BCS303 COMPUTER ORGANIZATION AND ARCHITECTURE

UNIT – I ARCHITECTURE FUNDAMENTALS

Functional units – CPU registers – CPU organization – Different Types of Memory – Memory Hierarchy – Memory operations – Instruction format – Addressing modes – Basic I/O(4) operations – Evaluating CPU performance

ARCHITECTURE FUNDAMENTALS

Basics of computer Architecture & organization

* Computer organization refers the operational units of the system & their interconnection
* Computer organization explains how all the hardware components are organized and how they are functioning
* The list of instructions entered in a computer through input device is called computer program and the internal storage is called computer memory
* Based on the size, speed and cost, computers can be classified as follows.
  - Desktop computers
  - Notebook computers.
  - Workstations
  - Main frame
  - Super computer

Scanned by CamScanner

* Desktop computer:

- It's a personal computer, which has used in homes, schools and office.
- It has processing unit, cathode-ray tube display and keyboard that can be located easi

* Notebook computer (portable):

- It's mainly used for word processing and easily fit into our bag just like a tab
- The keyboard, flat panel display and proces are all packaged into a single unit.

* Workstations (High performance)

- It has graphics input/output capabilit
- Computational power is higher than personal computer
- Workstations are used in engineering applic

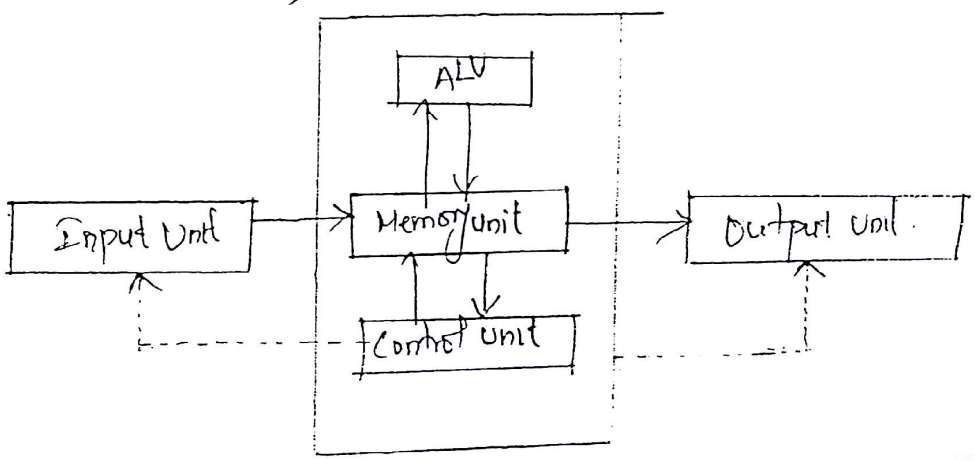
* Mainframe

- It's a large and powerful computer system exist at the lower end, used for business data processing.

* Supercomputer

- It exists in the high end, used for larg scale numerical calculation such as weather forecas

I. FUNCTIONAL UNIT



CPU

Scanned by CamScanner

computer consist of five functionally independent parts.

* Input Unit
* Memory unit
* Arithmetic & logic unit
* output Unit
* control Unit

Short notes on how functional unit works:

- The input unit accepts coded information from human operators, from electromechanical devices such as keyboard or from other devices through lines.
- The information received is either stored in the memory for later use or immediately used by ALU to perform desired operation.
- The processing steps are determined by the program stored in the memory. Finally results are sent back through output Unit.
- All of these actions are co-ordinated by control unit.

Input Unit

- i/p used to read the data, computer accepts coded information through i/p unit.
- Mostly used i/p device is keyboard, which is electronically connected to the processing part of a computer.
- Whenever the key is pressed, the corresponding letter or digit is automatically translated to its code and sent directly to either the memory or the processor.
- The other i/p devices are trackball, joystick & mouse.

Scanned by CamScanner

It is used to store programs and data

- There are two classes of memory device called primary storage and secondary storage.

Primary storage (main memory)

- Its a fast memory that operates at electronic speeds.
- Programs are stored in the main memory while they are being executed.

Secondary storage:

- It is cheap and additional device.
- It is used when large amount of data have to be used and when some of the data need not be accessed frequently.

Arithmetic and logic unit:

- Computer operations are executed in the ALU unit of the processor.
- Ex: Suppose two numbers located in the main memory are to be added. They are brought into the arithmetic unit where actual addition is carried out. The sum may be stored in the memory or retained in the processor for immediate use.
- For other operation, such as multiplication & division the required operands will be stored in ALU for +, -
- We cannot say all the operands are stored in the main memory. Processor has number of high speed storage elements called registers, which is used for storing frequently used operands.

Registers [0 0 0 0 0 1 1 1] => 2 is stored in register.

one word of data contains n bit

[2,3,4 => operands
*, ÷ => operator]

Eg: 2 * 3 & 2 ÷ 4 In a single program, a = b and a - d ; a = 2

frequently used operand is 2 which is stored in register.

Scanned by CamScanner

Output unit:

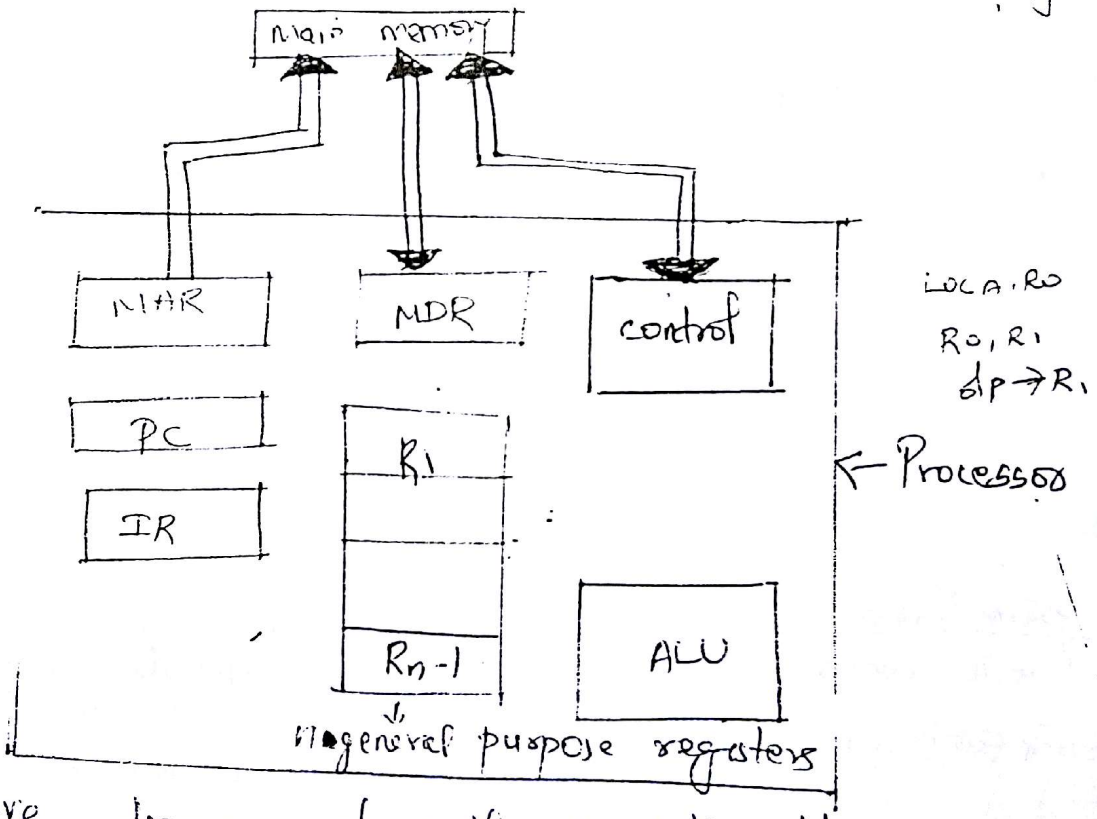
- Function of the output unit is to send processed results to the outside world.
- (ex. Printer. Claser printer, inkjet printer).
- Monitor is also one of the output device which supports video terminals and graphic display.

Control unit

- The memory, ALU, I/O units store and process information and perform input & output operations.
- The operations of these units are co-ordinated by control unit
- Control unit is the nerve centre that sends control signal to other units and sense their states.

II. CPU REGISTERS

Processor contains number of registers in addition to the ALU and the control unit for various purposes. The registers or



The above diagram shows the connection b/w CPU & main memory

Scanned by CamScanner

Instruction register

IR holds the instruction executed by the processor. Its o/p is available to the con

Program counter:

Its a special purpose register that contains the address of the next instruction to be fetched and executed. During the execution of the one instr., pc updated and point the address of the next instr. to be executed.

Memory Address Register (Read cycle)

The MAR holds the address of the memory location to be accessed.

Memory Data register: (Write cycle)

The MDR contains the data to be written into or read from memory location, that is pointed by MAR. MAR & MDR facilitates the communication b/w memory & processor.

III. CPU ORGANIZATION

* CPU organization done by bus
* It's a communication pathway connecting two or more devices, It's a shared transmission medium
* Multiple devices are connected by the bus, but the signal transmitted by any one device is taken by the bus
* A bus consists of multiple lines. Each line is transmitting signal represented by binary 0 or 1.

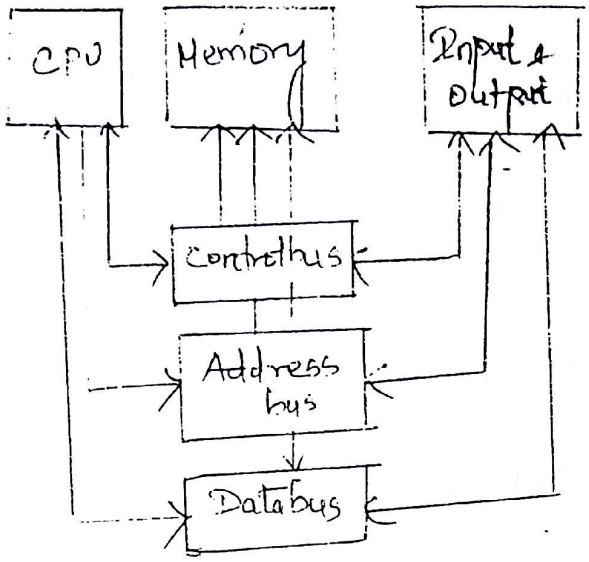
  Eg: 8 bit data transmitted over eight bus lines.
* Computer contains number of different buses.
* A bus that connects major computer components is called a system bus.

System bus:

* It consists of 50 to hundreds of seperate lines
* Each line is assigned a particular meaning or function.

Scanned by CamScanner

In every bus design, the bus line can be classified into 3 groups



Data bus:

* Data bus is a bidirectional path. It carries the actual data between the processor (CPU), the memory and the peripherals (Input and output).
* Data bus consists of 32 to 100 separate lines, no. of lines is called width of the data bus.

Address bus:

* Address bus used to specify the memory locations for the data being transferred.
* One device passes memory address to another device through this bus.
* Eg: If the processor wishes to read a word (8,16 or 32) of data from memory, it puts the address of the desired word on the address lines.

Control bus:

* It has the set of instructions about the data.
* It also carries the control, timing and co-ordination signals to manage the various functions across the system.
* It includes Memory R/W, I/O Read & write, Transfer ACK, Bus request & grant, Interrupt Req & Ack, clock & reset.

Scanned by CamScanner

Bus Width

* The size of the bus is known as width.
* It determines how much data can be transfer at a time

Bus speed

* This refers to the no. of bits or bytes the bus can send per unit time.

Internal bus:

* Its also known as internal data bus, memory bus, system bus or front side bus
* It connects all the internal components of a computer so it is called local bus

External bus:

* Its also known as expansion bus, it connects the different external devices such as printer.

Different types of memory

* Memory is divided into large numbers of cells.
* Each cell has a unique address which varies from zero to memory size minus one.
* Eg- If computer has 64k words then memory unit has 64 * 1024 = 65536 memory locations. The address of these locations varies from 0 to 65535.
* It has two types
  * Internal memory
    Cache memory and primary memory
  * External memory
    Magnetic disk / optical disk

Scanned by CamScanner

Primary memory

→ It communicates directly with the CPU.

→ Storage area in which all the programs are executed.

→ It has two types : RAM & ROM

RAM:

- Its a random access Memory, Volatile memory.
- It contains a large number of semiconductor storage cells, each capable of storing 1 bit of information
- These cells are grouped into fixed size called word
- The number of bits in each word is often referred as the word length of the computer.
- Typical word length ranges from 16 to 64 bits.
- Data stored in RAM will be lost, when system got switched off.

RAM has 2 types

↓ STATIC RAM

↓ DYNAMIC RAM

Static RAM:

It consists of internal flipflops that store binary information, it has shortest read & write cycles.

Dynamic RAM:

Stores information in the form of electric charges that are applied to capacitors, It offers reduced power consumption and larger storage capacity in a single memory chip.

Scanned by CamScanner

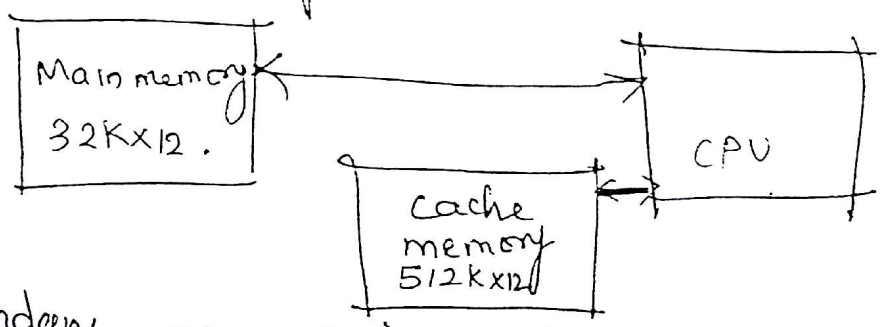
* Its a read only memory
we can read but cannot write.
* Its non-volatile, The information is stored perm
* It stores such instructions as are required to start computer when system get started.
* Its also used in washing machine, oven etc.

Types of ROM:

* MROM (Masked ROM)
* PROM (Programmable ROM)
* EPROM (Erasable PROM)
  The contents of this RAM can be deleted by showing ultra violet rays on a chip.
* EEPROM (Electrically Erasable PROM)
  The contents of this RAM can be deleted by giving sub instructions or program.

Cache memory:

Its a fast and small memory. Its used to reduce average memory access time. It is placed between CPU and memory.



Secondary memory (or) Auxiliary memory

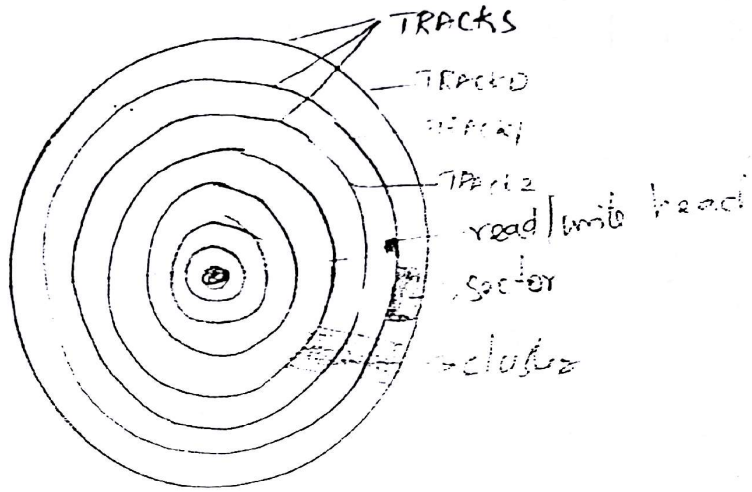
Magnetic Disk:

* Circular plate coated / constructed of a metal or plastic coated with magnetizing material

Scanned by CamScanner

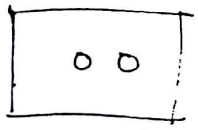
sides of the disk are used and several disk may be staked on one spindle with read/write head available on each surface.

* Bits are stored in spots along concentric circle called tracks
* Tracks are commonly divided into sectione called sectors.
* Disks that are permanently attached to the unit and cannot be removed by occasional uses is called hard disk.



Magnetic tape:

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



* The tape is a strip of plastic coated with a magnetic recording medium.
* Bits are recorded as magnetic spots on the tape along several tracks.

Scanned by CamScanner

Characteristics of memory device in memory hierarchy.

Component	CPU	cache memory	main memory	Disk Memory	Tape Memory
Access type	Random Access	Random Access	Random access	Direct access	Sequential access
Capacity (Bytes)	64-1024	8-256 KB	8-64 MB	1-10 GB	1TB
Latency	1-10 ns	20 ns	50 ns	10 ms	10 ms-10 s
Blocksize	1 word	16 words	16 words	4 KB	4 KB
Bandwidth	System clock rate	8 MB/s	1 MB/s	1 MB/s	1 MB/s
Cost /MB.	High	$ 500	$ 30	$ 0.25	$ 0.02.

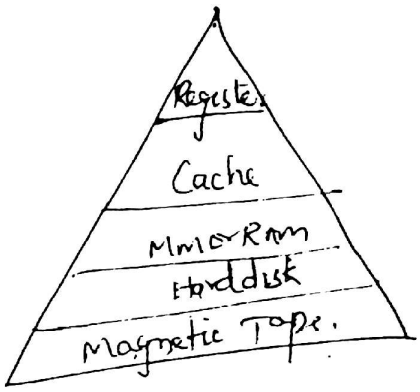
Scanned by CamScanner

* The hierarchial arrangement of storage in current computer architecture is called Memory hierarchy.

* Each level of hierarchy is of higher speed, lower latency & of smaller size c lower levels.

* Memory hierarchy in most computer as follow,

- Processor registers
- Cache memory
- Main memory (DRAM)
- Hard disk
- Magnetic tape

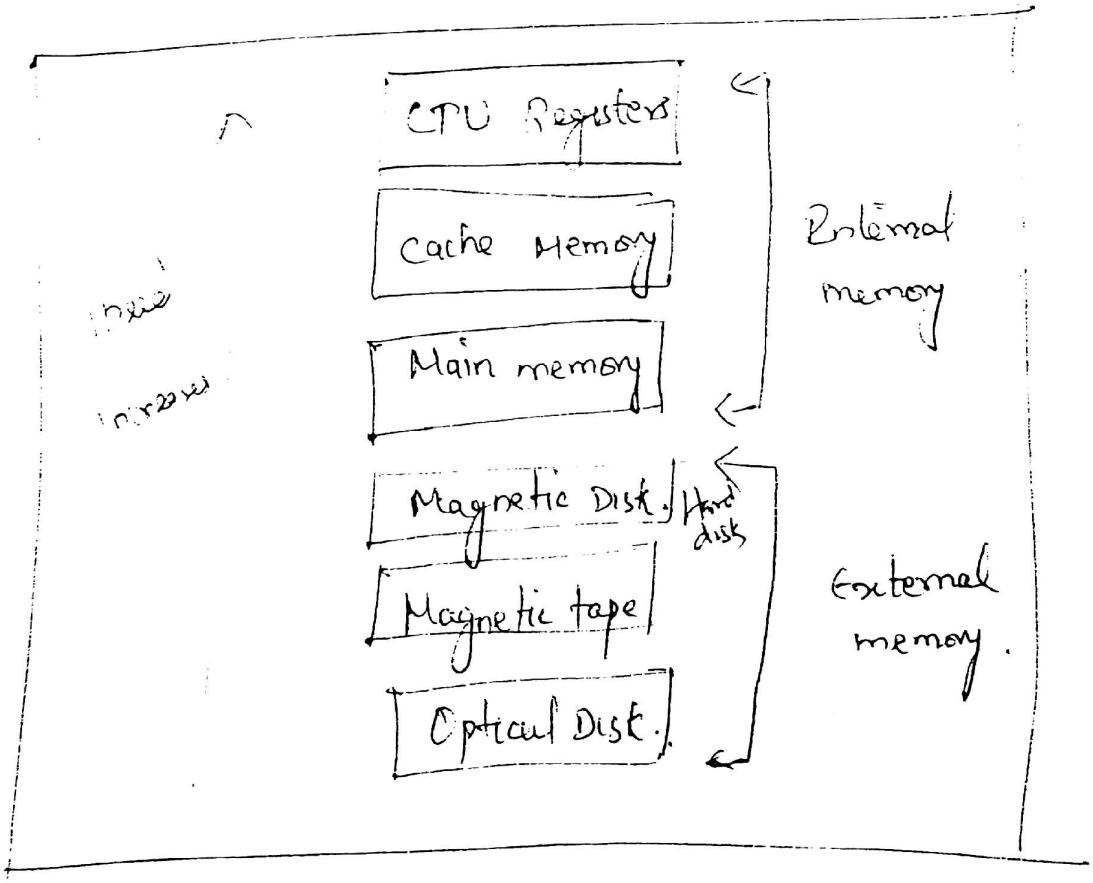


Cache memory has 3 types:

L1 cache: It is accessed without any delay (10KB)

L2 cache: It takes more clock cycles to access than L1 cache (512KB)

L3 cache: It takes more clock cycles to access than L2 cache (more than 1 MB).



Scanned by CamScanner

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

Three Address Instruction

ADD	R1, A, B	$R_1 \leftarrow M[A] + M[B]$
ADD	R2, C, D	$R_2 \leftarrow M[C] + M[D]$
MUL	X, R1, R2	$M[X] \leftarrow R_1 * R_2$

Two Address Instruction

MOV	R1, A	$R_1 \leftarrow M[A]$
ADD	R1, B	$R_1 \leftarrow R_1 + M[B]$
MOV	R2, C	$R_2 \leftarrow R_2 + M[D]$
MUL	R1, R2	$R_1 \leftarrow R_1 + R_2$
MOV	X, R1	$M[X] \leftarrow R_1$.

One Address Instruction.

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

LOAD	A	$[AC \leftarrow M[A]]$
ADD	B	$[ACC \leftarrow AC + M[B]]$
STORE	T	$[M[T] \leftarrow ACC]$
LOAD	C	$[AC \leftarrow M[C]]$
ADD	D	$[ACC \leftarrow ACC + M[D]]$
MUL	T	$[ACC \leftarrow ACC * M[T]]$
STORE	X	$[M[X] \leftarrow AC]$

Zero - Address Instruction

PUSH A

PUSH B

ADD

PUSH C

PUSH D.

