LECTURE NOTES

ON

COMPUTER SYSTEM ARCHITECTURE(3RD SEM,CSE)

PREPARED BY

SUNITA MAHAPATRA

SR.LECT. IN DEPT.OF CSE (PKAIET, BGH)

CONTENTS

6. I/O Interface & Bus architecture

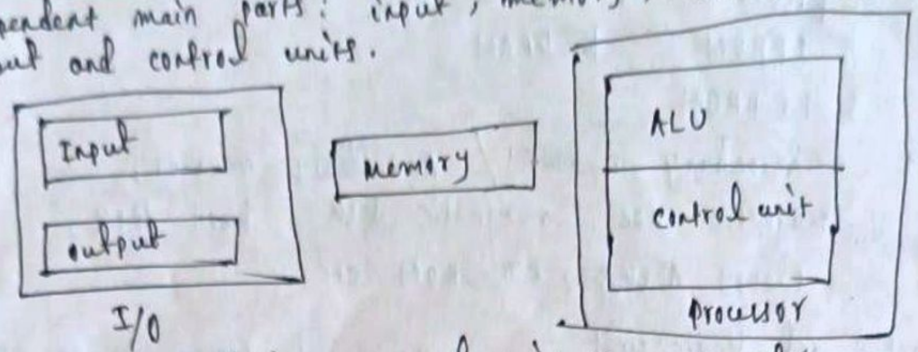
- Bus and System Bus
- Types of System Bus • Data • Address •Control
- Bus Structure
- Basic Parameters of Bus design
- SCSI
- USB

7. Parallel Processing

- Parallel Processing
- Linear Pipeline
- Multiprocessor
- Flynn’s Classification

Basic Organization of a computer :-

Functional units :- A computer consists of 5 functionally independent main parts: input, memory, arithmetic and logic, output and control units.



Basic Functional units of a computer

→ The i/p unit accepts coded information from human operators from electromechanical devices such as keyboards or from other computers over digital communication lines.

→ The information received is either stored in computer's memory for later reference or immediately used by the arithmetic and logic circuitary to perform the desired operations.

→ The processing steps are determined by a program stored in the memory.

→ Finally, the results are sent back to the outside world through the output unit.

→ All these actions are coordinated by the control unit.

→ A list of instructions that performs a task is called program. usually the program is stored in the memory.

→ The processor then fetches the instructions that make up the program from the memory, one after another and performs the desired operations.

Input unit :- computers accept coded information through input units, which read the data. The most well-known input device is the keyboard.

→ whenever a key is pressed, the corresponding letter or digit is automatically translated into its corresponding binary code and transmitted over a cable to either the memory or the processor.

→ Other i/p devices are mouse, joystics, trackballs, scanner etc.

Memory unit :- The function of the memory unit is to store programs and data. There are 2 classes of storage, called primary and secondary.

→ primary storage is a fast memory that operates at electronic speeds. Programs must be stored in the memory while they are being executed.

→ Secondary memory is the memory that provides huge amount of storage capacity but it is slower than primary memory.

→ Ex :- primary memory / main memory

- * ROM * RAM
- * PROM * SRAM
- * EPROM * DRAM
- * EEPROM

→ Ex :- Secondary memory / Auxillary memory

magnetic tape, magnetic disk, hard disk, floppy disks, CD-ROM etc..

Arithmetic and logic unit :- Most computer operations are executed in ALU of the processor. operations like addition, sub, mul, divisio or comparison of numbers is initiated by bringing the required operands into the processor, where the operation is performed by ALU

→ when operands are brought into the processor they are stored in high speed storage elements called registers.

→ Each register can store one bit of data.

output unit :- The output unit send processed results to the outside world. The most familier o/p devices are monitor, printer, speaker etc.

control unit :- The control unit controls all the operations of the computer. It guides the memory unit, I/O unit, ALU.

→ The I/O operations are controlled by the timing signals which is generated by the control unit.

→ Timing signals are signals that determine when a given action is to take place.

→ Data transfers between the processor and the memory are also controlled by the control unit through timing signals.

* The operations of a computer can be briefly described as :

→ The computer accepts information in the form of programs and data through an input unit and stores it in memory.

→ Information stored in the memory is fetched, under program control into an arithmetic and logic unit where it is processed.

→ processed information leaves the computer through an o/p unit.

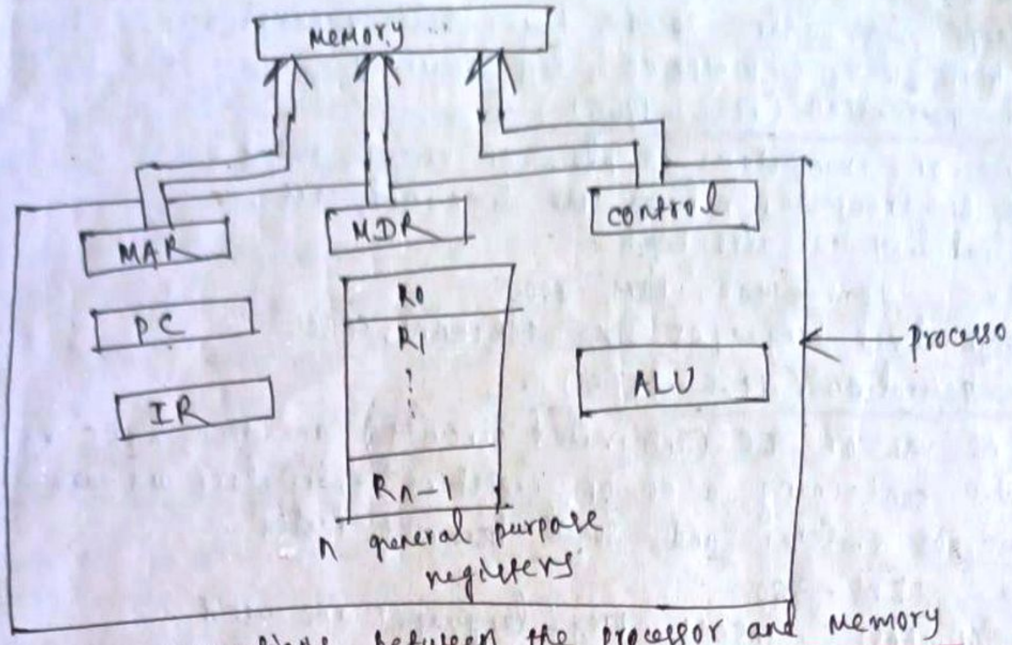
→ All activities inside the machine are directed by the control unit.

Basic operational concepts :- In addition to ALU and CU, the processor contains a no. of registers used for several diff. purpose.

→ The instruction register (IR) holds the instruction that is currently being executed.

→ The program counter (PC) is a register contains the address of the next instructions to be fetched from memory.

→ processor also having n-general purpose registers.



connections between the processor and memory

→ MAR and MDR are two registers performs communication with the memory.

→ MAR holds the address of the location to be accessed.

→ MDR contains the data to be written into or read out of the addressed location.

→ programs reside in the memory and usually get there through the input unit. Execution of the program starts when the PC is set to point to the 1st instructions of the program. Then the contents of the PC is transferred to MAR and a read signal is sent to memory.

→ Then the addressed word is read out of the memory and loaded into the MDR, next the contents of MDR are transferred to IR.

→ At this point, the instruction is ready to be decoded and executed.

→ If the instruction involves an operation to be performed by the ALU, it is necessary to obtain the required operands.

→ If operands are reside in memory then that could be obtained by the same way as the instructions.

→ After obtaining the operand then the ALU perform the operation and then the result is sent back to the memory through a write signal and through the MDR and MAR.

Generation of computers

First generation (1942 -1955) :- First generation computers were marked by use of vaccum tubes.

→ punched cards was common i/p media.

→ prog. are written in machine level language.

→ These computers suffers from the drawbacks line large size, slow speed, heavy heat generation, high power consumption, less accuracy.

Second generation (1956 – 1965) :-

→ use of transistors in place of vaccum tubes.

→ use of transistor reduces the size and cost.

→ Reliability is also high.

→ Ex :- IBM – 1400, IBM – 7000

→ High level languages like FORTRAN, COBOL

Third generation (1966 – 1975) :-

→ Here use of IC (integrated circuits) are there. IC is a chip containing a no. of registers, transistors and capacitors.

→ Size is smaller and speed becomes faster.

→ Ex: IBM – 360.

→ COBOL, BASIC, PASCAL prog. languages are used.

Fourth generation (1975 – onwards) :-

→ use of LSI and VLSI techniques and also microprocessor.

→ Small size, lower cost, faster and high reliability.

→ New S/w are used.

→ Floppy disks and optical disks are used in these computers.

Fifth generation :-

→ These computers are characterised by use of parallel processing & presence of artificial intelligence.

→ use of hard disks, CD-ROM are there.

Introduction to program translation :-

Assembler :- Assembler is a program that translates the assembly language to machine level language so that the computer can carry out the instructions. In assembly language programmers use short letter codes.

compilers :- It is a system S/w that is used to translate high level language to machine level language.

Interpreter :- It is also a system S/w that is used to convert high level language to machine level language.

compiler	Interpreter
→ Translates entire prog. at a time.	→ Translates the prog. line by line.
→ Require more memory.	→ uses less memory.
→ If error is detected nothing is executed.	→ If error is encountered execution stops but previous statements are already executed.
→ Execution time is less.	→ Execution time is more.

Bus architecture :- A typical digital computer has many registers and paths must be provided to transfer information from one register 2 another.

→ An efficient scheme for transferring information between registers in a multiple-register configuration is a common bus system.

→ A bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one at a time.

→ control signals determine which register is selected by the bus during each particular register transfer.

→ Typically, a bus consists of multiple communication pathways or lines. Each line is capable of transmitting signals representing binary 1 and binary 0.

→ computer systems contain a no. of different buses that provide pathways between components at various levels of the computer system hierarchy.

→ A bus that connects major computer components (processor, memory, I/O) is called a system bus.

Bus structure :- A system bus consists, typically of from about 50 to 100 separate lines. Each line is assigned a particular meaning or function.

→ On any bus the lines can be classified into 3 functional groups.

They are address bus, data bus and control bus.

Data bus

→ The data lines provide a path for moving data between system modules. These lines are collectively called the data bus.

→ The no. of lines being refferred as width of data bus. Each line can carry only 1 bit of data at a time.

Address bus :- The address lines are used to designate the source or destination of the data on the data bus. For e.g., if the processor wishes to read a word of data from memory, it puts the address of the desired word on the address lines.

control bus :- The control lines are used to control the access to and the use of the data and address lines.

→ control signals transmit both command and timing information between system modules.

Typical control lines include the following:-

Memory write :- causes data on the bus to be written into the addressed location.

Memory read :- causes data from the addressed location to be placed on the bus.

I/O write :- causes data on the bus to be O/P to the addressed I/O port.

I/O read :- causes data from the addressed I/O port to be placed on the bus.

Transfer ACK :- Indicates that data have been accepted from or placed on the bus.

Bus request :- Indicates that a module needs to gain control of the bus.

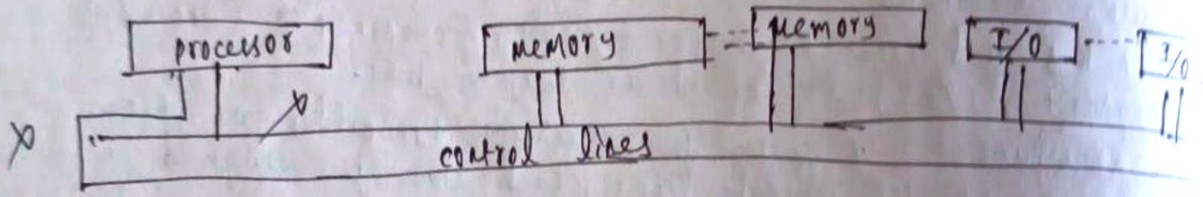
Bus grant :- Indicates that a requesting module has been granted control of the bus.

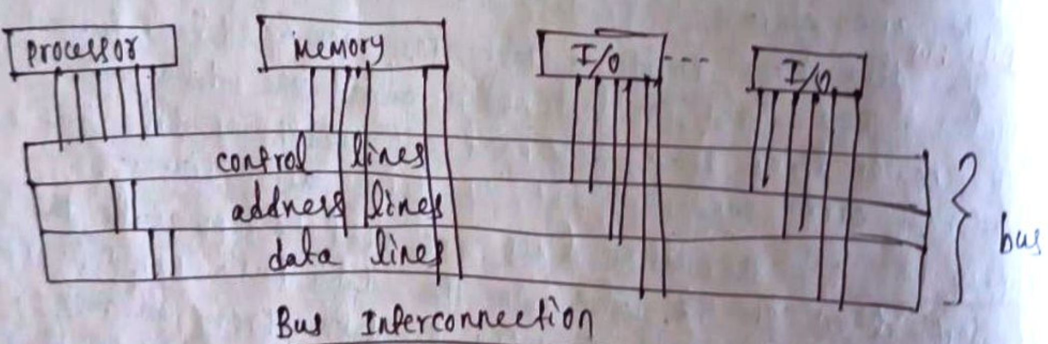
Interrupt request :- Indicates that an interrupt is pending.

Interrupt ACK :- Acknowledges that the pending interrupt has been recognized.

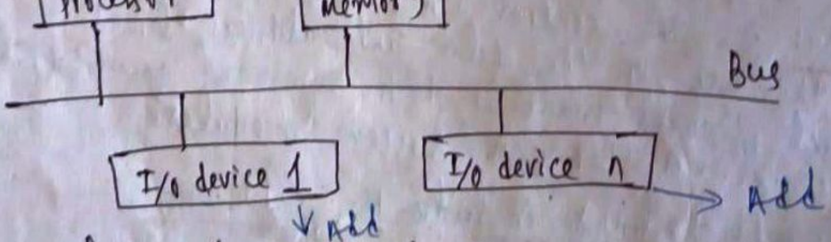
clock :- used to synchronize operations.

Reset :- Initializes all modules.





Bus Interconnection



→ If one module wishes to send data to another it must do two things (1) obtain the use of bus (2) transfer data via the bus.

→ Physically, the system bus is actually a no. of parallel electrical conductors.

→ A Simple arrangement to connect I/o devices to a computer is to use a single bus arrangement. The bus enables all the devices connected to it to exchange information.

→ Each I/o device is assigned a unique set of addresses. when the processor places a particular address on the address lines the device that recognize the address respond to the commands issued on the control lines.

→ The processor requests either a read or a write operation and the requested data are transferred over the data lines.

→ when the I/o devices and the memory share the same address space, the arrangement is called memory-mapped I/o.

CHAPTER-2 Functional units

Combinational circuits :- A combinational circuit is a connected arrangement of logic gates with a set of inputs and outputs. At any given time, the binary values of the outputs are a function of the binary combination of the inputs.

Ex:- Half adder, full adder, multiplexers, decoders etc.

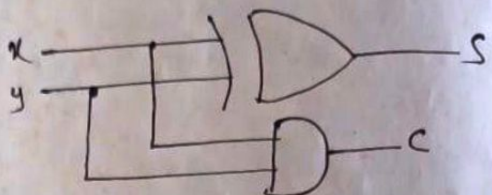
Half adder :- It is a combinational circuit that performs the arithmetic addition of two bits.

