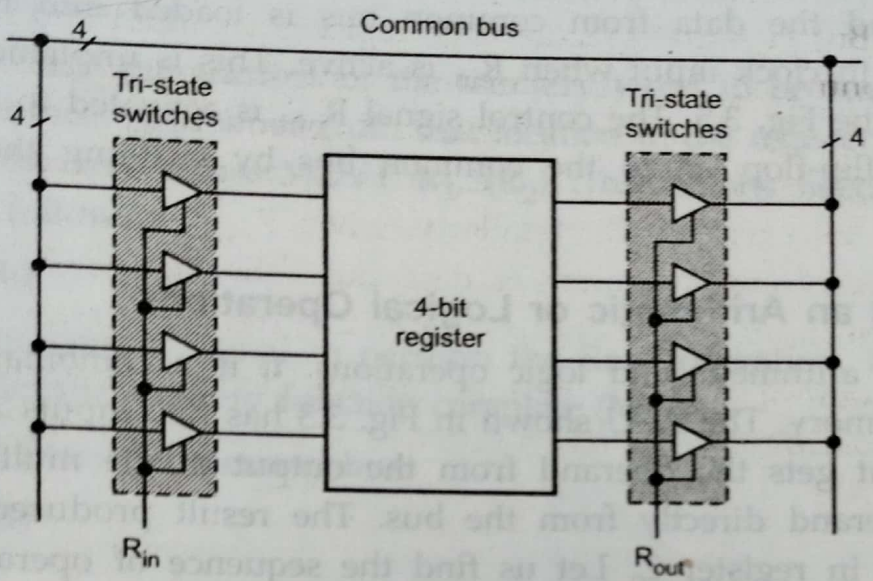


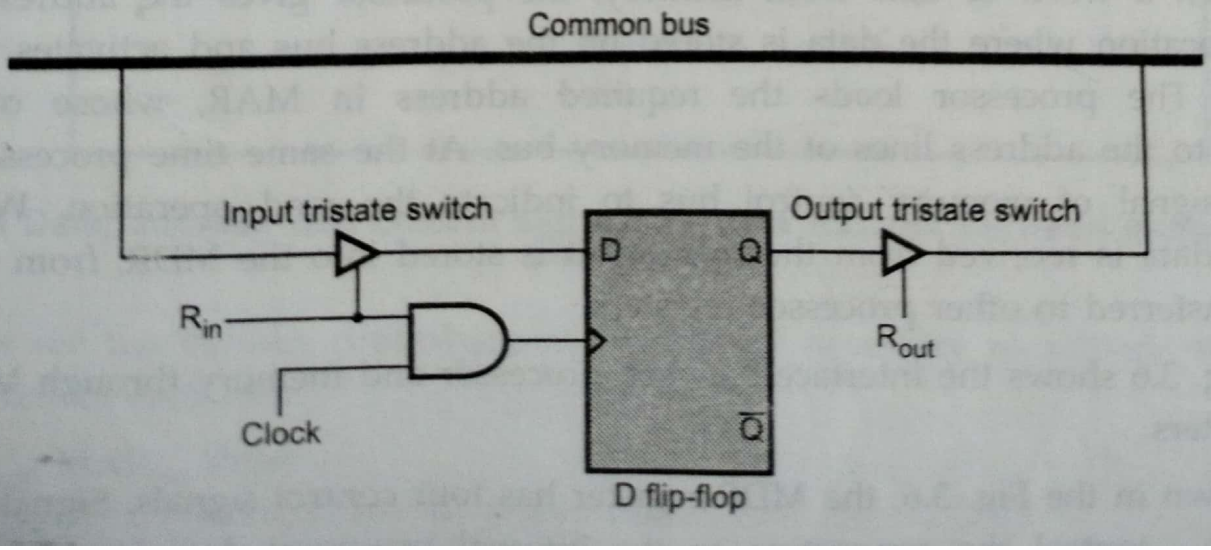
Fig. 3.4

Now consider that we have transfer data from register R_1 to R_2. This can be complished as follows :

- Activate the output enable signal of R_1, R_{1out} = 1. This places the contents of R_1 on the common bus.
- Activate the input enable signal of R_2, R_{2out} = 1. This loads data from the common bus into the register R_2.

All operations and data transfers within the processor take place in nchronisation with the clock signal. The control signals which controls a particular nsfer are activated either at the rising edge or at the falling edge of the clock cycle.

The Fig. 3.5 shows the implementation of one bit register. The edge triggered D p-flop which stores the one-bit data is connected to the common bus through



tri-state switches. Input D is connected through input tri-state switch and output Q is connected through output tri-state switch. The control signal R_{in} enables the input tri-state switch and the data from common bus is loaded into the D flip-flip in synchronisation with clock input when R_{in} is active. This is implemented using AND gate as shown in the Fig. 3.5. The control signal R_{out} is activated to load data from Q output of the D flip-flop on to the common bus by enabling the output tri-state switch.

3.1.2 Performing an Arithmetic or Logical Operations

ALU performs arithmetic and logic operations. It is a combinational circuit that has no internal memory. The ALU shown in Fig. 3.3 has two inputs A and B, and one output. Its A input gets the operand from the output of the multiplexer and its B input gets the operand directly from the bus. The result produced by the ALU is stored temporarily in register Z. Let us find the sequence of operations required to subtract the contents of register R_2 from register R_1, and store the result in register R_3. The sequence can be given as follows :

1. R_{1out}, Y_{in}
2. $R_{2\ out}$, Select Y, Sub, Z_{in}
3. Z_{out}, $R_{3\ in}$

Step 1 : In this step contents from register R_1 are loaded into register Y.

Step 2 : In this step contents from Y and from register R_2 are applied to the A and B inputs of ALU, respectively, subtraction is performed, and result is stored in the Z register.

Step 3 : The contents of Z-register (result) is stored in the R_3 register.

3.1.3 Fetching a Word from Memory

To fetch a word of data from memory, the processor gives the address of the memory location where the data is stored on the address bus and activates the read operation. The processor loads the required address in MAR, whose output is connected to the address lines of the memory bus. At the same time processor sends the read signal of memory control bus to indicate the read operation. When the requested data is received from the memory it is stored into the MDR, from where it can be transferred to other processor registers.

The Fig. 3.6 shows the interface between processor and memory through MAR and MDR registers.

As shown in the Fig. 3.6, the MDR register has four control signals. Signals MDR_{in} and MDR_{out} control the connection to the internal processor data bus and signals Read and Write control the connection to the memory data bus. The MAR register has

one control signal. The signal MAR_{in} controls the connection to the internal processor address bus. Control signals read and write from the processor controls the operation read and write, respectively.

Let us assume that the address of the memory word to be read is in register R_2 and we have to transfer data word from that location to the register R_3. This operation can be indicated by instruction MOVE R_3, (R_2). The actions needed to execute this instruction are as follows :

1. MAR $\leftarrow [R_2]$
2. Activate the control signal to perform the Read operation. If memory is slow, activate wait for memory function complete (MFC).
3. Load MDR from the memory bus
4. $R_3 \leftarrow$ [MDR]

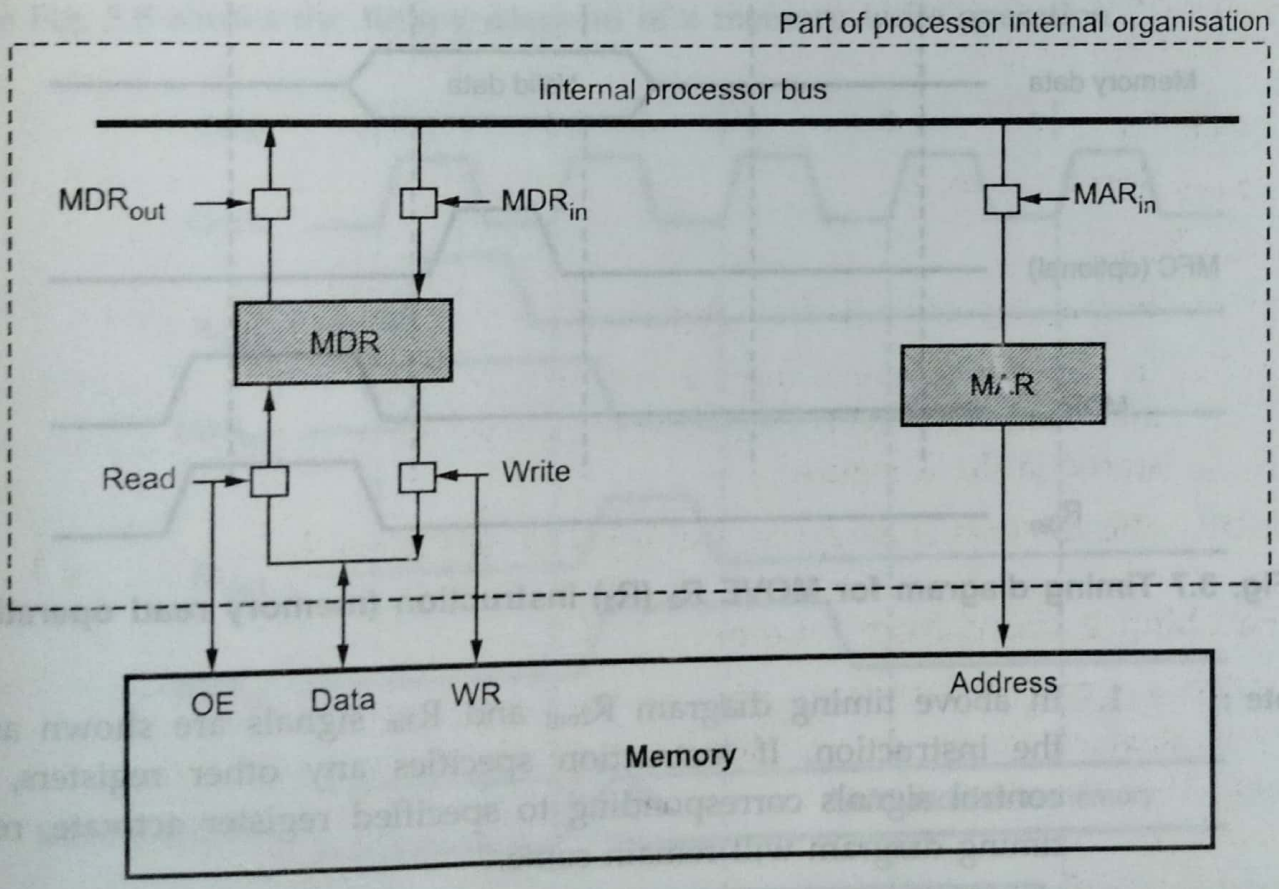


Fig. 3.6 Data, address and control signals for data transfer between memory and processor

Let us see the various control signals which are necessary to activate to perform given actions in each step.

1. R_{2out}, MAR_{in}, Read
2. WMFC

The Fig. 3.7 shows the timing diagram of a memory read operation.

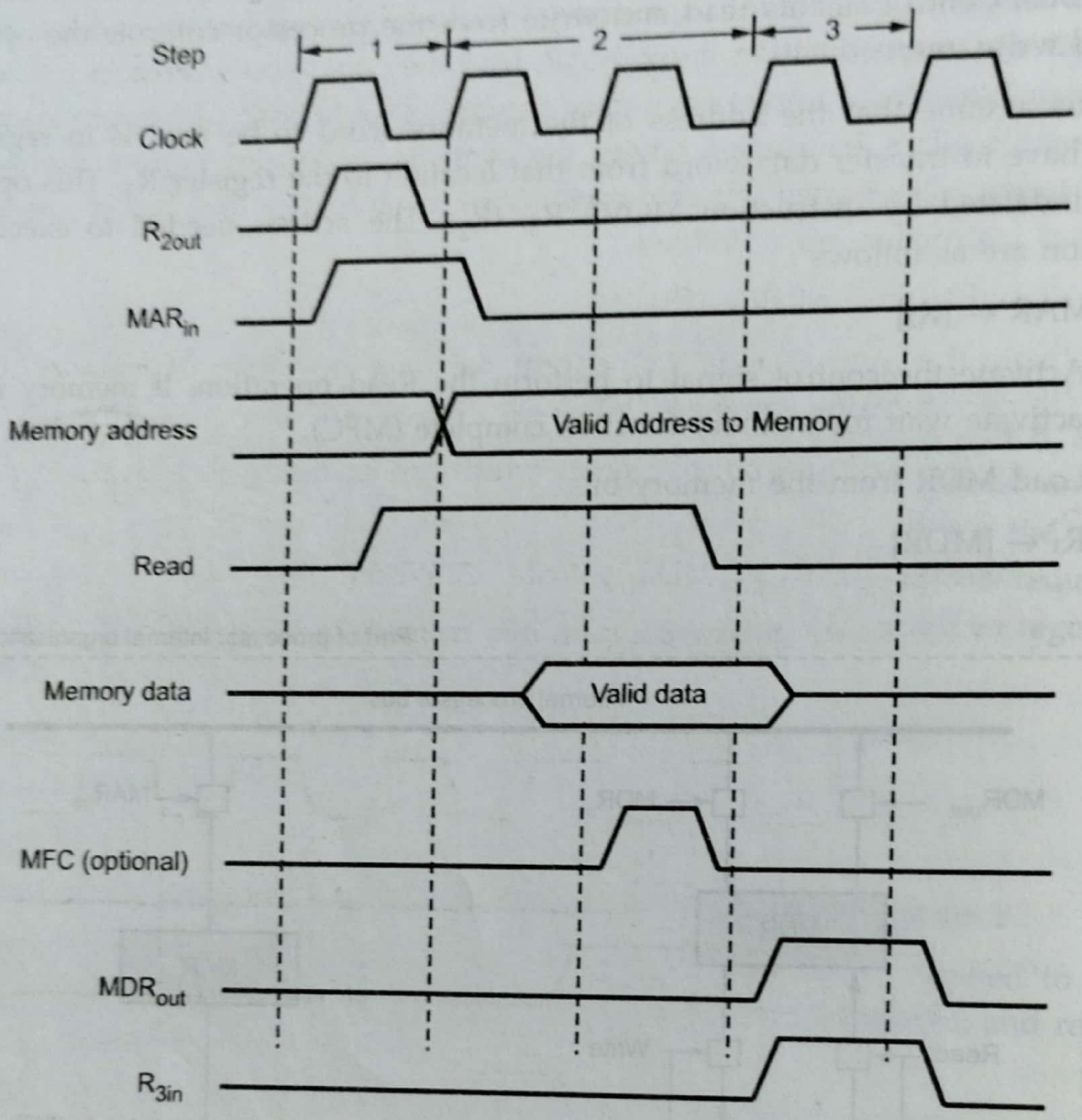


Fig. 3.7 Timing diagram for MOVE R_3 (R_2) instruction (memory read operation)

Note :

1. In above timing diagram R_{2out} and R_{3in} signals are shown as per the instruction. If instruction specifies any other registers, only control signals corresponding to specified register activate, rest of timing diagram will remain same.
2. In case of slow memory, the processor has to wait until the completion of memory read operation before going to perform next operation. In such cases control signal called Memory-Function Completed (MFC) is used. With MFC signal the step 2 becomes MAR_{out}, MDR_{in}, Read, WMFC (Wait for memory function complete).

3.1.4 Storing a Word in Memory

To write a word of data into a memory location processor has to load the address of the desired memory location in the MAR, load the data to be written in memory, in MDR and activate write operation. Let us assume that we have to execute instruction MOVE (R_2), R_1. This instruction copies the contents of register R_1 into the memory whose location is specified by the contents of register R_2. The actions needed to execute this instruction are as follows.

1. MAR $\leftarrow$ [R_2] 2. MDR $\leftarrow$ [R_1]

3. Activate the control signal to perform the write operation. If memory is slow, wait for memory function complete (MFC).

Let us see the various control signals which are necessary to activate to perform given actions in each step.

1. R_{2out}, MAR_{in} 2. R_{1out}, MDR_{in}, Write 3. WMFC

The Fig. 3.8 shows the timing diagram of a memory write operation.