Scanned by CamScanner

operation)

ADD LOCA, R0

→ steps

Load LOCA, R1 → Memory Access operation

Add R1, R0 → ALU operation.

The first instn transfers the contents of memory location LOCA into processor register R1, and the second instn adds the contents of registers R1 & R0 & places the sum into R0.

→ To execute an instn, the CPU must cause the instn. to be transfered from the main memory to the CPU. Operants & results must also be moved between the main memory & CPU.

→ The basic two operations are called fetch (read) and store (write).

→ fetch operation transfers a copy of the content of a specified memory location to CPU.

→ The store operation transfers the words of infm from the CPU to a main memory location.

VII INSTRUCTION FORMAT

* To perform an execution, infm can be transfered from one device to another in computer.
* Possible locations are memory location, CPU Registers,
* location can be denoted by LOC.
* The contents of a location or register are denoted by placing the corresponding name between square brackets.
* Ex: R1 ← [LOC], The contents of memory location LOC are transfered into Register R1.
* The operation of adding two number, C = A+B.

C = [A] + [B] ⇒ The contents of memory location A & B are fetched from main memory & transfered to CPU, where sum is computed. The result is send back to memory & stored in Loc.

Scanned by CamScanner

- Single machine instruction contains three fields each provides the address of one of three operands A, B and C.
- The operation field of the instruction specifies that addition is to be performed.

→ The three adress instn. can be represented as follows,

Add A, B, C.

Operands A & B are source operand & C is destination

- General instn. of this type has the format,

Operation Source 1, Source 2, Destination.

→ The two-address instruction can be specified as follows

Operation Source, Destination.

Add A, B. ⇒ [mov B, C
ADD A, C].

- It performs the operation, B ← [A] + [B].
- When sum is calculated, the result is sent to the main memory and stored in location B, replacing the original contents of B, i.e. B is both source & destination.

→ One-address Instruction

- The CPU register usually called accumulator used for one-address instn

Eg. Add A, Add the contents of memory location A to the contents of acc & store the sum into the accumulator

- The other one address instructions are LOAD A, Store A
- LOAD A ⇒ Load instn copies the contents of memory location into accumulator.
- Store A ⇒ Store instn. copies the contents of accumulator into memory location A.
- Eg. C = [A] + [B] ⇒ Load A
  Add B
  Store C
- In the above instn. Address A specifies source & C specifies destination

→ Zero address instruction,

PUSH & POP. Example in pg no: 7

Scanned by CamScanner

VI. ADDRESSING MODES

* Addressing mode is used to specify the location of operand which is used to execute the instruction.

* Addressing modes gives the programmer, a flexibility for writing program that are efficient with respect to number of instructions and execution time.

* General format of addressing mode given below.

Opcode	Mode	Address

Opcode: Operation code specifies the operation to be performed.

Mode: Locate the operands needed for operation.

Address field: Specify memory/register address of operand.

Different types of addressing mode.

1) Immediate addressing mode:

The operand is given expliutly in the instructions.

ie, the operand is specified in the instruction itself.

Eg1: MOVE #200, R0, this instn. copies the value 200 to register R0.

is used to indicate the value is an immediate operand.

Eg2: A=B+6

MOVE B, R1
ADD #6, R1
MOVE R1, A.

ADD	6 (8 bits)	R1

MOVE	200	R0

Effective address:

EA provides information from which the memory address of the operand can be deduced. Every addressing mode specified by EA.

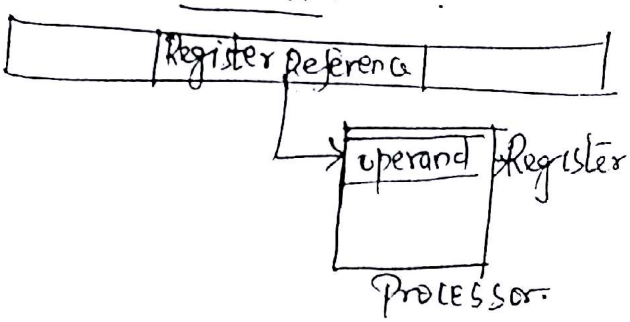
Scanned by CamScanner

Register addressing mode.

* The operand is the contents of CPU register the name of the register is given in the instruction.

* The address field refers to a register. particular - register selected from a register field of instruction

$EA = R$.



(3) Register Indirect addressing mode.

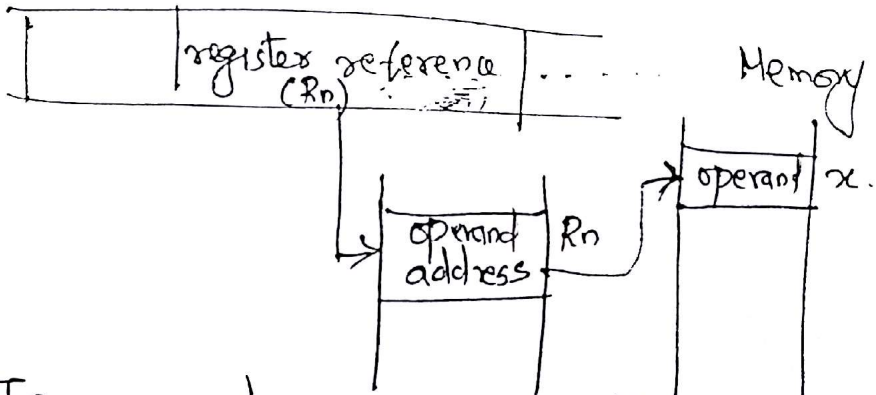
* EA is the contents of register or location whose address appears in the instruction.

* Operand is in memory cell pointed by the contents of the register R;

$$EA = (R)$$

ADD (B) ⇒ Add the contents of B to accumulator, EA (B

ADD $R_H, (R_1)$ $[\because R_H \leftarrow R_H + [R_1]]$.



The operand x obtained through the memory location which is specified by register Rn. ie, the operand accessed indirectly by register.

Scanned by CamScanner

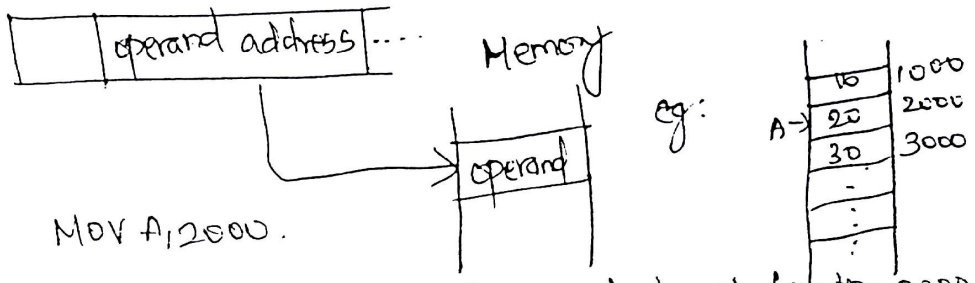
Addressing Modes:

* Address field contains the address of operand.
* operand is in a memory location, the address of this location is given explicitly in the instruction

$$\boxed{EA = (A).}$$

* EA specifies the memory address of an operand.

ADD R1, X EA ⇒ R1 ← R1+[X]

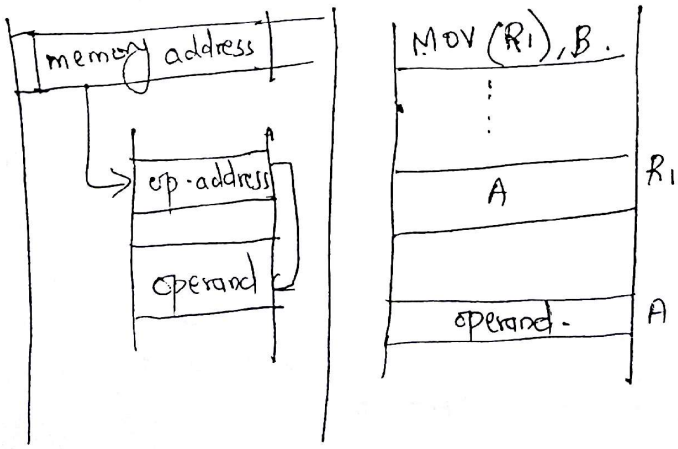


MOV A, 2000.

The above instruction copies the contents of location 2000 i.e, 20 to the register A. Address is given explicitly.

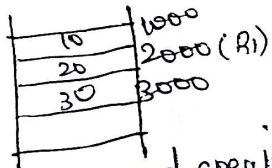
(5) Indirect Addressing mode.

* The address field of the instn. gives the address where the operand is stored in memory.
* The EA of the operand is the contents of register or the main memory location whose address is given explicitly in the instruction.



R1 holds the address A which has operand.

eg: R1 has 2000.



20 is the operand specified by A moved to B.

Scanned by CamScanner

* The effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of the register are incremented to address the next location.

Eg: MOV (R2)+, R0 R2 = 2000, R0 has 20.
After copy, R2 = 2001.

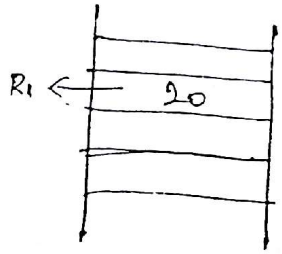
* Copies the contents of register R0 into the memory location whose address is specified by the contents of register R2. After copy operation, the contents of register R2 are automatically incremented by 1.

(T) Auto decrement mode:

The contents of a register specified in the instruction are decremented & they are used as an effective address to access a memory location.

Eg: MOV R1, -(R0)

R0 = 2000
After decrement
R0 = 1999



R1 = 20.

The instruction initially decrements the contents of register R0 and then the decremented contents of register R0 is used to address the memory location. Finally the contents from the addressed memory location are copied on to the register R1.

Auto Increment mode: EA = (Ri)+

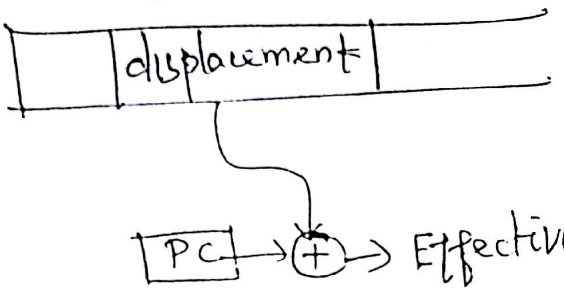
Decrement mode: EA -(Ri)

Scanned by CamScanner

Relative addressing mode

* Its a PC (Program Counter), relative addressing base address is implicitly given in programcounter.

$$\boxed{EA = A + (PC)}$$



* Displacement is a value, also known as offset, used to obtain the EA of few modes.

Base register addressing mode comes under relative addressing, where other registers can play the role of PC.

$$\boxed{EA = A + (R)}$$

R may be explicit or implicit.

3. Implied Addressing Mode

* Also known as STACK addressing mode.
* Special operations that work on the stack (PUSH, POP, CALL, RETURN) implicitly refer to the top of the stack.
* So they are called stack addressing, where the effective address is stored in the stack point registe
* operand is on the top of stack, PUSH(x), POP(x)

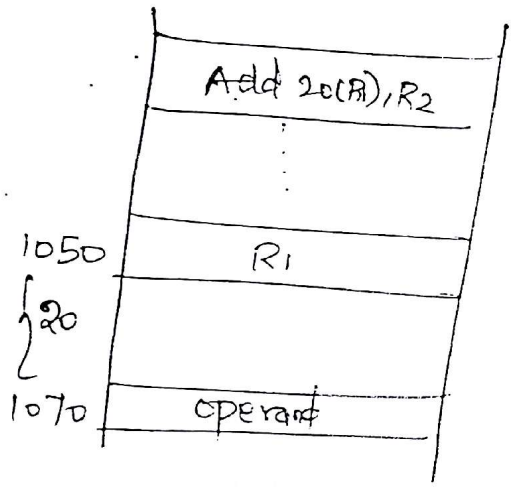
Scanned by CamScanner

(10) Indexed Addressing mode

* The effective address of the operand is generated by adding a constant value (specified in the instruction) to the contents of register.
* That constant value is known as displacement.
* The register used may be any one of a set of general purpose registers in CPU, it is referred as index register, index mode specified by,

$X(R)$
↓
displacement

EA of operand is given by $EA = X + [R]$.



In the figure, the index register R_1 contains the address of a memory location & the value X defines the distance i.e from the address to the location where the operand is found.

IX. BASIC I/O OPERATIONS.

* To initate an I/O operation, the CPU issues an address (specify the I/O module & device) and I/O command
* There are 4 types of I/O commands.
  * Control commands
  * Test status
  * Read commands
  * Write commands.

Scanned by CamScanner

Basic I/O Operations

Consider the task that requires character input from the keyboard of a video terminal and produces character output that is displayed on the screen of same terminal. A simple way of performing such I/O task is use a method known as program controlled I/O. The video terminal consisting of a keyboard unit & a display unit is a simple I/O device that we can use to illustrate the basic principles of program controlled I/O.

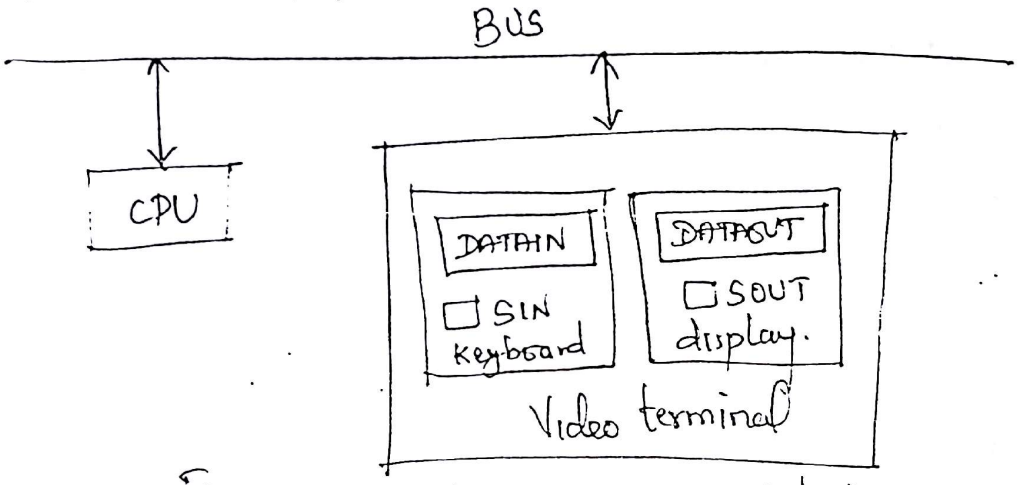


Fig: A video terminal connected to CPU.

* When a character is entered through keyboard, the key stores the corresponding character code in an 8-bit buffer register associated with the keyboard.
* Buffer register is called DATAIN, the sync control flag SIN is set to 1.
* A program monitors SIN, when SIN equals 1, it reads the contents of DATAIN.
* When the character is transferred to the CPU, SIN is automatically cleared to 0. If a second character is entered at the keyboard, SIN is again set to 1 and the process repeats.

The terminal can transfer 1000 characters per second. Processor

Scanned by CamScanner

transferred from CPU to display.

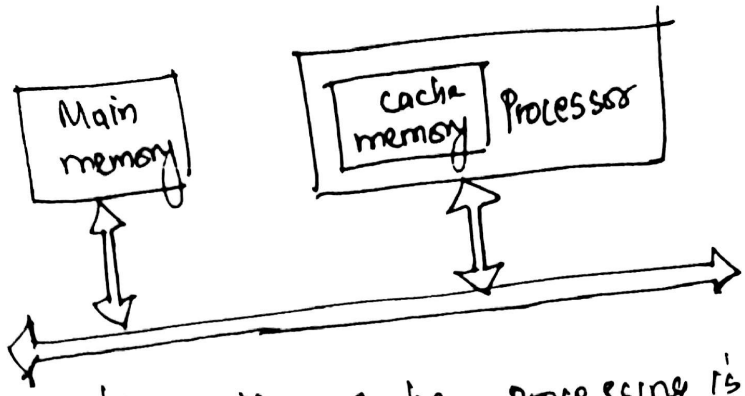
* A buffer register, DATAOUT & a synchronization SOUT are used for this transfer.
* When SOUT equals 1, the display is ready to receive a character, the CPU monitors SOUT, when SOUT set to 1, the CPU transfers a character code to DATAOUT.
* The transfer of a character to DATAOUT clears SOUT when the character has been displayed, and the display device is ready to receive a second character SOUT is again set to 1.
* The buffer registers DATAIN and DATAOUT and the Sync. flags SIN & SOUT are part of the circuit is known as device interface, which connects video terminal to CPU.
* The circuit is connected to the CPU via bus.

X. Evaluating CPU performance.

* The important measure of computer performance is the total time needed to execute application program.
* All the components must work properly for faster execution.
* Cache memory is used for faster data access that leads to execution in fast.
* Modern processors are constructed on chips large enough to include a high speed memory called cache.

Scanned by CamScanner

Processor cache Memory



In a system with a cache, processing is done by bringing instructions & data from the main memory into the cache when they are first needed.

To evaluate the performance, we dissuss about,

Processor clock

- Processor circuits are controlled by timing signal called clock.
- A clock is a circuit that generates, a signal that defines regular time intervals during basic steps of executi
- Processor clock rate is specified by clock rate in cycles per second.
- Today's personal computer and workstation clockrate ranges from 20 million to 2000 million cycles per second.
- cycles per second denoted by 'hertz', million denoted by mega mHz, 200 million cps is 200mHz.

Basic performance Equation:

- Lets T be the time required for the processor.
- Complete execution of program requires execution of N machine language instruction.
- Average number of basic steps needed to execute one Instn. is S, each step completed in one clock cycle.
- If clock rate is R cycles per second, then execution time $T = \frac{(N \times S)}{R}$ is called basic performance equation

Scanned by CamScanner

of the ... of N is reduced, time reduced, if ...
increased, T reduced.

3 Pipelining:

* ADD R1, R2, R3, while performing this operation, Contents of R1 & R2 added and output will be moved to R3.
* When o/p is moved to R3, at the same time another ADD instruction follows immediately.
* When i/p operand transferred to ALU, at the same time previous o/p is being transferred to R3.
* Overlapping of 2 instruction is called pipelining.
* If overlapping is done, execution will be completed fastly.

4. Parallelism:

It is possible to perform transfers to and from secondary storage disks & tapes in parallel with program execution in the processor or with activity in other I/O devices, this is called parallelism.

In summary, the use of caches, pipelining & parallelism leads to increased program execution rates, that gives the better performance of CPU.

5. Basic performance measures:

Response time: Time spent to complete an event or an operation. Also referred to as execution time.

Through put: Amount of work done at give time. Also referred as bandwidth

Scanned by CamScanner

CPU performance equation

$$\text{CPU time} = \frac{\text{CPU clock cycles for a program}}{\text{clock rate}}$$

$$\text{clock cycles per instruction (CPI)} = \frac{\text{CPU clock cycles for a program}}{\text{IC (Instruction Count)}}$$

$$\text{CPU time} = IC \times CPI \times \text{clock cycle time}$$

(or)

$$\text{CPU time} = \frac{IC \times CPI}{\text{clock rate}}$$

CPU performance depends upon three characteristics clock cycle, clock cycles per instruction (CPI), and Instruction count (IC)

$$\text{CPU clock cycles} = \sum_{i=1}^{n} CPI_i \times IC_i$$

IC_i = Represents no. of times instruction i is executed in a program

CPI_i = Represents average number of clock cycle for instruction i.

$$\text{CPU time} = \left(\sum_{i=1}^{n} CPI_i \times IC_i\right) \times \text{clock cycle time}$$

$$\text{overall CPI} = \frac{\sum_{i=1}^{n} CPI_i \times IC_i}{\text{Instruction count}}$$

$$= \sum_{i=1}^{n} CPI_i \times \left(\frac{IC_i}{\text{Instruction count}}\right)$$

Scanned by CamScanner

- A standard by which something can be measured.

* Standard Performance Evaluation Corporation (SPEC) created benchmark application suites in 1980's to deliver better benchmarks for workstation.

* SPEC created a benchmark set focussing on processor performance.

* SPEC rating = $\frac{\text{Running time on the ref. computer}}{\text{Running time on the computer under test}}$

* The test is repeated for all the programs in the SPEC suite, and the result is computed.

* Let $SPEC_i$ be the rating for program i in the suite. Overall rating is given by,

$$\text{SPEC rating} = \left(\prod_{i=1}^{n} SPEC_i\right)^{1/n}$$

n is the number of programs.

Scanned by CamScanner

UNIT-II

Instruction sets of different machines.

An instruction set is a group of commands for a CPU in machine language. The term can refer to all possible instructions for a CPU or a subset of instructions to enhance its performance in certain situations. All CPUs have instruction sets that enable commands to the processor directing the CPU to switch the relevant transistors. The eg. commands are read, write & Move.

A computer must have instructions capable of performing four types of basic operations, such as

1. * **Data transfer Instructions:**

Move - Transfer data from source location to destination location (Ex: MOV X1 R1)

LOAD - Transfer data from a memory location to a CPU register (Ex: LOAD R1, X)

STORE - Transfer data from CPU register to memory location (Ex: STORE X)

PUSH - Transfer data to the stack (Ex: PUSH X)

POP - Transfer data from the stack (Ex: POP X)

XCHG - Exchange or swap the contents of source and destination.

2. * **Arithmetic Instruction:**

ADD - Addition operation, ADD R1, B

SUB - Subtraction operation, SUB R2, C

MUL - Multiplication operation, MUL R1, R2

DIV - Division operation, DIV R1, R2

INC - Increment operation, INR A

DEC - Decrement operation, DCR B

Scanned by CamScanner

3) Logical Instructions:

NOT - Complement the operand
OR - Perform bitwise logical OR operation
AND - Perform bitwise Logical AND operation
XOR - Perform bitwise 'exclusive OR' of operand.