$s = sum = x'y + xy' = x \oplus y$

$C = carry = xy$

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(truth table)



(logic diagram)

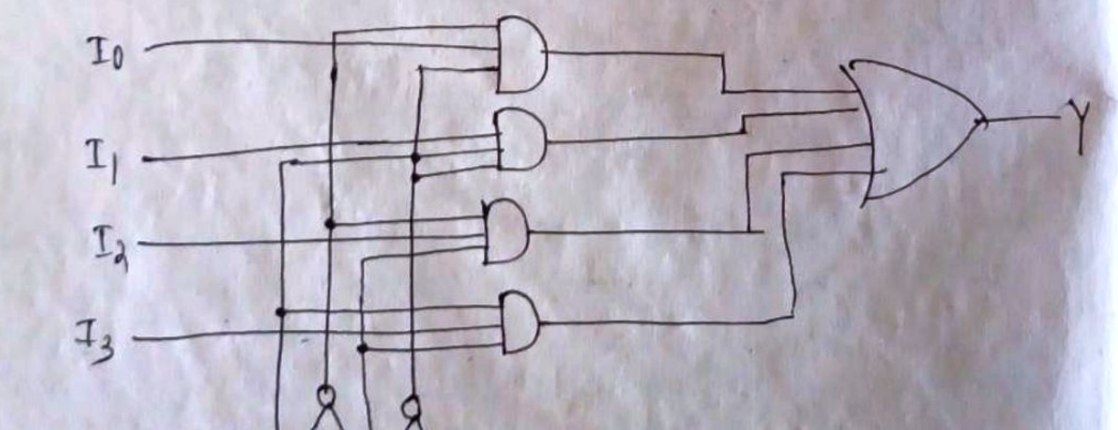
Full adder :- It is a combinational circuit that performs the arithmetic sum of three input bits. It is having three inputs and two outputs.

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

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

Multiplexers :- It is a combinational circuit that receives binary information from one of 2^n input data lines and directs it to a single output line. The selection of a particular input data line for the output is determined by a set of selection inputs.

→ A 2^n-to-1 multiplexer has 2^n input data lines and n input selection lines whose bit combinations determine which input data are selected for the o/p.



4-to-1 line multiplexer

Select		O/P
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

← Function table

→ Here It has 4 inputs I_0, I_1, I_2, I_3 and 2 selection i/p S_0 and S_1 and the 4 i/ps are applied to one i/p of an AND gate.

→ S_1 and S_0 are decoded to select a particular AND gate.

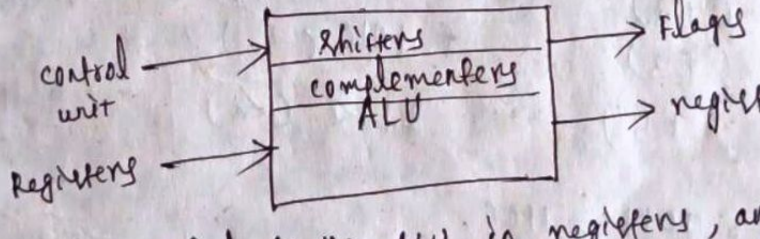
→ The O/Ps of the AND gates are applied to a single OR gate to provide the single O/P.

Decoders :- A decoder is a combinational circuit that converts binary information from the n coded i/ps to a maximum of 2^n unique outputs. If the n-bit coded information has unused bit combinations the decoder may have less than 2^n outputs.

Arithmetic and logic unit :- The ALU is the part of the computer that actually performs arithmetic and logical operations on data.

→ All of the other elements of the computer system - control unit, registers, memory, I/O - are there mainly to bring data into the ALU for its processing and then take the results back.

→ ALU is based on the use of simple digital logic devices that can store binary digits and perform simple boolean logic operations.



→ Data are presented to the ALU in registers, and the results of an operation are stored in registers.

→ These registers are temporary storage locations within the processor that are connected by signal paths to the ALU.

→ The ALU may also set flags as the result of an operation. for e.g. an overflow flag (condition code) is set to 1 if the result of a computation exceeds the length of the register into which it is to be stored.

→ The flag values are also stored in registers within the processor.

→ The control unit provides signals that control the operation of the ALU and the movement of the data into and out of ALU.

→ The ALU also performs the complement operations as well as the shift operations.

→ The ALU performs both the preshift and postshift operations

Encoding of ALU operations

Operation	Symbol	Operation	Symbol
Transfer A →	TSFA	OR A and B →	OR
Increment A →	INCA	XOR A and B →	XOR
ADD A+B →	ADD	complement A →	COMA
Subtract A−B →	SUB	shift right A →	SHRA
Decrement A →	DECA	shift left A →	SHLA
AND A and B →	AND		

Number representation :— Sign magnitude form

positive number → +5 → 0101
+6 → 0110

negative number → −6 → 1110
−5 → 1101

1's complement form → +5 → 0101
1's comp. form is → 1010 → −5

+3 → 0011
1's comp → 1100 is equivalent to −3.

2's comp → 1101

Addition & substraction :—

```
①   0010  (+2)          ②   0100 (+4)          +6 → 0110
    0011  (+3)             + 1010 (−6)             → 1001
    ---------              -----------                +1
    0101  (+5)               1110 (−2)               ----
                                                     1010

                                                     1101
                                                       +1
                                                     ----
                                                     1110
```

```
③  +3 → 0011 → 1100
                  +1
                ----
                1101 → −3

   +7 → 0111 → 1000
                  +1
                ----
                1001 → −7

     1101  (−3)                 1101
   − 1001  (−7)   =>          + 0111
   ------------              --------
                             [1]0100 → (+4)
```

```
④  +7 → 0111 → 1's → 1000
                2's → 1001 (−7)

   +5 → 0101 → 1's → 1010
                2's → 1011  (−5)
```

$$\begin{array}{rl} & 1001 \; (-7) \\ - & 1011 \; (-5) \\ \hline \end{array} \Rightarrow \begin{array}{rl} & 1001 \\ + & 0101 \\ \hline & 1110 \; (-2) \end{array}$$

(5)
$$\begin{array}{rl} & 1011 \quad (-5) \\ + & 1110 \quad (-2) \\ \hline & \boxed{1}1001 \quad (-7) \end{array}$$

Instruction formats :- A computer will usually have a variety of instruction code formats. It is the function of the control unit within the CPU to interpret each instruction code and provide the necessary control functions needed to process the instruction.

→ The format of an instruction is usually depicted in a rectangular box symbolizing the bits of the instruction as they appear in memory words or in a control register.

→ The bits of the instruction are divided into groups called fields.

→ The common fields found in instruction formats are:

* An operation code field that specifies the operation to be performed.

* An address field that designates a memory address or a processor register.

* A mode field that specifies the way the operand or the effective addres is determined.

→ The opcode field of an instruction is a group of bits that define various processor operations such as add, subtract, complement, and shift.

→ The bits that define the mode field of an instruction code specify a variety of alternatives for choosing the operands from the given address.

→ Operations specified by computer instructions are executed on some data stored in memory or processor registers. Operands residing in memory are specified by their memory address.

→ Operands residing in memory are specified with a register address. A register address is a binary number of k bits that defines one of 2^k registers in CPU. Thus a CPU with 16 processor registers R_0 through R_{15} will have a register address field of four bits. The binary number 0101 designate register R_5.

→ computers may have instructions of several different lengths containing varying no. of addresses. The no. of address fields in the instruction format of a computer depends on the internal organization of its registers.

→ More computers fall into one of three types of CPU organization.

- * Single accumulator organization.
- * General register organization
- * Stack organization.

Single Accumulator organization :- Here all operations are performed with the accumulator register. The instruction format in this type of computer uses one address field.

Ex :- ADD X Here X is the address ~~[illegible]~~ of the operand. The result of this will be $AC \leftarrow AC + M[X]$

AC → Accumulator register

M[X] → Memory word located at address X.

General register organization :- The instruction format in this type of computer needs three register address fields.

Ex :- ADD R_1, R_2, R_3

result : $R_1 \leftarrow R_2 + R_3$

The no. of register could be reduced to two if the destination register is the same as one of the source registers

ADD R_1, R_2 result : $R_1 \leftarrow R_1 + R_2$

Stack organization :- ~~The Stack organized~~ computers with stack organization would have PUSH and POP instructions which require an address field.

Ex :- PUSH X will push the word at address X to the top of the stack.

The instruction in a stack computer consists of an operation code with no address field. This operation has the effect of popping the top 2 numbers from the stack, adding the numbers and pushing the sum into the stack.

→ There is no need to specify operands with an address field since all operands are implied to be in the stack.

Types of instructions

Three-address Instructions :- computers with three-address instructions formats can use each address fields to specify either a processor register or a memory operand.

Ex $X = (A+B) * (C+D)$

ADD R_1, A, B $R_1 \leftarrow M[A] + M[B]$

ADD R_2, C, D $R_2 \leftarrow M[C] + M[D]$

MUL X, R_1, R_2 $M[X] \leftarrow R_1 * R_2$

It is assumed that the computer has 2 processor registers R_1 and R_2. The symbol M[A] denotes the operand at memory address symbolized by A.

→ The advantage of the three-address format is that it results in short programs when evaluating arithmetic expressions.

→ The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

Two-Address instructions :- Two-address instructions are the most common in commercial computers. Here again each address field can specify either a processor register or a memory word.

Ex :- X = (A+B) * (C+D)

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

The MOV instruction moves or transfers the operands to and from memory and processor registers. Here the register is considered as both source and destination where the result of the operation is transferred.

One-Address Instructions :- One-address instructions use an accumulator (AC) register for all data manipulation. For mul and division there is a need for a second register. However, here we will neglect the second register and assume that the AC contains the result of all operations.

X = (A+B) * (C+D)

Ex :

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

All operations are done between the AC registers and a memory operand. T is the address of a temporary memory location required for storing the intermediate results.

Zero-address Instructions :- A stack-organized computer does not use an address field for the instructions ADD and MUL. The PUSH and POP instructions, however, need an address field to specify the operand that communicates with the stack.

Ex : X = (A+B) * (C+D)   TOS — Top of the stack

```
PUSH A     TOS ← A
PUSH B     TOS ← B
ADD        TOS ← (A+B)
PUSH C     TOS ← C
PUSH D     TOS ← D
ADD        TOS ← (C+D)
```

```
MUL   TOS ← (C+D)*(A+B)
POP   X    M[X] ← TOS
```

→ The name "zero address" is given to this type of computer because of the absence of an address field in the computational instructions.

RISC Instructions :- The instruction set of a typical RISC processor is restricted to the use of load and store instructions when communicating between memory and CPU.

→ A program for a RISC-type CPU consists of LOAD and STORE instructions that have one memory and one register address and computational-type instructions that have three addresses with all three specifying processor registers.

$$X = (A+B) * (C+D)$$

```
LOAD   R1, A        R1 ← M[A]
LOAD   R2, B        R2 ← M[B]
LOAD   R3, C        R3 ← M[C]
LOAD   R4, D        R4 ← M[D]
ADD    R1, R1, R2   R1 ← R1+R2
ADD    R3, R3, R4   R3 ← R3+R4
MUL    R1, R1, R3   R1 ← R1*R3
STORE  X, R1        M[X] ← R1
```

→ The load instructions transfer the operands from memory to CPU registers. Add and MUL operations are executed in registers and the result is then stored in memory with a store instruction.

Addressing Modes :- The way the operands are choosen during program execution is dependent on the addressing mode of the instruction.

→ The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.

→ computers use addressing mode techniques for the purpose of accommodating one or both of the following provisions:-

* To give programming versatility to the user by providing such facilities as pointers to memory, counters for loop control, indexing of data, and program relocation.
* To reduce the number of bits in the addressing field of the instruction.

→ An instruction cycle has three phases.

* Fetch the instruction from memory.
* Decode the instruction
* Execute the instruction.

Opcode	Mode	Address

(Instruction format with mode field)

→ The operation code specifies the operation to be performed.

→ The mode field is used to locate the operands needed for the operation.

→ The address field is optional and it may designate a memory address or a processor register.

→ Most addressing modes modify the address field of the instructions. But there are two modes that need no address field at all. These are implied and immediate modes.

Implied Mode :- In this mode the operands are specified implicitly in the definition of the instruction.

→ Zero-address instructions in a stack organized computer are implied-mode instructions since the operands are implied to be on top of the stack.

Immediate Mode :- Here operand is specified in the instruction itself. In other words, an immediate-mode instruction has an operand field rather than an address field.

→ Immediate-mode instructions are useful for initializing registers to a constant value.

Register Mode :- In this mode the operands are in registers that reside within the CPU. The particular register is selected from a register field in the instruction. A k-bit field can specify any one of 2^k registers.

Register Indirect Mode :- In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory. In other words the selected register contains the address of the operand rather than the operand itself.

→ The advantage of a register indirect mode instruction is that the address field of the instruction uses fewer bits to select a register than would have been required to specify a memory address directly.

Autoincrement or Autodecrement Mode :- It is similar to the register indirect mode except that the register is incremented or decremented after its value is used to access memory.

→ It uses the increment or decrement instructions. Some computers incorporate a special mode that automatically increments or decrements the content of the register after data access.

Effective Address :- The effective address is defined to be the memory address obtained from the computation dictated by the given addressing mode. The effective address is the address of the operand in a computational type instruction.

Direct Address Mode :- Here the effective address is equal to the address part of the instruction. The operand resides in memory and its address is given directly by the address field of the instruction.

Indirect Address Mode :- Here the address field of the instruction gives the address where the effective address is stored in memory.

→ control fetches the instruction from memory and uses its address part to access memory again to read the effective address.

→ The effective address in these modes is obtained from the following computation.

effective address = address part of instruction + content of CPU register.

Relative Address Mode :- Here the content of the program counter is added to the address part of the instruction in order to obtain the effective address.

→ The address part of the instruction is usually a signed number which can be either positive or negative.

→ when this number is added to the content of the program counter, the result produces an effective address whose position in memory is relative to the address of the next instruction.

Indexed Addressing Mode :- In this mode the content of an index register is added to the address part of the instruction to obtain the effective address.

→ The index register is a special CPU register that contains an index value. The index register stores the distance between the beginning address and address of the operand in the array.

Base Register Addressing Mode :- In this mode the content of a base register is added to the address part of the instruction to obtain the effective address.

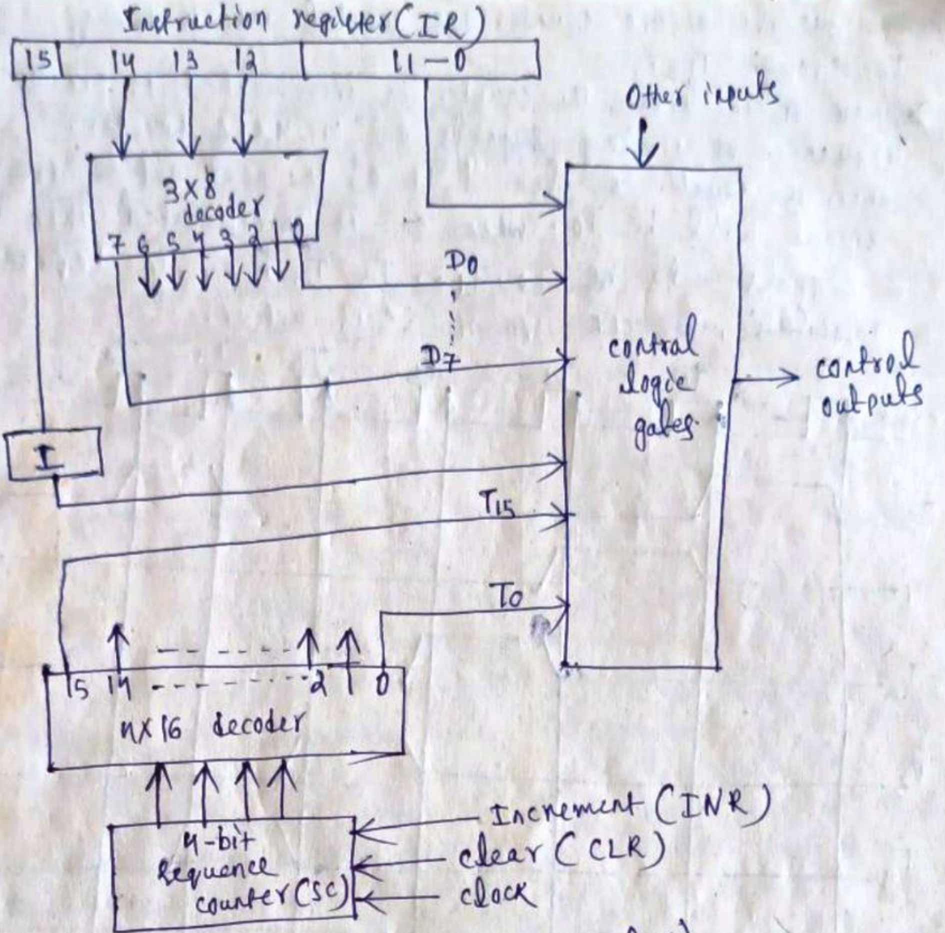
→ It is similar to the indexed addressing mode. The base register contains the base address and the address field of the instruction gives a displacement relative to this base address.