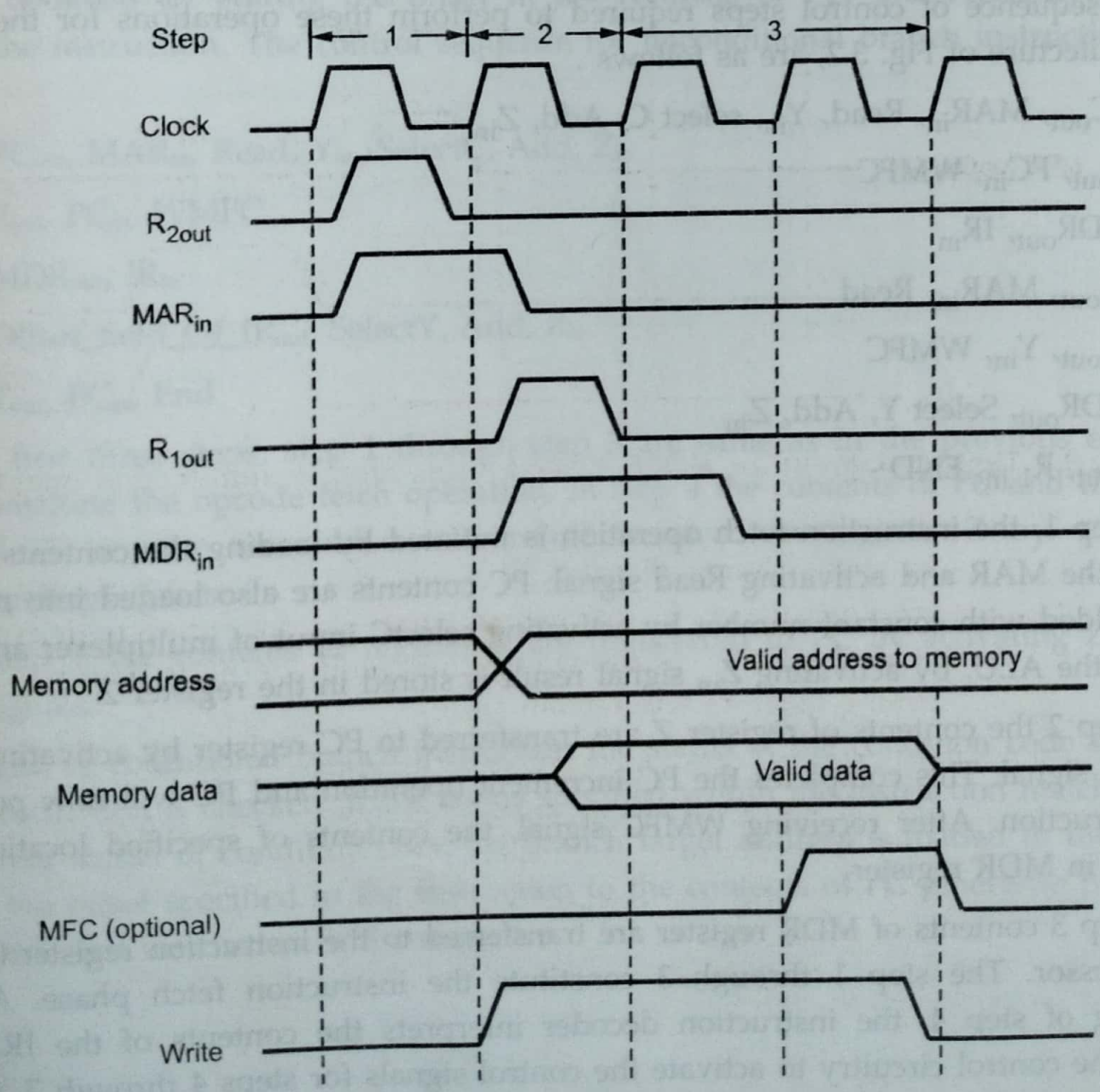


Fig. 3.8 Timing diagram for MOVE (R_2), R_1 instruction (Memory write operation)

Note :

In timing diagram signals R_{2out} and R_{1out} are shown as per the instruction. If instruction specifies any other registers, only control signals corresponding to specified register activate, rest of the timing diagram will remain same.

3.1.5 Execution of a Complete Instruction

Let us find the complete control sequence for execution of the instruction Add R_1, (R_2) for the single bus processor. This instruction adds the contents of register R_1 and the contents of memory location specified by register R_2 and store result in the registe R_1. To execute bus instruction it is necessary to perform following actions.

1. Fetch the instruction
2. Fetch the operand from memory location pointed by R_2.
3. Perform the addition
4. Store the result in R_1.

The sequence of control steps required to perform these operations for the single bus architecture of Fig. 3.2, are as follows .

1. PC_{out}, MAR_{in}, Read, Y_{in}, select C, Add, Z_{in}.
2. Z_{out}, PC_{in}, WMFC
3. MDR_{out}, IR_{in}
4. R_{2out}, MAR_{in}, Read
5. R_{1out}, Y_{in}, WMFC
6. MDR_{out}, Select Y, Add, Z_{in}
7. Z_{out}, $R_{1\ in}$, END

In step 1, the instruction fetch operation is initiated by loading the contents of the PC into the MAR and activating Read signal. PC contents are also loaded into register Y and added with constant number by activating selectC input of multiplexer and add input of the ALU. By activating Z_{in} signal result is stored in the register Z.

In step 2 the contents of register Z are transferred to PC register by activating Z_{out} and PC_{in} signal. This completes the PC increment operation and PC will now point to next instruction. After receiving WMFC signal, the contents of specified location are available in MDR register.

In step 3 contents of MDR register are transferred to the instruction register (IR) of the processor. The step 1 through 3 constitute the instruction fetch phase. At the beginning of step 4, the instruction decoder interprets the contents of the IR. This enables the control circuitry to activate the control signals for steps 4 through 7, which constitute the execution phase.

In step 4, the contents of register R_2 are transferred to register MAR by activating R_{2out} and MAR_{in} signals and Read signal is activated.

In step 5, the contents of register R_1 are transferred to register Y by activating R_{1out} and Y_{in} signals. After receiving WMFC signal, the contents of specified location are available in MDR register.

In step 6, MDR_{out}, select Y, Add and Z_{in} signals are activated to perform addition of contents of register Y and the contents of MDR. The result is stored in the register Z.

In step 7, the contents of register Z are transferred to register R_1 by activating Z_{out} and R_{1in} signals. The End signal causes a new instruction fetch cycle to begin by returning to step 1.

3.1.6 Branch Instruction

The branch instruction loads the branch target address in PC so that PC will fetch the next instruction from the branch target address. The branch target address is usually obtained by adding the offset in the contents of PC. The offset is specified within the instruction. The control sequence for unconditional branch instruction is as follows :

1. PC_{out}, MAR_{in}, Read, Y_{in},. SelectC, Add, Z_{in}
2. Z_{out}, PC_{in}, WMFC
3. MDR_{out}, IR_{in}
4. Offset_field_Of_IR_{out}, SelectY, Add, Z_{in}
5. Z_{out}, PC_{in}, End

The first three steps, step 1 through step 3 are same as in the previous example. They constitute the opcode fetch operation. In step 4 the contents of PC and the offset field of IR register are added and result is saved in register Z by activating corresponding signals.

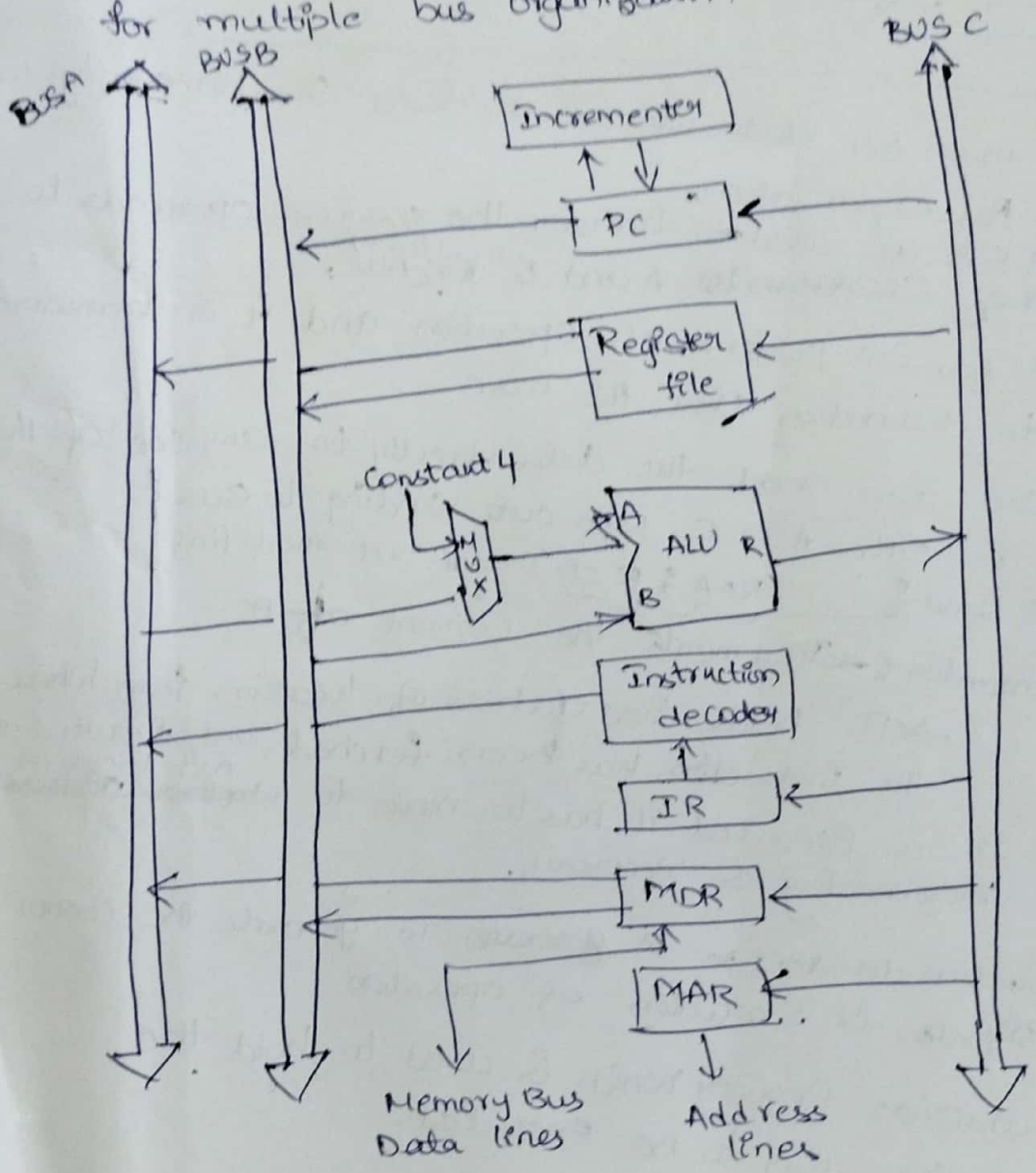
In step 5, the contents of register Z are transferred to PC by activating Z_{out} and PC in signals.

In case of conditional branch instruction the status of the condition code specified in the instruction is checked. If the status specified within the instruction matches with the current status of condition code the branch target address is loaded in the PC by adding the offset specified in the instruction to the contents of PC otherwise processor fetches the next instruction in the sequence.

* Multiple Bus organization :-

* Only one data can be transfered over the bus in a Clock cycle which makes us for more no. of Control signal.

→ To reduce the number of Control signals, then more than one data should be sent simultaneously for multiple bus organization.



→ All general purpose registers are stored in single file called register file.

↳ Have 3 ports

↳ 2 I/O Ports which allows the content of different registers to be accessed simultaneously that is put on BUS A & B.

→ 3rd Port ; which allows the data BUS C to be loaded into the 3rd register during the same clock cycle.

multiple Bus organization.

→ Buses A & B are used to transfer the source operands to i/p's of ALU.

where you can perform ALU operation and it is transfered to destination over the Bus C.

→ (It can also send the data directly to any one of the ip ie either A or C with out sending to Bus C.

Control Signals R=A ; R=B) with out modifing

→ Incrementer: increments the Content of PC

→ PC holds the address of location from where the Instruction has been fetched and Executed and nxt location is, once it is Executed it has to move to ~~another address~~ incrementing is necessary.

→ Instruction decoder → To generate the Control signals for Execution of operation

→ Instruction Register which is used to hold the current instruction to be Executed.

→ MDR - Memory data register
MAR - Memory Address Register.

→ Constant 4 → used to increment the address memory address in load multiple and store multiple instructions.

Add R4, R5, R6.

→ 4 Control Sequences.

Example: Add R4, R5 and store in R6. by using 3Bus organization.

Sol → 4 control sequences are required.

→ we have

1. PCout ; R=B ; MARin; Read ; Inc PC .

Contents of PC are passed through ALU and it sent through ALU so; R=B (Directly sent) and loaded into MAR and read is initiated and incremented the PC. (loaded) → we can go to nxt location.

The data in MAR is original content of PC and incremented value is loaded into PC at end of clock cycle.

2. WMFC (wait memory fn completed)

Processor waits for memory fn completed and loads the data into MDR

3. MDRout ; R=B ; IRin

Contents of MDR are transferred to instruction register

4. R4outA , R5outB, Select A, Add , R6in, End.

we have execution phase / Contents present A & B are added and store it in R6.

* Hardwired Control:

There are two different approaches used for the control unit design:

→ Hardwired Control

→ Microprogrammed Control.

→ In Hardwired Control, the control units are fixed logic circuits to interpret instructions and generate control signals from them.

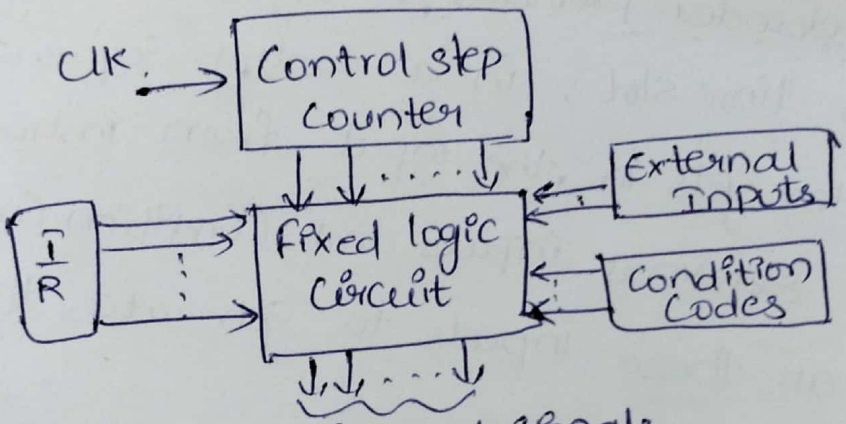


Fig: Typical Hardwired Control unit.

→ In this the fixed logic circuit block includes Combinational circuit that generates the required control outputs for decoding and Encoding functions.