4) Control Transfer Instruction

JUMP - Branch (Unconditional transfer)
JUMPIF - Branch on condition (conditional transfer)
INT - Interrupt (software Interrupt)
LOOP - Iteration
RET - Return (Restore)

i	MOVE A,R0
i+1	ADD B,R0
i+2	MOVE R0,C

5) Input - Output Instruction

IN - Input (Reads data from input device)
Ex: IN 8 bit port address
OUT - Output (Sends a databyte to an output device)
Start I/O - To start I/O program
Halt I/O - Abort the I/O program

6) String Manipulation

MOVS - Move byte or word of string
LODS - Load byte or word of string
CMPS - Compare byte or word of string
STOS - Store byte or word of string

7) Transfer Control Instructions (Processor control Instructions)

HLT - Halt (Processor stops executing & enters wait state)
NOOP - No operation
CMC - Complement carry flag
CLC - Clear carry flag
ESC - Escape (After escape instn. will be given to co-processor)
WAIT - Freeze instn. cycle till a specified condn is specified

8) Translate

XLAT - Translate (Convert given code into another)
PACK - Converts unpacked decimal into packed decimal
UNPACK - Convert packed decimal into unpacked decimal.

Scanned by CamScanner

II. PARALLEL PROCESSING

Instead of processing each instruction sequentially, a parallel processing system is able to perform concurrent data processing to achieve faster execution time.

Eg: Fetch the next instruction while the current instructions is in execution

Purpose:

1) To speed up the computer processing capability.
2) To increase the throughput

The amount of processing that can be accomplished during a given interval time is called throughput.

Disadvantages:

1. Amount of hardware increases
2. Cost of the system increases.
3. Complexity of the CPU also increases.

In general, the digital computer's may be classified into 4 types according to the multiplicity of instructions and data streams they deal with

This scheme of classifying the computer organization based on instruction and data stream was introduced by Michael J. Flynn and is called as Flynn's classification.

Instruction stream

An instruction stream is a sequence of instructions executed by the machine.

Data stream:

The data stream is a sequence of data including the input and the result.

Scanned by CamScanner

BRANCHING:

→ Consider the task contains list of n numbers. The addresses of the memory locations containing the n numbers are given as NUM1, NUM2 ... NUMn. Add each instruction used to add each number to the contents of register R0. After all the numbers have been added, result is placed into SUM.

→ Branch instruction loads new value to the PC. As a result, the CPU fetches and executes the instn. at this new address.

→ A conditional branch instruction also causes a branch, but only if a specified condition is satisfied. If the condn is not satisfied, the PC is incremented in the normal way, the next instruction in sequential address order is fetched & executed.

Program for adding n numbers

i	MOVE NUM1, R0
i+1	Add NUM2, R0
i+2	Add NUM3, R0
	⋮
i+n-1	Add NUMn, R0
i+n	Move R0, SUM
	⋮
SUM	
NUM1	
NUM2	
	⋮
NUMn	

Using a loop to add n numbers

	Move N, R1	101
	Clear R0	102
	Determine address of Next number and add Next number to R0	103
	Decrement R1	104
	Branch>0 LOOP	105
	Move R0, SUM	106
	⋮	
SUM		
N	n	
NUM1		
NUM2		
⋮		
NUMN		

Program loop

BRANCH Instruction allow the microprocessor to change the sequence of a program either unconditionally or under certain condition.

Branch Instructions.

- Jump Instn.
- Call & Return
- Restart

Unconditional Jump

Enables the programmer to set up continuous loops

Ex: JMP START
↳ Go back to the beginning.

Conditional Jump

Transfers the program sequence to the memory location specified under given condition

OPCODE	~~OPERAND~~ DESCRIPTION
JC	- Jump on carry
JNC	- Jump on no carry
JZ	- Jump on zero
JNZ	- Jump on nonzero
JP	- Jump on plus
JM	- Jump on minus
JPE	- Jump on even parity
JPO	- Jump on odd Parity.

Most commonly used flags are,
N (negative)
Z (zero)
V (overflow)
C (carry)

The flags are affected by certain instructions, but Z & N may be affected by data transfer instn.

Parity flag After an arithmetic operation, the result has even number of 1s, the parity flag is reset to 0, odd number of one's the parity flag will be set to 1.

Scanned by CamScanner

FLYNNS Classification: Its based on no of control units & processor available

SISD Computer System
SIMD
MISD
MIMD

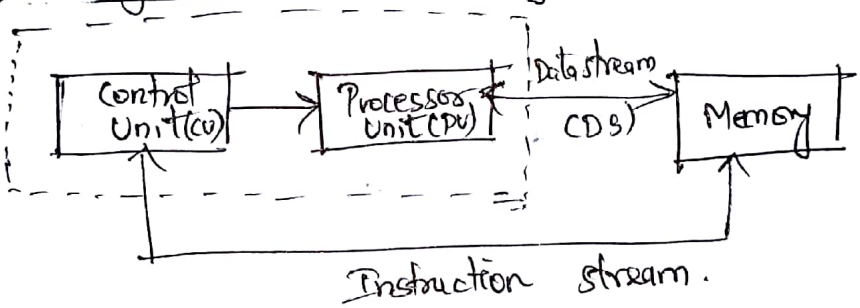
(6)

Flynn's Taxonomy: The basic computer may be classified as follows

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

* Instructions are decoded by the control unit & then ctrl unit send the instns to the processing unit for execution.
* Data stream flows between the processors and memory bi directionally.

SISD - Single Instruction Single Data Stream.



* This architecture has a single control unit producing single stream of instruction.
* These instructions are executed sequentially.
* They may overlap in their pipeline execution stages.
* An SISD computer may have more than one functional unit, under the supervision of one control unit.
* Examples of SISD architecture are the traditional uniprocessor machines like a PC or old mainframes.

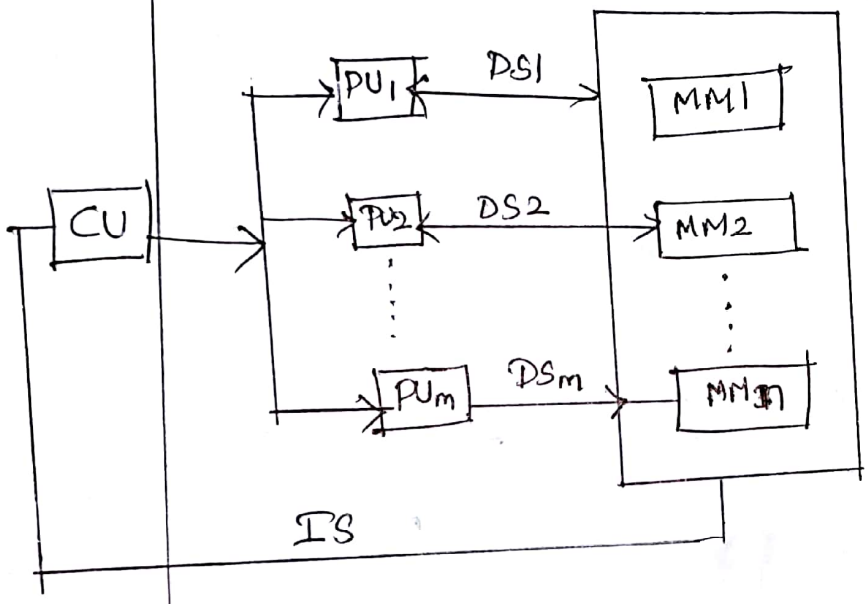
Scanned by CamScanner

SIMD: SINGLE INSTRUCTION MULTIPLE DATA STREAM

(7)

* A computer which exploits multiple data streams against a single instruction stream to perform operations which may be naturally parallelized.
* The single machine instruction controls the simult. execution of a processing elements.
* Each processing has its own associated data memory.
* This facilitates each instruction to be executed on diff. sets of data by different processors.

Eg: Array Processor.

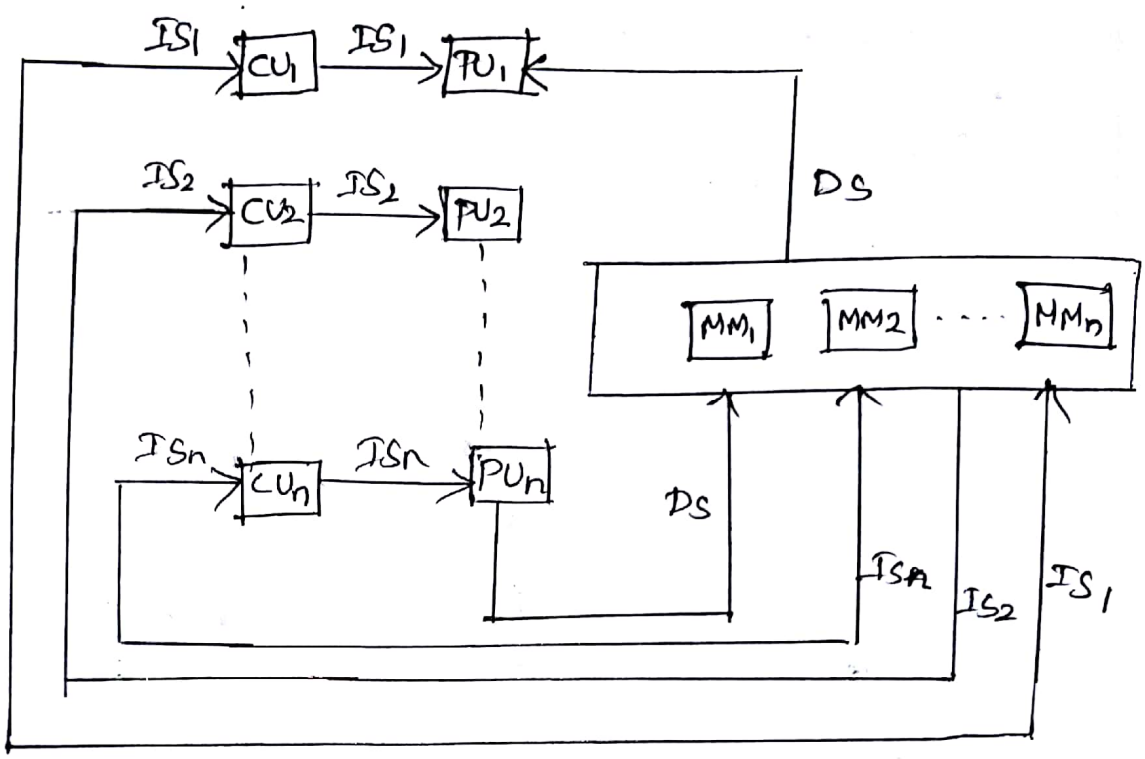


Multiple Instruction Single Data Stream (MISD).

* In this architecture, a single processor executes a single data stream of data using multiple instruction
* Multiple instructions operate on a single data stream
* Uncommon architecture which is used for fault tolerance
* Hetrogeneous system operate on the same data stream and must agree on the result.

Eg: Space Shuttle flight control computer

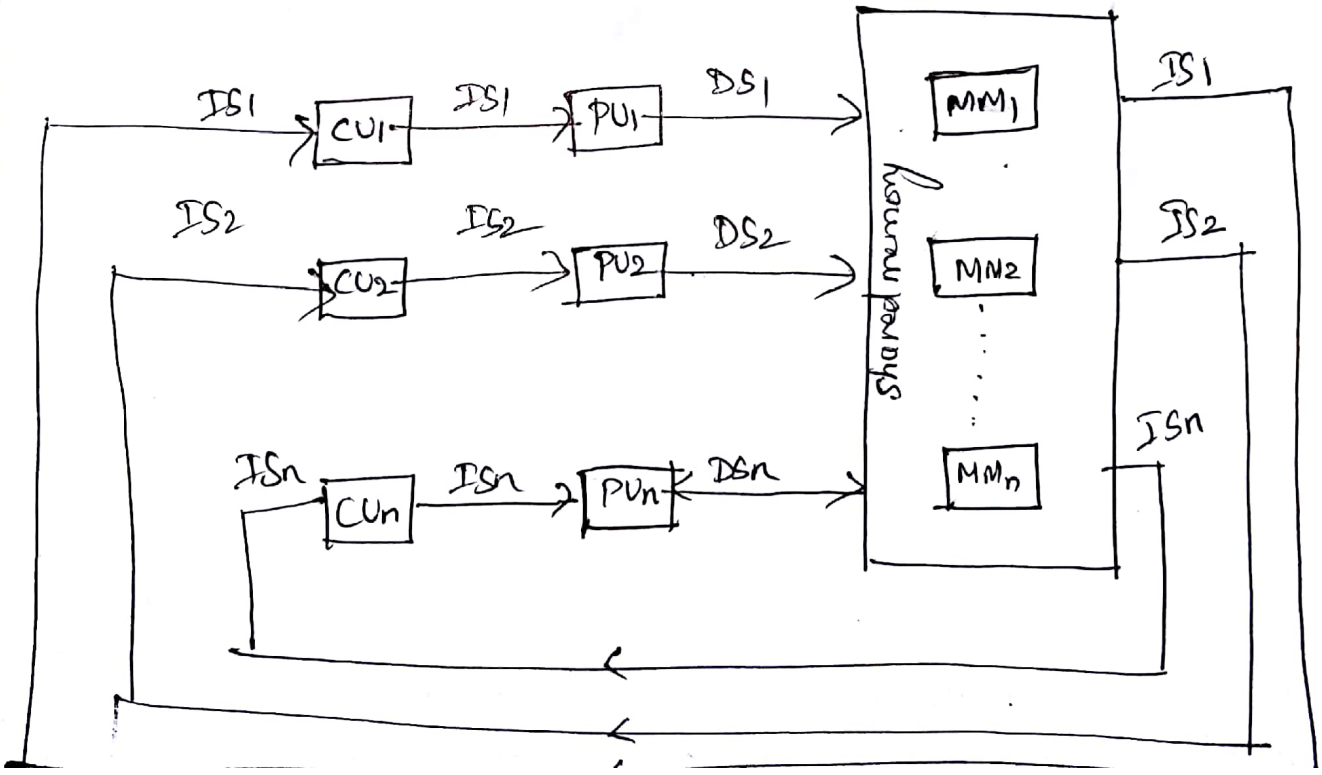
Scanned by CamScanner



4) <u>Multiple Instruction Multiple Data Stream (MIMD)</u>

It has set of processors, each executing a diff. stream of data using diff. set of instruction sequences.

Eg: Symmetric Multiprocessors.



Scanned by CamScanner

IV. PIPELINING

- Its an implementation technique where multiple Instructions are overlapped in execution. Its used to make fast CPUs.
- In a computer pipeline, each step in the pipeline completes a part of an instruction, different steps are completing different parts of instn. in parallel. Each of these step is called pipe stage or pipe segment.
- The time required between moving an instruction one step down the pipeline is a processor cycle.
- If the starting point is a processor that takes 1 clock cycle per instn, then pipelining decreases the clock cycle time
- Once a segement completes an operations, it passes the result to the next segment in the pipeline and fetches the next operations from the preceding segment.

The five stages of pipeline are

Instruction fetch cycle (IF)

IF is responsible for obtaining the requested instruction from memory. The instruction and the program counter are stored in the pipeline register as temporary storage so that it may be used in the next stage at the start of the next cycle.

Instruction decode (ID)

This stage is responsible for decoding the instruction and sending out the various control lines to the parts of the processor. The instruction is sent to the control unit where it is decoded and the registers are fetched from the register file.

Scanned by CamScanner

Execution (EX)

The execution stage is where any calculations are performed. The main component in this stage is the ALU. The ALU is made up of arithmetic, Logic & capabilities.

Memory and I/O (MEM)

The memory and I/O (MEM) stage is responsible for storing and loading values to and from memory. Its also responsible for input or output from processor. If the current instruction is not in this stage, then result from ALU is passed through to the write back stage.

Write back cycle (WB)

Write the result into the register file, whether it comes from the memory system (for load) or from the ALU (for an ALU operation)

Operating timings:

Estimated timings for each of the stages.

IF - 2ns, ID - 1ns, EX - 2ns, MEM - 2ns, WB - 1ns.

Pipelining segments shown in the below diagram.

	Clock number									
Instn. number	1	2	3	4	5	6	7	8	9	
Instruction i	IF	ID	EX	MEM	WB					
Instruction i+1		IF	ID	EX	MEM	WB				
Instruction i+2			IF	ID	EX	MEM	WB			
Instruction i+3				IF	ID	EX	MEM	WB		
Instruction i+4					IF	ID	EX	MEM	WB	

Scanned by CamScanner

Combined Multiply and add operations

$A_i * B_i + C_i$ for $i = 1, 2, 3 \ldots\ldots 7$

1st clock pulse : $R_1 \leftarrow A_i$, $R_2 \leftarrow B_i$ Input A_i & B_i

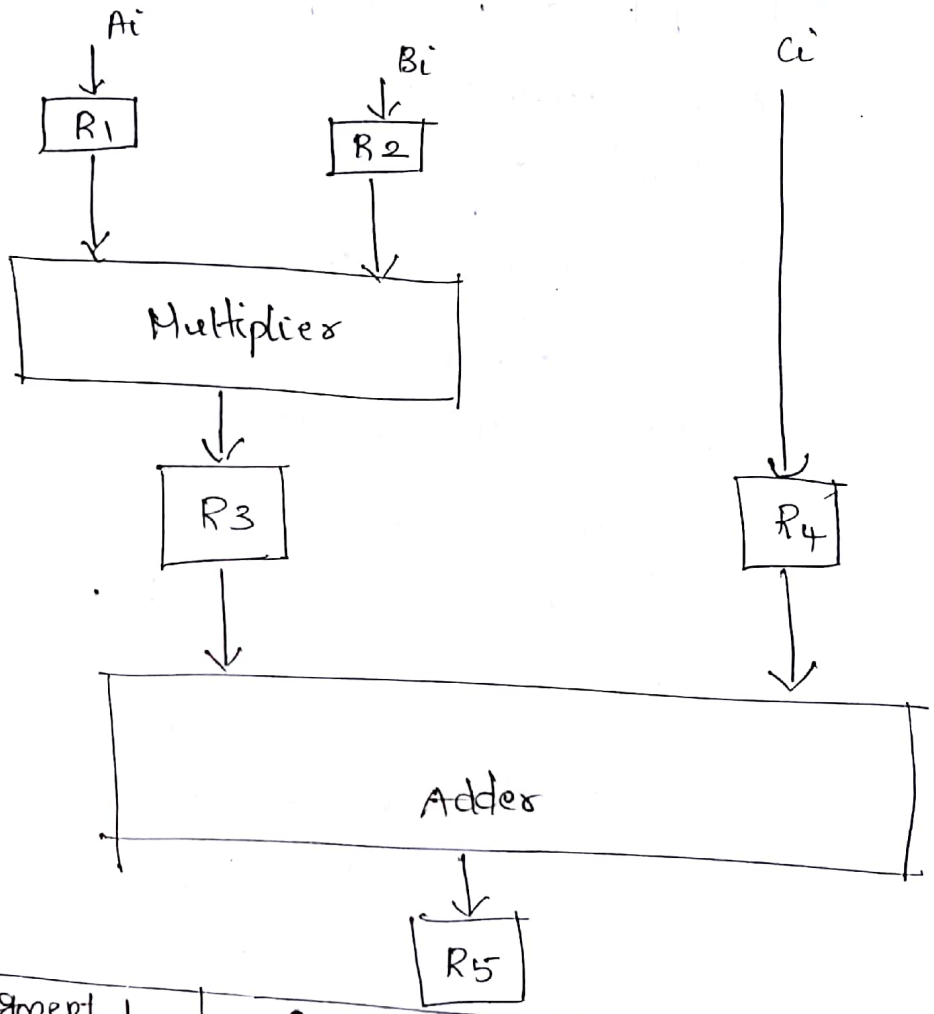
2nd clock pulse : $R_3 \leftarrow R_1 * R_2$, $R_4 \leftarrow C_i$ Multiply & input C_i

3rd clock pulse : $R_5 \leftarrow R_3 + R_4$

Segment 1 :

Segment 2 :

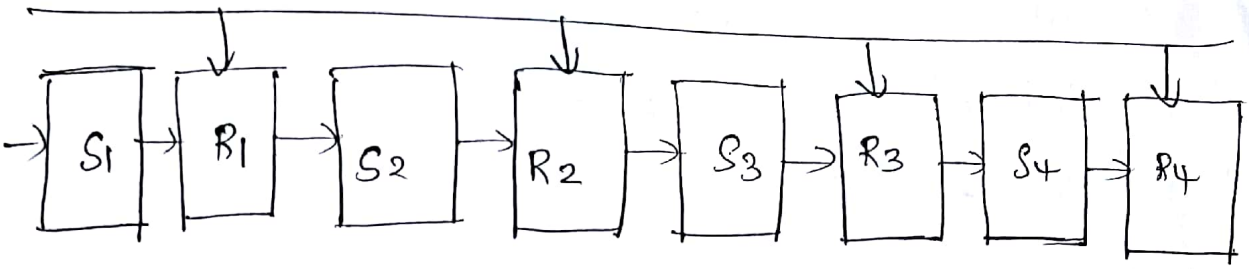
Segment 3 :



Clockpulse No.	Segment 1		Segment 2		Segment 3
	R_1	R_2	R_3	R_4	R_5
1.	A_1	B_1	—	—	—
2.	A_2	B_2	$A_1 * B_1$	C_1	—
3.	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4.	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5.	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6.	A_6	B_6	$A_5 * B_5$	C_5	$A_5 * B_5 + C_4$
7.	A_7	B_7	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
8.	—	—	$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$
9.	—	—	—		$A_7 * B_7 + C_7$

Scanned by CamScanner

Four Segment pipeline



* The operands pass through all four segments in a fixed sequence.
* Each segment consists of a combinational circuit S_i that performs a sub operation over the data stream flowing through the pipe.
* Segments are separated by the Registers R_i that holds the intermediate results between the stages.

Classification of pipeline processors

1. Unifunction Vs Multifunction pipelines
2. Static Vs dynamic pipelines
3. Scalar Vs Vector pipelines.

Advantages:

* More efficient use of processor
* Quicker time of execution of large number of instructions.

Disadvantages:

* Pipelining involves adding hardware to the chip.
* Inability to continuously run the pipeline at full speed because of pipeline hazards which disrupt the smooth execution of the pipeline

Scanned by CamScanner

ARITHMETIC PIPELINE

* Arithmetic pipeline units are usually found in high speed computers.