→ It is used in computers to fascilitate the relocation of programs in memory.

Control unit :- The timing for all registers in the basic computer is controlled by a master clock generator.

→ The clock pulses are applied to all flip-flops and registers in the system, including flip-flops and registers in the control unit

→ The control signals are generated in the control unit and provide control inputs for the multiplexers in common bus, control inputs in processor registers and microoperations for the accumulator.



(control unit of basic computer)

→ There are 2 major types of control organization: hardwired control and microprogrammed control.

→ In hardwired organization, the control logic is implemented with gates, flip-flops, decoders and other digital circuits. It has the advantage that it can be optimized to produce fast mode of operatin

→ In the microprogrammed organization, the control information is stored in a control memory. The control memory is programmed to initiate the required sequence of microoperations.

→ In hardwired control if we want any modifications then the changes in the wiring among the various components is required.

→ In microprogrammed control the updations and modifications are done through the microprograms.

Description :- The control unit consists of two decoders, a sequence counter, and a no. of logic gates and IR.

→ The instruction register is divided into three parts: the I bit, the operation code, and bits 0 to 11. The operation code in bits 12 through 14 are decoded with a 3X8 decoder. The o/ps of the decoder are designated by the symbols D0 through D7.

→ Bit 15 of the instruction is transferred to a flip-flop designated by the symbol I. Bits 0 through 11 are applied to control logic gates.

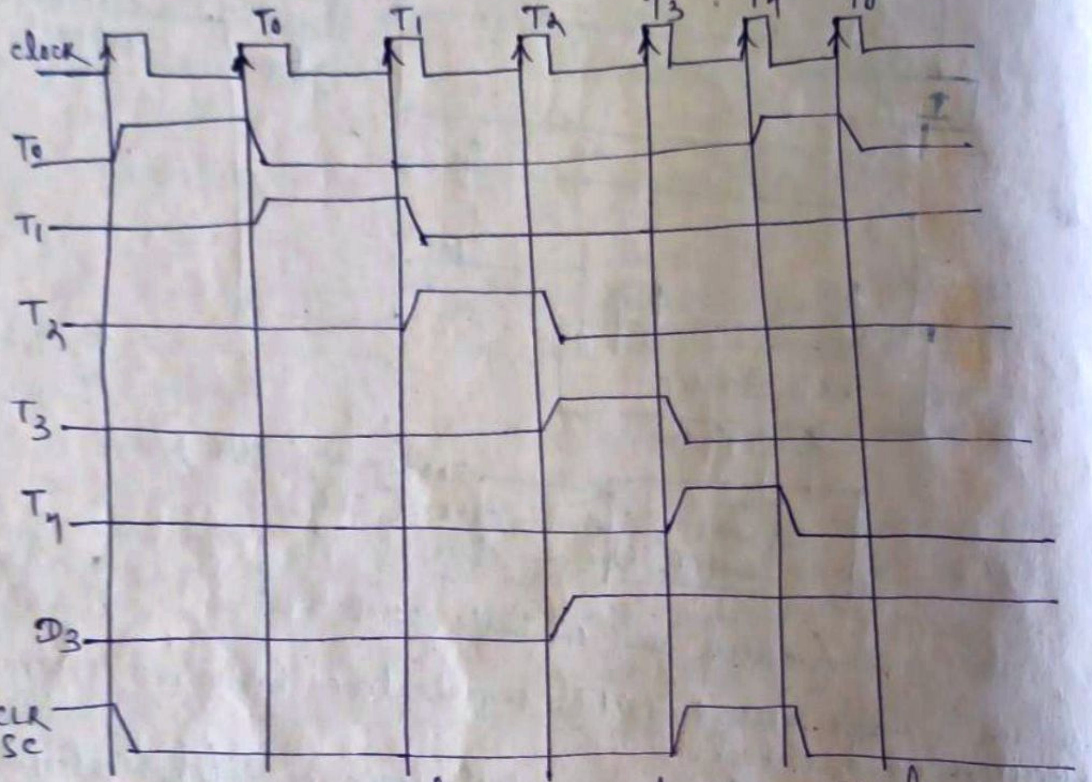
→ The 4-bit sequence counter can count in binary from 0 through 15.

→ The outputs of the counter are decoded into 16 timing signals T_0 through T_{15}.

→ Most of the time, the counter is incremented to provide the sequence of timing signals out of 4X16 decoder.

→ When the counter is cleared to 0, the next active timing signal will be T_0. When SC is incremented then the timing signals will be T_0, T_1, T_2, T_3, T_4 in sequence.

→ Initially, the CLR i/p of SC is active.



Example of control timing signals

→ The first positive transition of the clock clears SC to 0, which in turn activates the timing signals T_0 out of the decoder. T_0 is active during one clock cycle.

→ SC is incremented with every +ve clock transition, unless its CLR input is active. This produces timing signals $T_0, T_1 --- T_{15}$ and back to T_0.

→ A memory read or write cycle will be initiated with the rising edge of a timing signal. It will be assumed that a memory cycle time is less than the clock cycle time.

→ According to this assumption, a memory read or write cycle initiated by a timing signal will be completed by the time the next clock goes through its positive transition.

→ The clock transition is then be used to load the memory word into a register.

MEMORY ORGANIZATION chapter-3

Memory :— The memory unit is an essential component in any digital computer since it is needed for storing programs and data. There are broadly 2 types of memory are there.

* Main Memory * Auxiliary Memory

Main Memory :- The main memory is the central storage unit in a computer system. It can communicate directly with the CPU.

→ It is relatively large and fast memory used to store programs and data during the computer operation. It is based on semiconductor IC.

→ main memory consists of 2 parts RAM IC chips and ROM chips.

→ RAM :- Integrated circuit RAM chips are available in two possible operating modes, static and dynamic.

SRAM :— The static RAM consists of internal flip-flops that store the binary information. The stored information remains valid as long as power is applied to the unit. It is easier to use and has shorter read and write cycles.

DRAM :— 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 the capacitors must be periodically recharged by refreshing the dynamic memory.

→ The dynamic memory offers reduced power consumption and large storage capacity in a single memory chip.

ROM (Read only Memory) :— ROM is used for storing programs that are permanently resident in the computer.

→ It contains a permanent pattern of data that can't be changed.

→ A ROM is nonvolatile, that is no power source is required to maintain the bit values in memory.

→ It is possible to read a ROM but not possible to write new data into it.

→ The advantage of ROM is that the data or program is permanently in main memory and need never be loaded from a secondary memory.

→ A ROM is created like any other integrated circuit chip, with the data actually wired into the chip as part of the fabrication process.

Programmable ROM :— When only a small no. of ROMs with a particular memory content is needed, a less expensive alternative is PROM. Like ROM the PROM is nonvolatile and may be written into only once.

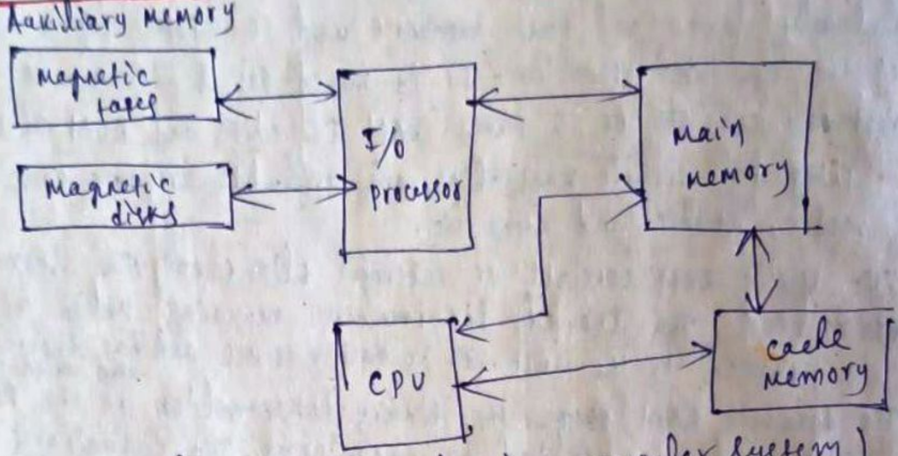
EPROM :— The erasable programmable read-only memory is read and written electrically, as PROM. However, before a write operation, all the storage cells must be erased to the same initial state by exposure of the packaged chip to UV radiation.

EEPROM :— This is a read-mostly memory that can be written into at any time without erasing prior contents only bytes address are updated. The write operation takes considerably longer than the read operation, on the order of several hundred microseconds per byte.

AUXILIARY MEMORY :- Devices that provide backup storage are called auxiliary memory. The most common devices used in computer systems are magnetic disks, magnetic tapes, hard disks, floppy disks etc.

→ They are used for storing system programs, large data files and other backup information. 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

Memory Hierarchy :-



(Memory Hierarchy in a computer system)

Cache Memory :- 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 to compensate the mismatch in operating speeds of main memory access time and processor logic.

→ It is a extremely fast memory whose access time is close to processor logic clock cycle time.

→ The cache is used for storing segments of programs currently being executed in the CPU and temporary data frequently needed in present calculations.

Description :- (Memory Hierarchy) The I/O processor manages data transfer between auxiliary memory and main memory and the cache organization is concerned with the transfer of information between main memory & CPU. Each involved with a different level in the memory hierarchy system.

→ Auxiliary and cache memories are used for different purposes.

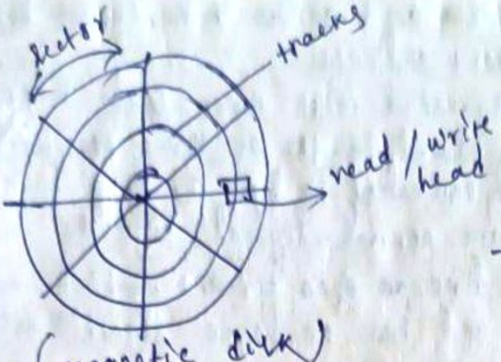
→ The cache holds those parts of the program and data that are most heavily used, while the auxiliary memory holds those parts that are not presently used by the CPU.

Auxiliary memory (magnetic disks)

A magnetic disk is a circular plate constructed of metal or plastic coated with magnetized material.

→ Both sides of the disk are used and several disks may be stacked on the spindle with read/write heads available on each surface.

→ All disks rotate together at high speed.



(magnetic disk)

→ Bits are stored in the magnetized surface in spots along concentric circles called tracks.

→ The tracks are commonly divided into sections called sectors.

→ In most systems the minimum quantity of information which can be transferred is a sector.

→ Some units use a single read/write head for each disk surface.

→ A disk system is addressed by address bits that specify the disk number, the disk surface, the sector no. and the track within sector.

→ Disks that are permanently attached to the unit assembly and cannot be removed by the occasional user are called hard disks.

→ A disk drive with removable disks is called a floppy disk.

Magnetic Tape :- A magnetic tape consists of electrical, mechanical and electronic components to provide the parts and control mechanism for a magnetic tape unit.

→ The tape itself is a strip of plastic coated with a magnetic recording medium. Bits are recorded as magnetic spots on the tape along several tracks. Usually, seven or nine bits are recorded simultaneously to form a character together.

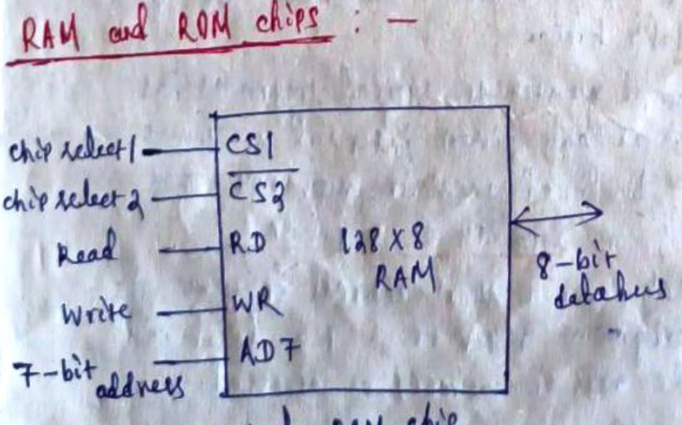
→ Read / write heads are mounted one in each track.

→ Magnetic tape units can be stopped, started to move forward or in reverse or can be rewind. However, they can't be started or stopped fast enough between individual characters. For this reason, information is recorded in blocks reffered to as records.

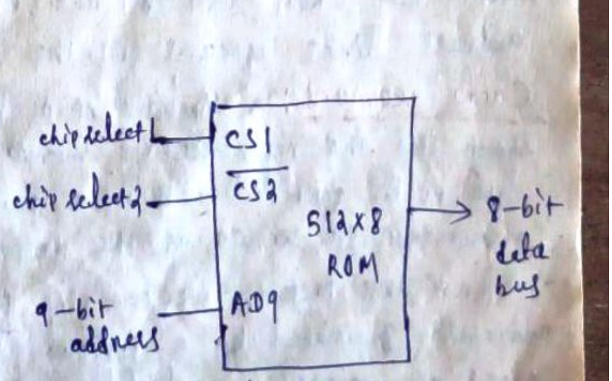
→ A tape unit is addressed by specifying the record number and the number of characters in the record.

→ Records may be of fixed or variable length.

RAM and ROM chips : -



Typical RAM chip



Typical ROM chip

RAM chip :- A RAM chip is better suited for communication with the CPU if it has one or more control inputs that select the chip only when needed.

→ Another common feature is a bidirectional data bus that allows both read and write operation between memory and CPU.

→ A bidirectional bus can be constructed with three-state buffers. A three-state buffer output can be placed in one of three possible states. a signal equivalent to logic 1, signal equivalent to logic 0, or a high impedance state. The logic 1 and 0 are normal digital signals.

→ The high impedance state behaves like an open circuit, which means that the output does not carry a signal and has no logic significance.

→ The capacity of the memory is 128 words of 8 bits per word. This requires a 7-bit address and an 8-bit bidirectional data bus.

→ The read and write inputs specify the memory operation and the two chips select (CS) control inputs are for enabling the chip.

→ The unit is in operation only when CS1 = 1 and $\overline{CS2} = 0$.

→ If the chip select inputs are not enabled or if they are enabled but the read or write are not enabled, the memory is inhibited and its data bus is in a high-impedance state.

→ when CS1 = 1 and $\overline{CS2} = 0$ and WR input is enabled, the memory stores a byte from the data bus into a location specified in address.

→ when the RD input is enabled, the content of the selected byte is placed into the data bus.

ROM CHIP :- A ROM chip is organized externally in a similar manner. However, since a ROM can only read, the data bus can only be in an output mode.

→ For the same-size chip, it is possible to have more bits of ROM than of RAM, because the internal binary cells in ROM occupy less space than in RAM.

→ The nine address lines in the ROM chip specify any one of 512 bytes stored in it. The two chip select inputs must be CS1 = 1 and $\overline{CS2} = 0$ for the unit to operate. Otherwise, the data bus is in a high-impedance state

→ There is no need for a read/write control because the unit can only read. Thus when CS1 = 1 and $\overline{CS2} = 0$ the byte selected by the address lines appears on the data bus.

Cache memory :- Analysis of a large number of typical programs has shown that the references to memory at any given internal of time tend to be confined within a few localized areas in memory. This phenomenon is known as the property of locality of reference.

→ From this property we can find that over a short internal of time, the addresses generated by a typical program refer to a few localized areas of memory repeatedly, while rest of memory is accessed relatively infrequently.

→ If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced, thus reducing the total execution time of the program.