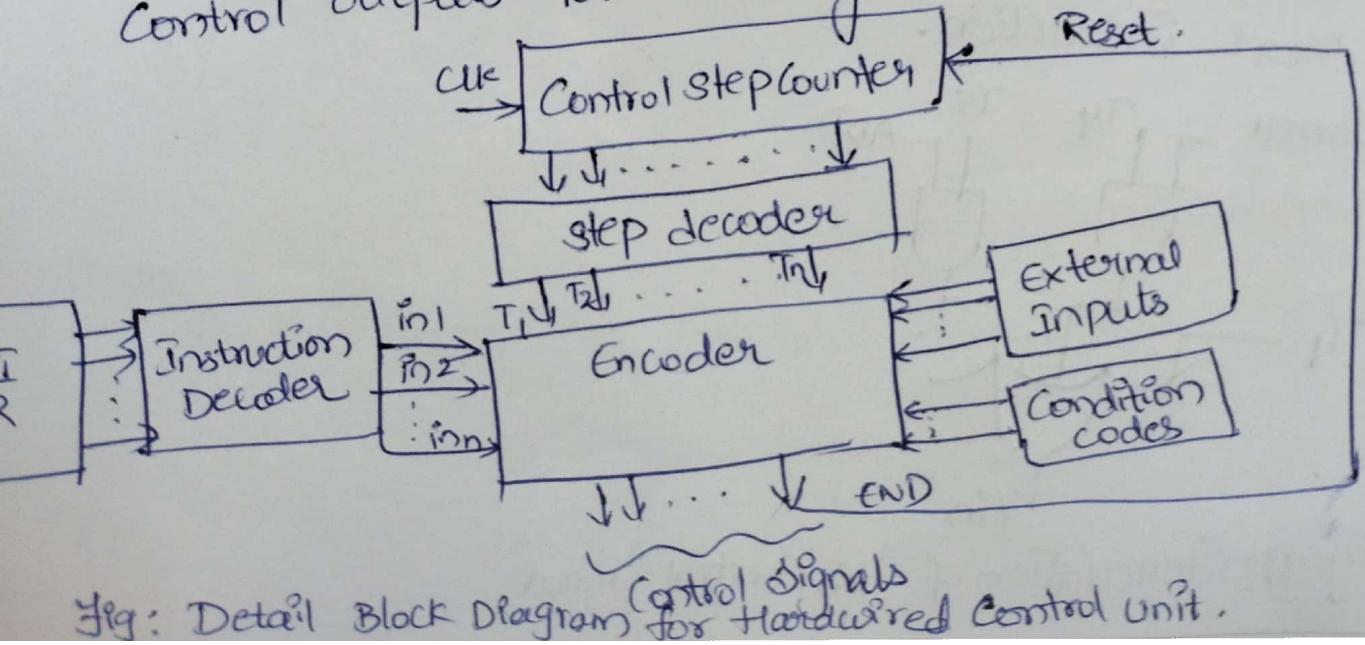


Fig: Detail Block Diagram for Hardwired Control unit.

→ The above figure ② is by separating the decoding and Encoding functions.

→ Instruction decoder decodes the instruction loaded in the IR. If IR is an 8 bit register then instruction decoder generates 2^8 ie 256 lines; one for Each instruction.

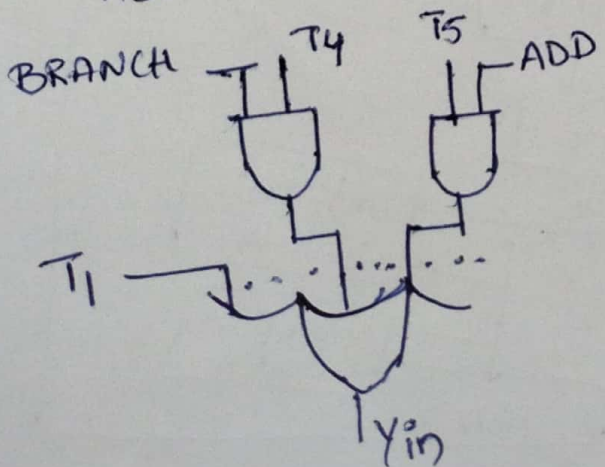
→ According to code in IR; only one line amongst all output lines of decoder goes high ie, set to 1 and all other lines are set 0.

→ The step decoder provides a separate signal line for Each step or time slot; in a control sequence.

→ The Encoder gets in the input from instruction decoder, Step decoder, External inputs and Condition codes.

→ It uses all these inputs to generate the individual Control signals.

→ After Execution of each instruction end signal is generated which resets control step Counter and make it ready for generation of Control step for next instruction.



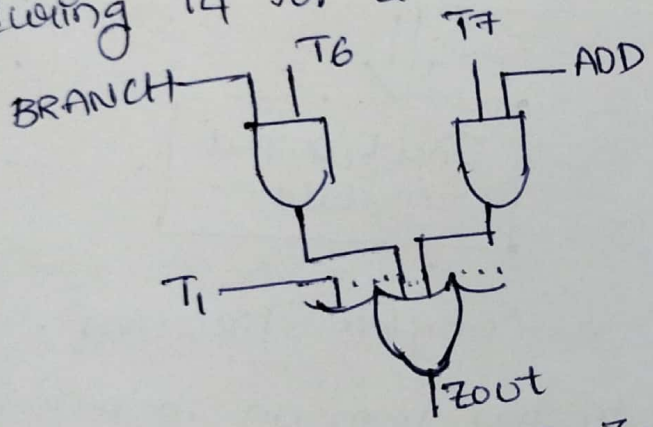
Fig(a): Generation of Yin Control Signal.

→ The encoder generates signal for single bus processor organization.

→ Consider Yin signal in above diagram.

$$Y_{in} = T_1 + T_5 \cdot ADD + T_4 \cdot BRANCH + \ldots$$

→ Yin signal is asserted during time interval T_1 for all instructions, during T_5 for ADD instruction, during T_4 for an unconditional branch instruction...



Fig(b): Generation of Zout control signal.

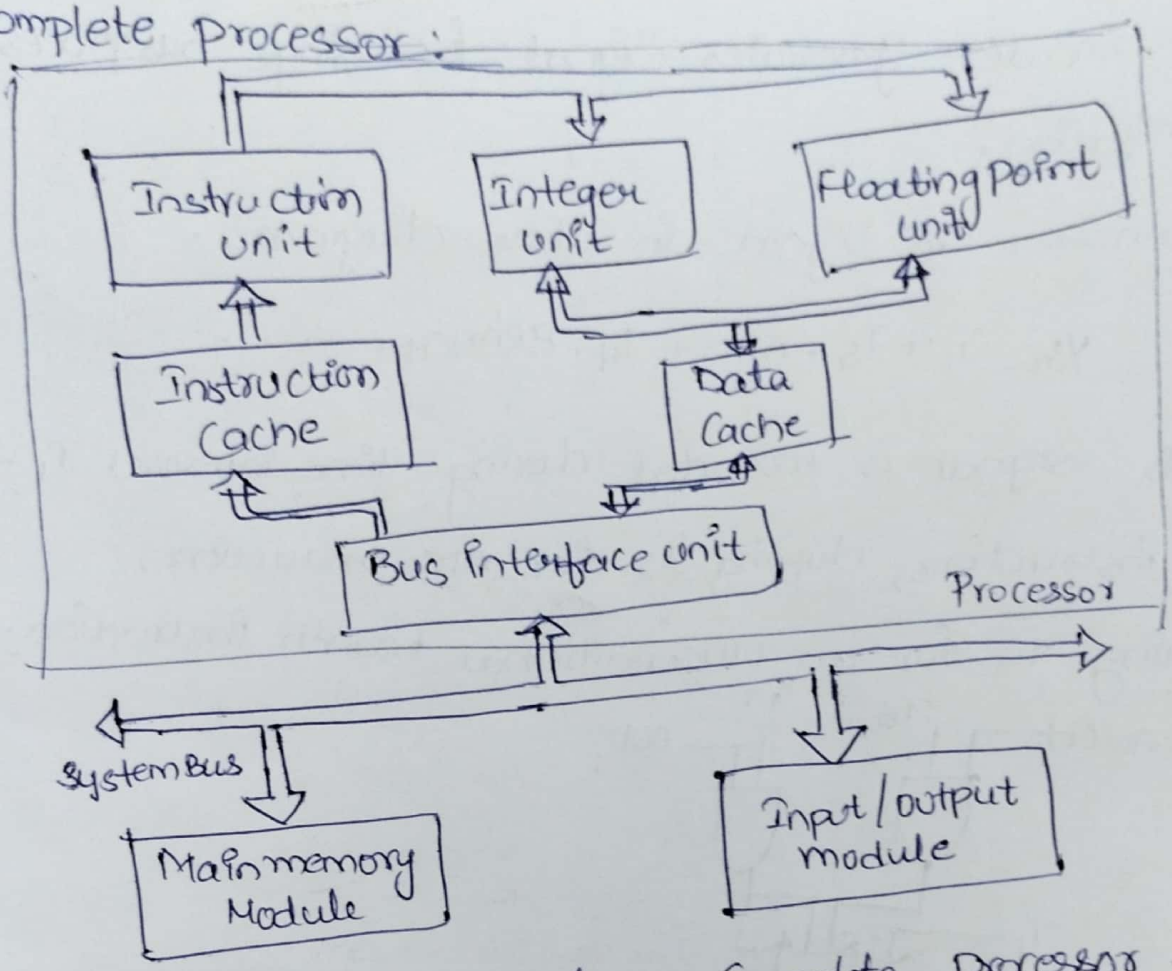
→ Another Example; logic fn' to generate Zout signal can be given by,

$$Z_{out} = T_2 + T_7 \cdot ADD + T_6 \cdot BRANCH + \ldots$$

→ Zout signal is asserted during time interval T_2 of all instructions, during T_7 for an ADD instruction, during T_6 for an unconditional branch instruction...

→ Fig (a) and (b) shows the hardware implementation of logic functions for Yin and Zout control signals.

→ A Complete processor:



Block diagram of a Complete processor.

→ Instruction unit fetches instruction from an instruction cache or from main memory when desired instructions are not available in the Cache.

→ Complete processor provides 2 processing units ↘ Integer unit, ↘ Floating point unit

→ These two units gets data from data Cache.

→ Use of seperate caches for instructions and data is common practice in many processors today.

→ the 80486 processor has 8-kbytes single Cache for both instructions and data whereas the Pentium processor has 2 separate 8Kbyte Caches for instruction and data.

→ the processor provides bus interface unit to control the interface of processor to System bus, main memory module and input/output module.

3.3 Microprogrammed Control

Every instruction in a processor is implemented by a sequence of one or more sets of concurrent microoperations. Each microoperation is associated with a specific set of control lines which, when activated, causes that microoperation to take place. Since the number of instructions and control lines is often in the hundreds, the complexity of hardwired control unit is very high. Thus, it is costly and difficult to design. Further more, the hardwired control unit is relatively inflexible because it is difficult to change the design, if one wishes to correct design error or modify the instruction set.

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 in much similar way an instruction is fetched from main memory. Each microinstruction also explicitly or implicitly specifies the next microinstruction to be used, thereby providing the necessary information for sequencing. A sequence of one or more microoperations designed to control specific operation, such as addition, multiplication is called a **microprogram**. The microprograms for all instructions are stored in the control memory.

The address where these microinstructions are stored in CM is generated by **microprogram sequencer/microprogram** controller. The microprogram sequencer generates the address for microinstruction according to the instruction stored in the IR.

Fig. 3.14 shows the microprogrammed control unit. It consists of control memory, control address register, micro instruction register and microprogram sequencer.

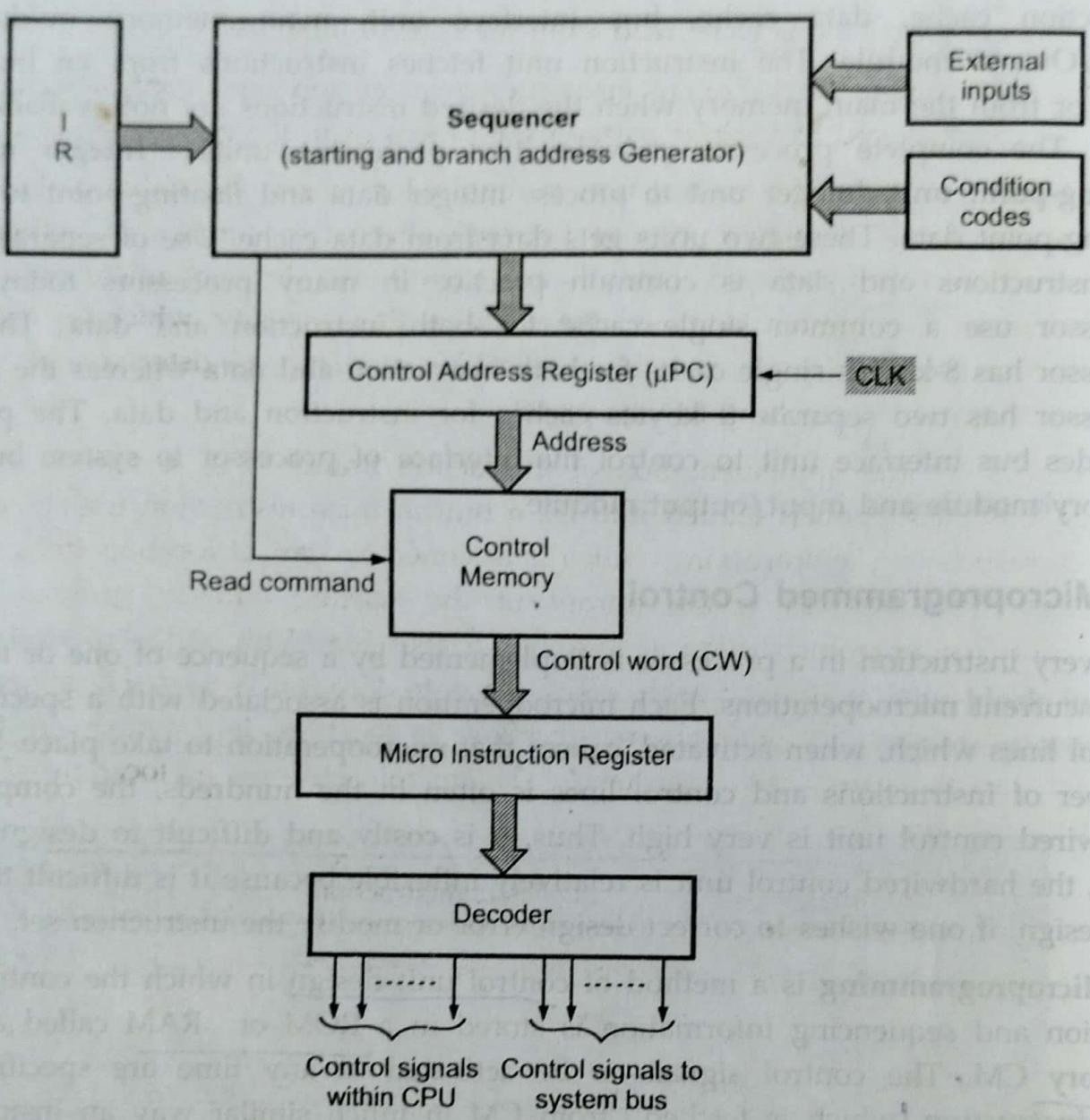


Fig. 3.14 Microprogrammed control unit

The components of control unit work together as follows :

- The control address register (μpc) holds the address of the next microinstruction to be read. Every time a new instruction is loaded into the IR, the output of the block labeled "starting address generator" is loaded into the μpc.
- When address is available in control address register, the sequencer issues READ command to the control memory.
- After issue of READ command, the word from the addressed location is read into the microinstruction register.

- The μpc is then automatically incremented by the clock, causing successive microinstructions to be read from the control memory.
- The content of the micro instruction register generates control signals which are delivered to various parts of the processor in the correct sequence.

Number of times the control unit is required to check the status of the condition codes or external inputs to choose between alternative courses of action. In such situation, microprogrammed control use conditional branch microinstructions. In additions to the branch address, these instructions specify which of the external inputs, condition codes, or, possibly bits of the instruction register should be checked as a condition for branching to take place.

Let us see the implementation of instruction Branch < 0, as shown in Fig. 3.15. When this instruction is loaded into IR, a branch microinstruction transfers control to the corresponding microroutine, which is assumed to start at location 45 in the control memory. This address is the output of the starting address generator block in Fig. 3.14. The microinstruction at location 45 tests the N bit of the condition codes. If this bit is equal to 0, a branch takes place to location 0 to fetch a new machine instruction. Otherwise, the microinstruction at location 46 is executed to put the branch target address into register Z. The microinstruction in location 47 loads this address into the PC.