* They are used to implement floating point operations, multiplication of fixed point numbers and similar computations encountered in scientific problems.

* A pipeline multiplier is essentially an array multiplier with special adders.

* Floating point operations are easily decomposed into sub operations, as in the foll. example of a pipeline unit for floating point addition & subtraction.

Floating point adder pipeline example

Add/subtract two normalized fp binary numbers

$X = A \times 2^a = 0.9504 \times 10^3$

$Y = B \times 2^b = 0.8200 \times 10^2$

4 segments sub operations:

1) Compare exponents by subtraction

$3 - 2 = 1$

$x = 0.9504 \times 10^3$

$y = 0.8200 \times 10^2 \times 10^1$

2) Align mantissas

$x = 0.9504 \times 10^3$

$y = 0.08200 \times 10^3$

3) Add mantissas

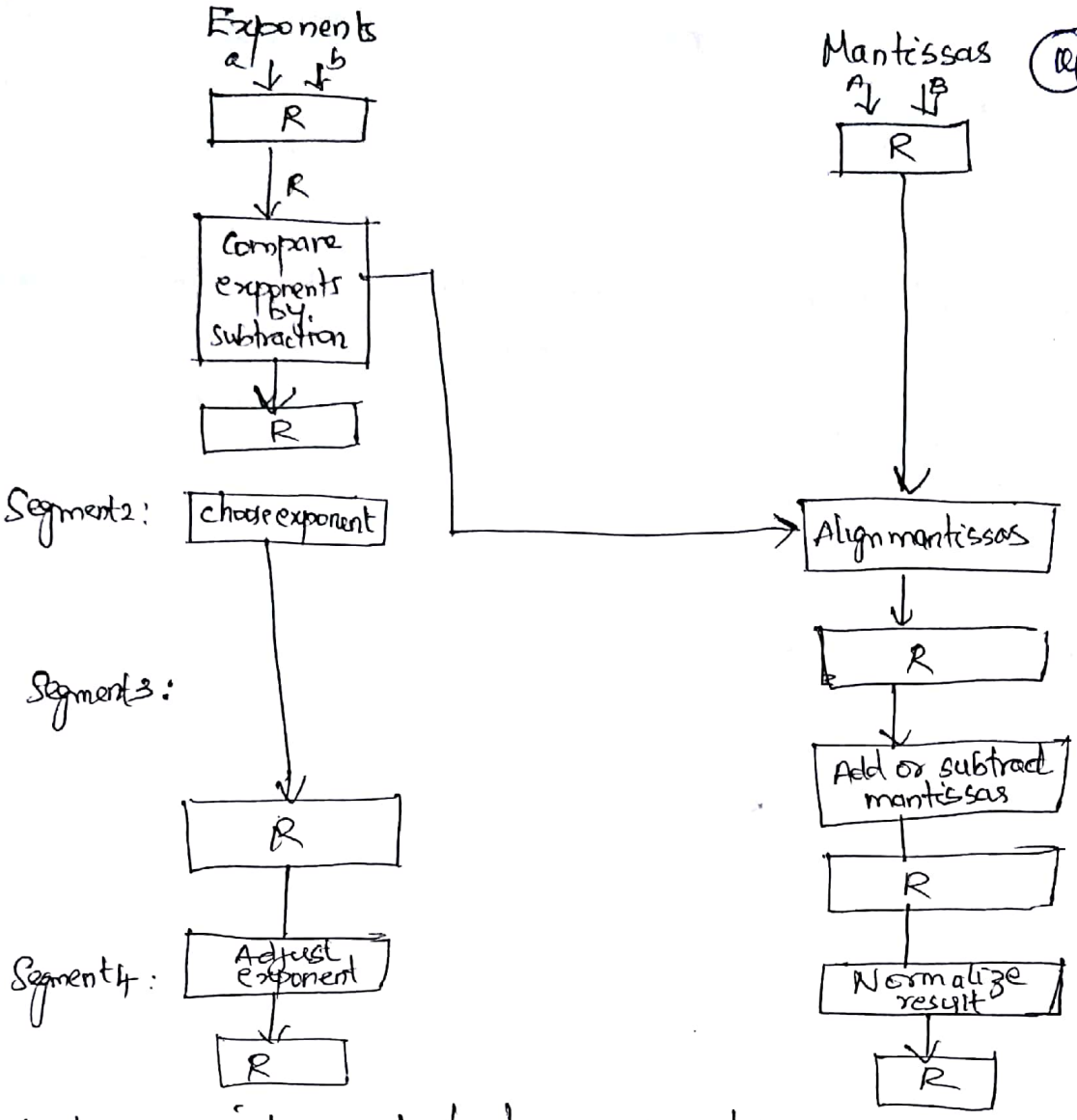
$Z = 1.0324 \times 10^3$

4) Normalize result

$z = 0.1324 \times 10^4$

Eg2:

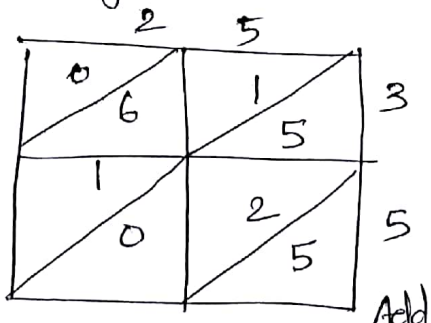
Scanned by CamScanner



Floating point multiplication pipeline

* A pipelined multiplier based on the digit product can be designed using digit product generation logic and the digit adders.

Eg: 25 * 35 = 875



Add numbers along diagonals.

$$
\begin{array}{r}
+\quad 0\ \ 1\ \ 5 \\
6\ \ 2\ \ 5 \\
1\ \ 0 \\
\hline
0\ \ 8\ \ 7\ \ 5
\end{array}
\Rightarrow \text{result}
$$

Scanned by CamScanner

Floating point multiplication pipeline

It involves following three major steps:

1. Multiplication of fractions.
2. Addition of exponents.
3. Normalization of the result.

$a^2 \times b^2$

a^2

Floating point division pipeline

Division operation appears less frequently in computer programs compared to addition, subtraction and multiplication and hence separate pipeline unit for the division is rarely implemented. It is common to schedule the division using adder and multiplier pipelines.

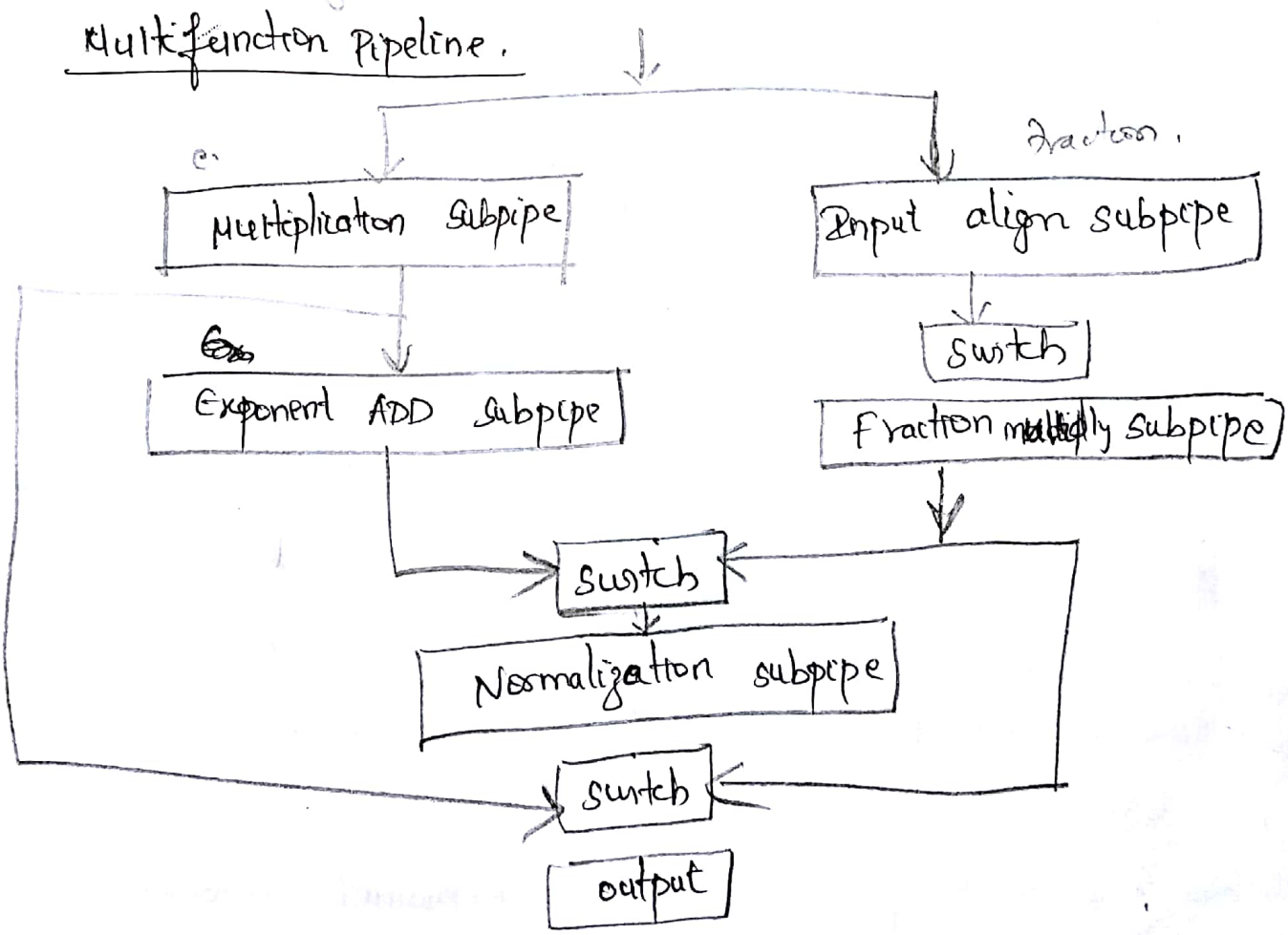
Multiplication

$x = 3.35 \times 10^2$

$y = 4.56 \times 10^3$

$\Rightarrow z = (3.35 \times 4.56)10^5 \Rightarrow z = 15 \ldots$ 10^5

Multifunction Pipeline.



Scanned by CamScanner

INSTRUCTION PIPELINING

* Pipelining can occur in the data stream + the instn. stre

An instruction cycle can be broken into the following stages.

- Fetch the Instruction from memory. (FI) MUL A,B
- Decode the instruction (DI) MOV R1,A
- Calculate the effective address (EA) MOV R2,B.
- Fetch operands (FO) MUL R1,R2,
- Execute Instruction (EI) MOV R3 → B3
- Store the result STORE B3, C.

EA = R1
MOV R1, X
EA = (A)
value
exe
dp

Each stage can be performed more or less independently from the rest.

* An instruction pipeline reads consecutive instructions from memory while previous instructions are being executed in other segments. This causes the instruction fetch and execute phases to overlap & perform simultaneous operations.

* There are 3 major difficulties (hazards) that cause the instruction pipeline to deviate from its normal operation.

* Any condition that causes the pipeline to stall is called a hazard. Stall is a obstacle in pipelining. A hazard is created whenever there is a dependence between instruction & they are close enough that the overlap caused by pipelining would change the order of access to an operand. Stall is commonly called pipeline bubble.

* When an instruction is stalled, all instruction later in the pipeline after the stalled instruction are also stalled.

* Types of hazards are explained below.

Scanned by CamScanner

PIPELINING HAZARDS

1. Structural hazard.
2. Data hazard.
3. Control hazard

STRUCTURAL HAZARD

* This hazard occurs when a certain resource is requested by more than one instruction at the same time.
* This situation occur due to resource conflicts.
* Example:

A process with single memory bus will generate a conflict whenever a memory reference occurs.

* In the following timing diagram, the load instn. uses memory bus in 4th clockcycle, causing pipeline to stall, so no instruction is initiated in clock cycle 4. But instn 3 has to be initiated at clock cycle 4. So no instruction completes on clock cycle 8, because Instn. 3 does not begin execution in cycle 4.

clock cycle number

Instruction	1	2	3	4	5	6	7	8	9	10	11	12
LOAD instn	IF	ID	EX	MEM	WB							
Instn i+1		IF	ID	EX	MEB	WB						
Instn i+2			IF	ID	EX	MEM	WB					
i+3				STALL	IF	ID	EX	MEM	WB			
i+4						IF	ID	EX	MEM	WB		
i+5							IF	ID	EX	MEM	WB	
i+6								IF	ID	EX	MEM	WB

Scanned by CamScanner

* If some combination of instructions cannot be accomodated because of resource conflicts, the process is said to have structural hazard.

* When a sequence of instruction encounters this hazards, the pipeline will stall one of the instruction until the required unit is available. Such stalls will increase CPI from its usual ideal value of 1. If the instn. has to be ~~extended~~ completed within 8th ~~cycle~~ clock cycle, will be extended upto 9th clock cycle. (See the above diagram).

* In the above example, when an instruction contains a data memory reference, it will conflict with the instruction reference for later instruction. To resolve this hazard, we stall the pipeline for 1 clock cycle when the data memory access occurs. A stall is called bubble, because it floats through the pipeline taking space but carrying no useful work.

* As an alternative to this structural hazard, cache memory can be used, it can be splitted into data cache and instruction cache.

* But usage of cache is costly. The processors that support both an instruction and a data cache access every cycle require twice as much total memory bandwidth and often having higher bandwidth.

* The average instruction time can be calculated as,

Average instruction time = CPI x clock cycle time.

Scanned by CamScanner

Data hazard:

* This hazard occurs when an instruction depends on the result of a previous instruction, but the result is not yet available.

* It also occurs when the pipeline changes the order of read/write accesses to operands, so that the order differs from the order seen by sequentially executing instructions on an unpipelined processor.

Eg:

ADD R1, R2, R3
SUB R4, R3, R5
AND R6, R3, R7
OR R8, R3, R9
XOR R10, R3, R11

* All the instruction after the ADD, use the result of the ADD instruction. The ADD instruction writes the value of R3 in the WB pipe stage, but SUB instruction reads the value during its ID stage. This problem is called data hazard. The problem is DSUB instruction will read the wrong value and try to use it.

* The AND instruction is also affected by this hazard. The write of R3 does not completed until the end of clock cycle 5. So the AND instruction that reads the registers during clock cycle 4 will receive wrong results.

* The XOR operation works properly because its register read occurs in clock cycle 6, after a register write. The OR instruction also operates without affected by hazard

Scanned by CamScanner

Minimizing data hazards by Forwarding:

- The above problem (data hazard) can be solved by a simple hardware technique called forwarding (bypassing & sometimes it is called short-circuiting)
  - * Forwarding can be done as follows.
    - The ALU result from both the Ex/MEM and MEM/WB pipeline registers is always given back to the ALU units.
    - If the forwarding hardware detects that the previous ALU operation has written the register corresponding to a source for the current ALU operation, circuit selects the forwarded result as the ALU input.
  - * Example diagram shown below (forwarding).

clock number

Instn	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10
ADD R1,R2,R3	IF	ID	EX	MEM	WB					
SUB R4,R1,R5		IF	ID	EX	MEM	WB				
AND R6,R1,R7			IF	ID	EX	MEM	WB			
OR R8,R1,R9				IF	ID	EX	MEM	WB		
XOR R10,R3,R11					IF	ID	EX	MEM	WB	

Reg.

Scanned by CamScanner

CONTROL HAZARDS.

* Control hazard occurs while executing branch Instructions.
* An attempt to make a decision before condition is evaluated

Eg: Washing uniforms and need to get proper detergent level; need to see after dryer before next load in

Branch Instn For eg. Ref. pg. no 4.

Move N, R1	100	
Move # NUM1, R2	101	1 MOVE
clear R0	102	2 MOVE
Add (R2), R0	loop 103	3 Add
Increment R2		4 INR
Decrement R1	104	5 DCR
Branch>0 loop	105	6 Branch>0
Move R0, SUM	106	7 MoveR0, Sum.
⋮		
SUM 200		
N 201		
NUM1 202		
⋮		
Num n 501		

BTA = Address of loop.

* Control hazard ~~occurs~~ caused by irregular execution path, branch taken or untaken
* If the branch is to be taken, the PC is not normally change until the end of the MEM stage.
* The simplest way to deal with control hazard is to stall pipe

Branch Instruction	IF	ID	Ex	MEM	WB					
Branch Successor		IF	Stall	Stall	IF	ID	ex	MEM	WB	
Branch Successor+1						IF	ID	Ex	MEM	WB
Branch Successor+2							IF	ID	Ex	ME

* 2 clock cycle wasted per instruction.
* Compute the address of the branch target BTA = earlier in the pipeli
* This can be solved by the technique branch prediction, branch instn. predicted earlier to avoid the irregular seq.

Scanned by CamScanner

Initially the prediction not taken, the pipeline sequence with branch shown below.

Taken branch Instn.	IF	ID	Ex	MEM	WB									
Instn. i+1		IF	idle	idle	idle	idle								
Branch target			IF	ID	Ex	MEM	WB							
Branch target+1				IF	ID	Ex	MEM	WB						
Branch target+2					IF	ID	Ex	MEM	WB					

* The another scheme is delayed branch, the execution cycle with a branch delay of one is

branch instruction
sequential successor
branch target if taken.

* The sequential successor is the branch delay slot. This instn. is executed whether or not branch is taken. Almost all the processors with delayed branch have a single instruction delay.

Branch Instn	IF	ID	Ex	MEM	WB									
Branch delay instn (i+1)		IF	ID	Ex	MEM	WB								
Branch target			IF	ID	Ex	MEM	WB							
Branch target+1				IF	ID	Ex	MEM	WB						
Branch target+2					IF	ID	Ex	MEM	WB					

* The above figure shows the behaviour of the delayed branch. If the branch is not taken, execution continues with the instn. after the delay instn. If the branch is taken, execution continues at the branch target.

Notes: • The simplest scheme to handle branches is to freeze or flush the pipeline, holding or deleting any instructions after the branch until the branch destination is known.

• In the simple five stage pipeline, the predicted (taken & untaken) scheme is implemented by continuing to fetch instns as if the branch were a normal instn. The pipeline looks as if nothing happens. If the branch is taken, we need to turn the fetched instn. into a no-op and restart the fetch at the target address.

Scanned by CamScanner

CISC Vs RISC

CISC :

- Complex Instruction set computer.
- A computer with a large number of instructions.
- Goal of CISC architecture is attempt to provide a single machine instruction for each statement that is written in high level language.
- In CISC, hardware is always faster than software.

Characteristics

- A large number of instructions from 100 to 250 instructions.
- Some instructions that perform specialized tasks and are used infrequently.
- Large variety of addressing modes, 5 to 20 modes.
- variable length instruction formats.
- Instructions that manipulate operands in memory

CISC drawbacks

CPU complexity - Because instruction set is long.

- system size & cost - Because of CPU complexity,

clock speed - Due to increased circuits

Reliability - Heavy hardware leads to frequent failure

Maintainability - Troubleshooting & detecting is big task because of large number of huge task.

Scanned by CamScanner

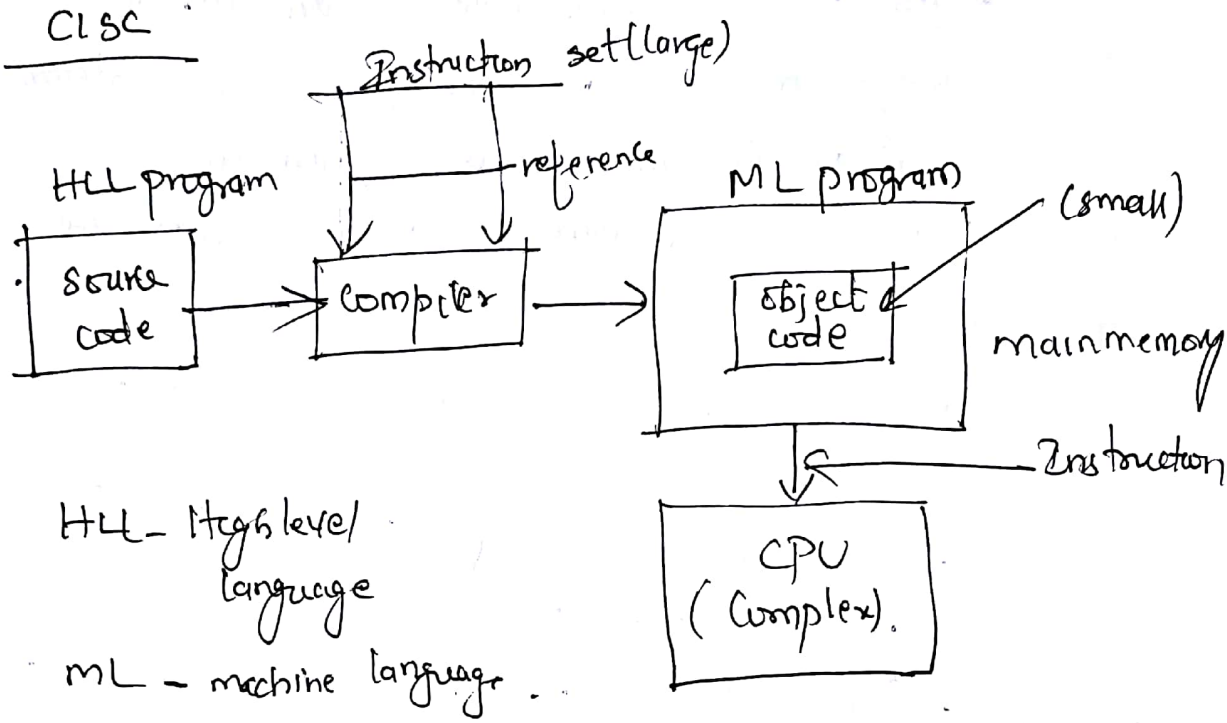
Initially the prediction not taken

Example of ~~CISC~~ CISC Arch: eg: intel pentium microprocessor

Digital equipment corporation VAX Computer

IBM 370 Computer, Data general nova, Motorola.