→ Such a fast small memory is referred to as a cache memory. It is placed between the CPU and main memory.

→ The cache memory access time is less than the access time of main memory. The cache is the fastest component in the memory hierarchy. The most frequently accessed instructions and data are there in cache memory.

→ when the CPU needs to access memory, the cache is examined. If the word is found in the cache, it is read from the fast memory. If the word is not found in cache then the main memory is accessed to read the word. A block of words containing the one just accessed is then transferred from main memory to cache memory.

→ The performance of cache memory is frequently measured in terms of a quantity called hit ratio. when the CPU refers to memory and finds the word in cache, it is said to produce a hit. If the word is not found in cache, it is in main memory and it counts as a miss.

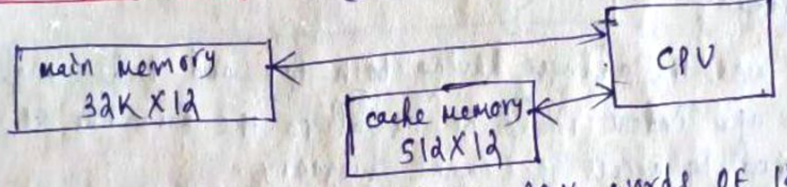
→ The ratio of the number of hits divided by the total CPU references to memory (hits plus misses) is the hit ratio.

→ The basic characteristic of cache memory is its fast access time. Therefore, very little or no time must be wasted when searching for words in the cache.

→ The transformation of data from main memory to cache memory is referred to as a mapping process.

→ Three types of mapping procedures are generally used.
* Associative mapping * Direct mapping * Set-associative mapping.

Associative mapping: — Ex:-

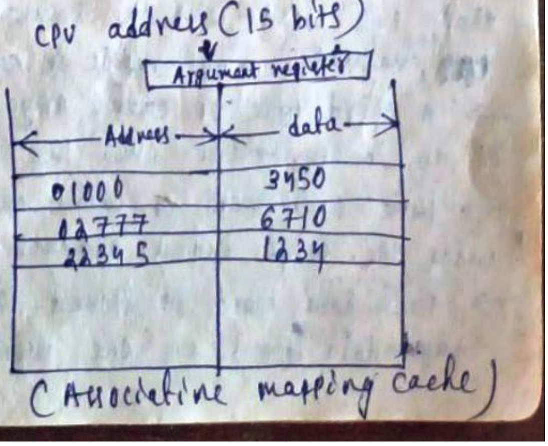


Here the main memory can store 32K words of 12 bits each. The cache
is capable of storing 512 of these words at any given time. For every
word stored in cache there is a duplicate copy in main memory. The
CPU communicate with both memories. It first sends 15-bit address to
cache if there is a hit, the CPU accepts the 12-bit data from the cache.
If there is a miss then the CPU reads the word from main memory
and the word is then transferred to the cache.

* → The fastest and most flexible cache organization uses an associative memory. The associative memory stores both the address and content of the memory word. This permits any location in cache to store any word from main memory.

→ The address value of 15 bits is shown as a five digit octal no. and its corresponding 12-bit word is shown as a four digit octal number.

→ A CPU ~~ref~~ address of 15-bits is placed in the argument register and the associative memory is searched for a matching address.



Address	data
01000	3450
02777	6710
22345	1234

(Associative mapping cache)

→ If address is found then the corresponding 12-bit data is read & sent to the CPU. If no match occurs, the main memory is accessed for the word.

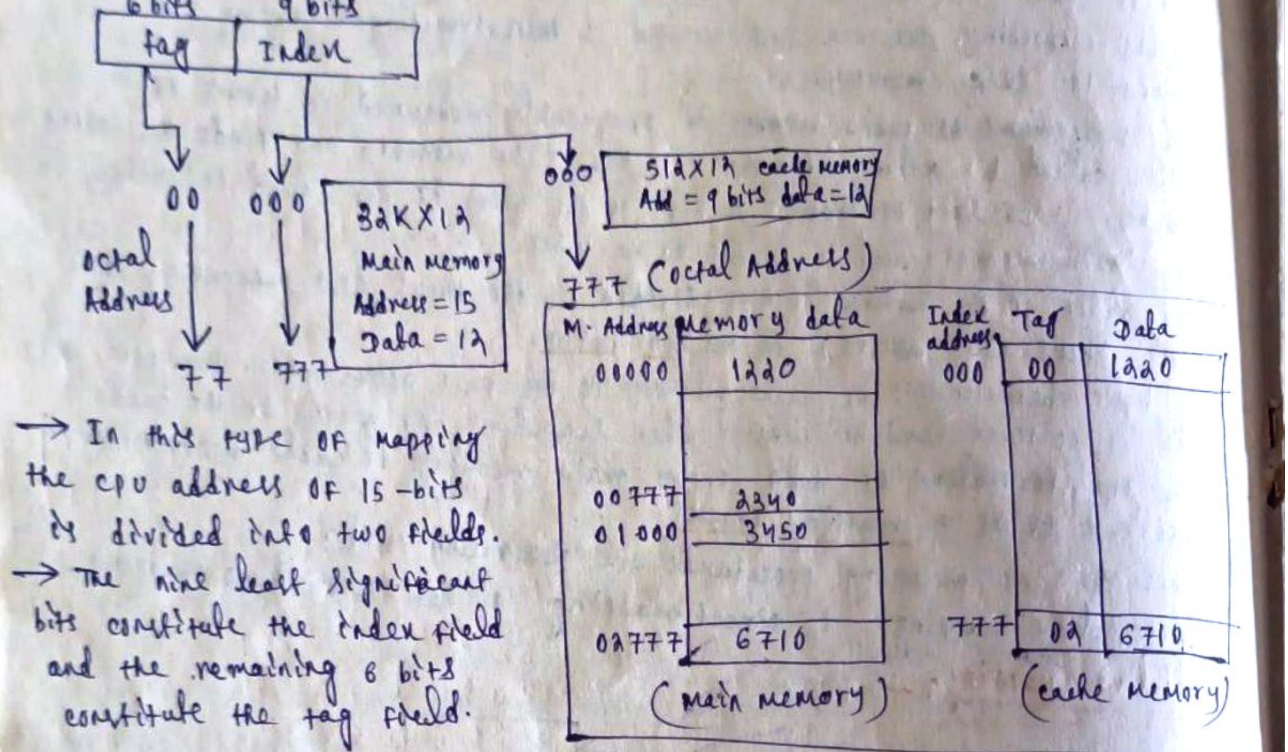
→ The address-data pair is then transferred to the associative cache memory.

→ If the cache is full, an address-data pair must be displaced to make room for a pair that is needed and not presently in the cache.

→ The decision for replacement is done by replacement algorithms.

DIRECT MAPPING :-

3bits → 1 digit in octal 15bits → 5 digits in octal



→ In this type of mapping the CPU address of 15-bits is divided into two fields.

→ The nine least significant bits constitute the index field and the remaining 6 bits constitute the tag field.

→ The main memory uses the address having both tag and index field.

→ The no. of bits in the index field is equal to the number of address bits required to access the cache memory.

→ Each word in cache consist of the data word and its associated tag.

→ When CPU generates a memory request, the index field is used for the address to access the cache. The tag field of the CPU address is compared with the tag in the word read from cache. If two tags match, there is a hit and the desired data word is in cache.

→ If there is no match, there is a miss and the required word is read from main memory. It is then stored in the cache together with the new tag, replacing the previous value.

SET-ASSOCIATIVE MAPPING :→

Direct mapping has one disadvantage that two words with the same index in their address but with different tag values cannot reside in cache memory at the same time.

→ A third type of cache organization, called set-associative mapping is an improvement over the direct mapping organization.

→ Here each word of cache can store two or more words of memory under the same index address.

→ Each data word is stored together with its tag and the no. of tag-data items in one word of cache is said to form a set.

→ when cpu generates a memory request, the index value of the address is used to access the cache.

→ The tag field of the cpu address is compared with both tags in cache to determine if a match occurs.

→ The comparison logic is done by an associative search of the tags in the set thus the name comes "set-associative".

Index	Tag	data	Tag	data
000	01	3450	02	5670
777	02	6710	00	2340

→ The hit ratio will improve as the set size increases.

VIRTUAL MEMORY :— portions of a program or data are brought into main memory as they are needed by the cpu. virtual memory is a concept used in some large computer systems that permit the user to construct programs as though a large memory space were available, equal to the totality of auxiliary memory.

→ virtual memory is used to give programmers the illusion that they have a very large memory space at their disposal, even though the computer actually has a relatively small main memory.

→ A virtual memory system provides a mechanism for translating program-generated addresses into correct main memory locations. This is done dynamically, while programs are being executed in the cpu.

Address Space and Memory Space :— An address used by a programmer will be called a virtual address, and the set of such addresses the address space.

→ An address in main memory is called a location or physical address. The set of such locations is called the memory space.

→ In most computers the address and memory spaces are identical.

→ The address space is allowed to be larger than the memory space in computers with virtual memory.

Address Mapping Using Pages :— The physical memory is broken down into groups of equal size called blocks, which may range from 64 to 4096 words.

→ The term page refers to groups of address space of the same size.

→ The page refers to the organization of address space while a block refers to the organization of memory space.

→ The programs are also considered to be split into pages. portions of programs are moved from auxiliary memory to main memory in records equal to the size of a page. The term page frame is sometimes used to denote a block.

→ The mapping from address space to memory space is fascilitated if each virtual address is considered to be represented by two numbers: a page number address and a line within the page.

Page replacement :— A virtual memory system is a combination of h/w and s/w techniques. The memory management s/w system handles all the s/w operations for the efficient utilization of memory space.

→ It must decide (1) which page in main memory ought to be removed to make room for a new page. (2) when a new page is to be transferred from auxilliary memory to main memory. (3) where the page is to be placed in main memory.

→ When a program starts execution, one or more pages are transferred into main memory and the page table is set to indicate their position. The program is executed from main memory until it attempts to reference a page that is still in auxilliary memory. This condition is called page fault.

→ When page fault occurs the execution of the present program is suspended until the required page is brought into main memory.

→ If main memory is full, it would be necessary to remove a page from a memory block to make room for the new page.

→ The policy for choosing pages to remove is determined from the replacement algorithm that is used. The goal of a replacement policy is to try to remove the page least likely to be referenced in the immediate future.

Page replacement Algorithms :-

FIFO (First-in, First-out) :- The FIFO algorithm selects for replacement the page that has been in memory the longest time. Each time a page is loaded into memory, its identification number is pushed into a FIFO stack.

→ FIFO will be full whenever memory has no more empty blocks. When a new page is required to be loaded then the page least recently brought in is removed. The page to be removed can be easily determined because its identification number is at the top of the FIFO stack. It is easy to implement but it has the disadvantage that under certain conditions pages are removed and loaded from memory too frequently.

LRU (Least recently used) :- It is difficult to implement but has been more attractive on the assumption that the least recently used page is a better candidate for removal than the least recently loaded page as in FIFO.

→ It can be implemented by associating a counter with every page that is in main memory. When a page is referenced then its associated counter is set to zero.

→ At fixed intervals of time, the counters associated with all pages presently in memory are incremented by 1. The least recently used page is the page with the highest count.

→ The counters are also called aging registers, as their count indicates their age, that is, how long ago their associated pages have been referenced.

INPUT/OUTPUT ORGANISATION chapter - 4

Peripheral Devices :— The input-output subsystem of a computer, referred to as I/O, provides an efficient mode of communication between the central system and the outside environment.

→ The most familiar means of entering information into a computer is through a key-board that allows a person to enter alphanumeric information directly.

→ Devices that are under the direct control of the computer are said to be connected on-line. These devices are designed to read information into or out of the memory unit upon command from the cpu and are considered to be part of the total computer system.

→ Input or output devices attached to the computer are called peripherals.

→ peripherals that provide auxiliary storage for the system are magnetic disks and tapes. peripherals are electromechanical and electromagnetic devices of some complexity.

Monitor and Keyboard :- video monitors are the most commonly used peripherals They consists of a keyboard as the input device and a display unit as the output device. The most popular monitor is a CRT monitor.

→ The CRT contains an electronic gun that sends an electronic beam to a screen in front of the tube. The beam can be deflected horizontally or vertically.

→ To produce a pattern on the screen, a grid inside make it glow at selected spots. A characteristic feature of display devices is a cursor that marks the position in the screen where next character will be inser

→ The display terminal can operate in a single-character mode where all characters entered on the screen through the keyboard are transmitted to the computer simultaneously.

→ In the block-mode, the edited text is first stored in a local memory inside the terminal. The text is transferred to the computer as a block of dat

Printer :— printers are most common output device. It provides a permanent record on paper of computer output data or text. The printed output are known as hard copy. There are three basic types of character printers Those are daisywheel, dot-matrix and laser printers.

→ The daisywheel printer contains a wheel with the characters placed along the circumference. To print a character, the wheel rotates to the proper position and an energized magnet then presses the letter against the ribbon.

→ The dot matrix printer contains a set of dots along the printing mechanism. For e.g. a 5X7 dot matrix printer that prints 80 characters per line has seven horizontal lines, each consisting of 5X80 = 400 dots.

→ The laser printer uses a rotating photographic drum that is used to imprint the character images. The pattern is then transferred onto paper in the same manner as a copying machine.

✱ Other inputs and output devices encountered are plotter, scanner, joystick bar code reader, optical and magnetic character readers etc.

Input - Output Interface :- Input - output interface provides a method for transferring information between internal storage and external I/o devices.

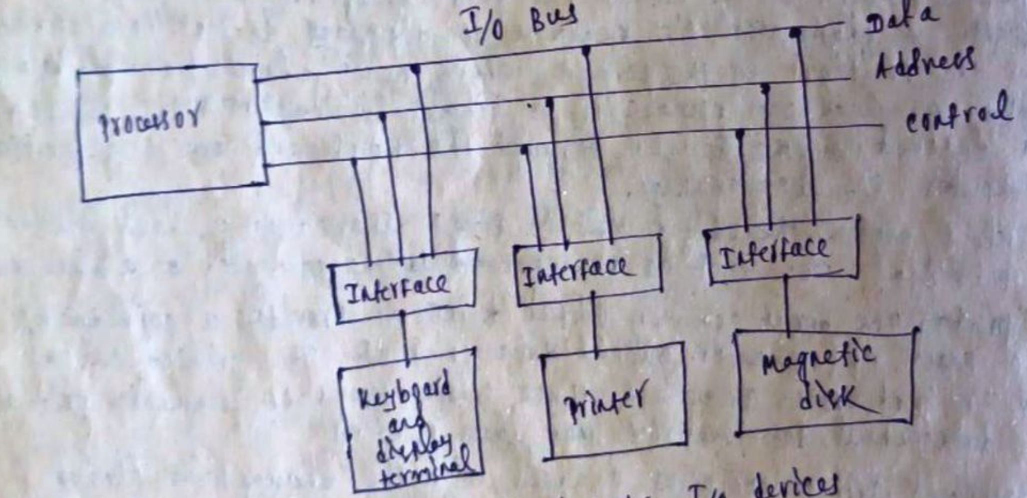
→ peripherals connected to a computer need special communication links for interfacing them with the CPU. The purpose of communication link is to resolve the differences between the peripheral and CPU.

→ The major differences are :

- * peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU.
- * The data transfer rate of peripherals is usually slower than the transfer rate of CPU.
- * Data codes and formats in peripherals differ from word format in CPU and memory.
- * The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

→ To resolve these differences, computer systems include special h/w components between the CPU and peripherals to supervise and synchronize all input and output transfers. These components are called interface units because they interface between the processor bus & the peripheral device.

I/o Bus and Interface Modules :-



connection of I/o bus to I/o devices

→ The I/o bus consists of data lines, address lines and control lines. The magnetic disks, printer and terminal are employed in practically any general purpose computer. The magnetic tape is used in some computers for backup storage.

Each peripheral device is associated with an interface unit. Each

→ Each peripheral interface decodes the address and control received from the I/o bus, interprets them for the peripheral and provides signal for the peripheral controller.