Address	Microinstruction
0	PC_{out}, MAR_{in}, Read, Y_{in}, SelectC, Add, Z_{in}
1	Z_{out}, PC_{in}, WMFC
2	MDR_{out}, IR_{in}
3	Branch to starting address of appropriate microroutine
45	If N=0, then branch to microinstruction 0
46	offset_field_of_IR_{out}, SelectY, Add, Z_{in}
47	Z_{out}, PC_{in}, End

Fig. 3.15 Microroutine for the instruction branch < 0

To support microprogram branching, the starting address and branch address generator block loads a new address into the μpc when a microinstruction instructs it to do so. To allow implementation of conditional branch, the external inputs, condition codes and the contents of IR are given to the starting address and branch address generator block. Let us see how the μpc is used at different situations.

1. When a new instruction is loaded into the IR, the μpc is loaded with the starting address of the microroutine for that instruction.
2. When a Branch microinstruction is encountered and the branch condition is satisfied, the μpc is loaded with the branch address.
3. When the End instruction is encountered, the μpc is loaded with the address of the first control word (CW) in the microroutine for the instruction fetch cycle, i.e. address 0.
4. In any other situation, the μpc is incremented every time a new microinstruction is fetched from the control memory.

Advantages of Microprogrammed Control

- It simplifies the design of control unit. Thus it is both, cheaper and less error prone to implement.
- Control functions are implemented in software rather than hardware.
- The design process is orderly and systematic.
- More flexible, can be changed to accommodate new system specifications or to correct the design errors quickly and cheaply.
- Complex function such as floating point arithmetic can be realised efficiently.

Disadvantages of Microprogrammed Control

- A microprogrammed control unit is somewhat slower than the hardwired control unit, because time is required to access the microinstructions from CM.
- The flexibility is achieved at some extra hardware cost due to the control memory and its access circuitry.

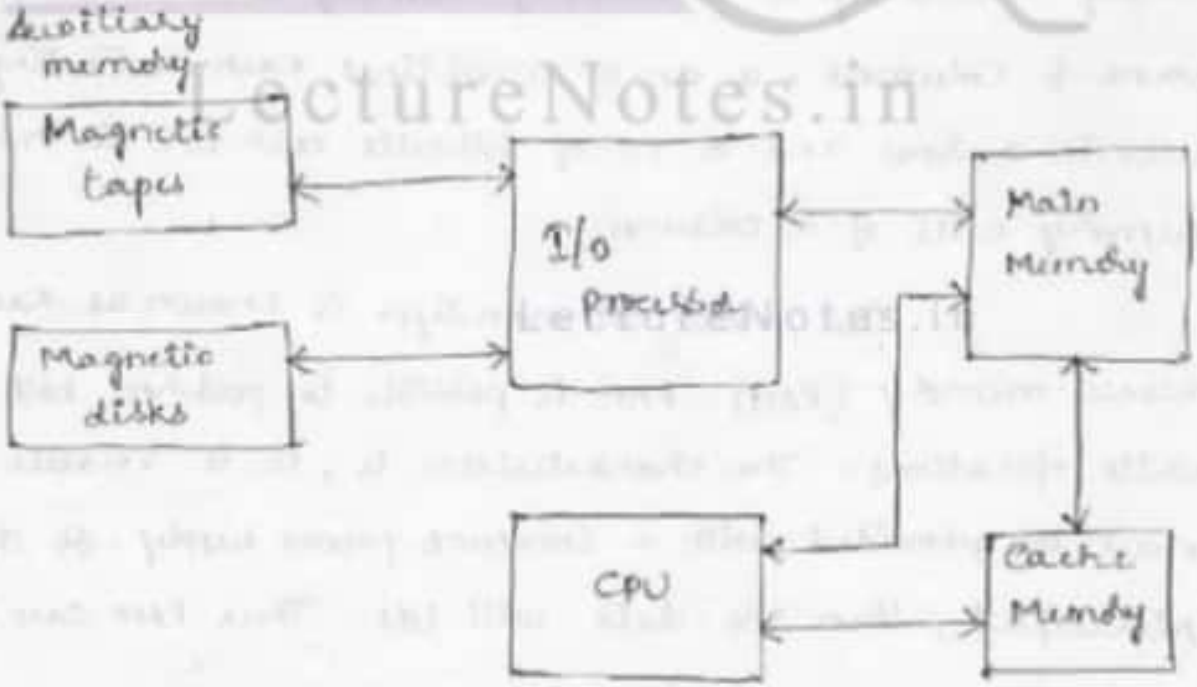
Besides these disadvantages, the microprogramming is the dominant technique for implementing control units.

MEMORY ORGANIZATION

Memory Hierarchy :-

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 devices used in computer systems are magnetic disks and tapes. They are used for storing system programs, large data files and other back up information. Only programs and data currently needed by the processor reside in main memory and all other information is stored in auxiliary memory and transferred to main memory when it is needed.

The below fig. shows the components in a memory hierarchy.



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 and

cache memory accessible to the high-speed processing logic.

At the bottom of the hierarchy are the relatively slow magnetic tapes used to store removable files. Next are the magnetic disks are used as backup storage. The main memory occupies a central position be 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.

A special very-high-speed memory called a cache. It is used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate.

Semiconductor RAM Memory :-

The semiconductor memory includes a memory cell array including a number of memory cells arranged in a form of rows & columns, a no. of word lines each connecting all memory cells in a row and a no of bit cells each one connecting all memory cells of a column.

The most common type is known as random access memory (RAM). RAM is possible to perform both read and write operations. The characteristic is, it is volatile. A RAM must be provided with a constant power supply. If the power is interrupted, then the data will lost. Thus RAM can be used only as temporary storage.

RAM technology has divided into two technologies

(i) Static RAM

(ii) Dynamic RAM.

Static RAM :-

In a static RAM, binary values are stored using traditional flip-flop logic gate configurations. A SRAM will hold its data as long as power is supplied to it. SRAM is used as Cache memory.

Dynamic RAM :-

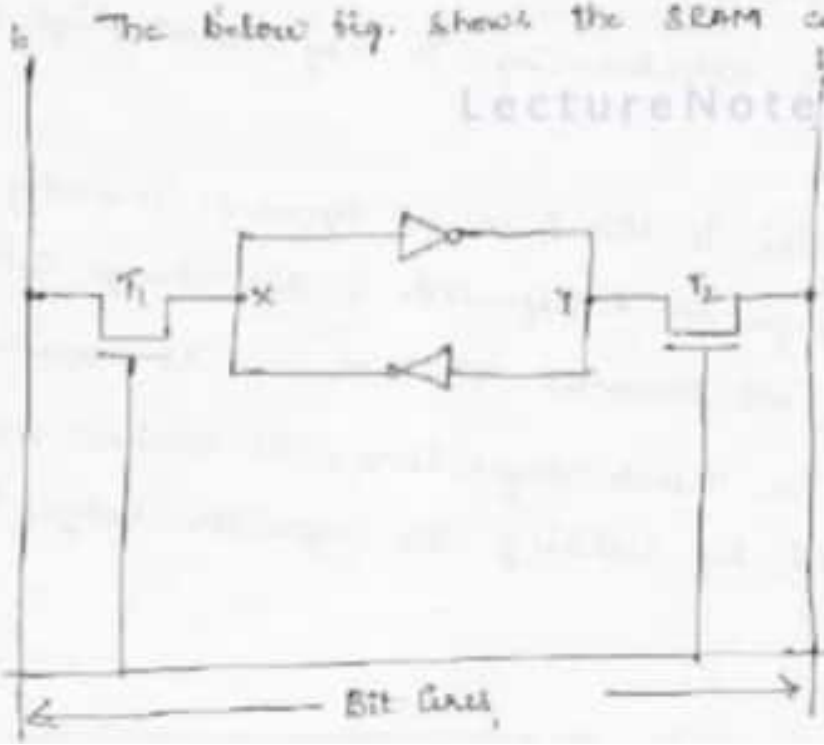
A DRAM is made with cells that store data as charge on capacitors.

DRAM's are among the densest VLSI circuit items of transistors per chip.

Both SRAM's and DRAM's are volatile, that is the stored information is lost when the power is removed. DRAM is used as main memory. A dynamic memory cell is simpler and hence smaller than a static memory cell. Thus, a dynamic RAM is more dense and less expensive than SRAM. But SRAM's are generally faster than DRAM's.

There is an another type of memory is known as Read-only memory (ROM).

The below fig. shows the SRAM can be implemented.



In the fig; the two inverters are cross-connected to form a latch. The latch is connected to two bit lines by transistors T_1 & T_2. These transistors act as switches that can be opened or closed under the control of the word line. When the word line is at ground level, the transistors are turned off and the latch retains its state. For eg, let us assume that the cell is in state 1 if the logic value at point X is 1 and at point Y is 0.

Read operation:

To read an SRAM cell, the word line is activated to close switches T_1 & T_2. If the cell is in state 1, the signal on bit line b is high & the signal on bit line b' is low.

Write operation:

The state of the cell is set by placing the appropriate value on bit line b and its complement on b' and then activating the word line.

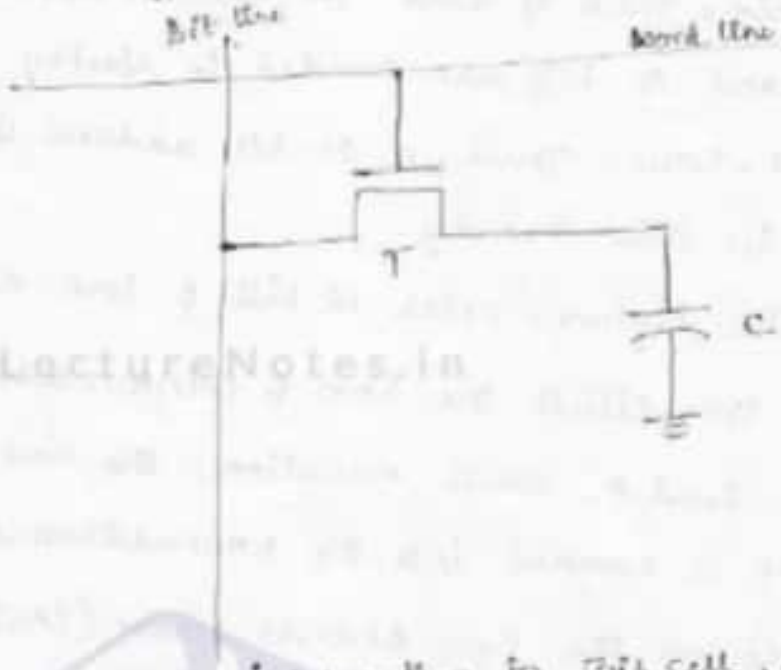
A major advantage of SRAM's is their very low power consumption because current flows in the cell only when the cell is being accessed and also they are fast.

The disadvantage is they are at high cost.

Dynamic RAM's:

Information is stored in a dynamic memory cell in the form of a charge on a capacitor & this charge can be maintained for only tens of milliseconds. Since the cell is required to store information for a much longer time, its content must be periodically refreshed by restoring the capacitor charge to its full value.

The below fig shows the dynamic memory cell that consists of a capacitor C and a transistor T.

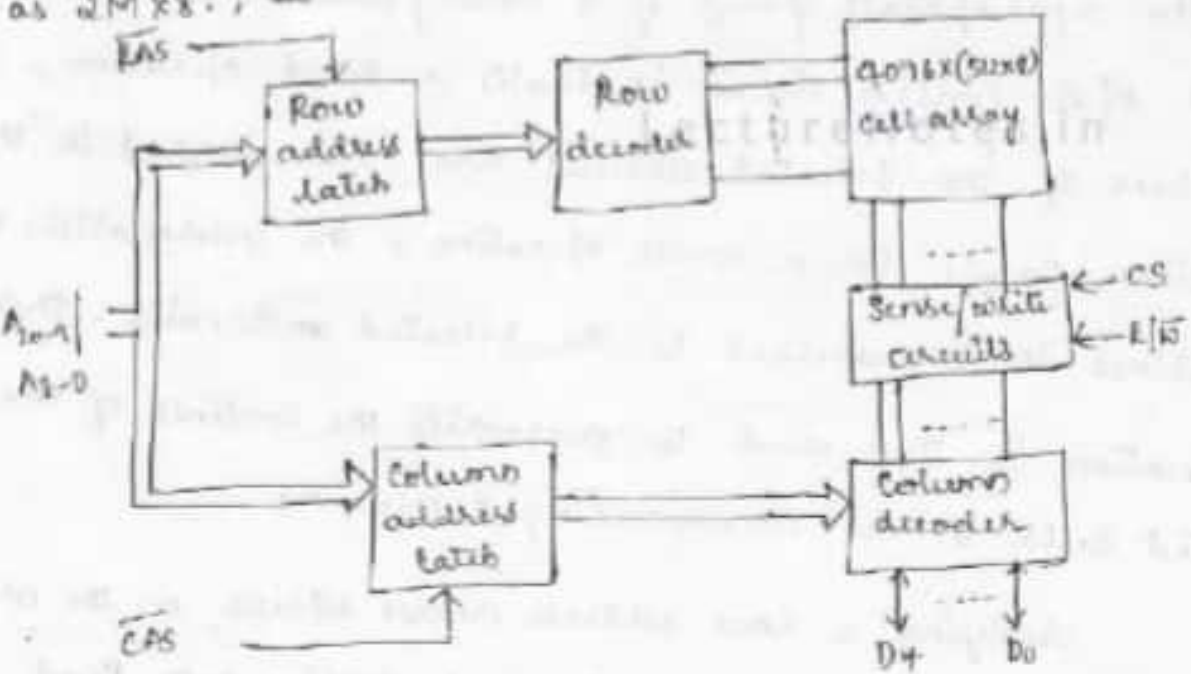


In order to store information in this cell, the transistor T is turned on and an appropriate voltage is applied to the bit line. This causes a known amount of charge to be stored in the capacitor.

After the transistor is turned off, the capacitor begins to discharge.

Asynchronous DRAM's :-

The below fig shows the 16-Megabit DRAM chip, configured as 2M x 8.



The cells are organized in the form of a 4K x 4K array. The 4096 cells in each row are divided into 512 groups of 8, so that each row can store 512 B of data. 12 address bits are needed to select a row and 9 bits are needed to specify a group of 8 bits in the selected row. Thus, a 21-bit address is needed to access a byte in this memory.

The higher-order 12 bits & lower order of 9 bits of the address constitute the row & column address of a byte.

During a Read or write operation, the row address is applied first. It is loaded into the row address latch in response to a signal pulse on the Row Address Strobe (RAS) of the chip. Then a Read operation is initiated, in which all cells on the selected row are read & refreshed.

After the row address is loaded, the column is applied to the address pins and loaded into the column address latch under the control of the column Address Strobe (CAS) signal.

The information in this latch is decoded and the appropriate group of 8 Sense/write are selected. If the R/W control signal indicates a Read operation, the O/P values of the selected circuits are transferred to the data lines D_{7-0}. For a write operation, the information on the D_{7-0} lines is transferred to the selected circuits. This information is then used to overwrite the contents of the selected cells in the corresponding 8 columns.

Applying a row address causes all cells on the corresponding row to be read and refreshed during both Read & Write operations.

while operations. To ensure that the contents of a DRAM are maintained, each row of cells must be accessed periodically. A refresh counter performs this function automatically.

The timing of the memory device is controlled asynchronously. A specialized memory controller circuit provides the necessary control signals, RAS & CAS signals. The processor must take into account the delay in the response of the memory. Such memories are referred to as asynchronous DRAM's.

Advantages :-

1) It has high density & low cost, these are widely used.
2) Available chips are of size 1M to 256 Mbits & larger chips are developed.
3) A DRAM chip is organized to read or write a number of bits in parallel.
4) It provides flexibility in designing memory systems.

Fast page Mode :-

The contents of all 4096 cells in the selected row are sensed, but only 8 bits are placed in the data lines D_{7-0}. This byte is selected by the column address bits A_{8-0}.

A simple modification can make it possible to access the other bytes in the same row with out having to reselect the row. A latch can be added at the O/P of the sense

amplifier in each column.