CISC

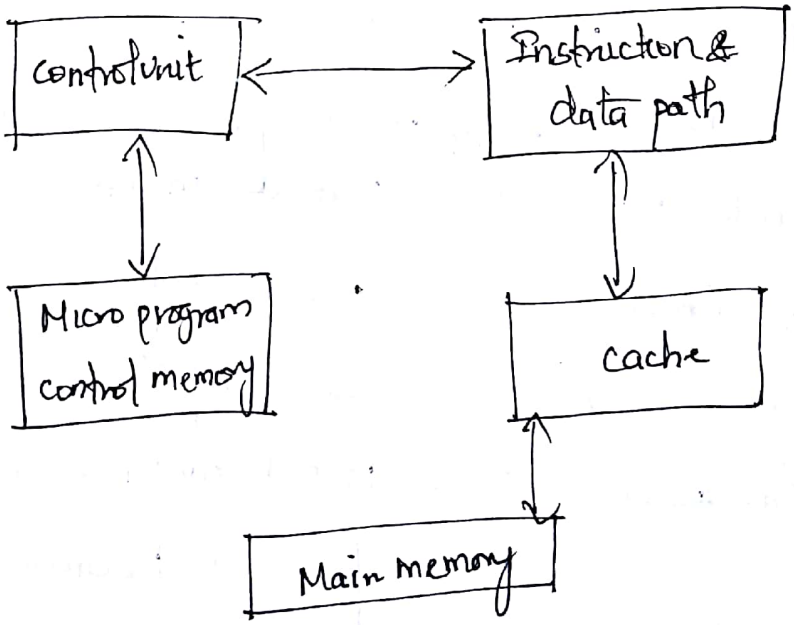


HLL - High level language

ML - machine language.

* Its a set of complex instructs that is capable of doing multi step operations (load/memory/CPU, arithmetic store)

CISC Architecture



Scanned by CamScanner

RISC

Reduced Instruction set Computer.

- This architecture involves an attempt to reduce execution time by simplifying the instruction set of the computer.
- It has fewer, simpler & faster instructions, so more instns. are needed to accomplish a task.
- RISC processors only use simple instructions that can be executed within one clock cycle.
- The MUL command divided into three separate commands.

```
LOAD A,2
LOAD B,3
MUL A,B
STORE B.
```

- RISC require less transistors of hardware space than the complex instructions
- More registers will be free, pipelining is possible.

Eg Apple's power macintosh line featured RISC based chips & windows NT.

Sample RISC CPU's

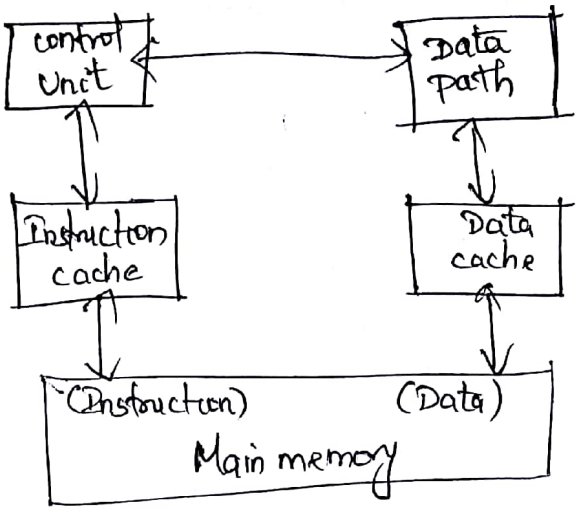
Intel 860
Power PC
MIPS series
IBM RS/6000.
SUN's SPARC family.

Features:

- It can operate upto higher speed (performs more millions of instructions per second MIPS)
- It is performed by overlapping the execution of several instructions in a pipeline fashion.

Scanned by CamScanner

Architecture.



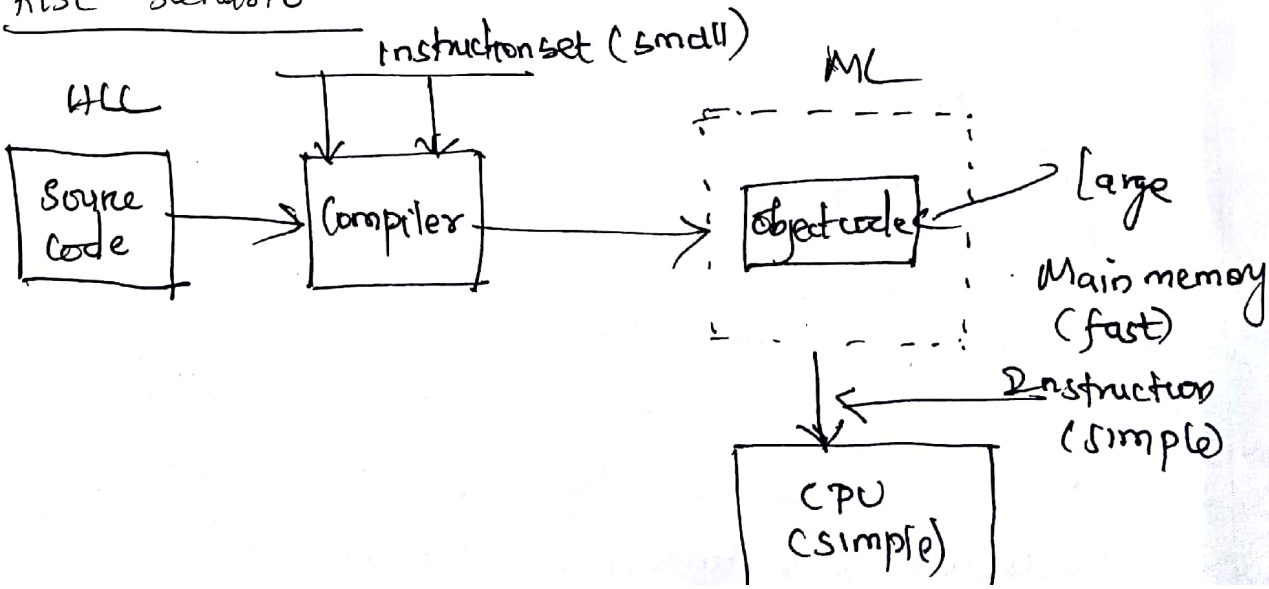
Advantages:

- It allows freedom of using space on microprocessors because of its simplicity.
- Many RISC processors use the registers for passing arguments and holding the local variables.
- The speed of the operation can be maximized and the execution time can be minimized

Disadvantages

- The performance of the RISC processor depends on the programmer or compiler, as compiler can change RISC to CISC
- It occupies more size.
- It has large memory caches on chip itself.

RISC scenario

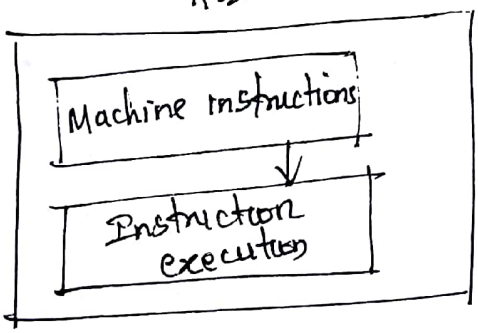


Scanned by CamScanner

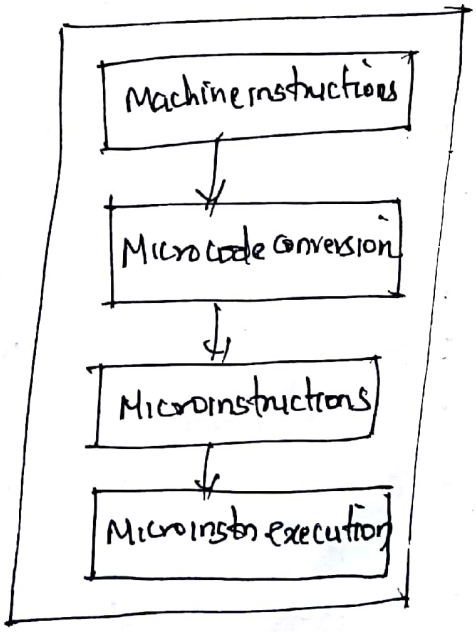
CISC & RISC Comparision.

CISC	RISC
1. It is prominent on Hardware	1. It is prominent on software.
2. It has high cycles per second	2. It has low cycles per second.
3. It has transistors used for storing instructions which are complex.	3. More transistors are used for storing memory.
4. LOAD & STORE memory-to-memory is included in the instruction.	4. LOAD & STORE registers are independent.
5. It has multi-clock cycle.	5. It uses multi clock cycle.
6. It access memory directly	6. It requires registers.
7. Complexity in a interpreter	7. Complexity in a compiler
8. Variety of addressing mode used	8. Few addressing mode used.
9. It has variable length format instruction set.	9. It has fixed length format instruction set.
10. CISC machines doesn't use pipelining	10. RISC uses pipelining.

RISC.



CISC



Scanned by CamScanner

SUPERSCALAR ARCHITECTURE

- It allows the execution of 2 or more successive instructions simultaneously in different pipelines.
- As a result, more than one instruction gets completed in every clock cycle.
- A superscalar processor may use RISC or CISC architecture

Requirements of superscalar processors

- The instruction fetching unit fetch more than one instruction at a time.
- Instruction decoding logic should decide whether the instructions are independent & executable simultaneously.
- There are sufficient number of execution units so that several instructions can be processed simutaneously.
- Cycle time for each pipeline stage should match the cycle time for fetching and decoding the instruction, so that pipeline works at optimum efficiency.
- In general it takes 8 clock cycles to complete an instruction
- Fetch, decode & store, each take 1 clock cycle, but execute stage takes two clock cycles. So the other stage should also wait for one more clock cycle.
- Totally it takes 8 clock cycles to complete the first instruction and then for every 2 clock cycles, one instruction is completed.
- The foll. figure contains two separate execution units
- Each execution unit takes two clock cycles.
- Odd instructions enters execution unit 1 & even instruction enters execution unit 2.
- It has 5 clock cycles to complete the first instructio and for every clock cycle, one instruction is completed.

Scanned by CamScanner

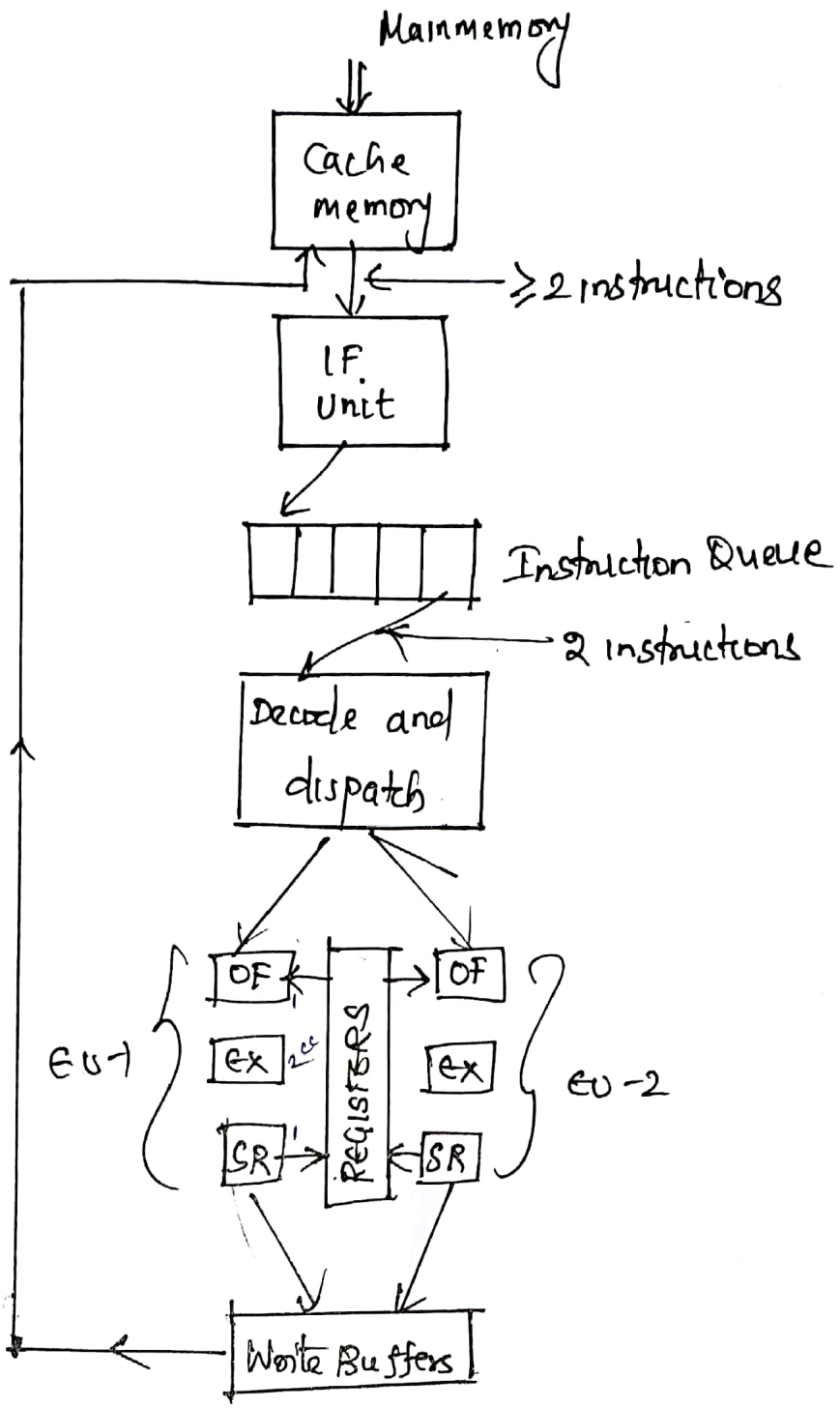


Figure 1.

OF — operand fetch

Ex — Execution

SR — store

EU — Execution Unit

IF — Instruction Fetch.

Superscalar architecture diagram

Scanned by CamScanner

Features of superscalar Architecture.

Branch Prediction

IF I=J then
m=m+1
else
m=m-1, "IF-THEN" is a branch

The processor uses the branch prediction to start only one of them based on previous Information. It uses special small cache called Branch Target Buffer (BTB)

Out of order Execution

- Super Scalar processors can complete the processing of instructions in the non program order.
- Before the completion of earlier instructions, later one can also be executed.

Register Renaming

- Eliminates the conflict between different execution units trying to use the same register as required by the instruction.
- Renaming a temporary requirement, once the execution is complete, the result is put in the originally intended register and the renaming is removed.

Write buffers:

Holds the result of the execution unit until they can be stored in the registers or memory locations.

Syllabus UNIT-II CPU ARCHITECTURE

Instruction sets of different machines - Parallel Processing - Flynn's classification - Pipelining - Arithmetic pipeline - Instruction pipeline - pipelining hazards - CISC V/s RISC - Superscalar architecture.

C. GEETHA
AP/CSE

Scanned by CamScanner

UNIT-III

MEMORY SYSTEM

Basic concepts - Semiconductor RAMs - ROMs - speed - size and cost - cache memory - Mapping Techniques - Virtual memory - Evaluating Memory Performance - Secondary Storage - Multiprocessor.

I. BASIC CONCEPTS

- Programs & data are stored in main memory.
- The execution speed of the program is dependent on the speed of the data transferred between memory & CPU.
- So the memory must be fast, large & inexpensive.
- But increase in speed & size leads to higher cost.
- So, the structure of the memory must be improved in such a way that speed & size increased without affecting cost.
  - → The speed of the memory achieved by caches
  - → Virtual memory, increases the size of the memory
  - → Secondary storage device used to produce large capaci

Main memory

- Maximum size of the memory determined by addressing scher
- n bit computer generates 2^n memory locations.

  Eg: 16 bit computer has $2^{16} = 64K$ memory location.

  32 bit, $2^{32} = 4GB$; 40 bits, $2^{40} = 1TB$.

- Data transfer between the memory & the CPU takes place through the use of 2 CPU registers, MAR & MDR.
- In the following diagram, MAR with k bits long, MDR with n bits long & memory unit may contain upto 2k addressable location.

Scanned by CamScanner

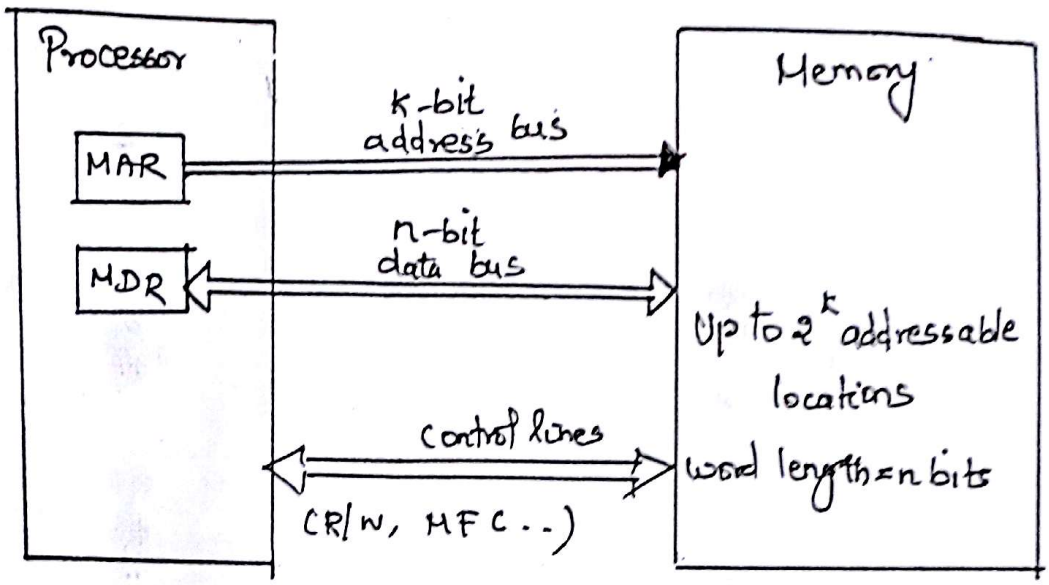


Fig: Connection of the memory to the processor

- During a memory cycle, n bits of data are transferred between the memory & the CPU. This transfer takes place processor bus which has k address lines & n data lines. The bus also includes the control lines read, write and Memory Function Completed (MFC) for coordinating data transfer.
- The processor reads the data from the memory by loading the address of the required memory location & set the R/W line to 1. The memory responds by placing the data from the addressed location onto the data lines & confirms this by asserting the MFC signal.
- After receiving the MFC signal, the processor loads the data on the data lines into the MDR register. The processor writes the data into a memory location by loading the address of the location into MAR & loading the data into MDR. A write operation set the R/W line to 0.

A memory unit is called random access memory (RAM) if any location can be accessed for read or write operation that is independent of the location address.

Scanned by CamScanner

II. SEMICONDUCTOR RAMs

- Its available in a wide range of speed.
- Its clock cycle time ranges from 100 ns to ≤ 10ns.

Internal organization of memory chips

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

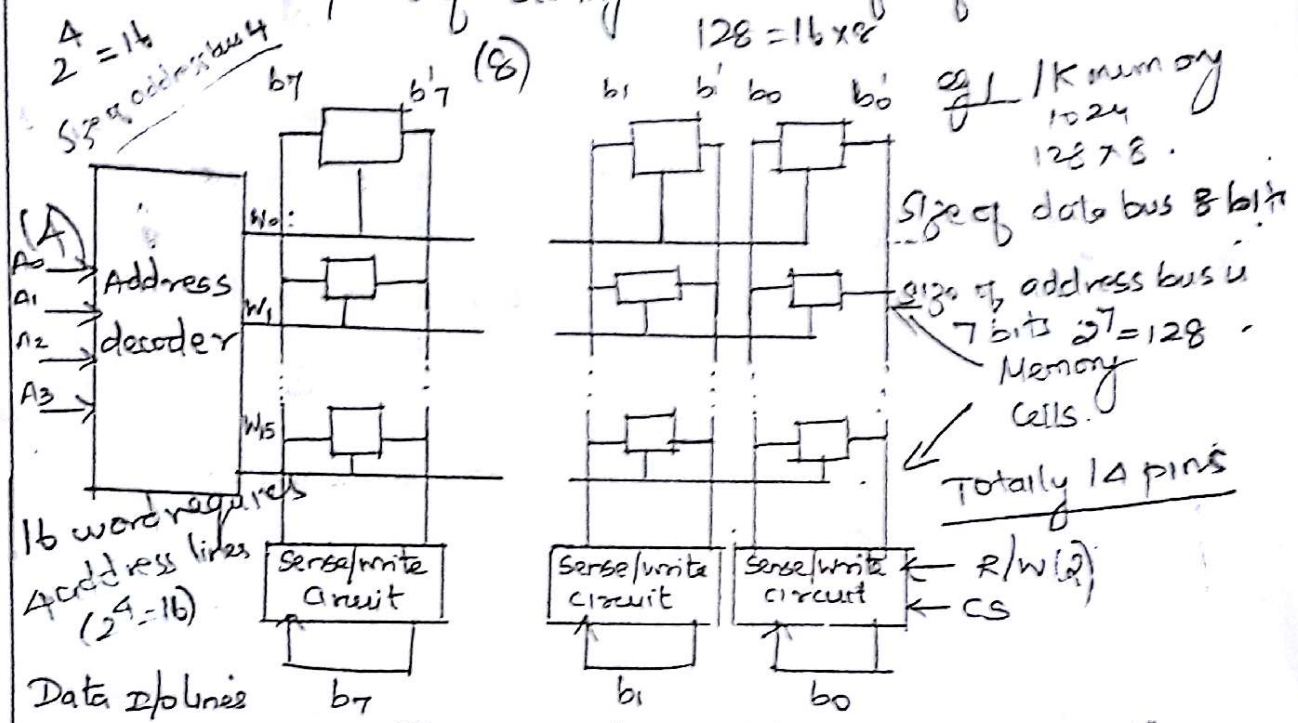


Fig: Organization of bit cells in memory chip.

- Each row of cells has a memory word, and all cells of a row are connected to a common line called word line, which is driven by address decoder on the chip.
- The cells in each column are connected to a sense/write circuit by 2 bit lines. The sense/write circuits are connected to the data I/O lines of the chip.
- During a Read operation, these circuits sense or read the information stored in the cells selected by a word line and transmit this information to the output data lines.
- During a write operation, the sense/write circuit receive input information and store it in the cells of the selected word.