→ It also synchronizes the data flow and supervises the transfer between peripheral and processor. Each peripheral has its own controller that operates the particular electromechanical device.

→ A controller may be housed separately or may be physically integrated with the peripheral.

→ The I/O bus from the processor is attached to all peripheral interfaces.

→ To communicate with a particular device, the processor places a device address on the address lines. Each interface attached to the I/O bus contains an address decoder that monitors the address lines.

→ when the interface detects its own address, it activates the path between the bus lines and the device that it controls.

I/O command :- The interface selected responds to the function code and proceeds to execute it. The function code is referred to as an I/O command and is in essence an instruction that is executed in the interface and its attached peripheral unit. There are 4 types of commands that an interface may receive. They are classified as control, status, data input and data output.

control command :- A control command is issued to activate the peripheral and inform it what to do. The particular control command issued depends on the peripheral.

Status command :- A status command is used to test various status conditions in the interface and the peripheral.

data input :- A data input command is the opposite of data output. In this case the interface receives an item of data from the peripheral and places it in its buffer register. The interface places the data on the data lines where they are accepted by the processor.

data output :- A data output command causes the interface to respond by transferring data from the bus into one of its registers. The interface responds to the address and command and transfers the information from the data lines in the bus to its buffer register. ~~The interface then communicates with the tape controller and sends the data to~~

In addition to communicating with I/O, the processor
I/O versus Memory Bus : In addition to communicating with I/O, the memory
must communicate with the memory unit. Like the I/O bus, the memory
bus contains data, address and read/write control lines. There are three
ways that computer buses can be used to communicate with memory & I/O :
1. Use two separate buses, one for memory and the other for I/O.
2. Use one common bus for both memory and I/O but have separate
control lines for each.
3. Use one common bus for memory and I/O with common control lines.

In the first method, the computer has independent
IOP (input-output processor) : sets of data, address and control buses, one for accessing memory and the
other for I/O. This is done in computers that provide a separate I/O
processor in addition to CPU.
→ The memory communicates with both the CPU and the IOP through a
memory bus. The IOP communicates also with the input and output
devices through a separate I/O bus with its own address, data & control lines
→ The purpose of the IOP is to provide an independent pathway for the
transfer of information between external devices and internal memory.
→ The I/O processor is sometimes called a data channel.

Isolated versus Memory-mapped I/O :- Many computers uses one common bus
to transfer information between memory or I/O and the CPU. The
distinction between a memory transfer and I/O transfer is made through
separate read and write lines.

→ The I/O read and I/O write control lines are enabled during an I/O transfer.

→ The memory read and memory write control lines are enabled during a memory transfer. This configuration isolates all I/O interface addresses from the addresses assigned to memory is referred to as the isolated I/O method for assigning addresses in a common bus.

ISOLATED I/O : In the isolated I/O configuration, the CPU has distinct input and output instructions, and each of these instructions is associated with the address of an interface register.

→ When the CPU fetches and decodes the operation code of an input or output instruction, it places the address associated with the instruction into the common address lines.

→ At the same time, it enables the I/O read or I/O write control lines.

→ This informs the external components that are attached to the common bus that the address in the address lines is for an interface register and not for a memory word.

→ On the other hand when the CPU fetches an instruction or operand from memory then it places a memory address on the address lines and enables the memory read or write control line. This informs the external components that the address is for a memory word.

→ The isolated I/O method isolates memory and I/O addresses so that memory address values are not affected by interface address assignment since each has its own address space.

Memory-mapped I/O : Here the same address space is used for both memory & I/O. This is the case in computers that employ only one set of read and write signals and do not distinguish between memory and I/O addresses.

→ This configuration is referred to as memory-mapped I/O.

→ The computer treats an interface register as being part of the memory system.

→ In a memory-mapped I/O organization there are no specific input or output instructions. The CPU can manipulate I/O data residing in interface registers with the same instructions that are used to manipulate memory words.

→ Each interface is organized as a set of registers that respond to read and write requests in the normal address space.

→ Typically, a segment of the total address space is reserved for interface registers, but in general, they can be located at any address as long as there is not also a memory word that responds to the same address.

→ computers with memory-mapped I/O can use memory-type instructions to access I/O data. It allows the computer to use the same instructions for either input-output transfers or for memory transfers.

→ The advantage is that the load and store instructions used for reading and writting from memory can be used to input and output data from I/O registers.

Asynchronous Data Transfer : The internal operations in a digital system are synchronized by means of clock pulses supplied by a common pulse generator. clock pulses are applied to all registers within a unit and all data transfers among internal registers occur simultaneously.

→ Two units, such as a CPU and an I/O interface are designed independently of each other. If the registers in the interface share a common clock with the CPU registers, the transfer between the two units is said to be synchronous.

→ In most cases, the internal timing in each unit is independent from the other in that each uses its own private clock for internal registers. In that case, the two units are said to be asynchronous to each other. This approach is widely used in most computer systems.

→ Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted. One way of achieving this is by means of a strobe pulse supplied by one of the units to indicate to the other unit when the transfer has to occur.

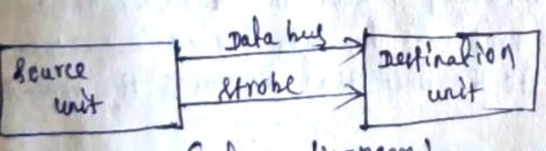
→ Another method commonly used is to accompany each data item being transferred with a control signal that indicates the presence of data in the bus.

→ The unit receiving the data item responds with another control signal to acknowledge receipt of the data. This type of agreement between two independent units is referred to as handshaking.

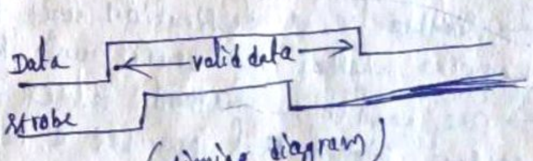
→ The sequence of control during an asynchronous transfer depends on whether the transfer is initiated by the source or by the destination unit.

→ The timing diagram shows the timing relationship that must exist between the control signals and the data in the buses.

STROBE CONTROL: The strobe control method of asynchronous data transfer employs a single control line to time each transfer. The strobe may be activated by either the source or destination unit.



(Block diagram)



(timing diagram)

(Source-initiated strobe for data transfer)

→ The data bus carries the binary information from source unit to the destination unit. Typically, the bus has multiple lines to transfer an entire byte or word.

→ The strobe is a single line that informs the destination unit when a valid data word is avilable in the bus.

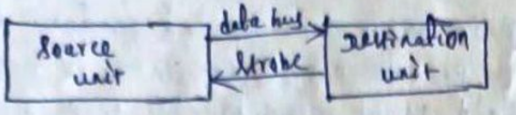
→ The source unit first places the data on the data bus then after some time the source activates the strobe pulse.

→ The information on the data bus and the strobe signal remain in the active state for a sufficient time period to allow the destination unit to receive the data.

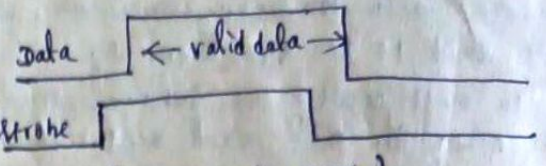
→ Often the destination unit uses the falling edge of the strobe pulse to transfer the contents of the data bus into one of its internal registers.

→ The source removes the data from the bus a brief period after it disables its strobe pulse and new valid data will be avilable only after the strobe is enabled again.

Destination-initiated : In this case the destination unit activates the strobe pulse, informing the source to provide the data.
→ The source unit responds by placing the requested binary information on the data bus.



(Block diagram)



(Timing diagram)

Destination-initiated strobe for data transfer

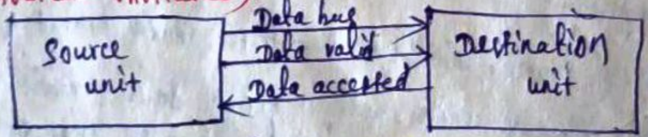
→ ~~The source unit responds by placing the requested binary information on the data bus~~. The data must be valid and remain in the bus long enough for the destination to accept it.
→ The falling edge of the strobe pulse can be used again to trigger a destination register. The destination unit then disables the strobe. The source removes the data from the bus after a predetermined time interval.

* In many computers the strobe pulse is actually controlled by the clock pulses in the CPU. The CPU is always in control of the buses and informs the external units how to transfer data.

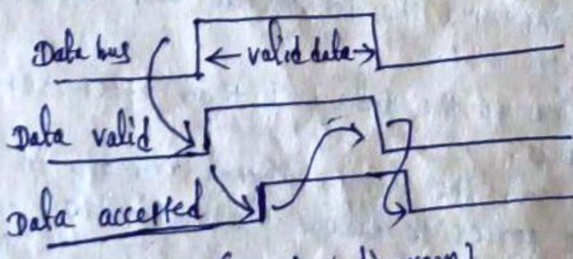
* The transfer of data between the CPU and an interface unit is similar to the strobe transfer. Data transfer between an interface and an I/O device is commonly controlled by a set of handshaking lines.

HANDSHAKING : The disadvantage of the strobe method is that the source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was placed in the bus.
→ Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has actually placed the data on the bus.
→ The handshake method solves this problem by introducing a second control signal that provides a reply to the unit that initiates the transfer.

Principle : (Source-initiated)

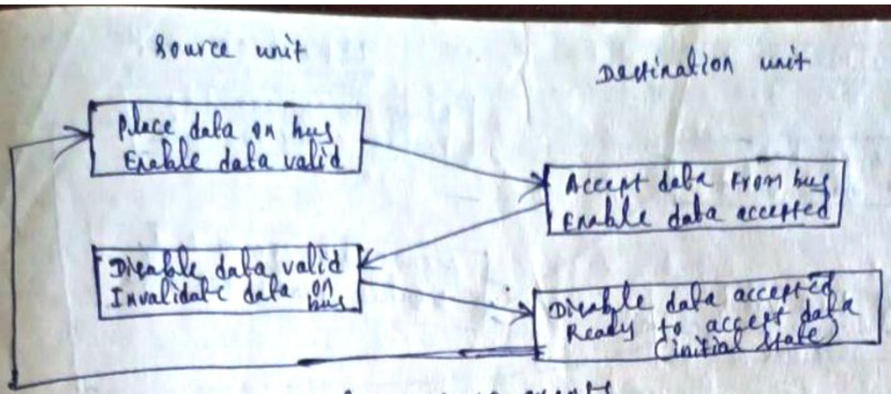


(a) Block diagram



(b) (Timing diagram)

→ Here one control line is in the same direction as the data flow in the bus from the source to the destination.
→ It is used by the source unit to inform the destination unit whether there are valid data in the bus. The other control line is in the

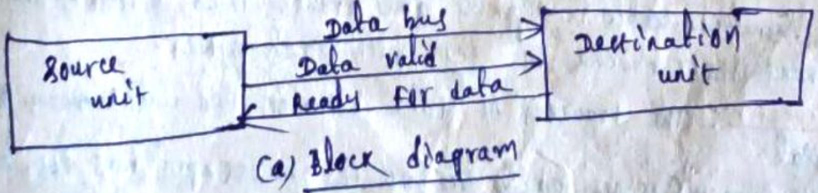


(C) Sequence of events

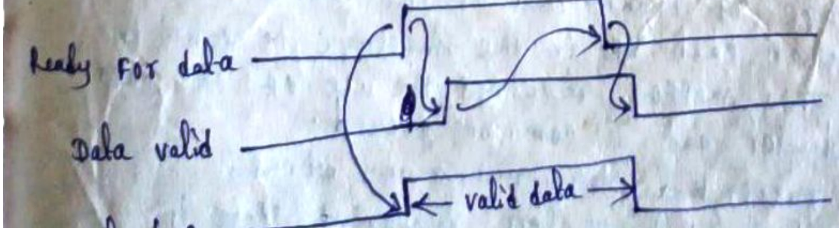
Source - initiated transfer using handshaking

other direction from the destination to the source. It is used by the
destination unit to inform the source whether it can accept data.
→ The sequence of control during the transfer depends on the unit
that initiates the transfer.
→ The two handshaking lines are data valid, which is generated by the
source unit, and data accepted, generated by the destination unit.
→ The sequence of events shows the four possible states that the system
can be at any given time.
→ The source unit initiates the transfer by placing the data on the bus and
enabling its data valid signal.
→ The data accepted signal is activated by the destination unit after it
accepts the data from the bus.
→ The source unit then disables its data valid signal, which invalidates
the data on the bus. The destination unit then disables its data accepted
signal and the system goes into its initial state.
→ The source does not send the next data item until after the destination
unit ~~then disables its data accepted~~ shows its readiness to accept new
data by disabling its data accepted signal.
→ This scheme allows arbitrary delays from one state to the next state.

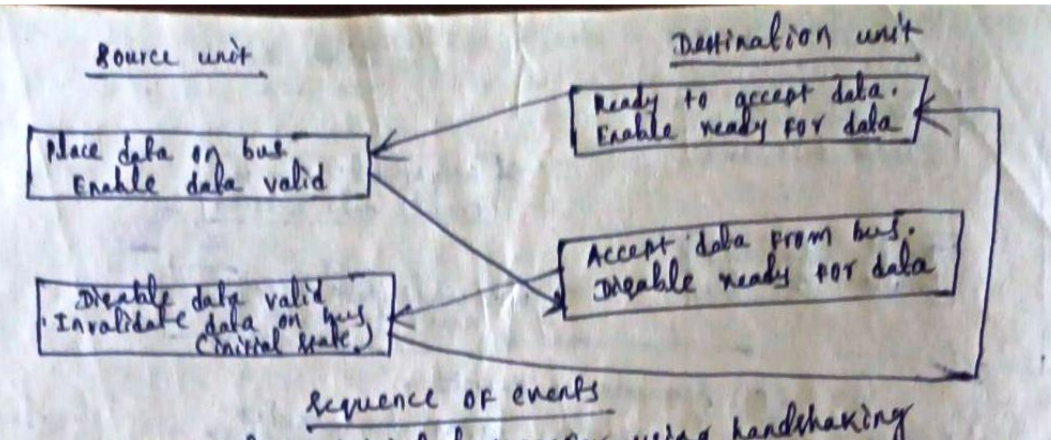
Destination - initiated : ~~The destination - initiated transfer using handshaking~~
~~lines~~ Here in this case the name of the signal generated by the
destination unit has been changed to ready for data to reflect its new
meaning. The source unit in this case does not place data on the bus
until after it receives the ready for data signal from the destination unit.



(a) Block diagram



(b) Timing diagram



Sequence of events

Destination-initiated transfer using handshaking

→ From there on, the handshaking procedure follows the same pattern as in the source-initiated case.

→ The sequence of events in both cases would be identical if we consider the ready for data signal as the complement of data accepted.

→ The only difference between the source-initiated and the destination initiated transfer is in their choice of initial state.

TIMEOUT: The handshaking scheme provides a high degree of flexibility and reliability because the successful completion of a data transfer relies on active participation of both the units.

→ If one unit is faulty then data transfer will not be completed. Such error can be detected by timeout mechanism, which produces an alarm if the data transfer is not completed within a predetermined time.

→ The timeout is implemented by means of an internal clock that starts counting time when the unit enables one of its handshaking control signals.

→ If the return handshake signal does not respond within a given time ~~then~~ period, then the units assumes that an error has occured.

→ The timeout signal can be used to interrupt the processor and execute a service routine that takes appropriate error recovery action.

Synchronous data transfer: In synchronous transmission the two units share a common clock frequency and bits are transmitted continuously at the rate dictated by the clock pulses.

→ ~~In long distant~~ Synchronization signals are transmitted periodically between the two units to keep their clocks in step with each other.

Asynchronous Serial Transfer: The transfer of data between two units may be done in paraller or serial.

→ In parallel data transmission each bit of the message has its own path and the total message it transmitted at the same time.