The application of a row address will load the latches corresponding to all bits in the selected row. Then, it is only necessary to apply different column address to place the different bytes on the data lines.

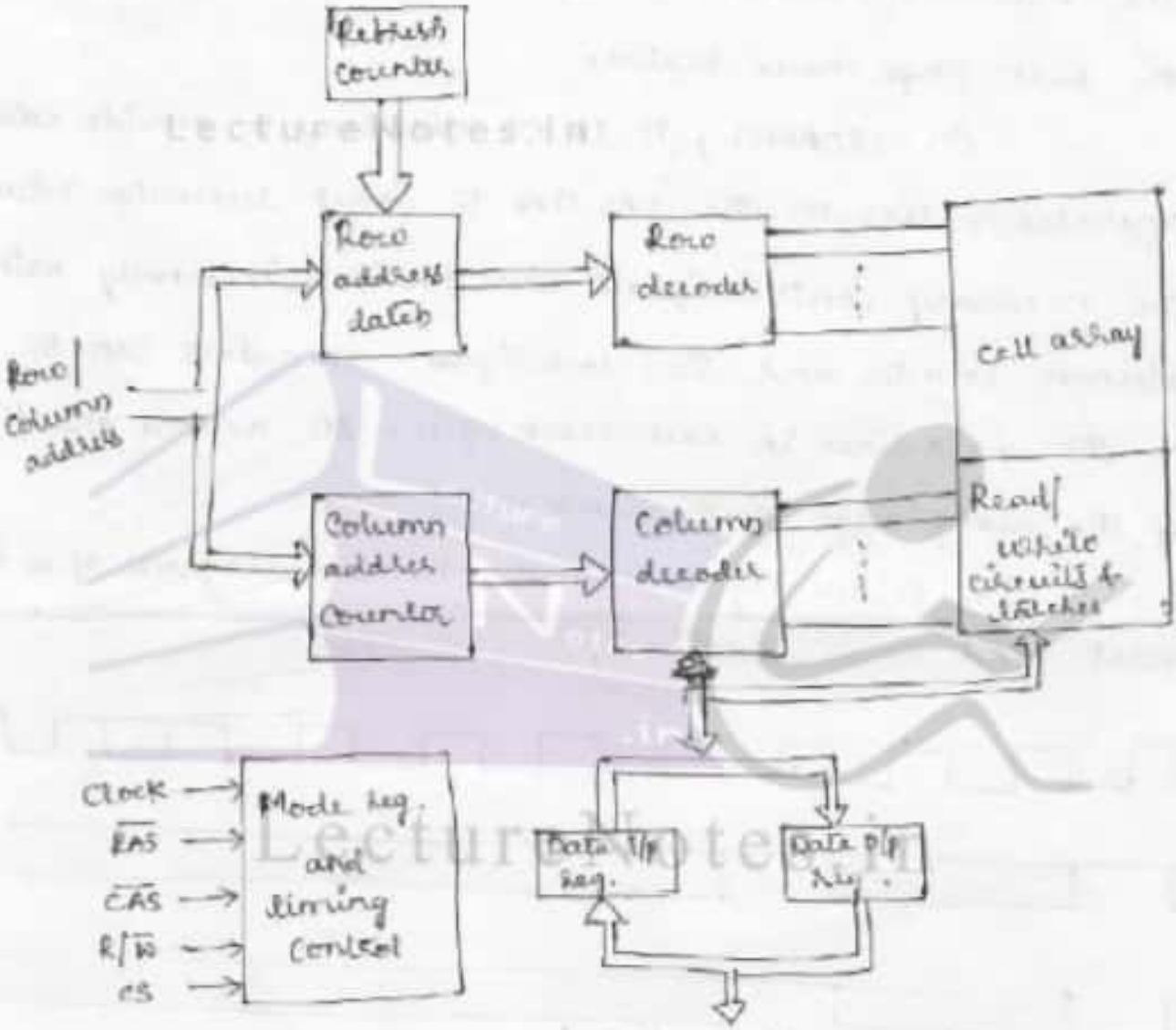
The most useful arrangement is to transfer the bytes in sequential order, which is applied by applying a consecutive sequence of column address under the control of successive CAS signals.

This scheme allows transferring a block of data at a much faster rate than can be achieved for transfers involving random addresses. The block transfer capability is referred to as 'fast page mode' feature.

Synchronous DRAM's :-

The operation is directly synchronized with a clock signal, such memories are known as Synchronous DRAM's.

The below figure shows the structure SDRAM.



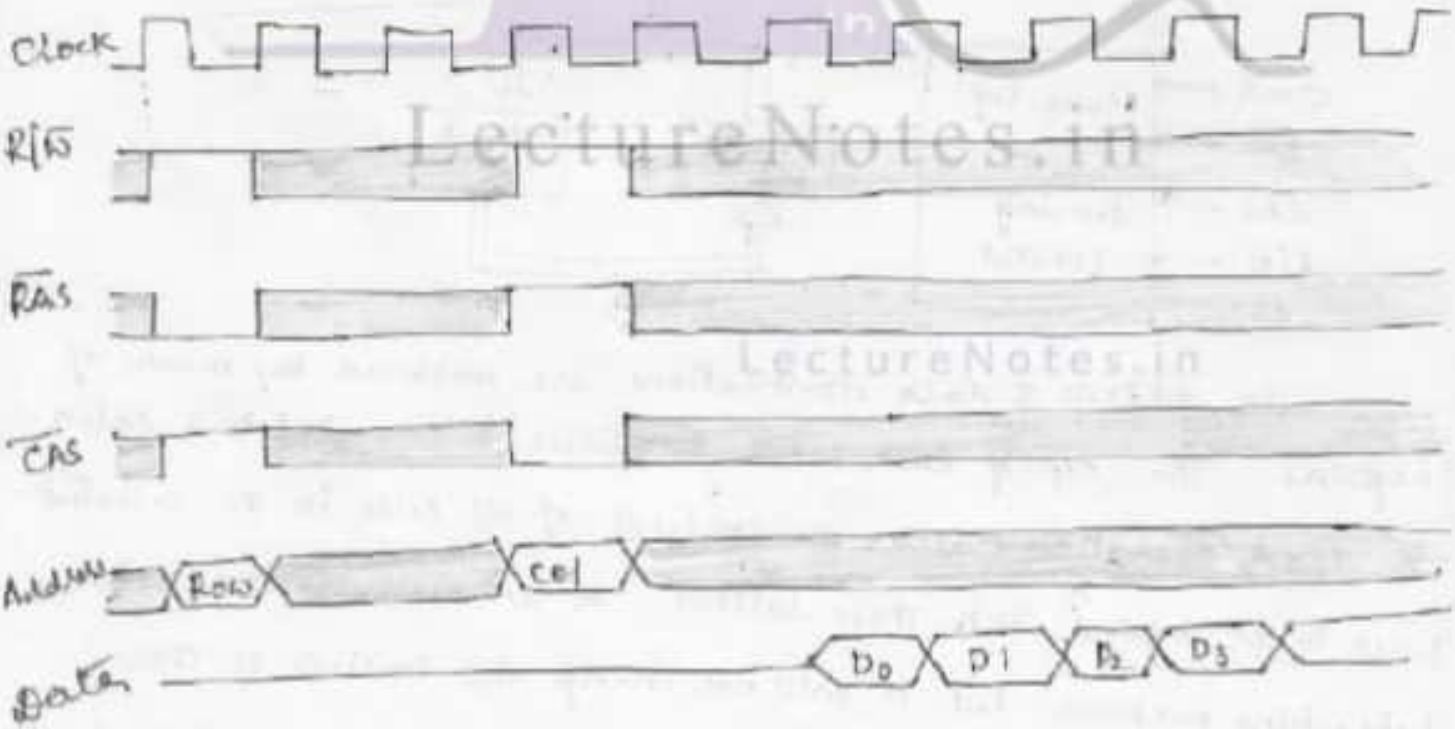
The address & data connections are buffered by means of registers. The O/p of each sense amplifier is connected to a latch. A read operation causes the contents of all cells in the selected row to be loaded into these latches. If an access is made for refreshing purposes but it will not change the contents of these latches and it will refresh the contents of the cells. Data held in the latches that correspond to the selected columns are transferred into the data o/p reg.

SRAM's have several different modes of operation, which can be selected by writing control information into a 'mode' register.

For eg, burst operations of different lengths can be specified. The burst operations use the block transfer capability as in the fast page mode feature.

In SDRAM's, it is not necessary to provide externally generated pulses on the CAS line to select successive columns. The necessary control signals are provided internally using a column counter and the clock signal. New data can be placed on the data lines in each clock cycle. All actions are triggered by the rising edge of the clock.

The below fig shows the timing diagram of a typical burst read



First, the row address is latched under control of the RAS signal. The memory typically takes 2 or 3 clock cycles to activate the selected row. Then, the column address is

latched under control of $\overline{CAS}$ signal. After a delay of one clock cycle, the first set of data bits is placed on the data lines.

The SDRAM automatically increments the column address to access the next three sets of bits in the selected row, which are placed on the data lines in the next 3 clock cycles.

SDRAMs have built-in refresh circuitry. A part of this circuitry is a refresh counter, which provides the addresses of the rows that are selected for refreshing.

The commercial SDRAM's can be used with clock speeds above 100 MHz.

~~Organization of SDRAM's~~ :-

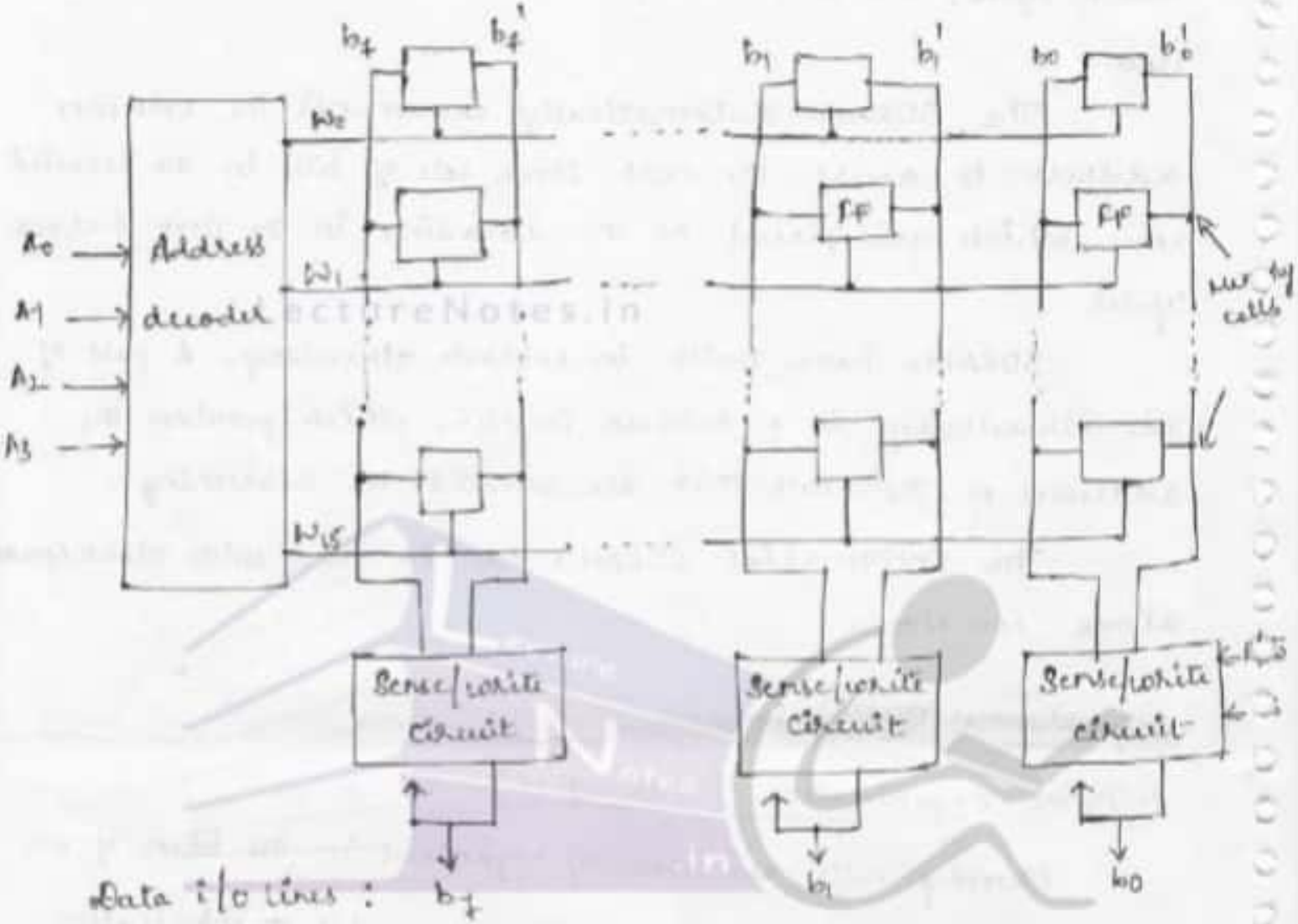
Internal organization of Memory chips :-

Memory cells are usually organized in the form of an array, each cell is capable of storing one bit of information.

The fig. is an example of a very small memory chip consisting of 16 words of 8 bits each.

Each row of cells constitutes a memory word and all cells of a row are connected to a common line is referred to as the word line, which is driven by the address decoder on the chip. The cells in each column are connected to a sense/write circuit by two bit lines.

The sense/write circuits are connected to the data i/o lines of the chip. During Read operation, these circuits sense, or read the information stored in the cells selected by a



Data i/o lines : b_7

Word line and transmit this information to the o/p data line. During a write operation, the sense/write circuit receive the information and store the cells of the selected word.

The data i/p & the data o/p of each sense/write circuit are connected to a single bidirectional data line that can be connected to the data bus of a computer.

Two control lines, $R/\bar{W}$ & CS are provided in addition to the address & data lines. The $R/\bar{W}$ input specifies the required operation & the CS (chip select) i/p selects a given chip in a multi chip memory system.

PROM :-

For small quantities, we are going to use a type of a ROM called programmable Read only Memory.

In this, the programmability is achieved by 'inserting a fuse at point p (shown in fig). Before it is programmed, the memory contains all 0's.

The user can insert 1's at the required locations by burning out the fuses at these locations using high-current pulses

PROM's provide flexibility & Convenience

PROM's provide faster and considerably less expensive because they can be programmed directly by the user. The ROM & PROM are irreversible.

EPROM :-

This an type of ROM, which allows the stored data to be erased and new data to be loaded. Such an erasable, reprogrammable ROM is usually called an EPROM.

EPROM's are capable of retaining stored information for a long time, they can be used inplace of ROM's.

The EPROM & ROM has similar structure. In EPROM cell, the connection to ground is always made at point 'p' and a special transistor is used, which has the ability to function as either as a normal transistor or as a disabled transistor that is always turned off. i.e., the transistor can be programmed to behave as a permanently open switch.

The advantage is that their contents can be erased and reprogrammed. The disadv. is that a chip is removed physically from the circuit for reprogramming and the entire contents is erased by exposure to U.V. light.

EEPROM :-

The another version of erasable PROM's that can be both programmed and erased electrically. Such chips are called EEPROM's. It is possible to erase the cell contents selectively.

The disadv is that different voltages are needed for erasing, writing and reading the stored data.

Flash Memory :

Flash Memory is intermediate b/w EPROM & EEPROM. Like EEPROM, flash memory uses an electrical erasing technology. An entire flash memory can be erased in one or a few seconds which is much faster than EPROM.

Flash Memory uses only one transistor per bit and so achieves the high density.

Cache memories :-

The speed of the main memory is very low, because for good performance, the processor cannot spend much of its time waiting to access instructions and data in main memory.

An efficient solution is to use a fast cache memory which essentially makes the main memory appear to the processor to be faster.

The effectiveness of cache mechanism is based on a property of computer programs called 'locality of reference'. Analysis of programs shows that most of their execution time is spent on routines in which many instructions are executed repeatedly.