Scanned by CamScanner

STATIC MEMORIES (SRAM)

- Memories that consist of circuits capable of retaining their state as long as power is applied are known as static memories.
- The following figure shows how SRAM cell is implemented.
  - Two inverters are cross connected to form a latch.
  - The latch is connected to two bit lines by transistors T_1 & T_2.
  - These transistors acts as switches that can be opened or closed under control of the word line.
  - When the word line is at ground level, the transistors are turned off and the latch retain its state.

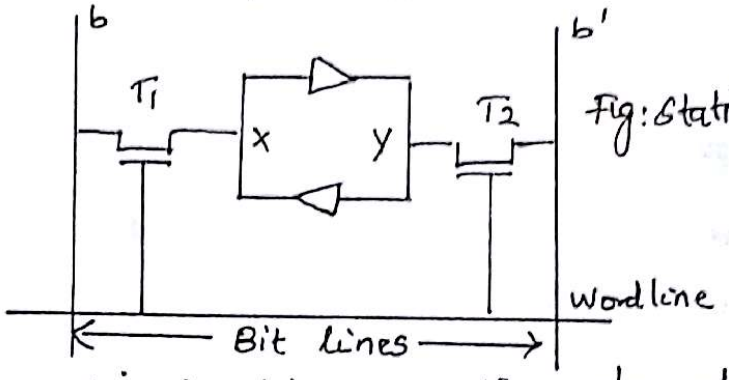


Fig: Static RAM cell.

If the cell is in state 1 then the value at point x is 1. If the cell is in state 0 then the value at point y is 0.

Read operation:

In order to read the state of the SRAM cell, the word line is activated to close switches T_1 & T_2.

- If the cell is in state 1, then the signal on bit line b is high and the signal on bit line b is low.
- If the cell is in state 0, then the signal on bit line b is low and the signal on bit line b is high.

Write operation

The state of the cell is set by placing the appropriate value on bit line b & its complement on b', and then activating the word line. This forces the cell into the corresponding state. The required signal on the bit lines are generated by the sense/write circuit.

Scanned by CamScanner

Dynamic RAM (DRAM)

- Static RAMs are fast, but they come at high cost because their cells require several transistors. Less expensive RAMs can be implemented if simpler cells are used. Such cells do not retain their state indefinitely, so they are called DRAM.
- Information is stored in a dynamic memory cell in the form of a charge on a capacitor, and this charge can be maintained for only tens of milliseconds.
- Since the cell is required to store information for a much longer time, its contents must be periodically refreshed by restoring the capacitor charge to its full value.

For eg: dynamic memory cell, which has capacitor C & Transistor T,

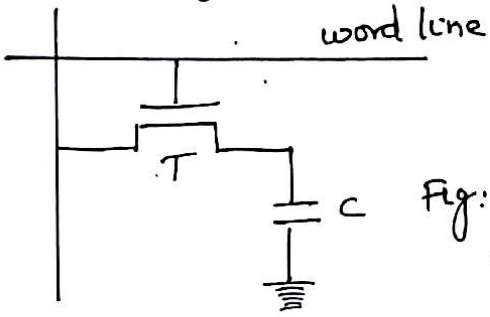


Fig: A single transistor dynamic memory cell

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

After the transistor is turned off, the capacitor begins to discharge. This is caused by the capacitor's own leakage resistance and by the fact that the transistor continues to conduct a tiny amount of current, after its turned off. The information stored in the cell can be retrieved correctly only if it is read before the charge on the capacitor drops threshold value. During a read operation, the transistor in a selected cell is turned on.

Types of DRAM → Refer pg. no. 7

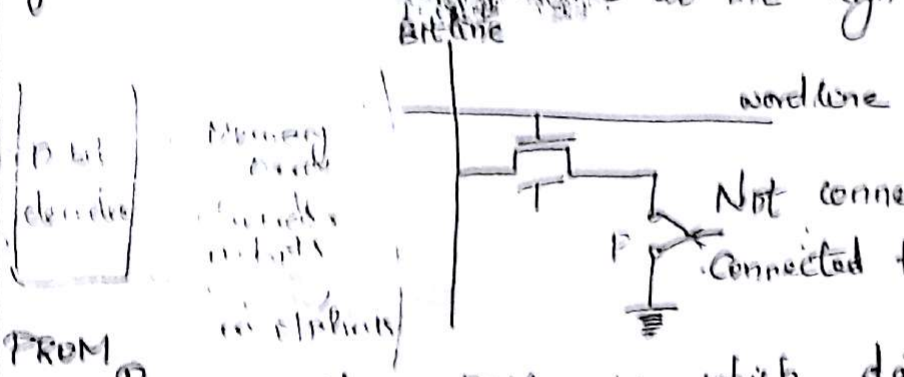
Scanned by CamScanner

III READ ONLY MEMORY (ROM)

It's a non-volatile memory which holds the data even if the power is turned off.

A logic value 0 is stored in the cell if the transistor is connected to ground at point P otherwise 1 is stored. The bit line is connected through a resistor to the power supply.

To read the state of the cell, the word line is activated. Thus the transistor switch is closed & the voltage on the bit line drops to near zero if there is a connection between the transistor and ground. If there is no connection to ground, the bit line remains at the high voltage indicating 1.



PROM: Programmable ROM in which data can be loaded by user. Program is done by inserting a fuse at point P. Before it is programmed, the memory contains all 0's. The user can insert 1s at the required locations by burning out the fuses at these locations using high current pulses.

EPROM: It allows the stored data to be erased & new data to be loaded. Such an erasable programmable ROM is called EPROM. Since EPROMs are capable of retaining stored information for a long time. In this way, memory changes & updates can be easily made. Contents can be erased by UV light & its reprogrammed.

EEPROM: It can be programmed & erased electrically.

It is possible to erase the cell contents selectively.

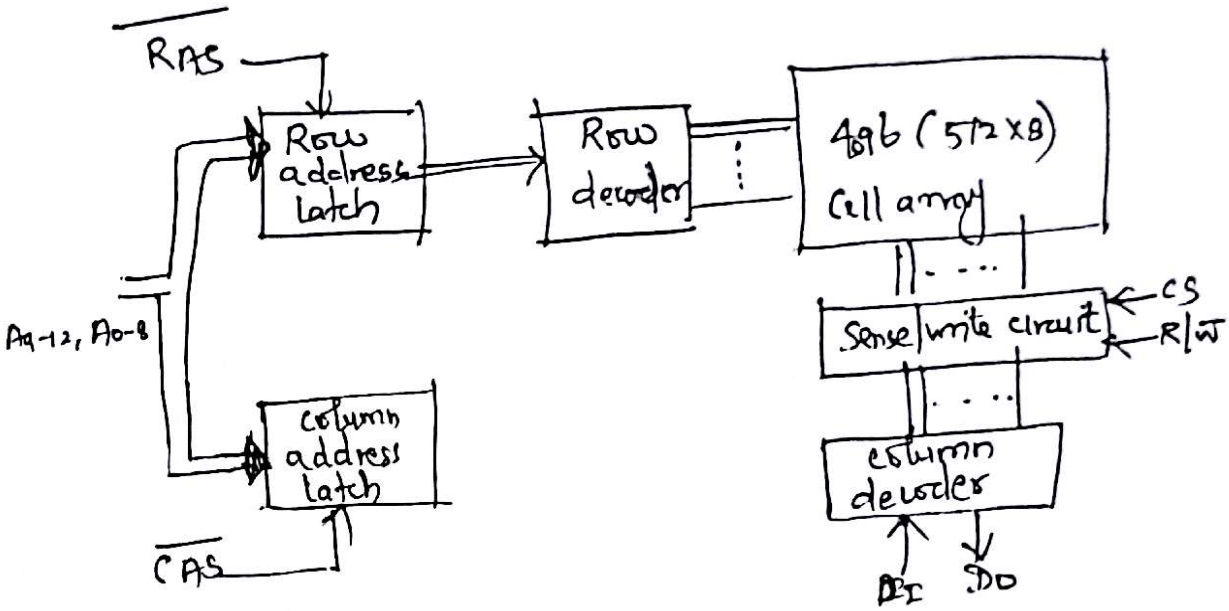
The disadvantage of EEPROM is that different voltages are needed for erasing, writing & reading the stored data.

Scanned by CamScanner

Two types of DRAM : Synchronous & Asynchronous DRAM

Asynchronous DRAM

In DRAM, the timing of the memory device is controlled asynchronously. A specialized memory controller circuit provides the necessary control signal, RAS & CAS that govern the timing. The processor must take into account the delay in the response of the memory, such memories referred as Asynchronous DRAM.



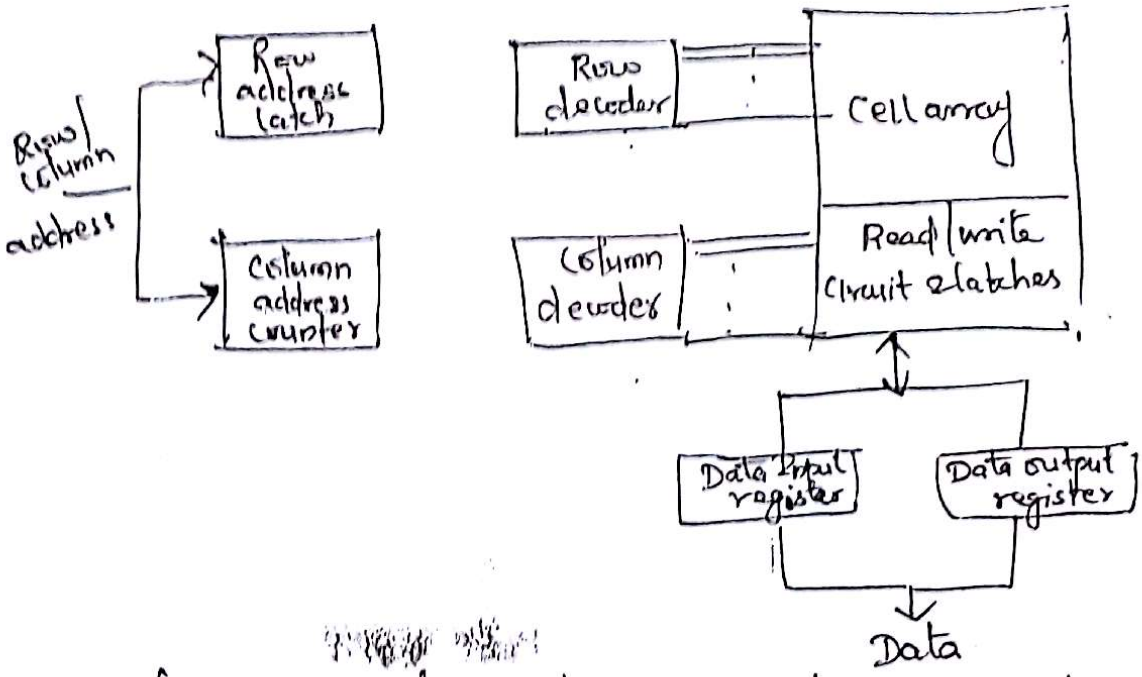
The cell organized in the form of a 4K×4K array. The 4096 cells in each row are divided into 512 groups of 8, So row can store 512 bytes of data, 12 address bits are needed to select a row. Another 9 bits needed to specify a group of 8 bits in the selected row. Thus 21-bit address is needed to specify ~~a group of 8 bits in the selected row~~ byte in memory. The high order 12 bits and the low order 9 bits of the address constitute the row & column addresses of a byte respectively.

Scanned by CamScanner

Synchronous DRAM

More recent developments in memory technology have resulted in DRAM's whose operation is directly synchronized with a clock signal. Such memories known as Synchronous DRAM.

The following figure indicates the structure of an SDRAM. The cell array is same as in asynchronous DRAM. The address and data connections are buffered by means of registers.



SDRAM has several different modes of operation, which can be selected by writing control information into a mode register.

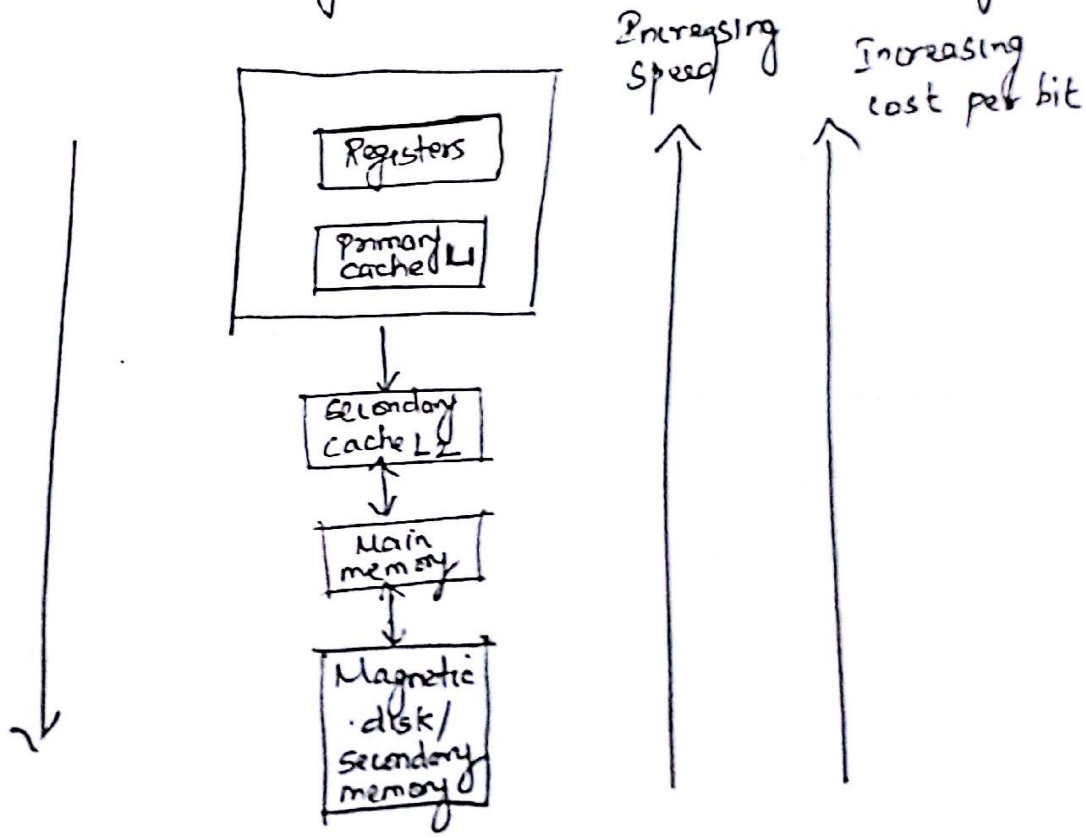
Commercial SDRAMs can be used with clock speeds above 100 MHz. These chips are designed to meet the requirements of commercially available processors that are used in large volume.

Scanned by CamScanner

IV SPEED, SIZE & COST

- A very fast memory can be implemented if SRAM chips are used. These chips are expensive because their basic cells has six transistors. For cost reasons, it is impractical to build a large memory using SRAM chips.
- Thus alternative is to use Dynamic RAM chips, which has simpler cells & inexpensive, but its slower.
- Solution is provided by using secondary storage, magnetic disk to implement large memory spaces.
- Large disks available at less price, but its slower, so we follows
  - A huge amount of cost-effective storage can be provided by magnetic disks.
  - A large, affordable, main memory can be built with DRAM
  - A smaller unit is the SRAM, where speed is more as cache memory.

The entire computer memory can be viewed as the hierarchy



Memory hierarchy

Scanned by CamScanner

The access is faster in processor registers & they are at the top in memory hierarchy. To reduce the time needed to access the info, efficient solution is to use a fast cache memory which makes the main memory appear to the processor to be faster.

At the next level of the hierarchy is a small amount of memory that can be implemented directly on the processor chip. This memory called a processor cache, holds copies of instructions & data stored in a much larger memory that is provided externally.

There are two levels of caches,

- A primary cache is always located on the processor chip. This cache is small because it competes for space on the processor chip, which must implement many other function. The primary cache is known as L1 cache (Level 1)
- A larger, secondary cache is placed between the primary cache & rest of the memory. It is referred as L2 cache (Level 2). It is usually implemented using SRAM chips.

The next level hierarchy is called the main memory. This rather large memory is implemented using dynamic memory locations. The main memory is larger but slower than the cache memory.

Magnetic disk devices provide a huge amount of inexpensive storage. They are slow compared to semiconductor devices used to implement the main memory.

Scanned by CamScanner

V Cache memory

A simple & fast memory which is used for additional storage and to help the processor when the main memory is busy.

The effectiveness of cache memory is based on locality of reference.

Many instructions in localized areas of the program are executed repeatedly during some time period, and the remainder of the program is accessed relatively infrequently. This is referred as locality of reference. It has 2 types,

Temporal locality:

Recently executed instructions are executed soon.

Spatial locality:

The instructions which are not executed recently is likely to be executed soon.

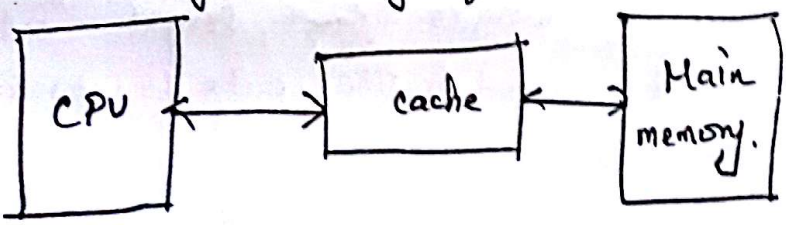
→ Temporal aspect of the locality suggests that whenever an information is needed, they should be brought into the cache will remain until needed again.

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

→ Block is referred to a set of contiguous address locations of some size. Another term is used to refer a block is cache line

→ Cache memory can store a reasonable number of blocks at any given time but this number is small compared to the total number of block in the main memory.

→ The communication between the main memory blocks & cache is specified by mapping function.



Scanned by CamScanner

- The processor need not known the presence of cache. It simply issues Read & Write request using addresses that refer to location in the memory. The cache control circuitry determines whether the requested word currently exists in the cache.
  - → If it exists, Read or Write operation is performed on the appropriate cache location. In a read operation, the main memory is not involved.
  - → For write operation, the system can proceed in two ways
    - * Write through protocol.
    - * Write back or copy back protocol.

<u>Write through protocol</u>

The cache & main memory locations are updated simultaneously.

<u>Write back</u> or <u>copy back protocol</u>

Here, Cache is only updated and to mark it as updated with an flag bit called modified bit. The main memory location is updated when the location is removed from cache.

→ When the addressed word in a read operation is not in the cache, a read miss occurs.

→ During a write operation, if the addressed word is not in the cache, a write miss occurs.

If the write through protocol is used, the information is written directly into the main memory.

If the write protocol is used, the block containing the addressed word is first brought into the cache, then the desired word in the cache is overwritten with the new information.

Scanned by CamScanner

VI Mapping functions

To discuss about the possible methods for specifying where memory blocks are placed in the cache.

Eg: Consider a cache memory consisting of 128 blocks of 16 words each, for a total of 2048 (2K) words, and assume that the main memory is addressable by 16 bit addres. The main memory has 64K words, which we will view as 4K blocks of 16 words each, We will use consecutive address.

Direct mapping

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

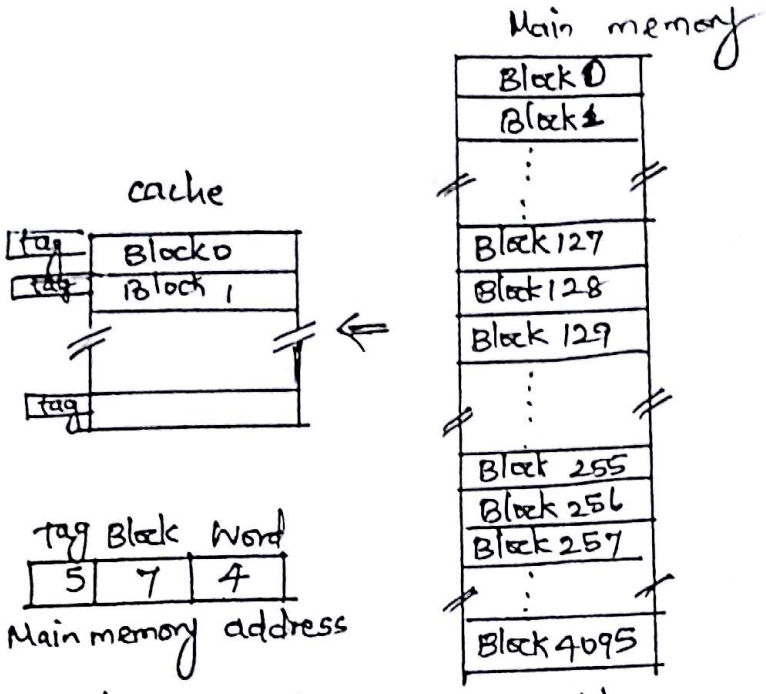
(Block address) MOD (Number of blocks in cache)

Consider a cache of 128 blocks. The jth block in main memory is mapped onto block j % 128 of the cache. Thus whenever one of the main memory blocks 0, 128, 256 .... is loaded into the cache, it is stored in cache block 0. Block 1, 129, 257 .... are stored in cache block 1 and so on.

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

- The low order 4 bits select one of 16 words in a block.
- When a new block enters the cache, The 7 bit cache block field determines the cache position in which This block must be stored.
- The high order 5 bits of The memory address of the blocks are stored in 5 tag bits associated with its location in the cache.