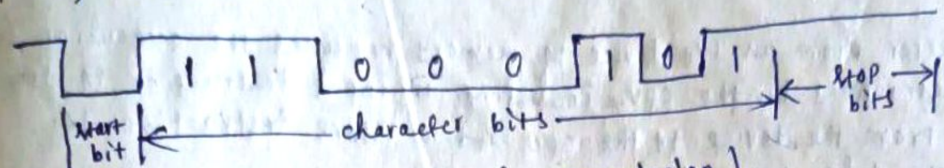
→ In serial data transmission each bit in the message is sent in sequence one at a time. This method requires the use of one pair of conductors or one conductor and a common ground.

→ parallel transmission is faster but requires many wires.

→ Serial transmission is slower but is less expensive since it requires only one pair of conductors.

→ Serial transmission can be synchronous or asynchronous.

* In asynchronous transmission, binary information is sent only when it
is available and the line remains idle when there is no information
to be transmitted.
→ with this technique, each character consists of three parts: a
start bit, the character bits and stop bits.



(Asynchronous serial transmission)

→ At the 1-state the transmitter rests and no characters are transmitted.
→ The first bit, called the start bit is always 0 and is used to
indicate the begining of a character. The last bit called the stop
bit is always a 1.
→ A transmitted character can be detected by receiver from knowledge
of transmission rules:
1. when a character is not being sent, the line is kept in the 1-state.
2. The initiation of a character transmission is detected from the
start bit, which is always 0.
3. The character bits always follow the start bit.
4. After the last bit of the character is transmitted, a stop bit is
detected when the line returns to the 1-state for at least one bit time.
→ using these rules, the receiver can detect the start bit when the line goes
from 1 to 0. A clock in the receiver examines the line at proper bit times.
→ The receiver knows the transfer rate of the bits and the no. of character
bits to accept.
→ After the character bits are transmitted, one or two stop bits are sent. The
stop bits are always in the 1-state and signify the idle or wait state.
→ Some older electromechanical terminals use two stop bits but newer terminals
use one stop bit. The line remains in the 1-state until another character
is transmitted.
→ The baud rate is defined as the rate at which serial information is
transmitted and is equivalent to the data transfer in bits per second.
Ten characters per second with an 11-bit format has a transfer rate of 110 baud.

MODES OF TRANSFER Data transfer between the central computer and I/O devices
may be handled in a variety of modes. Some modes use cpu as an
intermediate path others transfer the data directly to and from the memory unit.
→ Data transfer to and from peripherals may be handled in one of
three possible modes.

* programmed I/O

* Interrupt-initiated I/O

* Direct memory access (DMA)

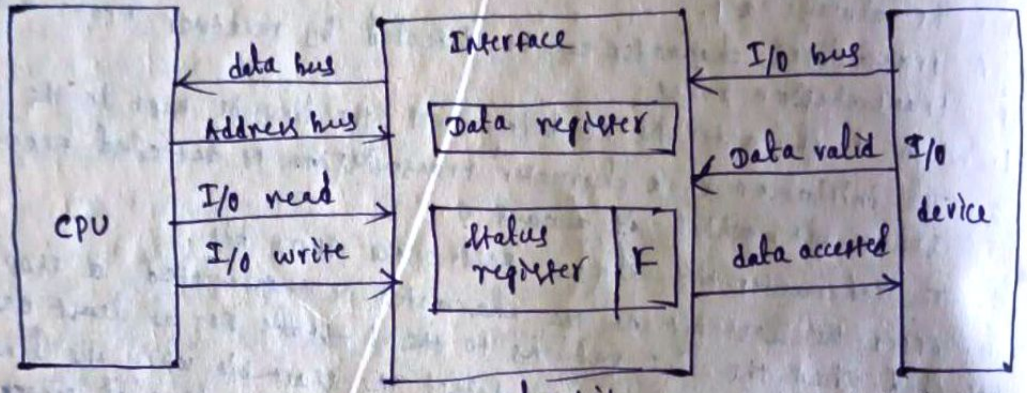
Programmed I/O: programmed I/O operations are the result of I/O instructions
written in the computer program. Each data item transfer is initiated by
an instruction in the program.
→ usually, the transfer is to and from a cpu register and peripheral.
→ Other instructions are needed to transfer the data to and from
cpu and memory.

→ Once a data transfer is initiated, the cpu is required to monitor the interface to see when a transfer can again be made. It is done by the programmed instructions executed in the cpu.

→ In the programmed I/o method, the I/o device does not have direct access to memory.

→ A transfer from an I/o device to memory requires the execution of several instructions by the cpu, including an input instruction to transfer the data from the device to the cpu and a store instruction to transfer the data from the cpu to memory.

Example: Here the device transfers bytes of data one at a time as they are avilable. when the byte of data is avilable, the device places it in the I/o bus and enables its data valid line.



F = flag bit

Data transfer from I/o device to cpu

→ The interface accepts the byte into its data register and enables the data accepted line. The interface then sets a bit in the status register that will refer to as an F or "flag" bit.

→ A program is written for the computer to check the flag in the status register to determine if a byte has been placed in the data register by the I/o device.

→ This is done by reading the status register into a cpu register and checking the value of the flag bit. If the flag is equal to 1, the cpu reads the data from the data register.

→ The flag bit is then cleared to 0 by either the cpu or the interface. Once the flag is cleared, the interface disables the data accepted line and the device can then transfer the next data byte.

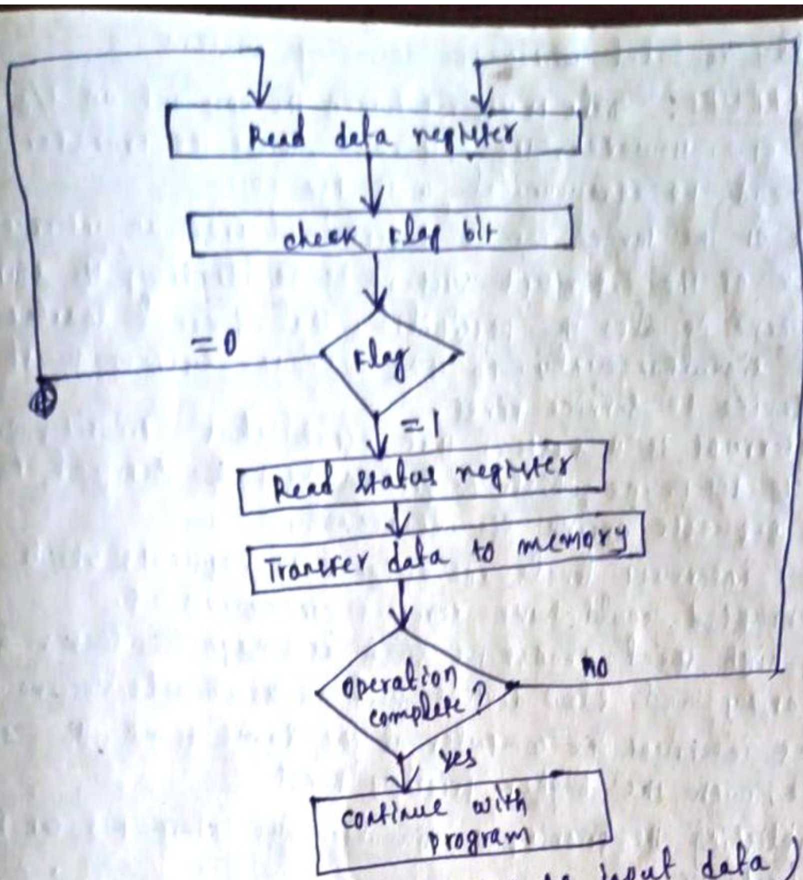
→ The transfer of each byte requires three instructions.

1. Read the status register.
2. check the status of the flag bit and branch to step1 if not set or to step 3 if set.
3. Read the data register.

→ Each byte is read into a cpu register and then transferred to memory with a store instruction.

→ The programmed I/o method is particularly useful in small low-speed computers or in systems that are dedicated to monitor a device continuously.

→ The difference in information transfer rate between the cpu and the I/o device makes this type of transfer inefficient.



(Flowchart for CPU program to input data)

Interrupt-Initiated I/O: An alternative to the CPU constantly monitoring the flag is to let the interface inform the computer when it is ready to transfer data.

→ This mode of transfer uses the interrupt facility. While the CPU is running a program, it does not check the flag. However, when the flag is set, the computer is momentarily interrupted from proceeding with the current program and is informed of the fact that the flag has been set.

→ The CPU then deviates from what it is doing to take care of the input or output transfer. After the transfer is completed, the computer returns to the previous program to continue what it was doing before the interrupt.

→ The CPU responds to the interrupt signal by storing the return address from the program counter into a memory stack and then control branches to a service routine that processes the required I/O transfer.

→ Two types of interrupts are there vectored Interrupt and Non-vectored interrupt. In a nonvectored interrupt, the branch address is assigned to a fixed location in memory.

→ In a vectored interrupt, the source that interrupts supplies the branch information to the computer. This information is called the interrupt vector.

SOFTWARE CONSIDERATIONS: Along with the hardware a computer must also have software routines for controlling peripherals and for transfer of data between the processor and peripherals.

→ I/O routines must issue control commands to activate the peripheral and to check the device status to determine when it is ready for data transfer.

→ In interrupt-controlled transfers, the I/O S/W must issue commands to the peripheral to interrupt when ready and to service the interrupt when it occurs.

→ S/W control of input-output equipment is a complex undertaking. For this reason I/O routines for standard peripherals are provided by the manufacturer as part of the computer system.

→ They are usually included within the operating system.

PRIORITY INTERRUPT: Data transfer between the CPU and an I/O device is initiated by the CPU. However, the CPU can't start the transfer unless the device is ready to communicate with the CPU.

→ The readiness of the device can be determined from an interrupt signal.

→ The first task of the interrupt system is to identify the source of the interrupt. There is also the possibility that several sources will request service simultaneously. In this case the system must also decide which device to service first.

→ A priority interrupt is a system that establishes a priority over the various sources to determine which condition is to be serviced first when two or more requests arrive simultaneously.

→ Higher-priority interrupt levels are assigned to requests which, if delayed or interrupted, could have serious consequences.

→ Devices with high speed transfers such as magnetic disks are given high priority and slow devices such as keyboards receive low priority

→ when a devices interrupt the computer at the same time, the computer services the device, with the higher priority first.

POLLING: Establishing the priority of simultaneous interrupts can be done by software or hardware.

→ A polling procedure is used to identify the highest-priority source by S/w means. In this method there is one common branch address for all interrupts.

→ The program begins at the branch address and polls the interrupt sources in sequence. The order in which they are tested determines the priority of each interrupt.

→ The highest-priority source is tested first, and if its interrupt signal is on, control branches to a service routine for this source. Otherwise, the next-lower-priority source is tested and so on.

→ The disadvantage of S/w method is that if there are many interrupts then the time required to poll them is much more.

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

→ Here no polling is required because all the decisions are established by the hardware priority-interrupt unit.

→ The h/w priority function can be established by either a serial or a parallel connection of interrupt lines. The serial connection is also known as the daisy-chaining method.

I/O INTERFACE AND BUS ARCHITECTURE

A system bus is **a facet of computer architecture that transmits and shares data throughout the computer and between devices**. It's the primary way for a computer to process information because it connects the main processor to all other internal hardware components of a computer.

TYPES OF SYSTEM BUS

Following are the three components of a bus: –

- The **address** bus, a one-way pathway that allows information to pass in one direction only, carries information about where data is stored in memory.
- The **data** bus is a two-way pathway carrying the actual data (information) to and from the main memory.
- The **control** bus holds the control and timing signals needed to coordinate all of the computer's activities.

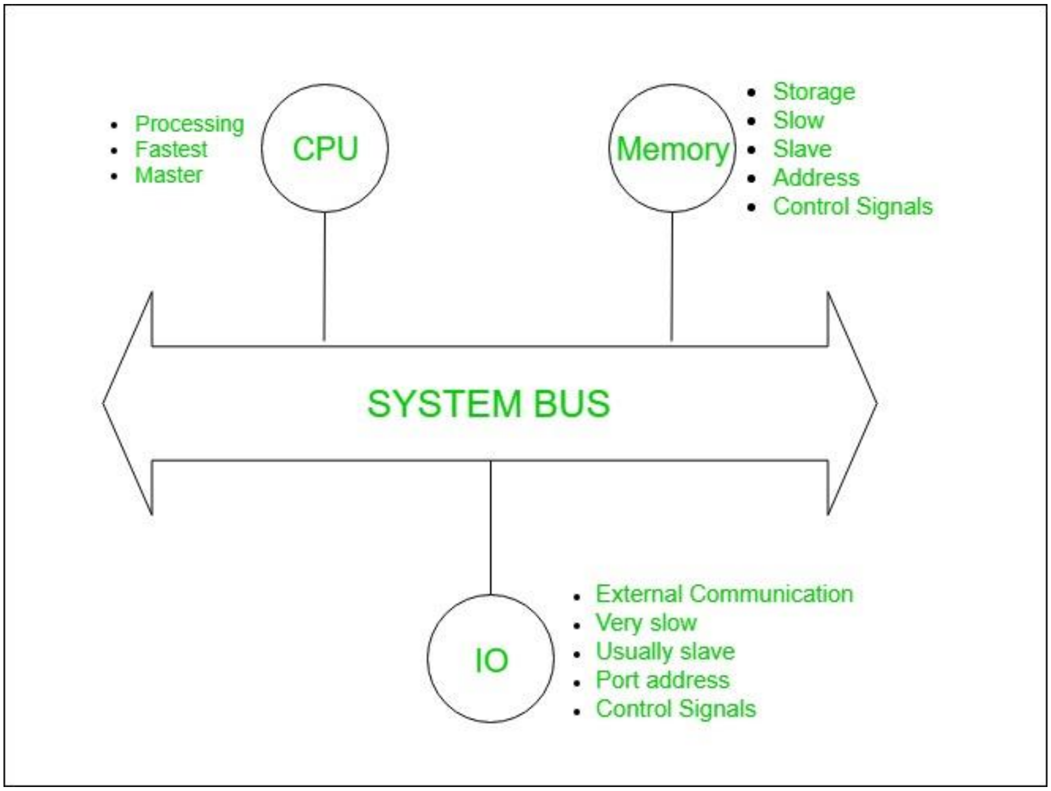
Functions of a computer bus

Below are a few of the functions of a computer bus:-

- **Data sharing** – All types of buses used in the network transfer data between the connected computer peripherals. The buses either transfer or send data in serial or parallel transfer methods. This allows 1, 2, 4, or even 8 bytes of data to be exchanged at a time. (A Byte is an 8-bit group). Buses are classified according to how many bits they can move simultaneously, meaning we have 8-bit, 16-bit, 32-bit, or even 64-bit buses.
- **Addressing** – A bus has address lines that suit the processors. This allows us to transfer data to or from different locations in the memory.
- **Power** – A bus supplies the power to various connected peripherals.

STRUCTURE OF SYSTEM BUS

A system bus is a single computer bus that connects the major components of a computer system, combining the functions of a data bus to carry information, an address bus to determine where it should be sent or read from, and a control bus to determine its operation.



System bus contains 3 categories of lines used to provide the communication between the CPU, memory and IO named as:

1. Address lines (AL)
2. Data lines (DL)
3. Control lines (CL)

1. Address Lines:

- Used to carry the address to memory and IO.
- Unidirectional.
- Based on width of a address bus we can determine the capacity of a main memory

2. Data Lines:

- Used to carry the binary data between the CPU, memory and IO.
- Bidirectional.
- Based on the width of a data bus we can determine the word length of a CPU.
- Based on the word length we can determine the performance of a CPU.

3. Control Lines:

- Used to carry the control signals and timing signals
- Control signals indicates type of operation.
- Timing Signals used to synchronize the memory and IO operations with a CPU clock.