The instructions in localized areas of the program are

executed repeatedly during some time period and the remainder of the program is accessed relatively infrequently. This is referred to as 'locality of reference'. This is divided into two types i) Temporal ii) Spatial

The temporal means that a recently executed instruction is likely to be executed again.

The spatial means that instructions in close proximity to a recently executed instruction (Based on "instruction address")

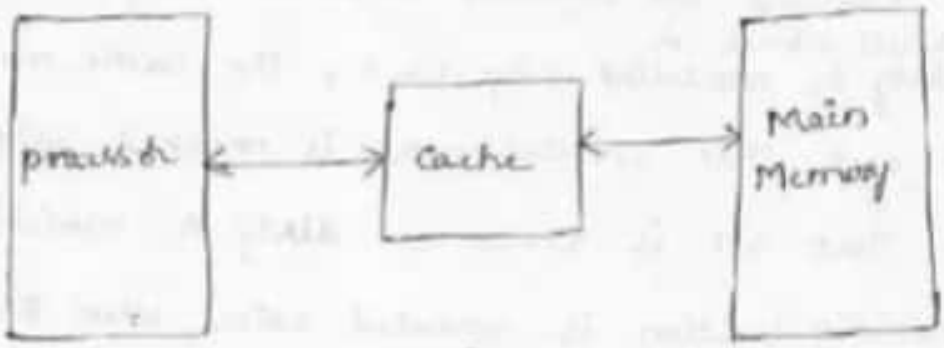
If the active segments of a program can be placed in a fast cache memory, then the total execution time is reduced.

The memory control circuitary is designed to take adv. of the property of locality of reference. The temporal aspect of the locality of reference, that when ever an information item is first needed, this item should be brought into cache where it is needed.

The spatial aspect suggests that instead of fetching one item from the main memory to the cache, it is useful to fetch several items that reside at adjacent addresses.

The term 'block' to refer to a set of contiguous address locations of some size. Another term is refer to a cache block is 'cache line'.

The below fig. shows the use of a cache memory



When a Read Request is received from the processor, the contents of a block of memory words containing the location specified are transferred into the cache one word at a time. When the program references any of the locations in this block, the desired contents are read directly from the cache. The cache memory can store a reasonable number of blocks at any given time, but this number is small compared to the total number of blocks in the main memory.

The correspondence b/w the main memory blocks and those in the cache is specified by a mapping function.

When the cache is full and a memory word that is not in the cache is referenced, the cache control h/w must decide which block should be removed to create space for the new block that contains the referenced word. The collection of rules for making this decision is "replacement algorithm".

The cache control circuitry determines whether the requested word currently exists in the cache. The Read & Write operation is performed on the appropriate cache location. In Read operation, the main memory is not involved.

The write operation may be proceed into two ways

1) Write-Through protocol 2) Write-back or copy-back

In write-through protocol, both cache memory and the main memory locations are updated simultaneously.

In write-back or copy-back, the cache memory is only updated and that updated part is marked with its associated flag bit. That bit is known as dirty or modified bit. The main memory location is updated later, when the block containing

this marked word is to be removed from the cache to make room for a new block.

When the addressed word in a Read operation is not in the cache, a "read miss" occurs. The block of words that contains the requested word copies from the main memory into the cache. After the entire block is loaded into the cache, the particular word requested is forwarded to the processor.

The latter approach, which is called load-through or early restart, reduces the processor's waiting period, but it is more complex circuitary.

During a write operation, if the addressed word is not in the cache, a "write miss" occurs. Then, the write-through is used, the information is written directly into the main memory.

In the case of write-back protocol, the block containing the addressed word is first brought into the cache, and then the desired word in the cache is overwritten with the new information.

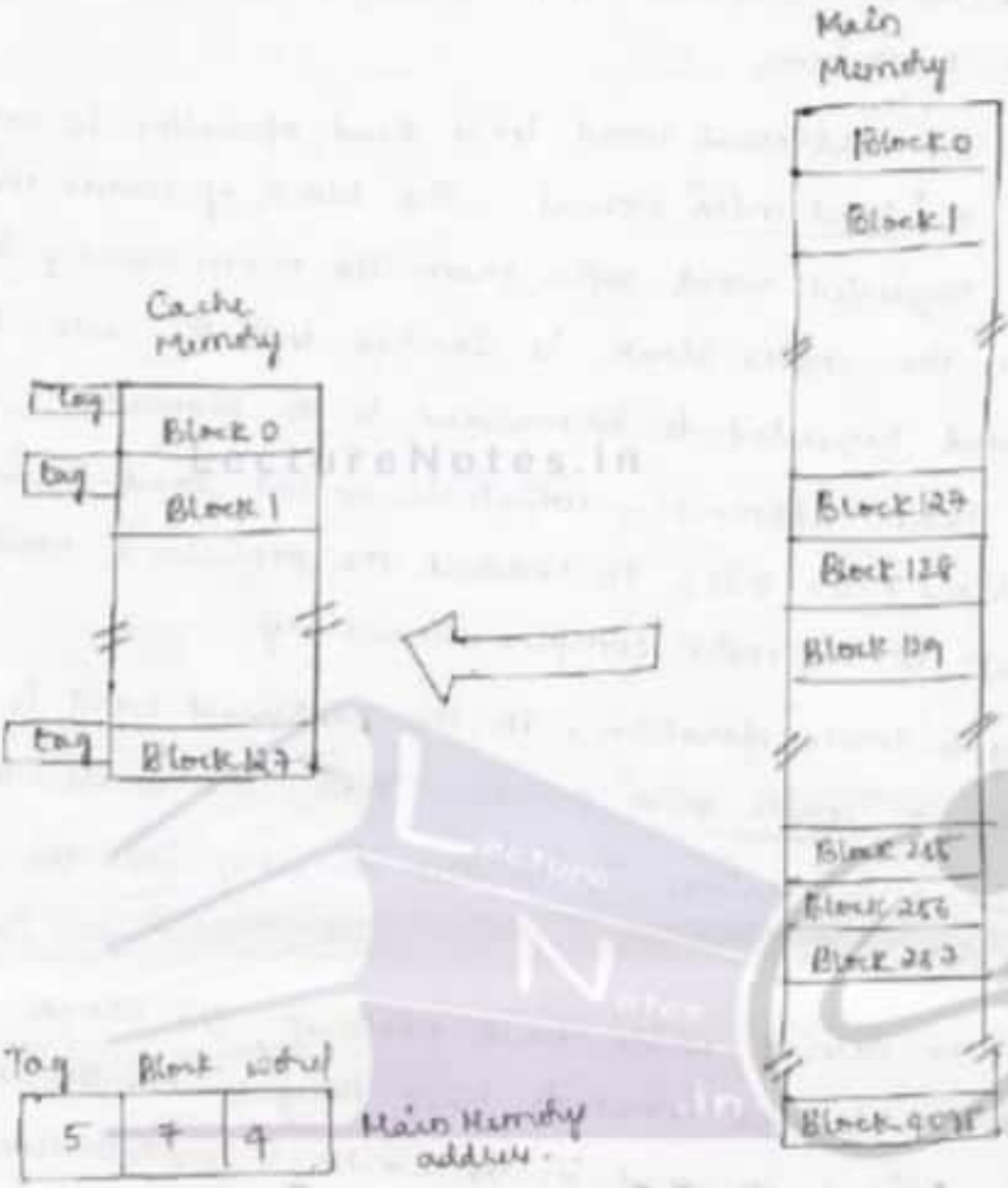
Mapping Functions :-

Direct Mapping :-

The simplest way to determine cache locations in which to store memory blocks is the direct-mapping technique.

In this technique, block j of the main memory maps onto block j modulo 128 of the cache, as depicted & shown in fig.

One of the main memory blocks 0, 128, 256 ... are loaded into main memory cache, it is stored in cache block 0. 1, 129, 257 ... are stored in cache block 1 and so on.



Since more than one memory block is mapped onto a given cache block position, contention may arise for that position even when the cache is not full.

For eg, instructions program may start in block 1 and continue in block 129, after a branch. As this program is executed, both of these blocks are transferred to the block-1 position in the cache. Contention is resolved by allowing the new block to overwrite the currently resident block.

Placement of a block in the cache is determined from the memory address. The memory address can be divided into three fields as shown in fig.

The low-order 4 bit select one of 16 words in a block.

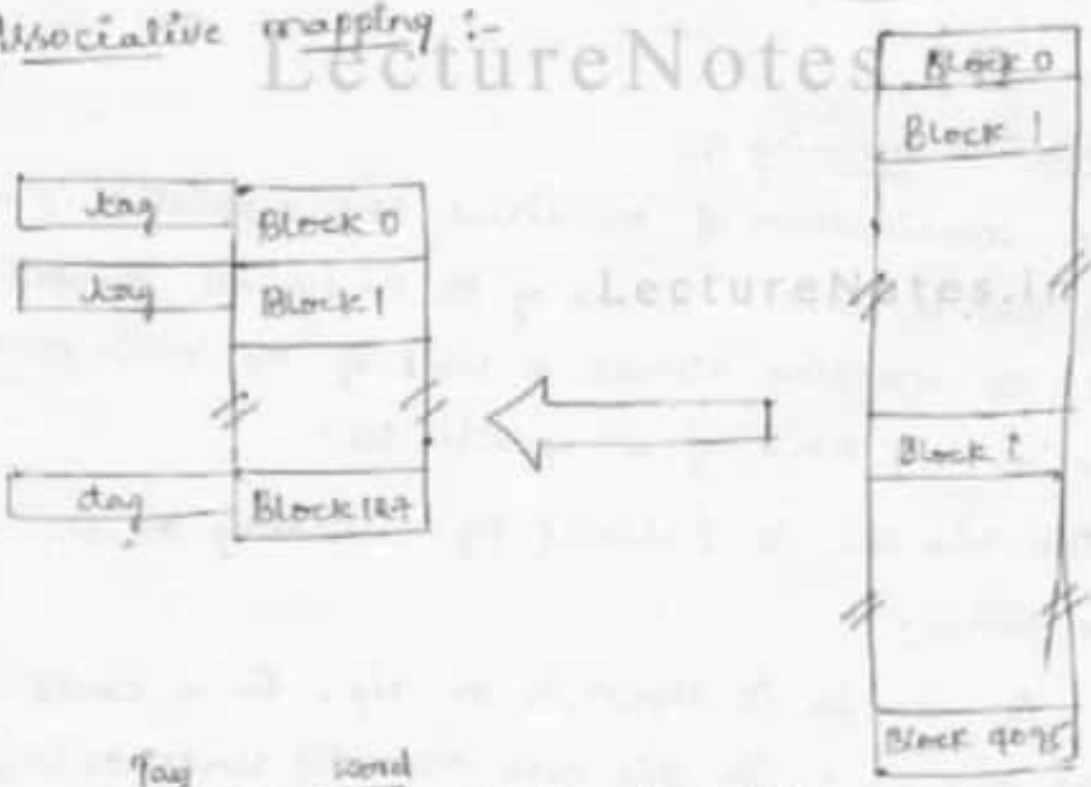
When a new block enters the cache, the 7-bit cache block field determines the cache position in which this block must be stored. The high-order 5 tag bits associated with its location in the cache. They identify the 32 blocks that are mapped into cache.

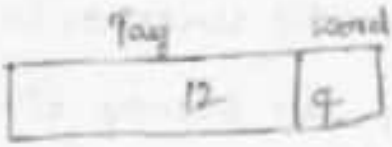
As execution proceeds, the 7-bit cache block field of each address generated by the processor points to a particular block location in the cache. The high-order 5 bit of the address are compared with the tag bit associated with the cache location. If they match, then the desired word is in that block of cache.

If there is no match, then the block containing the required word must first be read from the main memory and loaded into the cache.

The direct-mapping is easy to implement but it is not very flexible.

Associative mapping :-





This is a much more flexible mapping method, in which a main memory block can be placed into any cache block position.

In this 12 tag bits are required to identify a memory block when it is resident in the cache. The tag bits of an address received from the processor are compared to the tag bits of each block of the cache, to see if the desired block is present. This is called the 'associative-mapping' technique.

A new block that has to be brought into the cache has to replace an existing block only if the cache is full. In this case we need an algorithm to select that particular block to be replaced.

The cost of an associative cache is higher than the cost of a direct-mapped cache because of the need to search all 128 tag patterns to determine whether a given block is in the cache.

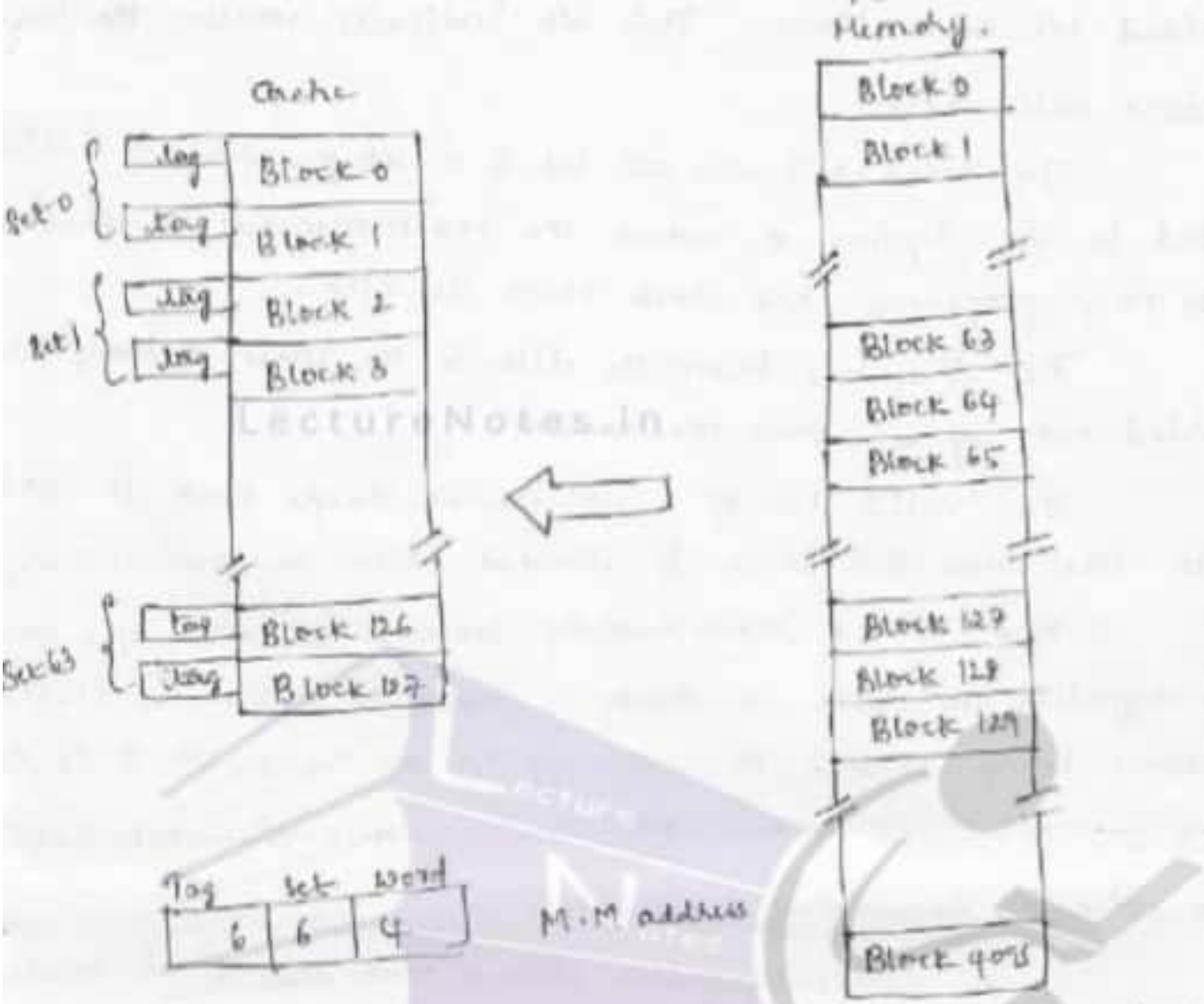
A search of this kind is called an associative search.