Scanned by CamScanner



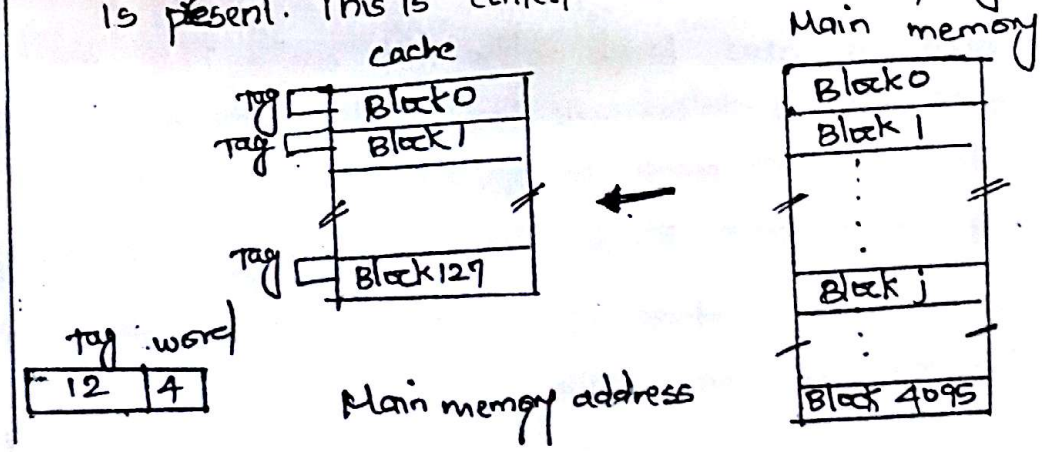
The high order 5 bits of the address are compared with the tag bits associated with that cache location.

- If they match, then the desired word is in that block of the cache.
- If there is no match, then the block containing the required word must first be read from the main memory & loaded into the cache.

Associative mapping

→ The main memory blocks can be placed into any cache block position. Here, 12 tag bits are required to identify a memory block when it is resides in the cache.

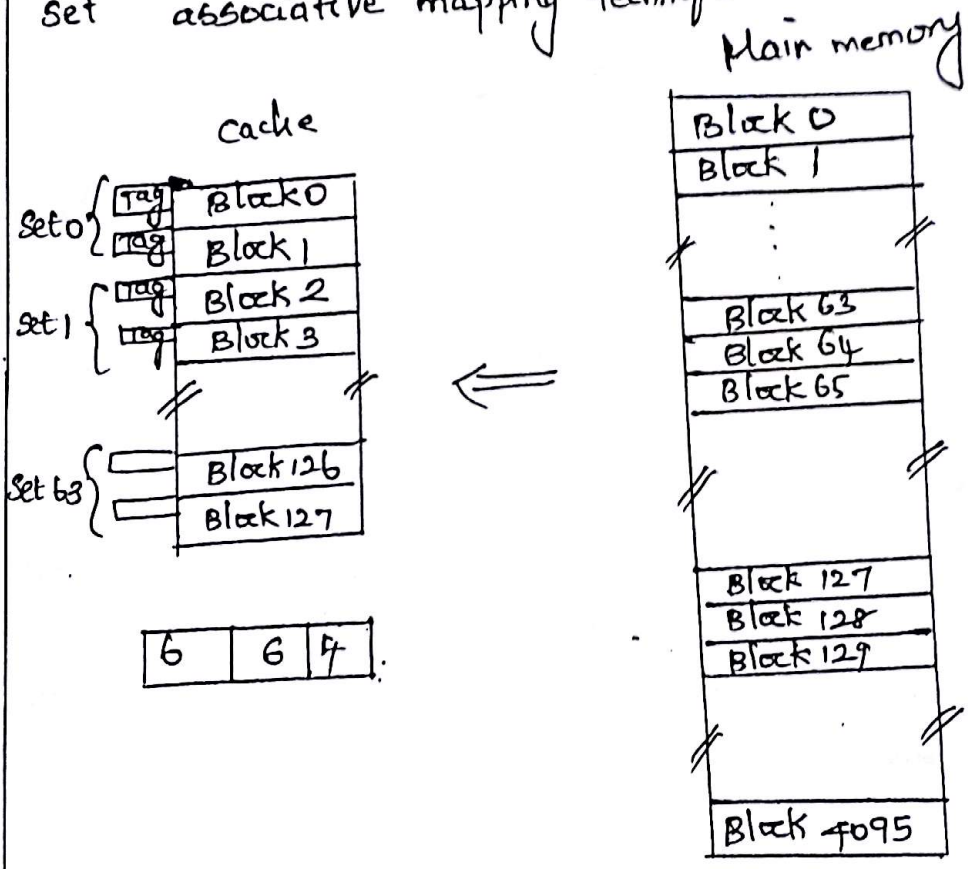
→ The tag bits of an address received from the processor are compared to the tag bits of each block of the cache to see if the desired block is present. This is called associative mapping techniques.



Scanned by CamScanner

Set Associative Mapping

A combination of the direct and associative mapping techniques can be used. Blocks of the cache are grouped into sets, & the mapping allows a block of the main memory to reside in any block of a specific set. At the same time, the hw cost is reduced by decreasing the size of the associative search. This is called the set associative mapping technique



Set-associative mapped cache with two blocks per set.

- Memory blocks; 0, 64, 128 ... 4032 map into cache set 0, and they can occupy either of the two block positions within this set. Having 64 set means, the 6-bit set field of the address determines which set of the cache might contain the desired block.
- Four blocks per set can be accomodated by a 5-bit set field, 8 blocks per set by a 4-bit set field. Another condition, 128 blocks per set requires no set bits & corresponds to the associative technique with 12 tag bits.

Scanned by CamScanner

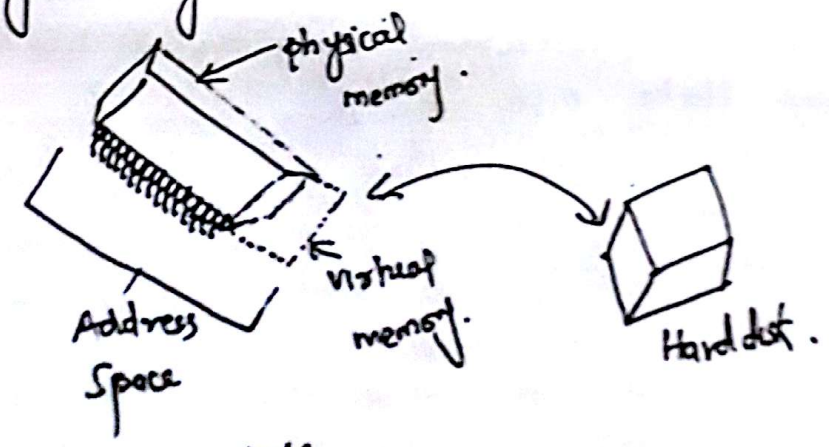
VII VIRTUAL MEMORY

VM is a memory is a memory management capability of an OS that uses h/w & s/w to allow a computer to compensate for physical memory (RAM) to disk storage.

Since computer has finite amount of RAM, it is possible to run out of memory when too many programs are running at one time. So virtual memory used. It increases the memory by enlarging address space. So HDD is used. During execution, data must be given back to main memory. This process takes more time.

VM is an imaginary memory area supported by OS. Its a alternate set of memory addresses. A processor that issues 32 bit addresses has an addr. space (2^{32}) 4 bytes. When a program does not completely fit into the main memory, the parts of its not being executed are stored on secondary storage devices, such as magnetic disks.

In modern computers, virtual memory moves program & data b/w main memory and secondary storage.



VM

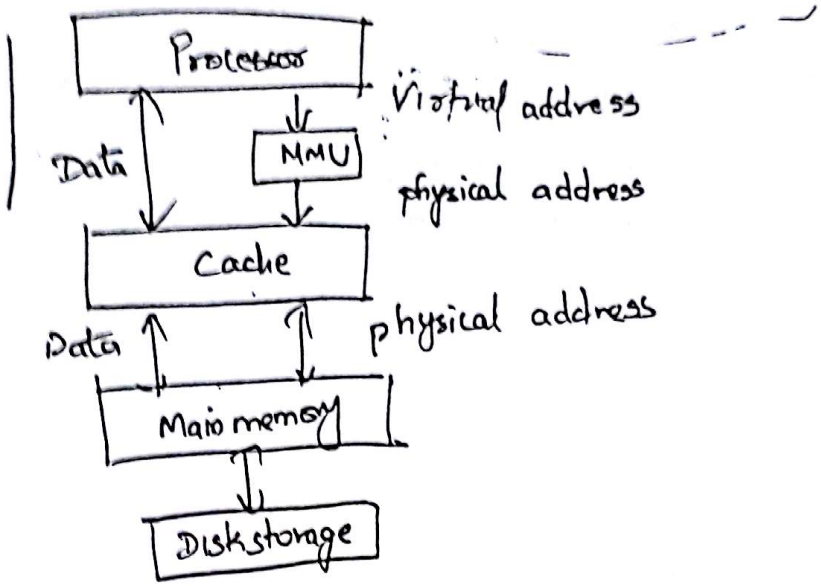
Scanned by CamScanner

The foll. diagram shows a typical organization that implements virtual memory. (17)

· In this diagram, special h/w unit, called the MMU translates virtual address into physical address.

· If the data is not in the main memory, the MMU causes the operating system to bring the data into the main memory from the disk.

~~Address~~ translation



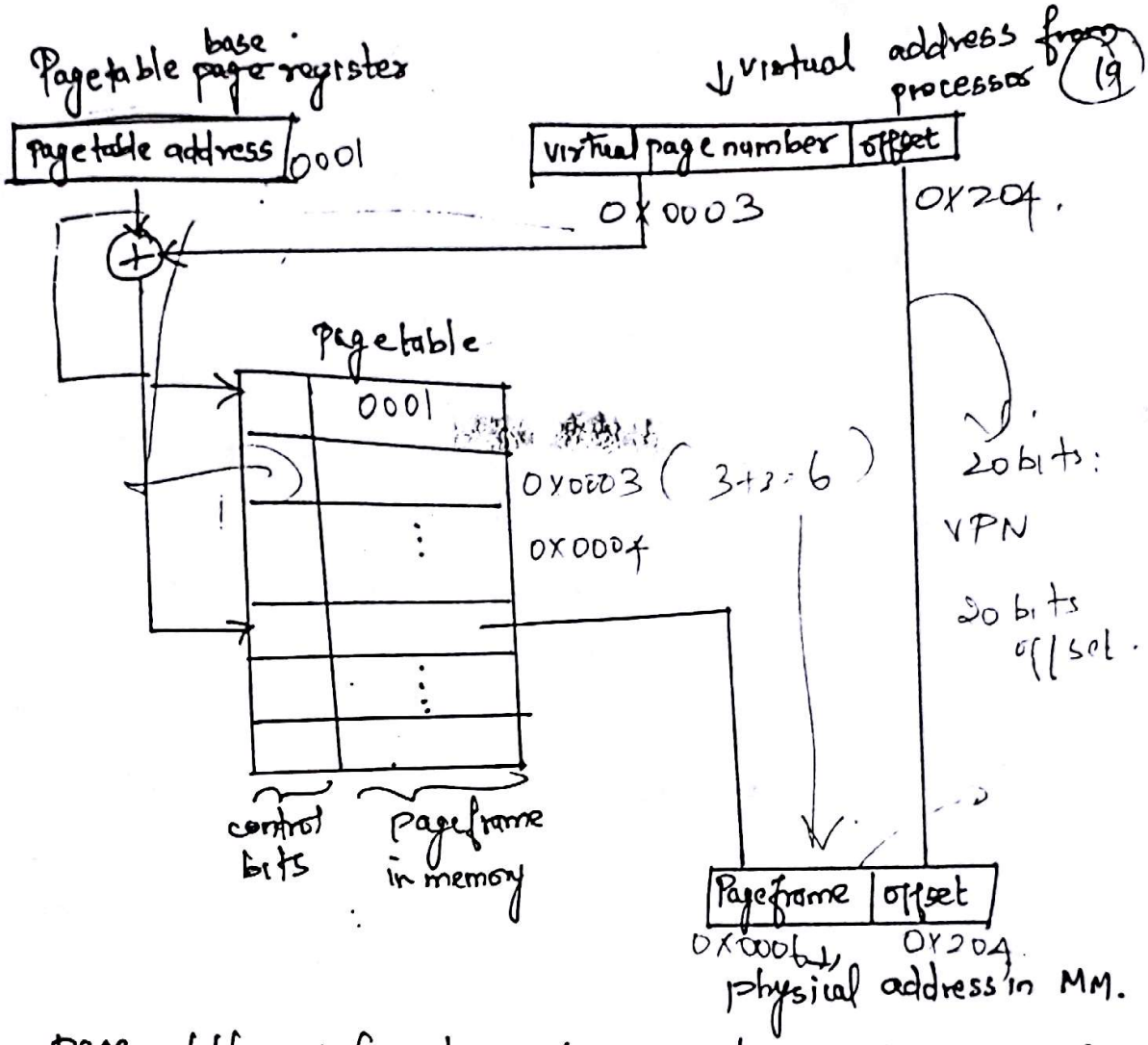
Address translation

- Virtual address space splitted into pages.
- Assume that all programs & data are composed of fixed-length units called pages, each consists of a block of words that occupy contiguous locations in the main memory.
- page commonly ranges from 2K to 16K bytes in length.
- Pages should not be too small, because the access time of a magnetic disk is much longer than the access time of the main memory.
- The cache bridges speed up the gap between the processor & main memory & implemented in h/w.

Scanned by CamScanner

- A virtual memory address translation based on the concept of fixed length pages.
- Each virtual address generated by the processor, whether it is for an instn. fetch or an operand fetch/store operation is implemented as a virtual page number (high order bits) followed by an offset (low order bits) that specify the location of a byte within a page.
- Information about the main memory location of each page is kept in page table. This information include the main memory address where the page is stored & the current status of the page.
- An area in the main memory that can hold one page is called page frame, The starting address of the page table is kept in a page table base register.
- By adding the virtual page number to the contents of this register, the address of the corresponding entry in the page table is obtained
- The contents of this location give the starting address of the page if that page currently resides in memory.
- Each entry in the page table includes control bits that describes the status of the page.
- one bit indicates the validity of the page, where the page is actually loaded in the memory. Another bit indicates whether the page has been modified,
- As in cache memories, the mm needed to determine whether the page should be written back to the disk before it is removed from main memory to make place for another page.

Scanned by CamScanner



The page table information is used by the MMU for every read & write access, Sometimes the pagetable may be large, and since MMU implemented as part of cpu chip. Page table kept in main memory. A copy of a small portion of the page table can be ~~implemented~~ accomodated within MMU.

The portion consists of the page table entries that corresponds to the most recently accessed pages. Small cache is called TLB (Translation lookaside buffer), is incorporated with the MMU for this purpose.

VA → 16 bit long 2^{16} addresses. 12 bit

Transfer of data between the disk and the main memory is called Direct Memory Access.

Scanned by CamScanner

VIII Evaluating memory performance.

- Two key factors in the computer are performance & cost
- Performance must be improved without affecting the cost
- Performance depends on
  - how fast machine instruction can be brought into the processor for execution &
  - how fast they can be executed.
- The speed & efficiency of data transfer between various levels of the hierarchy are also of great significance.
  - It is beneficial if transfers to and from units can be done at a rate equal to that of faster unit.
  - Its not possible when slow & fast units are accessed in the same manner, but it can be done by doing parallelism in the slower devices.
  - An effective way to introduce parallelism is to use interleaved organization.

① Interleaving

- If the main memory of the computer is structured as a collection of physically separate modules, each with its own address buffer register (ABR) and data buffer register (DBR), memory access operations may proceed in more than one module at the same time. Thus transmission rate is increased.
- 2 methods of address layout are indicated.
- In the first method, The high order k bits name one of n modules & the low order m bits name a particular word in that module.
- Its shown in the 2 figures, shown below..

Addressing multiple-module memory system.

Scanned by CamScanner

- When consecutive locations are accessed as happens when a block of data is transferred to a cache, only one module is involved.

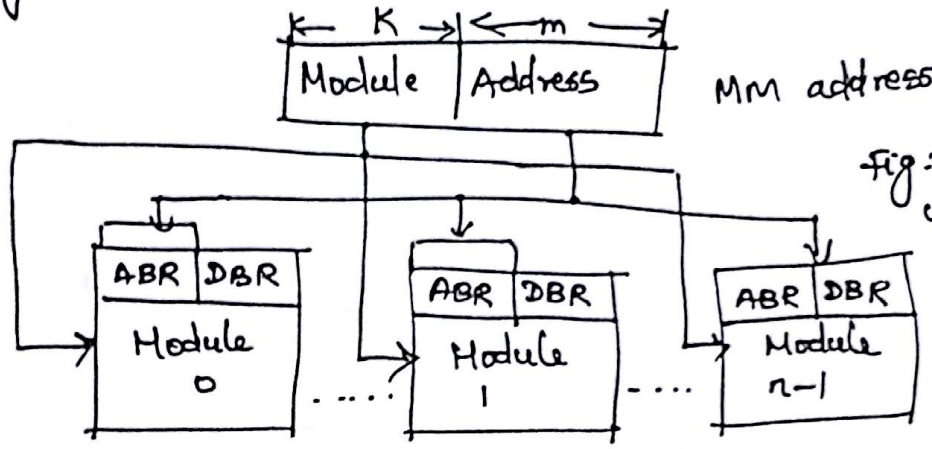


Fig: consecutive words in a module

When one module is being accessed & the other module can be accessed via DMA & hence increasing the performance.

- The second method for memory modules is called memory interleaving is shown in the foll. figure. The low order K bits of the memory address select a module, and the high order m bits name a location within that module.

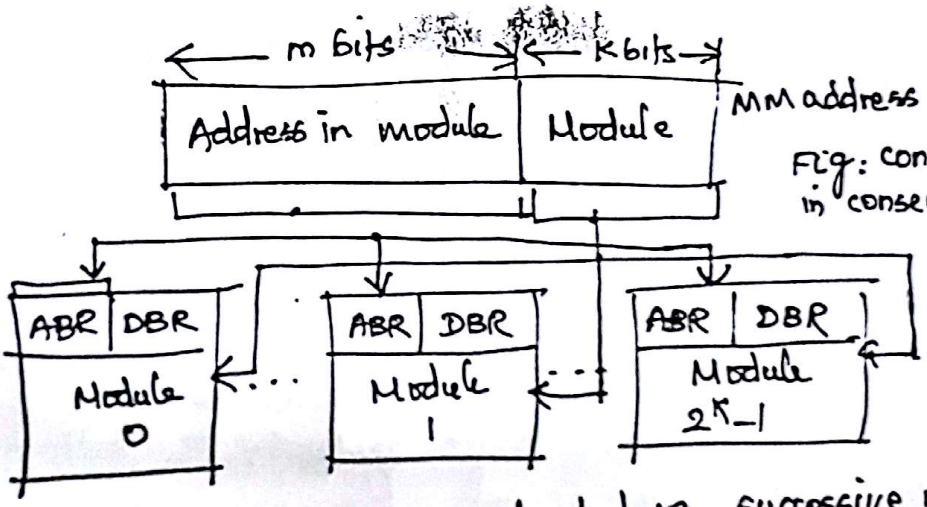


Fig: consecutive words in consecutive modules

Here, consecutive addresses are located in successive modules. Thus, any component of the system that generates request for access to consecutive memory locations can keep several modules busy at one time. This result in both faster access to a block of data & high average utilization of the M as whole.

Scanned by CamScanner

(2) Hit rate & Miss penalty

- A successful access to data is called a hit.
- The number of hits stated as a fraction of all attempted accesses is called hit rate.
- Miss rate is the number of misses stated as a fraction of attempted accesses.
- High hit rate measured as 0.9 in computers.
- Performance affected by the actions that must be taken after a miss.
- Extra time needed to bring the desired information into the cache is called the miss penalty.
- The miss penalty is the time needed to bring a block of data from a slower unit in the memory hierarchy to a faster unit.
- Let h be the hit rate, M the miss penalty, that is the time to access information in the main memory and C the time to access information in the cache.
- The average accesse time experienced by the CPU is,

$$t_{ave} = hC + (1-h)M \Rightarrow \text{Time} = \text{Hit time} + \text{Miss Rate} \times \text{Miss penalty}.$$

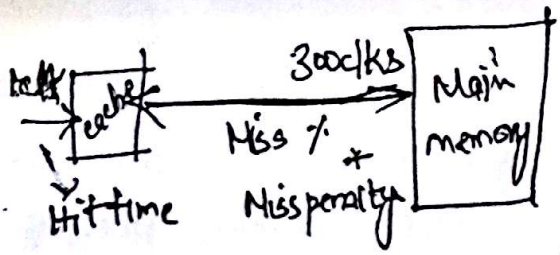
Eg1: CPU with 1ns clock, hit time = 1 cycle, Miss penalty = 20 cycles, cache miss rate = 5%.

$$t_{ave} = 1 + 0.05 \times 20 = 2 \text{ ns}.$$

2 cycles / 2 instruction.

Eg2: Cache Hit = 1 cycle,
Miss rate = 10% = 0.1
Miss Penalty = 300 cycles

$$t_{ave} = 1 + 0.1 \times 300$$
$$= 31 \text{ cycles}.$$



Scanned by CamScanner

X. MULTIPROCESSOR

→ Multiprocessing is the use of two or more central Processing Unit within a single computer systems. The term also refers to the ability of a system to support more than one processor or the ability to allocate tasks between them.

→ In a MP system, all CPUs may be equal or some may be used for special purposes.

→ Systems that treat all CPUs equally are called Symmetric multiprocessing systems.

→ If all CPUs are not equal, its called Asymmetric multiprocessor

→ Multiprocessor can be used in the following ways.

- SISD
- SIMD
- MISD
- MIMD.

→ Multiprocessor communication Architectures.

Message passing

- separate address space for each processor.
- Processors communicate via message passing

Shared memory.

- Processor communicate with shared address space.
- Processors communicate by memory read/write
- Easy on small scale machines

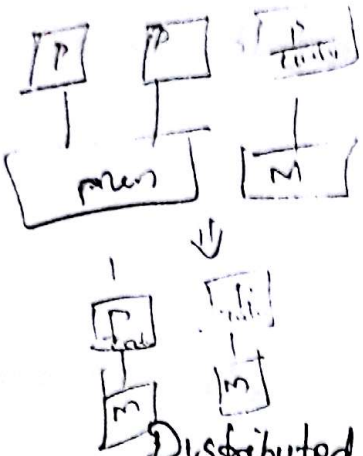
Based on processor coupling MP split into the foll. 2 types.