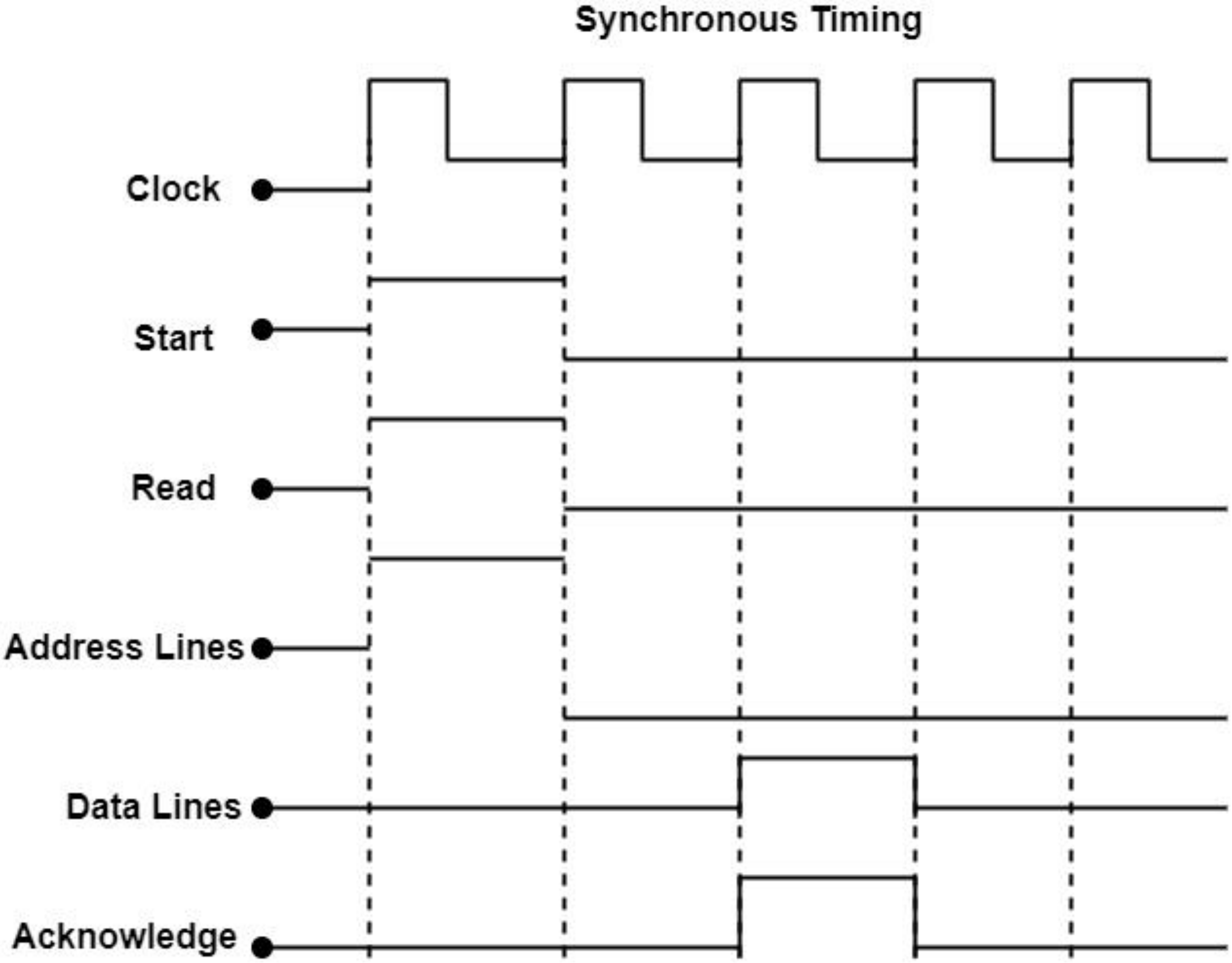
BASIC PARAMETERS IN BUS DESIGN

Bus Width

The width of the data has an impact on system execution. The wider the data bus, the higher the number of bits moved at one time. The width of the address bus has an impact on system capacity, that is, the wider the address bus, the higher the dimension of locations that can be referenced.

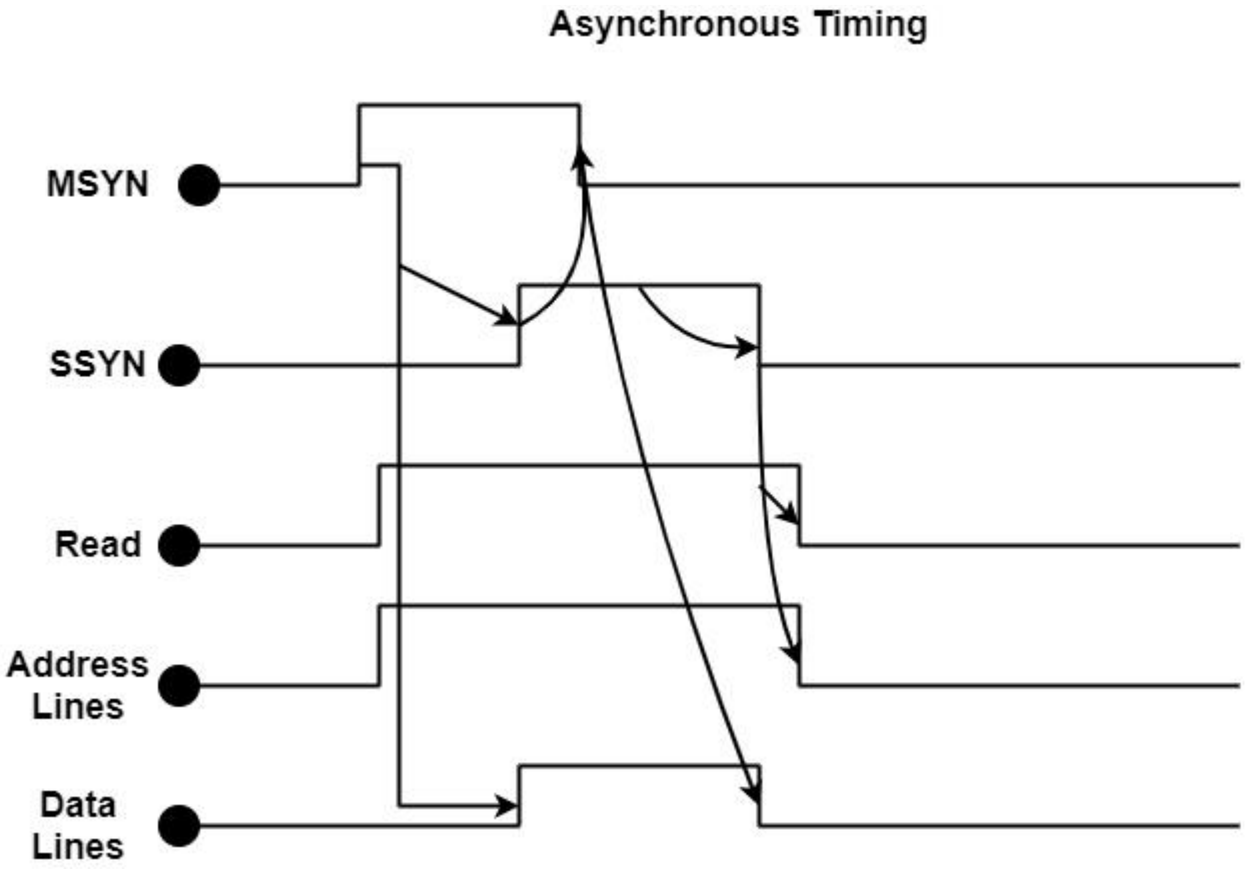
Timing

Timing defines how events are integrated on the bus. With synchronous timing, the circumstances of events on the bus are persistent by a clock. The figure shows the timing diagram for a synchronous read operation.



With asynchronous timing, the circumstances of one event on a bus follows and are based on the circumstances of a previous event. In this example, the CPU places address and read signals on the bus.

After pausing for these signals to maintain, it declares an MSYN (master sync) signal. It indicating the presence of valid current address and control signals. The memory module responds with data and an SSYN (slave sync) signal, indicating the response.



SMALL COMPUTER SYSTEMS INTERFACE (SCSI)

A small computer systems interface (SCSI) is a standard interface for connecting peripheral devices to a PC. Depending on the standard, generally it can connect up to 16 peripheral devices using a single bus including one host adapter.

SCSI is used to increase performance, deliver faster data transfer transmission and provide larger expansion for devices such as CD-ROM drives, scanners, DVD drives and CD writers. SCSI is also frequently used with RAID, servers, high-performance PCs and storage area networks SCSI has a controller in charge of transferring data between the devices and the SCSI bus.

It is either embedded on the motherboard or a host adapter is inserted into an expansion slot on the motherboard. The controller also contains SCSI basic input/output system, which is a small chip providing the required software to access and control devices.

Each device on a parallel SCSI bus must be assigned a number between 0 and 7 on a narrow bus or 0 and 15 on a wider bus. This number is called an SCSI ID. Newer serial SCSI IDs such as serialattached SCSI (SAS) use an automatic process assigning a 7-bit number with the use of serial storage architecture initiators.

SCSI WORKING PRINCIPLE

To implement SCSI on a system, you use a SCSI adapter to interface with the system bus, suitable SCSI devices such as SCSI hard drives, SCSI cables to daisy-chain the devices, and SCSI terminators for the ends of the bus. Each device on a SCSI bus must have a SCSI device ID number assigned to it, allowing SCSI to be used for daisy-chaining a number of devices together on a single parallel bus. You can change SCSI IDs by using dip switches or jumpers, or by using configuration software.

SCSI devices come in two basic types:

- **Single-ended devices:** Use one data lead and one ground lead to establish single-ended signal transmission over the bus. This type of device is more prone to the effects of noise and is less forgiving of cable lengths beyond specifications.
- **Differential devices:** Use two data leads, neither of which are at ground potential. These devices are generally more expensive but are resistant to the effects of noise and can often function over distances that exceed the SCSI specifications.

Universal Serial Bus (USB)

A Universal Serial Bus (USB) is a common interface that enables communication between devices and a host controller such as a personal computer (PC) or smartphone.

It connects peripheral devices such as digital cameras, mice, keyboards, printers, scanners, media devices, external hard drives and flash drives. Because of its wide variety of uses, including support for electrical power, the USB has replaced a wide range of interfaces like the parallel and serial port.

A USB is intended to enhance plug-and-play and allow hot swapping. Plug-and-play enables the operating system (OS) to spontaneously configure and discover a new peripheral device without having to restart the computer.

As well, hot swapping allows removal and replacement of a new peripheral without having to reboot.

There are several types of USB connectors. In the past the majority of USB cables were one of two types, type A and type B.

The USB 2.0 standard is type A; it has a flat rectangle interface that inserts into a hub or USB host which transmits data and supplies power. A keyboard or mouse are common examples of a type A USB connector.

A type B USB connector is square with slanted exterior corners. It is connected to an upstream port that uses a removable cable such as a printer. The type B connector also transmits data and supplies power.

Some type B connectors do not have a data connection and are used only as a power connection.

A Universal Serial Bus (USB) is basically a newer port that is used as a common interface to connect several different types of devices such as:

- Keyboards.
- Printers.
- Media devices.
- Cameras.
- Scanners.
- Mice.

It is designed for easy installation, faster transfer rates, higher quality cabling and hot-swapping. It has conclusively replaced the bulkier and slower serial and parallel ports.

One of the greatest features of the USB is hot swapping. This feature allows a device to be removed or replaced without the past prerequisite of rebooting and interrupting the system. Older ports required that a PC be restarted when adding or removing a new device.

Another USB feature is the use of direct current (DC). In fact, several devices use a USB power line to connect to DC current and do not transfer data. Example devices using a USB connector only for DC current include a set of speakers, an audio jack and power devices like a miniature refrigerator, coffee cup warmer or keyboard lamp.

USB Version 1 allowed for two speeds: 1.5 Mb/s (megabits per second) and 12 Mb/s, which work well for slow I/O devices. USB Version 2 allows up to 480 Mb/s and is backward compatible with slower USB devices. The first USB version 3 (USB 3.0 or SuperSpeed USB) was released in 2008, and allowed for a speed of 500 Mb/s. In 2013 and 2017, two new USB version 3 were released: USB 3.1 and USB 3.2, which allowed for 1.21 Gb/s and 2.42 Gb/s, respectively.

PARALLEL PROCESSING

Parallel processing is **a method in computing of running two or more processors (CPUs) to handle separate parts of an overall task**. Breaking up different parts of a task among multiple processors will help reduce the amount of time to run a program.

In parallel processing, **we take in multiple different forms of information at the same time**. This is especially important in vision. For example, when you see a bus coming towards you, you see its color, shape, depth, and motion all at once. If you had to assess those things one at a time, it would take far too long.

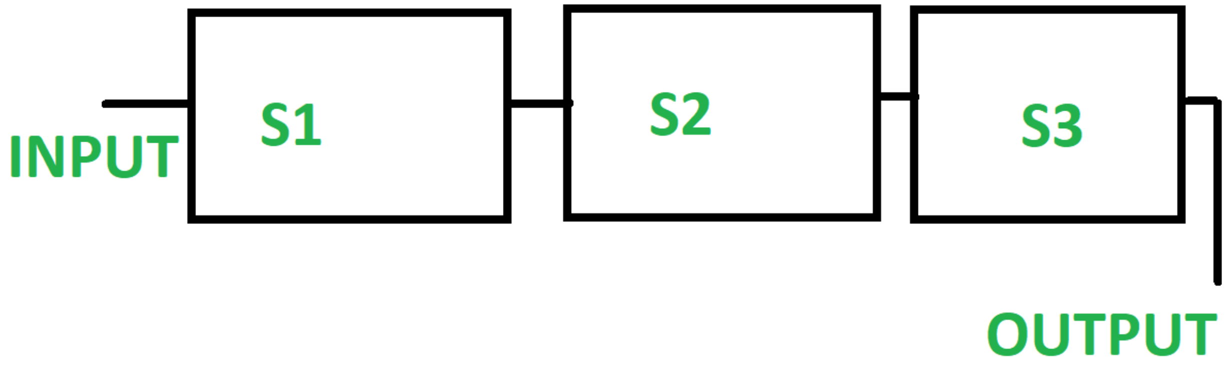
The advantages of parallel computing are that **computers can execute code more efficiently, which can save time and money by sorting through "big data" faster than ever**. Parallel programming can also solve more complex problems, bringing more resources to the table.

Notable applications for parallel processing (also known as parallel computing) include computational astrophysics, geoprocessing (or seismic surveying), climate modeling, agriculture estimates, financial risk management, video color correction, computational fluid dynamics, medical imaging and drug discovery.

LINEAR PIPELINE

Linear pipeline is **a pipeline in which a series of processors are connected together in a serial manner**. In linear pipeline the data flows from the first block to the final block of processor. The processing of data is done in a linear and sequential manner.

A linear pipeline processor is **a cascade of Processing Stages which are linearly connected to perform fixed function over a stream of data flowing from one end to the other**. Linear pipeline are static pipeline because they are used to perform fixed functions.