Set-Associative mapping :-

A combination of the direct and associative mapping techniques can be used. Blocks of the cache are grouped into sets and the mapping allows a block of the main memory to reside in any block of a specific set.

The h/w cost is reduced by decreasing the size of the associative search.

An example is shown in the fig. For a cache with two blocks per set. In this case memory blocks 0, 64, 128 ... 4032 map into cache set 0 and they can occupy either of the two block positions with in this set.



Having 64 to sets means that the 6-bit set field of the address determines which set of the cache might contain the desired block. The tag field of the address must then be associatively compared to the tags of the two blocks of the set to check if the desired block is present.

For the main memory and cache sizes, four blocks per set can be accommodated by a 5-bit set field, eight blocks per set can be accommodated by a 4-bit set field and so on. The extreme condition of 128 blocks per set requires no set bits & corresponds to the fully associative technique, with 12 tag bits. The other extreme of one block per set is the direct mapping method. A cache that has k blocks per set is referred to as a k-way set-associative cache.

one more control bit, called the valid bit, must be provided for each block. This bit indicates whether the block contains valid data.

The valid bits are all set to 0 when power is initially applied to the system or when the main memory is loaded with new programs and data from the disk.

The transfers from the disk to the main memory are carried out by a DMA mechanism.

The valid bit of a particular cache block is set to '1' the first time this block is loaded from the main memory.

Whenever a main memory block is updated by a source that bypasses the cache, a check is made to determine whether the block being loaded is currently in the cache. If it is, its valid bit is cleared to '0'. This ensures that 'stale' data will not exist in the cache.

A difficulty arises when a DMA transfer is made and the cache uses the write back protocol.

One solution to this problem is to flush the cache by forcing the dirty data to be written back to the memory before the DMA transfer takes place.

The need to ensure that two different entities (processor & DMA) uses the same copies of data is referred to as Cache - Coherence problem.

Replacement algorithms :-

When a block is to be overwritten, it is sensible to overwrite the one that has gone the longest time without being referenced. This block is called the least recently used (LRU) block and the technique is called the LRU Replacement algorithm.

LRU Algorithm :-

To use the LRU algorithm, the cache controller must track references to all blocks. To track the LRU block of a four-block set in a set associative cache. A 2-bit counter can be used for each block. When a 'hit' occurs, the counter of the block that is referenced is set to 0.

When a 'miss' occurs and the set is not full, the counter associated with the new block loaded from the main memory is set to 0 and the values of all other counters are increased by one.

When a 'miss' occurs and the set is full, the block with the counter value 3 is removed, the new block is put in its place and its counter is set to '0'. The other three block counters are incremented by one.

The LRU algorithm has been used extensively. Although it performs well for many patterns, it can lead to poor performance in some cases. For eg, LRU produces bad results, when accesses are made to sequential elements of an array that is too large to fit into the cache. performance of the LRU algorithm can be improved by introducing a small amount of randomness in deciding which block to replace.

Performance Considerations :-

Two key factors of a computer are performance & cost. performance depends on how fast machine instructions can be brought into the processor for execution & how fast they can be executed.

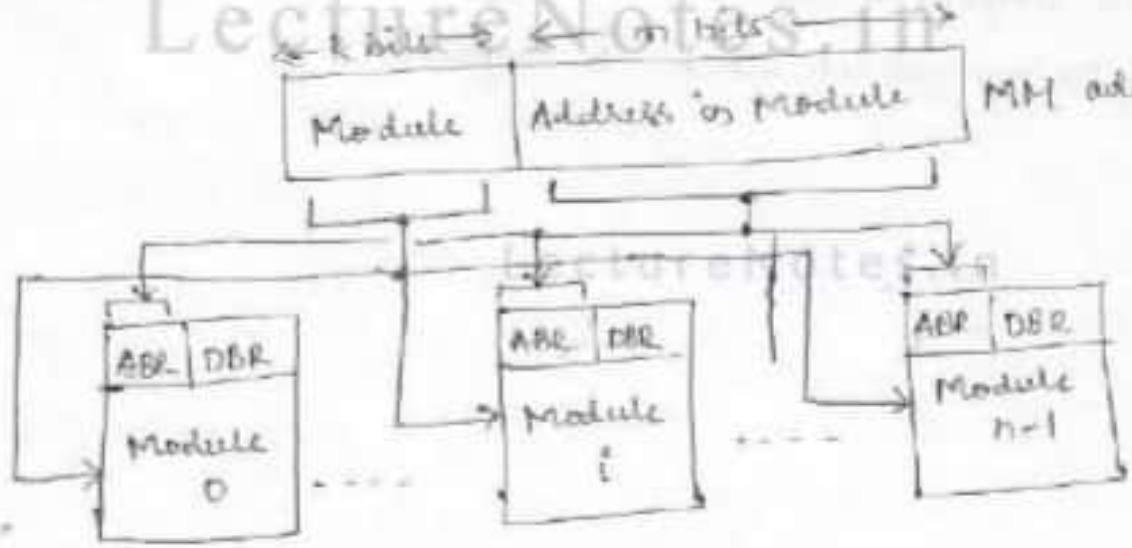
An effective way to introduce parallelism is to use an interleaved organization.

Interleaving :-

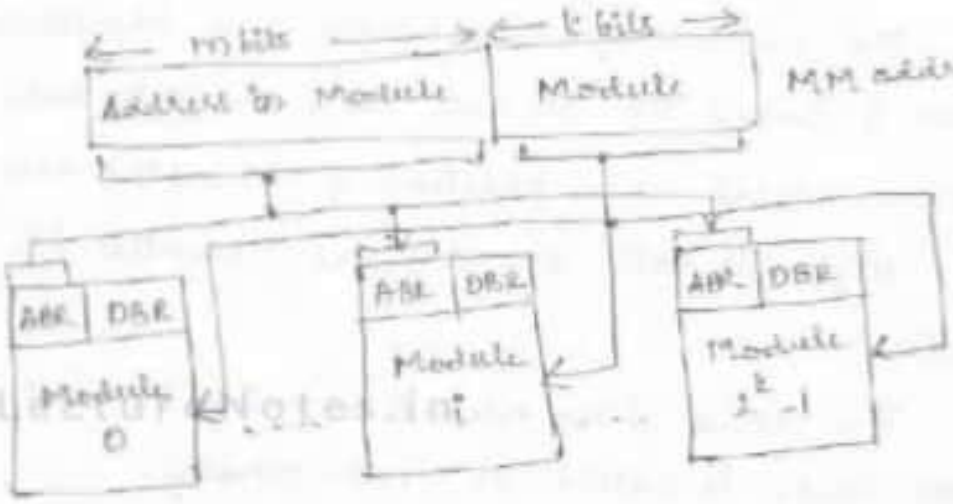
In the main memory of a computer is structured as collection of physically separate modules, each with its own address buffer register (ABR) and data buffer register (DBR).

The memory access operations may proceed in more than one module at the same time. Thus, the aggregate rate of transmission of words to and from the main memory system can be increased.

Two methods of address layout are shown below.



a) Consecutive words in a module.



b) Consecutive words in consecutive Modules

In fig (a), the memory address generated by the processor is decoded. The high-order k bits name one of 'n' modules and the low-order m bits name a particular word in that module. When consecutive locations are accessed, as when a block of data is transferred to a cache, only one module is busy at a time. However, devices with DMA ability may be accessing information in other memory modules.

The fig (b) is a more effective way to address the modules. It is called Memory interleaving. The low-order k bits of the memory address select a module, and the high-order m bits name a location within that module. The consecutive addresses are located in successive modules.

Any component of the system that generates requests for access to consecutive memory locations can keep several modules busy at any one time.

This results in both faster access to a block of data & higher avg. utilization of the memory system as a whole.

To implement the interleaved structure, there must be 2^k modules otherwise, there will be gaps of nonexistent locations in the memory address space.

Hit rate & Miss penalty :-

The number of hits stated as a fraction of all attempted accesses is called the 'hit rate' and the 'miss rate' is the number of misses stated as a fraction of attempted accesses.

High hit rates are 0.9 are essential for high-performance computers.

The extra time needed to bring the desired information into the cache is called the 'miss penalty'.

In general, the miss penalty is the time needed to bring a block of data from a slower unit in the memory hierarchy to a faster unit.

The miss penalty is reduced if efficient mechanisms for transferring data b/w the various units of the hierarchy are implemented.

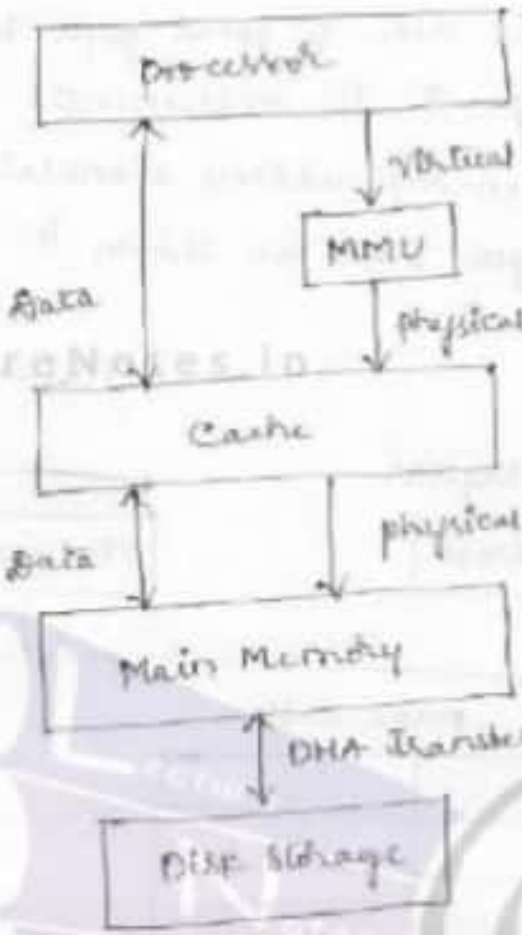
Virtual memories :-

Techniques that automatically move program and data blocks into the physical main memory, when they are required for execution is called Virtual Techniques.

The binary addresses that the processor issues for either instructions & data are called virtual or logical addresses. These addresses are translated into physical addresses by a combination of h/w & s/w components.

If a virtual address refers to a part of the program or data space that is currently in the physical memory, then the contents of the appropriate locations in the main memory are accessed immediately. If the referenced address is not in the main memory, its contents must be brought into a suitable location in the memory.

The below fig. shows a organization that implements virtual memory. A special hw unit called the memory Management



unit (MMU), translates virtual address to physical address. When the desired data (& instructions) are in the main memory the data is fetched and transfer to cache memory.

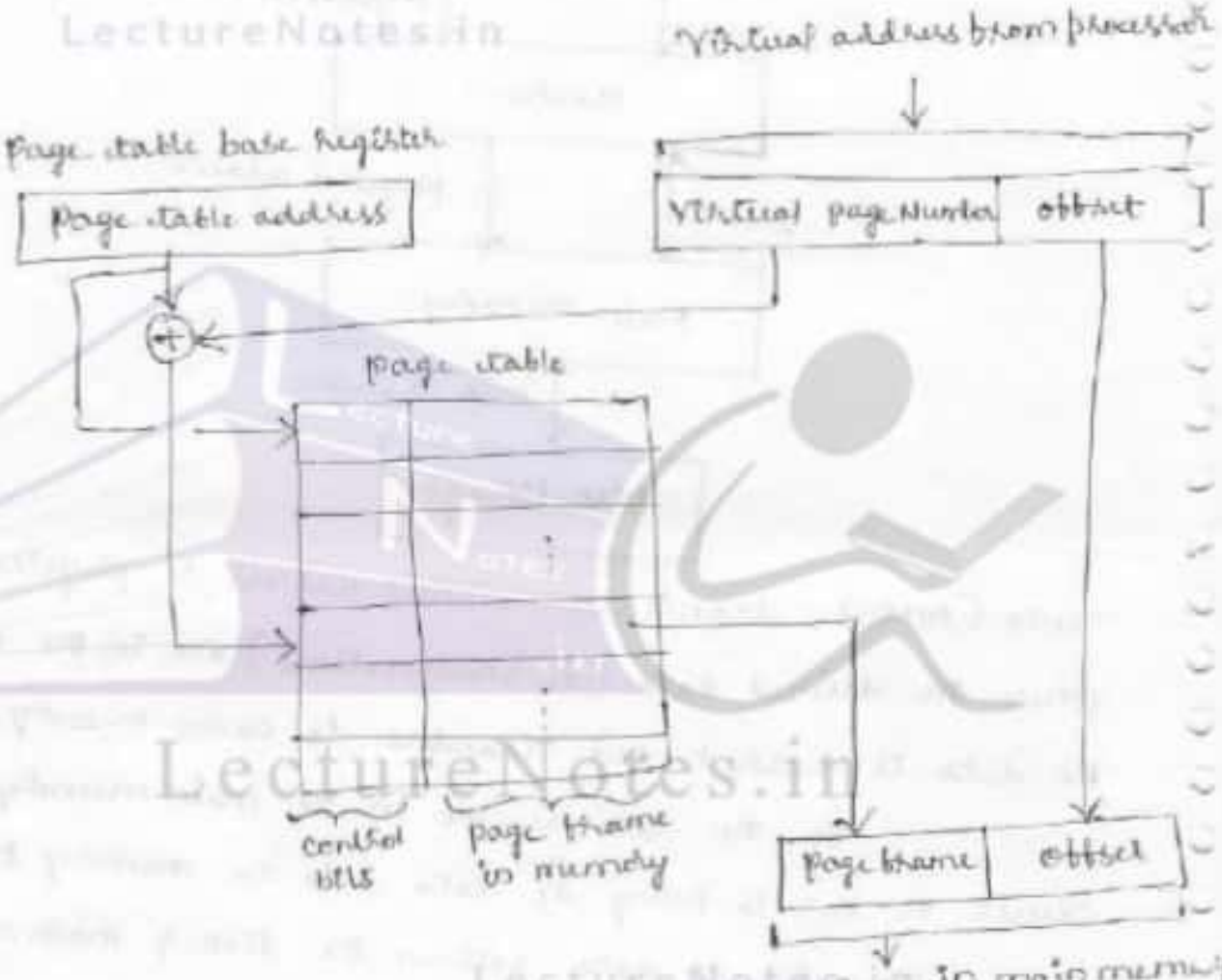
If the data are not in the main memory, the MMU causes the O.S to bring the data into the memory from the disk. Transfer of data between the disk & main memory is performed using DMA.

Address Translation :-

A simple method for translating virtual addresses into physical addresses is to assume that all programs and data are composed of fixed-length units called pages, each of which consists of a block of words that occupy contiguous locations in the main memory.

The cache memory bridges the speed gap b/w the processor and the main memory and is implemented in H/w. The virtual-memory mechanism bridges the size & speed gaps b/w the main memory and secondary storage & is implemented in S/w.

A virtual-memory address translation is based on the concept of fixed-length pages as shown in fig.



Each virtual address generated by the processor, whether it is for an instruction fetch or an operand fetch/store operation is interpreted as a virtual page Number (High-order bits) followed by an offset (Low-order bits) that specifies the location of a particular byte (or word) within a page.

This information tells the memory that can hold one page is called a 'page frame'.

The starting address of the page table is kept in a 'page table base reg'.

By adding the virtual page number to the contents of this register, the address of the corresponding entry in the page table is obtained.

Each entry in the page table also includes some control bits that describe the status of the page while it is in the main memory.

One bit indicates the validity of the page, whether the page is loaded in main memory.

Another bit indicates whether the page has been modified during residency in the memory.

Other control bits indicates various restrictions that may be imposed on accessing the page.

The page table information is used by the MMU for every read & write access, so the page table should be kept with in the MMU.