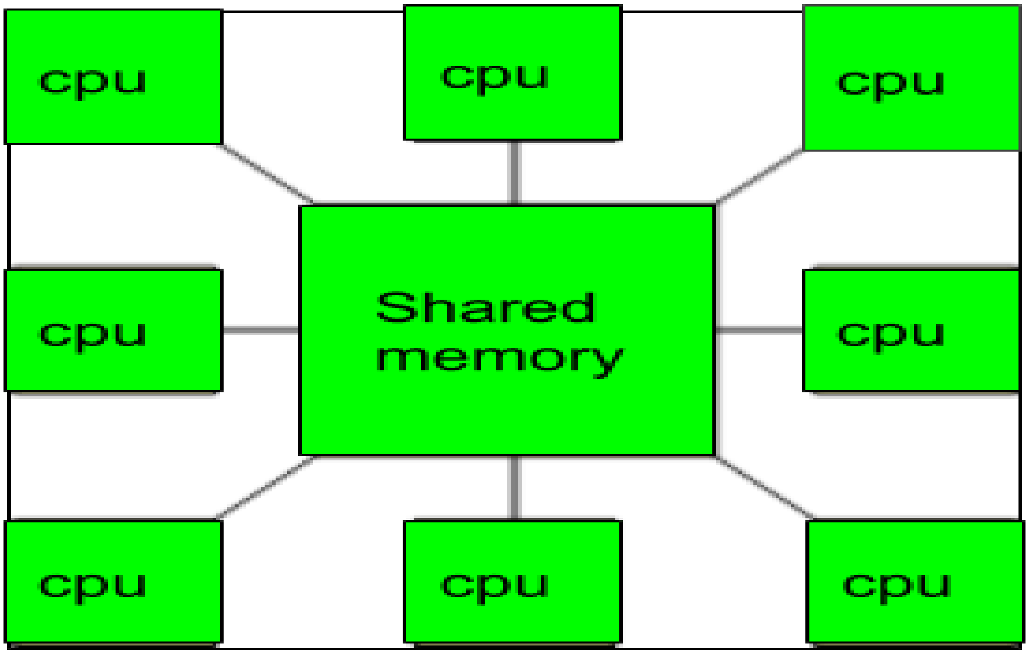


Linear pipeline is a pipeline in which a series of processors are connected together in a serial manner. In linear pipeline the data flows from the first block to the final block of processor. The processing of data is done in a linear and sequential manner. The input is supplied to the first block and we get the output from the last block till which the processing of data is being done. The linear pipelines can be further be divided into synchronous and asynchronous models.

MULTI PROCESSOR

Multiprocessing, in computing, a mode of operation in which two or more processors in a computer simultaneously process two or more different portions of the same program (set of instructions).

A multiprocessor is **a computer system with two or more central processing units (CPUs), with each one sharing the common main memory as well as the peripherals**. This helps in simultaneous processing of programs.



There are two types of multiprocessors, one is called **shared memory multiprocessor and another is distributed memory multiprocessor**. In shared memory multiprocessors, all the CPUs shares the common memory but in a distributed memory multiprocessor, every CPU has its own private memory.

The key objective of using a multiprocessor is to boost the system's execution speed, with other objectives being fault tolerance and application matching.

A good illustration of a multiprocessor is a single central tower attached to two computer systems. A multiprocessor is regarded as a means to improve computing speeds, performance and cost-effectiveness, as well as to provide enhanced availability and reliability.

FLYNN’S CLASSIFICATION

Flynn's Classification refers to **a classification of parallel computer architectures**. Parallel computers can be classified by the concurrency in processing sequences (streams), data, or instructions from the perspective of an assembly language programmer.

M.J. Flynn proposed a classification for the organization of a computer system by the number of instructions and data items that are manipulated simultaneously.

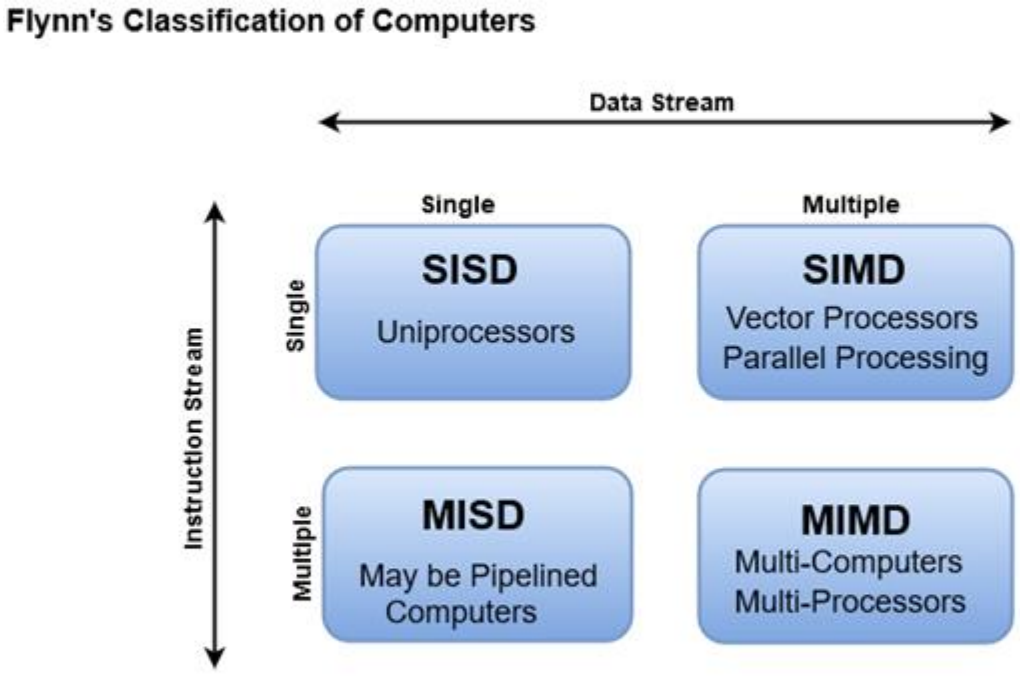
The sequence of instructions read from memory constitutes an **instruction stream**.

The operations performed on the data in the processor constitute a **data stream**.

Flynn's classification divides computers into four major groups that are:

1. Single instruction stream, single data stream (SISD)
2. Single instruction stream, multiple data stream (SIMD)
3. Multiple instruction stream, single data stream (MISD)
4. Multiple instruction stream, multiple data stream (MIMD)

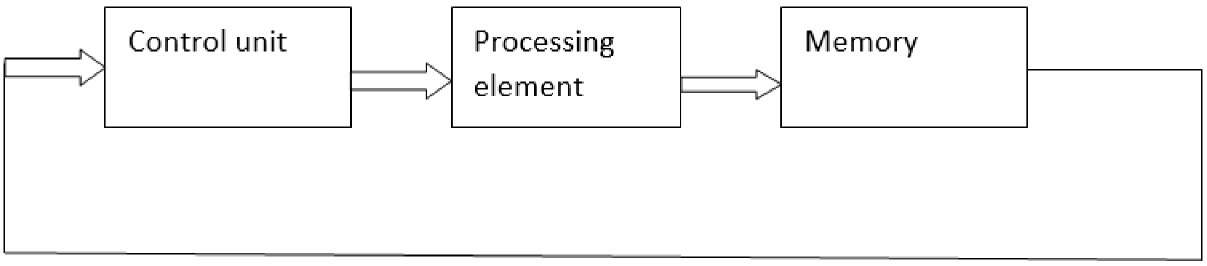
Flynn's classification divides computers into four major groups that are:



1) SISD (Single Instruction Single Data Stream)

Single instruction: Only one instruction stream is being acted or executed by CPU during one clock cycle.

Single data stream: Only one data stream is used as input during one clock cycle.

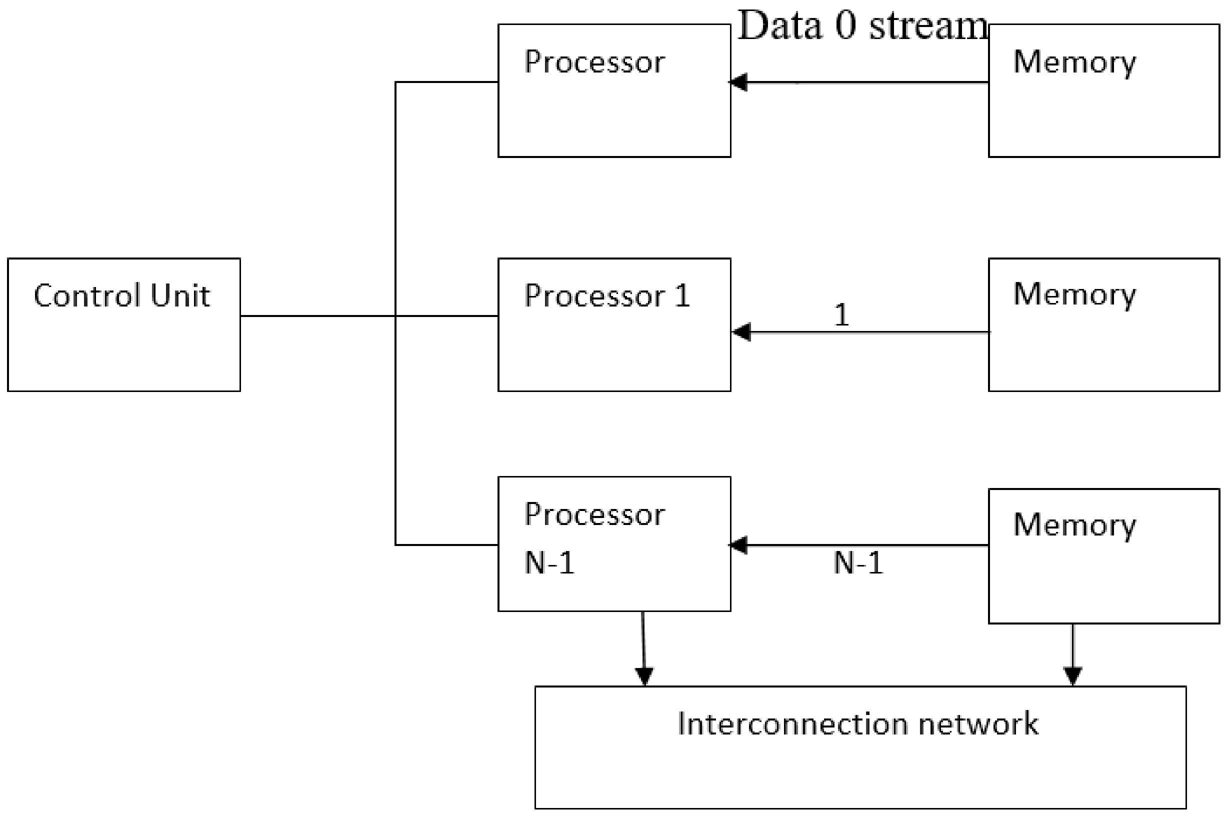


A SISD computing system is a uniprocessor machine that is capable of executing a single instruction operating on a single data stream. Most conventional computers have SISD architecture where all the instruction and data to be processed have to be stored in primary memory.

- It has one instruction stream one data stream.
- It does one thing at a time.
- It has capability of manipulating one data stream at a time by executing a single instruction stream.
- Most serial computers are based on SISD.
- Instructions may get overlapped during their execution
- Most SISD computers are pipelined. For example- IBM 370 computers.

2) SIMD (Single Instruction Multiple Data Stream)

A SIMD system is a multiprocessor machine, capable of executing the same instruction on all the CPUs but operating on the different data stream.

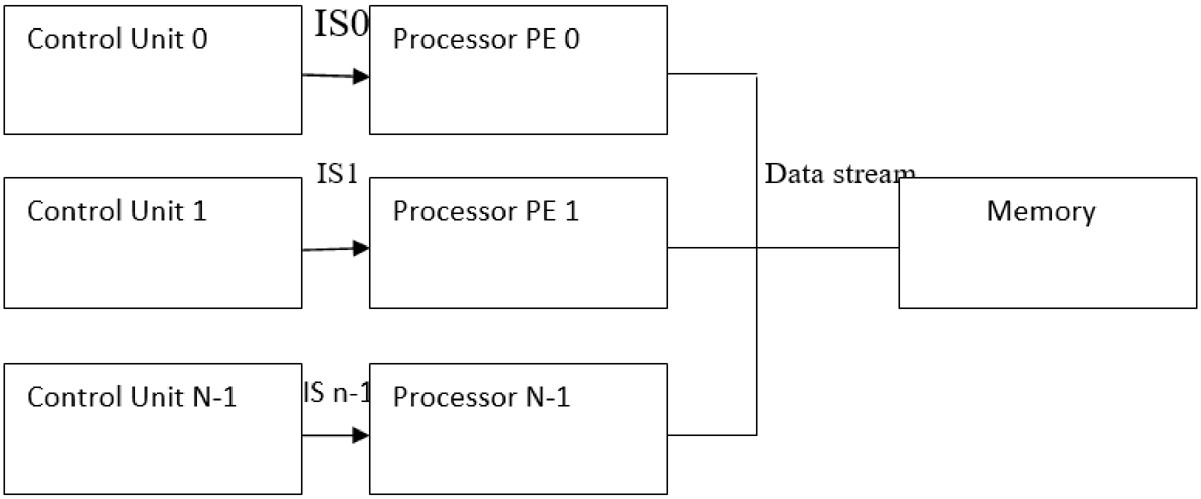


- It has a single control unit to generate one instruction stream at a time.
- A single control unit have multiple ALUs (Arithmetic and logic units) to work on multiple data streams simultaneously.
- It has capability to execute a single instruction stream on multiple data streams.
- Its also known as vector or array processors machine.
- In SIMD multiple processing units are supervised by a single control unit.

For example- ILLIAC-IV

3) MISD (Multiple Instruction Single Data stream)

An MISD computing is a multiprocessor machine capable of executing different instructions on processing elements but all of them operating on the same data set.

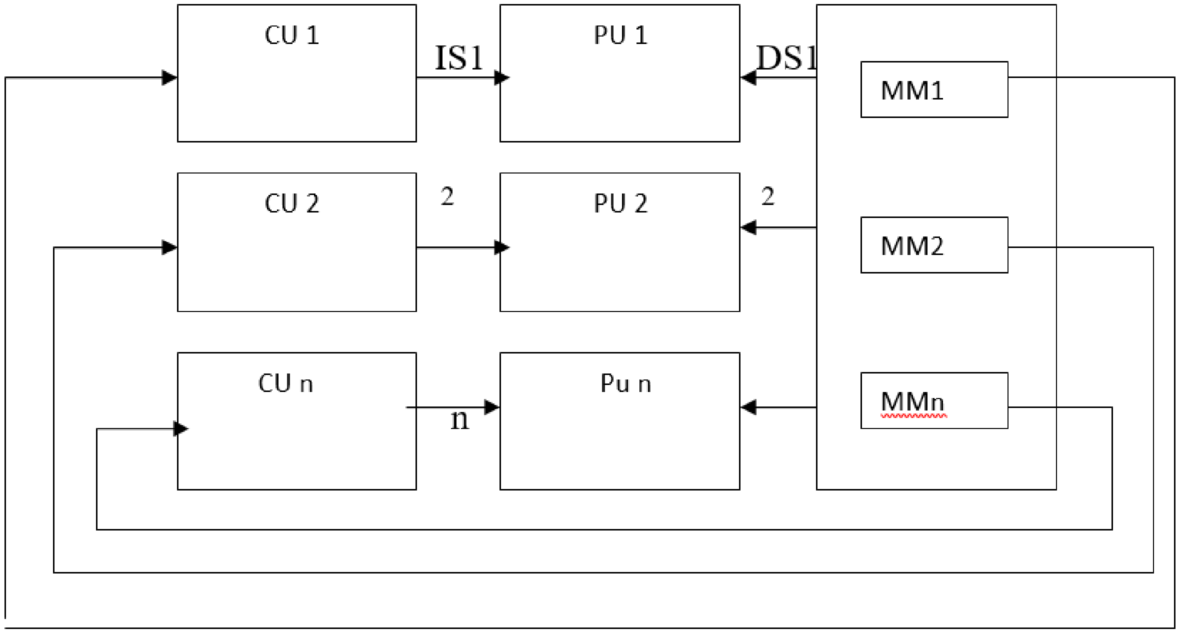


MISD computers have multiple instruction stream to execute on single data stream.

- This type of system is not to build practically, it’s a theoretical approach.
- It has multiple instruction stream, which operate on sama data stream.
- The output of one processor become the input of next processor.

4) MIMD (Multiple Instruction Multiple Data Stream)

A MIMD system is a multiprocessor machine that is capable of executing multiple instructions over multiple data streams. Each processing element has a separate instruction stream and data stream.



- It has capability of performing several programs simultaneously.
- It is similar to multiprocessor, in which multiple CPUs are operating independently to be a part of large system.
- Both multiprocessor and multi computer comes under MIMD.
- When multiple SISD works together than its called MSISD, which comes under category of MIMD.
- If number of instructions are high than it’s known as tightly coupled else known as loosely coupled.

For example- Cray-2 computers.