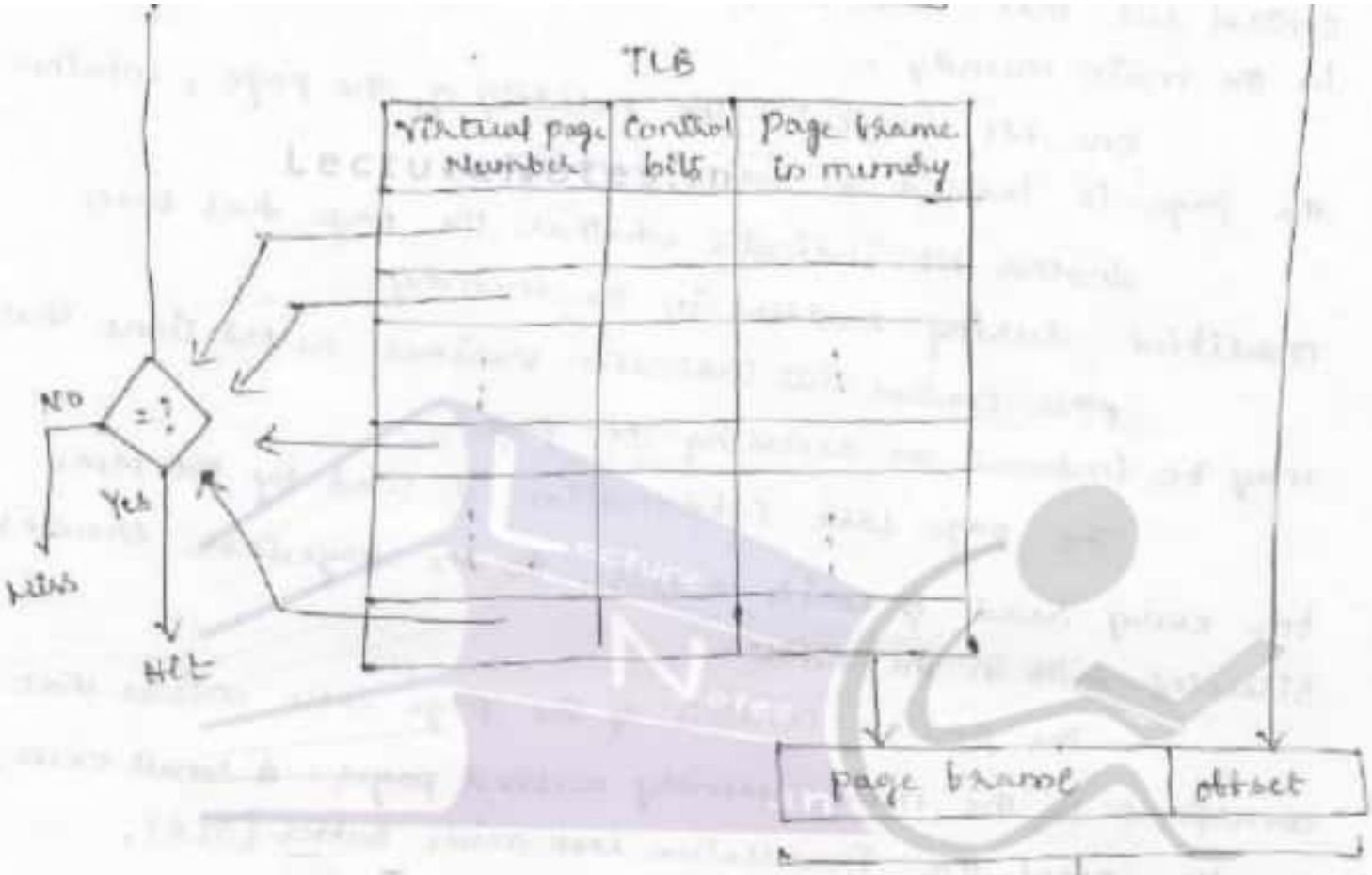
The portion consists of the page table entries that correspond to the most recently accessed pages. A small cache, usually called the Translation look aside Buffer (TLB).

The operation of the TLB with respect to the page table in the main memory is same as the cache memory. The information that constitutes a page table entry, the TLB must also include the virtual address of the entry.

The fig. shows the organization of a TLB where the associative mapping technique is used.

To contents of the TLB be coherent with the content of page tables in the memory. When the O.S changes the contents of page tables, it must simultaneously invalidate the corresponding entries in the TLB.

One of the control bits in the TLB is provided for this purpose. When an entry is invalidated, the TLB will



acquire the new information as a part of the MMU's normal response to access misses.

Address translation may follows in this way

1) Given a virtual address, the MMU looks for this referenced page in the TLB.
2) If the page table entry for this page is found in the TLB, the physical address is obtained immediately.
3) If there is a miss in the TLB, then the required entry is obtained from the page table in the main memory and the TLB is updated

When a program generates an access request to a page, if that page isn't in the main memory, then a 'page fault' is occurred.

A page fault can be detected, by the MMU asks the O.S. to intervene by raising an exception (interrupt). processing of the active task is interrupted, the control is transferred to the O.S.

The O.S. then copies the requested page from the disk into the main memory and returns control to the interrupted task. A long delay occurs while the page transfer takes place.

If a new page is brought from the disk when the main memory is full, it must replace one of the resident pages.

The problem of choosing which page to remove is critical, because main memory has larger memory. We should large portions in main memory. This will reduce the frequency of transfers to & from the disk.

Here also, the LRU replacement algorithm is used and the control bits in the page table entries can indicate usage.

One simple scheme is, the control bit is set to '1' when-ever the corresponding pages is referenced. The O.S, clears this bit in all page-table entries, thus providing a simple way of determining which have not been used recently.

A modified page has to be written back to the disk before it is removed from the main memory.

The address translation process in the MMU requires some time to perform, mostly dependent on the time needed to look up entries in the TLB.

We can reduce the average translation time by including one or more special registers that retain the virtual page number's and the physical page frame of the most recently performed translations. The information in these registers can be accessed more quickly than the TLB.

Magnetic disks :-

- Magnetic disk is a circular plate constructed of metal or plastic coated with magnetized material. The both sides of the disk are used & several disks may be stacked on one spindle with read/write heads available on each surface.

All disks rotate together at high speed and are not stopped. Bits are stored in the magnetized surface in spots along concentric circles called tracks. The tracks are commonly divided into sections called sectors. The min quantity of information which can be transferred is a sector. The subdivision of one disk surface into tracks & sectors are shown in below fig

Some units use a single read/write head for each disk surface. The track address bits are used by mechanical assembly to move the head into the specified track position before read/write. In other disk systems, separate read/write heads are provided for each track in each surface. The address bits can then select a particular track electronically through a decoder circuit.

This type of unit is more expensive & is found only in very large computer systems.

permanent Storing tracks are used in disks to synchronize the bits & recognize the sectors. A disk system is addressed by address bits that specify the disk number, disk surface, the sector number and track with in the sector.

Information transfer is very fast disks may have multiple heads & simultaneous transfer of bits from several tracks at the same time.

Disks that are permanently attached to unit & can't be removed by the user are called hard disks. A disk drive with removable disks is called a floppy disk.

Magnetic tape :-

A magnetic tape transport consists of the electrical, mechanical and electronic components to provide the parts & control mechanism for a magnetic tape unit.

The tape itself is a strip of plastic coated with a magnetic recording. Bits are recorded as magnetic spot on the tape along several tracks. Seven or nine bits are recorded simultaneously to form a character together with a parity bit. Read/write heads are mounted one in each track so that data can be recorded & read as a sequence of characters.

Magnetic tape units can be ~~started~~ stopped, started to move forward or in reverse. The information is recorded in blocks referred to as records. Gaps of unrecorded tape are inserted b/w records where the tape can be stopped. The tape starts moving while in a gap and attains its constant speed.

By reading the bit pattern at the end of the record, the control recognizes the beginning of a gap.

A tape unit is addressed by specifying the record numbers & no. of records. Records may be fixed or variable length.

RAID :-

RAID (Redundant Array of Inexpensive ~~independent~~ Disks) is a Technology where multiple independent disks are maintained. In this, a single large file is partitioned & each partition is placed in each of these independent disks. When ever any process requires these partitions, it can easily access these disks parallely & acquire the required data.

When these disks are accessing parallely, hence the given accessing time is decreased where as bandwidth of data transfer is increased.

The data belonging to a single large file, when gets partitioned then, each partition is referred as 'Stripe' and the way each of these stripes are placed in these disks is referred as Stripping.

Hence RAID is dependent on two factors.

1) Stripping of data in order to access all the disks parallely &
2) Data redundancy to increase reliability.

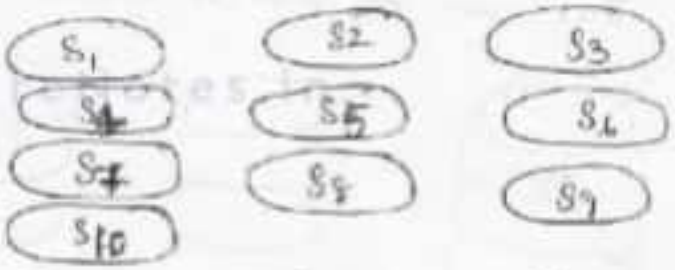
RAID has different levels which are explained below.

RAID Level 0 :-

In this level, stripes of data are belonging to the files is introduced in each disks, such that first strip is added in first disk, second strip is added in disk2 and so on. As the no. of disks ends, the same procedure is repeated by considering disk1 as initiating disk.

The stripping procedure ends only when all strips are filled in these disks. In Raid level 0, there is no provision for data redundancy.

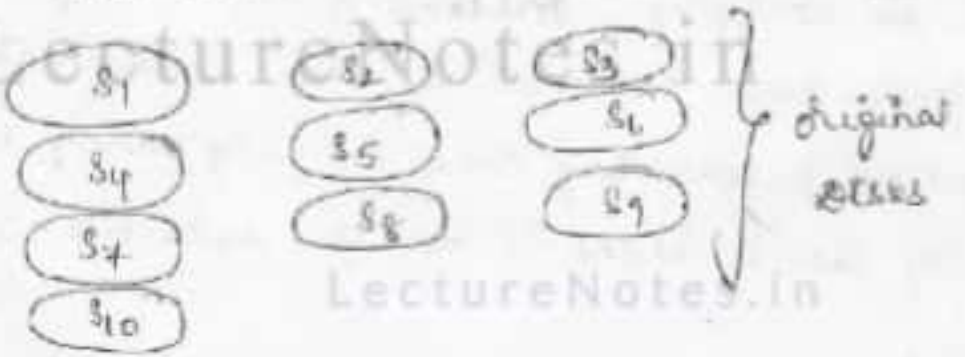
Various consequences occuring at RAID level 0 can be represented below:



RAID level 1:-

At this level data redundancy is considered to be of primary focus. RAID level 0 is applied here, inorder to achieve redundancy each physical disk is maintained twice. As a result each stripe of data will have it copy in a separate identical disk. Hence reliability can be achieved.

This can be shown below:



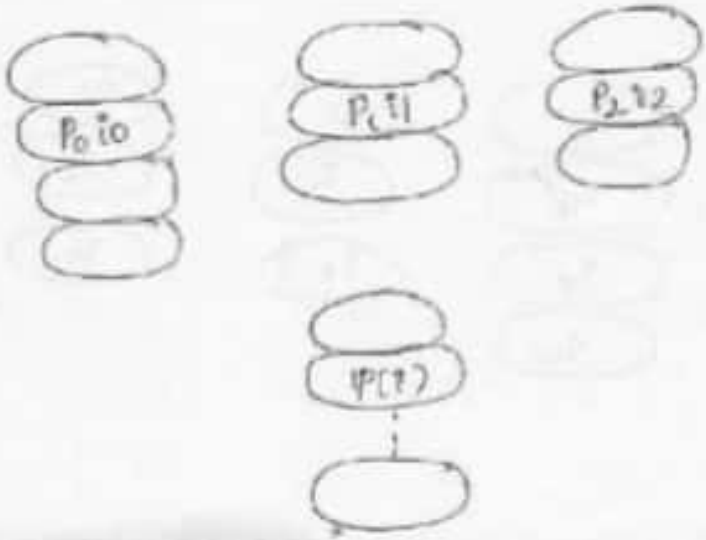
RAID level 2:

RAID level 2 is a combination of level 0 & level 1, instead of looking the frequently occuring errors, an error detecting as well as correcting code is maintained across all the disks. The frequently used error detecting code is "Hamming code.

This can be shown below

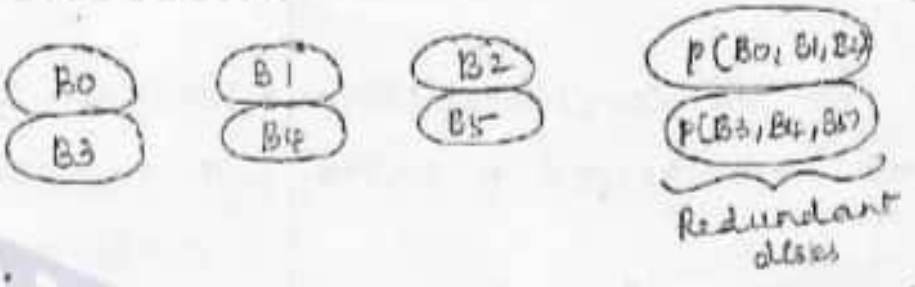
RAID level 3:

RAID level 3 resembles other levels in terms of redundancy as well as stripping. Instead of maintaing equal no. of redundant a single large redundant disk is maintained. Also, the error detecting & correcting code by using parity bit concept (extra bit) parity bit is added to all the data bits corresponding to each disk.



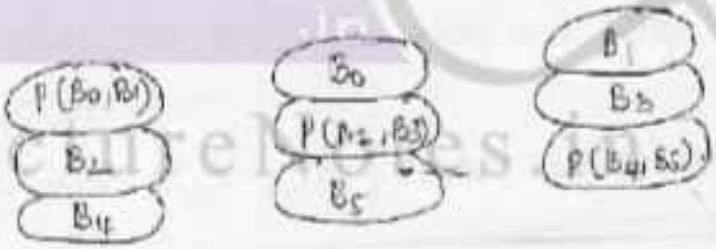
RAID level 4 :

Most of the data strips are combined to form a t of data. A single redundant disk is maintained. For ea Combination of blocks, a parity bit is added and the code placed in the redundant disk. In this level, the redunda disk with parity added data blocks turn out to be a fun



RAID level 5 :

In order to overcome the limitation of level 4 i.e., instead of placing all the parity enabled blocks in one disk, we are distributed to all the disk. Hence all the disks now responsible for maintaining parity enabled blocks.



SRAM

1) Complex ckt
2) Costlier
3) Use latch & translational FFs
4) Can be used as Cache
5) Temp storage — permanent
6) Current flow when it is accessed only
7) Hard latch
8) Less pwr consumption
9) We store automatically in latch

DRAM

1) Simple ckt
2) Cost effective
3) Capacitors used
4) Can be used as MM.
5) Temp storage
6) Charging Capacitors.
7) Hard latch
8) High pwr Consumption.
9) Store data in auxiliary memory after discharging

Synchronous DRAM	Asynchronous DRAM
* In this it is directly synchronize with clock signal	* In this it is controlled asynchronously by using timing signal
* The complexity of ckt is retained	* The ckt complexity is here
* High cost for synchronous DRAM	* Low cost for asynchronous DRAM
* Refresh counter is used to refresh the cell	* It provides flexibility in designing memory system
* Low density	* High density
* Time delay is present	* Time delay is not here
* No data lines are used	* Data lines D_0 to D_7 are used
* Over write doesn't occur	* Over write occurs
* Here add is diff but not data	* Here also the add is diff but not data.
* If over write doesn't takes place	* If data is var then only the over write of data takes place
* Here data i/p & data o/p regs are present.	* Here data i/p & data o/p regs are not present.

SRAM	DRAM
10) Manipulated by selection of bits	10) Manipulated by giving Power supply
11) Continuous pwr supply is req'd	11) Charging is req'd to capacitors
12) Fastest manipulation speed	12) Slower manipulation