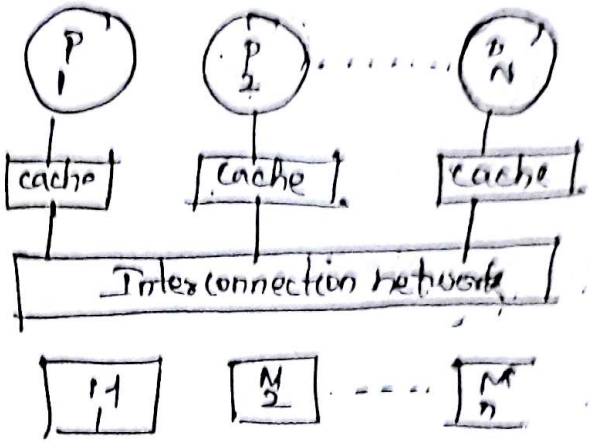
- Tightly coupled multiprocessor systems (Its connected at the bus level
- Loosely coupled multiprocessor systems.

Scanned by CamScanner

Shared-memory processors (SMP (eg - IBM P690))

- Global memory accessible by all processors.
- Relatively easy to program & difficult to scale to large number of processors.





P - Processor
M - Memory.

Distributed Memory (local memory)

- All memory units are associated with the processor.
- To retriere information from another processors memory a message must be sent there.

Uniform Memory (UMA)

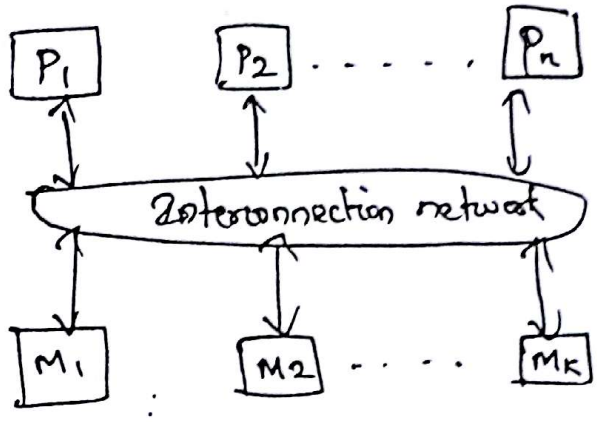
- All processors take the same time to reach all memory locations.
- It has uniform memory access.

Non uniform memory access

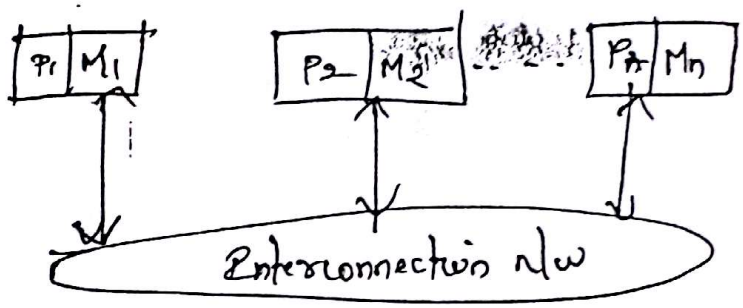
- Its distributed shared memory.
- Access to some parts of memory is faster for some processors than other parts of memory.
- Harder to program, but scales to more processors
- Memory access is not uniform.

Scanned by CamScanner

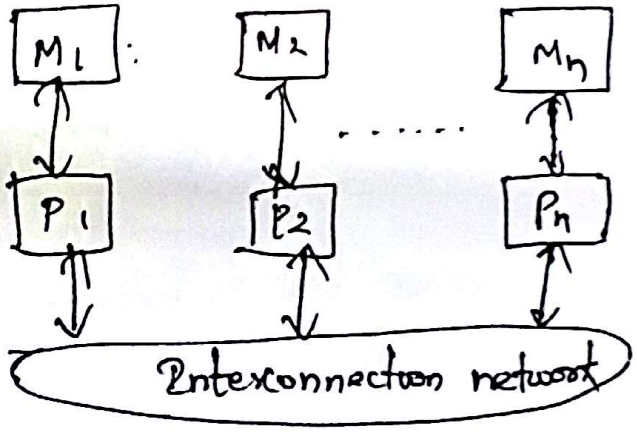
UMA multiprocessor



NUMA multiprocessor.



A Distributed memory system



IX Secondary storage devices.

Refer UNIT-I notes.

C. GEETHA
Asst Prof

Scanned by CamScanner

Input/output interface - Input output performance measures - I/O Modes of Transfer - Direct memory access - Input Output Processor - Interfacing to different types of I/O devices.

I. INPUT OUTPUT INTERFACE

An interface consists of the circuitry required to connect an I/O device to a computer bus.

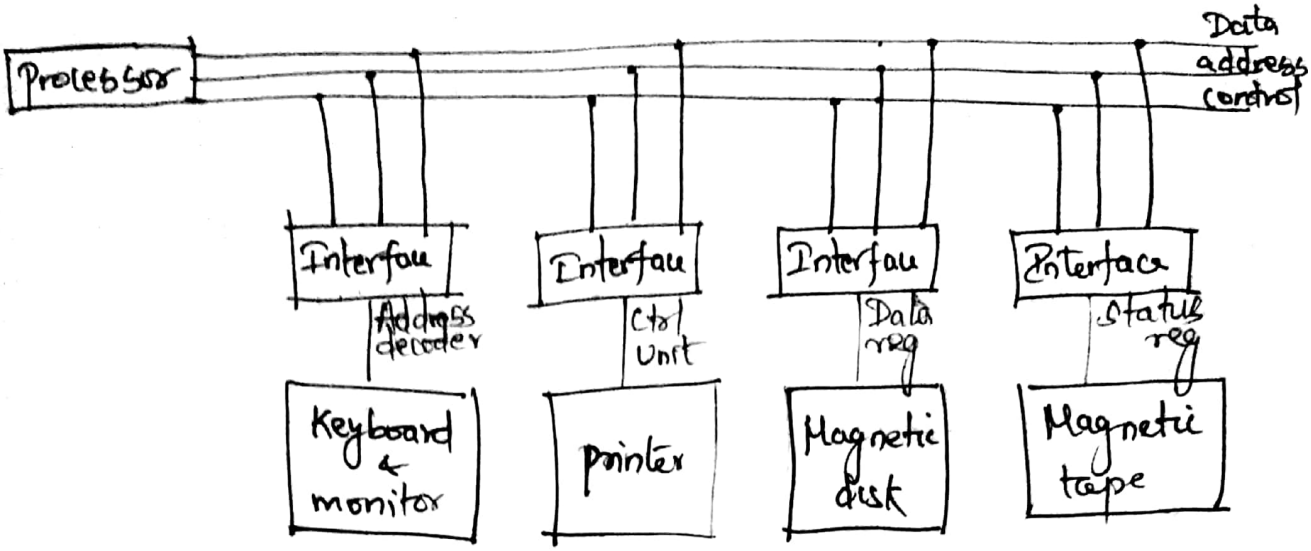
On one side of the interface, the bus signals for address, data & control. On the otherside we have a path which its associated controls to transfer data between the interface & the I/O devices

Input or Output devices attached to a computer called peripherals. Eg: Monitor, keyboard.

Interface.

- Special hardware components between CPU & peripherals to supervise & synchronize all i/o transfers.
- I/O bus consists of data line, address line & control line.
- Each peripheral device has an interface unit.
- The interface do the following
  - → Decodes the address & control received from I/O bus.
  - → Interprets them for peripheral.
- I/O bus from the processor is attached to all the peripheral interfaces.

Scanned by CamScanner



4 types of I/O commands

* Control
* Status
* Data Input
* Data output

Control command

Its used to activate the peripheral and to inform it what to do.

Status command (Status register has the ctrl flags SIN & SOUT).

Its used to test various status conditions in the interface and peripheral.

Data output command

It causes interface to respond by transfering data from bus into one of its register.

Data input command

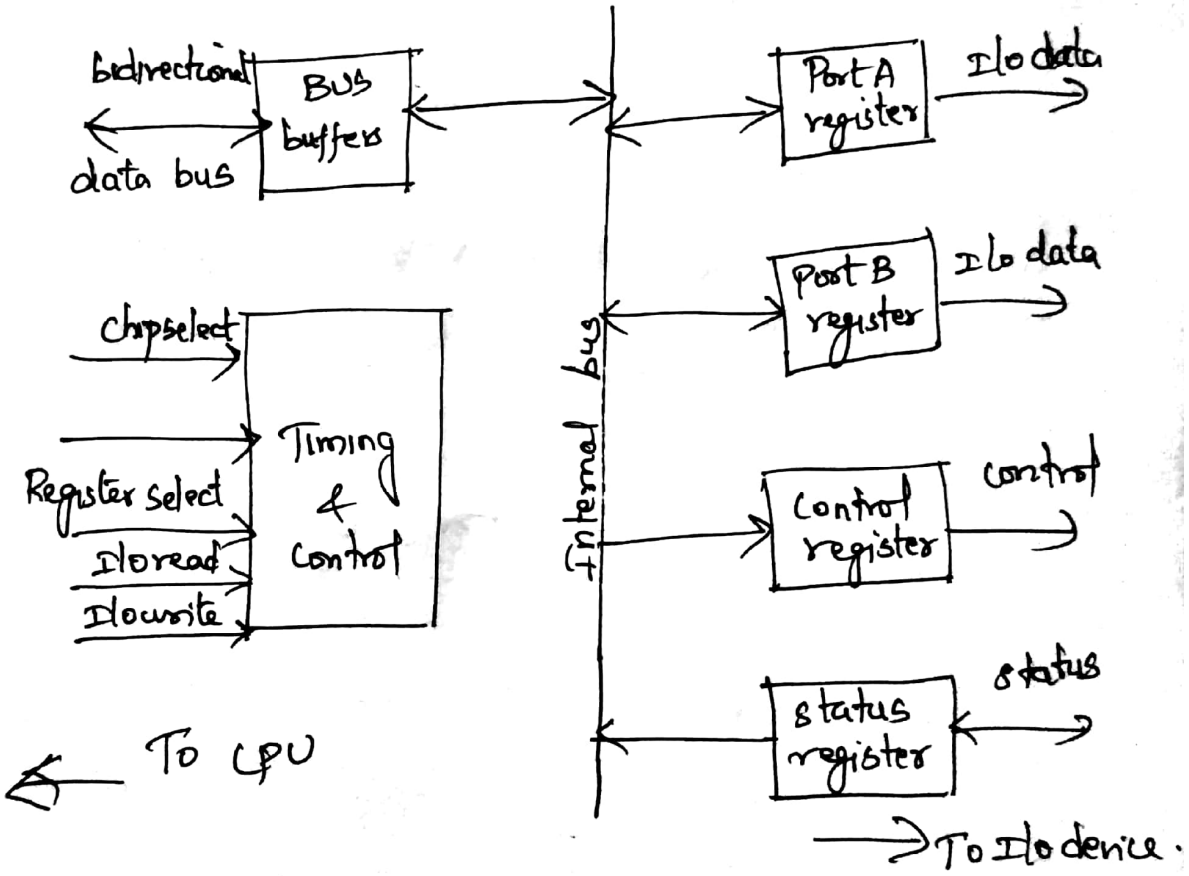
Its the reverse of data output command.

Scanned by CamScanner

I/O versus memory bus

Three ways that computer buses used to communicate with memory & I/O.

1. Use 2 separate buses (one for memory & one for I/O).
2. Use one common bus for both memory & I/O bus separate control lines for each.
3. Use one common bus for memory & I/O with common control lines.

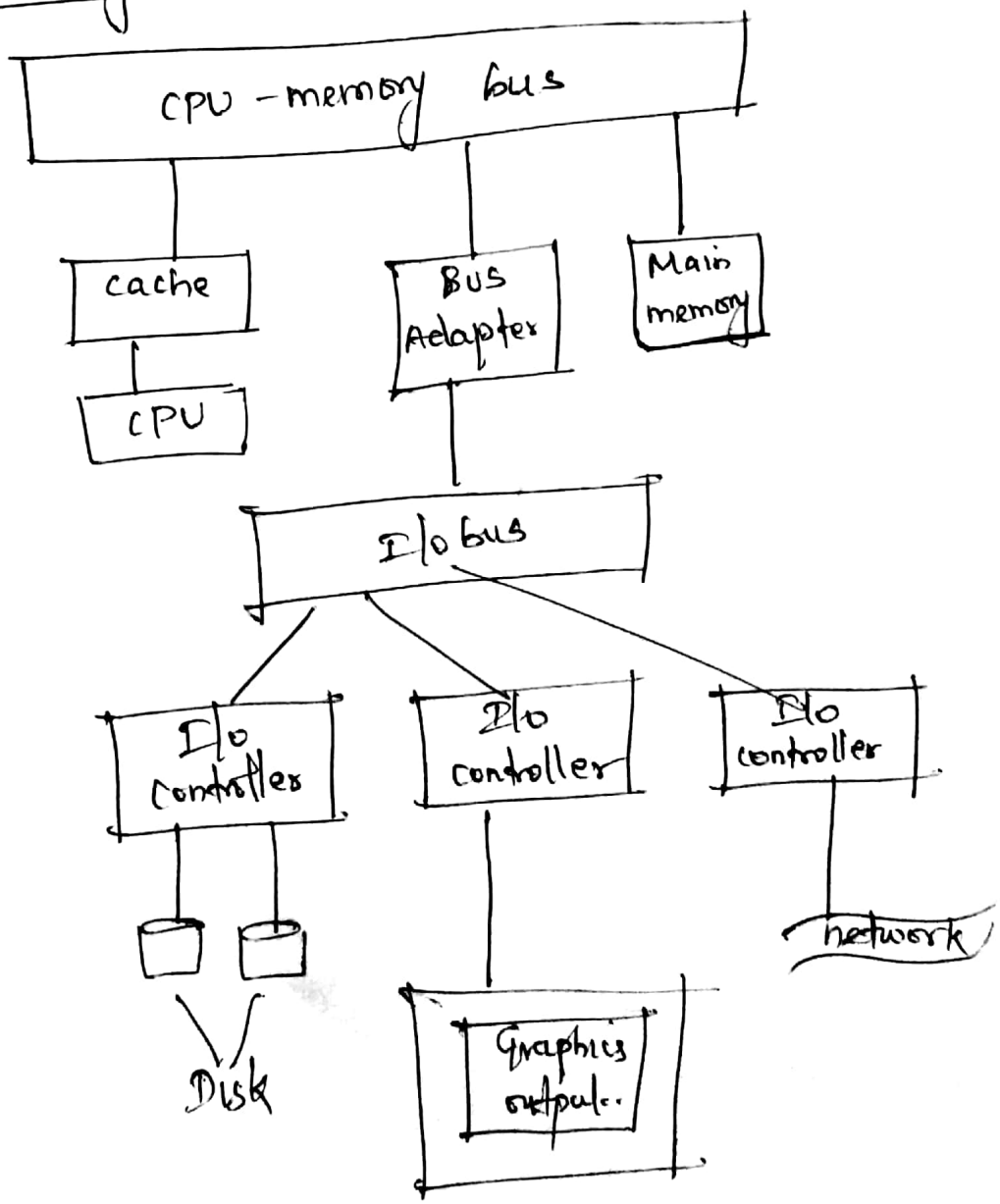


- It consists of 2 data registers called ports, a control register, a status register, bus buffers, timing & control circuits.
- Interface communicates with CPU through data bus
- I/O read & write are two control lines chipselect & register select determines the address.

Scanned by CamScanner

- I/O data from the device can be transferred into either port A or port B.
- Interface may operate with an output or an input device.
- Control register receives control information from CPU.
- Interface registers communicate with CPU through bidirectional data bus.

Typical Interface of I/O devices & I/O bus to CPU memory bus.



Scanned by CamScanner

II. Input Output Performance measures

- Input Output Throughput is called bandwidth.
- Response time is called latency.

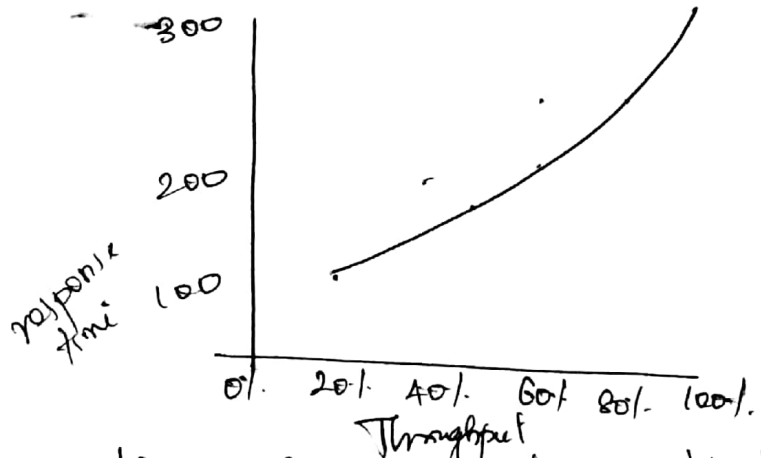
Response time:

Its the time, a task takes from the moment it is placed in the buffer until the server finishes the task.

Throughput

Its the average number of tasks completed by the server over a time period.

- Another measure is the interference of I/O with CPU execution



An interaction with a computer is divided into 3 parts.

Entry time: The time for the user to enter the command. (Graphic system needs 0.25 s to enter a command).

System Response time: The time between the user enters the command & the complete response is delayed.

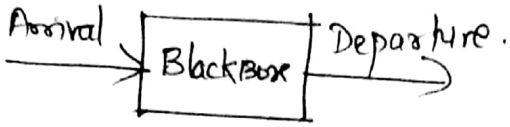
Thinks time: The time from the reception of the response until the user begins to enter the next command.

The sum of these three part is called transaction time.

Scanned by CamScanner

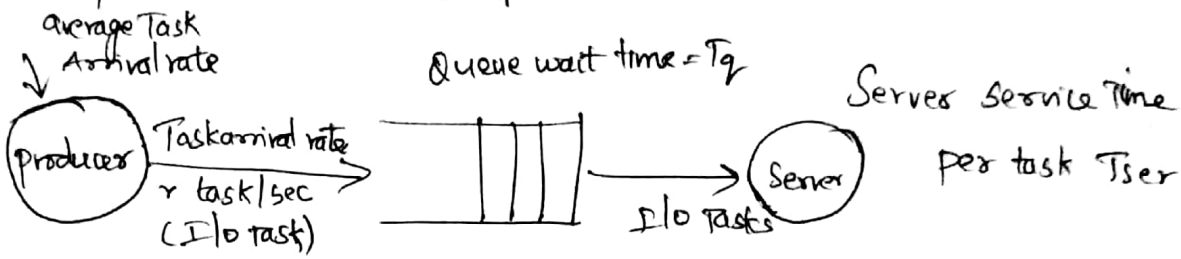
Queuing Theory

- CPU makes I/O request that arrive at the I/O device and the request "depart" when the device fulfills them.



I/O Modeling:

Producer – Server Model



(Single queue + Single Server.

Time system = Time in System for a task =

Response Time = Queuing Time + Service Time.

To increase the throughput,

- The server should never be idle
- The queue should never be empty.

To decrease response time,

- The queue should be empty
- The server will be idle at times.

Little's law

cycle times are proportional to queue lengths.

The long-term average number L of customers in a system is equal to the long-term average effective arrival rate λ multiplied by the average time W that a customer spends in the system.

The law is expressed as,

$$L = \lambda W.$$

Scanned by CamScanner

Its a formula shown below...

Mean number of tasks = Arrival rate × Mean response time in system.

$Time_{Server}$ → Average time to service a task

$Time_{Queue}$ → Average time per task in the queue.

$Length_{Server}$ → Average no. of tasks in service.

$Length_{Queue}$ → Average length of Queue

$Length_{System}$ → Average no. of tasks in system.

$Length_{System}$ = Arrival rate (λ) × Time system (W).

$$\text{Server Utilization} = \frac{\text{Arrival rate } r \text{ task/sec}}{\text{Service rate } (\mu)}$$

$LQ = 4$
$TS = 0.4ms$
$ST = 0.3ms$

$$Time_{System} = \underbrace{Length_{Queue} \times Time_{server}}_{\text{Queuing time}} + \underbrace{\text{Mean time to complete service of tasks whose new task arrive.}}_{\text{Service time}}$$

$$\text{weighted mean time} = \frac{f_1 \times T_1 + f_2 \times T_2 + \dots f_n \times T_n}{\sum_{i=1}^{n} f_i}$$

$$\text{Variance} = \frac{f_1 \times T_1^2 + f_2 \times T_2^2 + \dots f_n \times T_n^2}{\sum_{i=1}^{n} f_i} - \text{weighted mean time.}$$

$$C = \frac{\text{variance}}{\text{weighted mean time}^2}.$$

$$Time_{Queue} = Time_{Server} \times \frac{\text{Server Utilization}}{(1 - \text{Server utilization})}$$

Scanned by CamScanner

eg: (Little's Law)

consider the following scenario: The inter-arrival time is exponentially distributed with a mean of 10 minutes & the service time is distributed with a mean of 8 mins.

i) mean wait in the queue ii) mean number in the queue

iii) the mean wait in the system (iv) mean number in the system

(v) proportion of time the server is idle.

$\lambda = 1/10$, $\mu = 1/8$, $\rho = 8/10$.

Number in the queue, $L_q = \frac{\rho^2}{1-\rho} = \frac{0.8^2}{1-0.8} = 3.2$

Wait in the queue $= W_q = L_q / \lambda = 32$ mins

Wait in the system $= W = W_q + 1/\mu = 40$ mins.

Number in the system $= L = \lambda W = 4$.

Proportion of time the server is idle $= 1 - \rho = 0.2$

Values:

Server utilization, $\rho = \frac{\lambda}{\mu} = 0.8$

$W_q = 3.2 \times 10 = 32$.

$W = W_q +$ service time

$= 32 + 1/\mu = 32 + 8 = \underline{40 \text{ mins}}$

$L = \lambda \cdot W \Rightarrow \frac{1}{40} \times 40 = 4$.

$\underline{L = 4}$

$\underline{\rho = 0.8}$

$\underline{L_q = 3.2}$.

Scanned by CamScanner

III. I/O Modes of Transfer

* Data transfer between central computer and I/O device may be handled in a variety of modes.

Modes

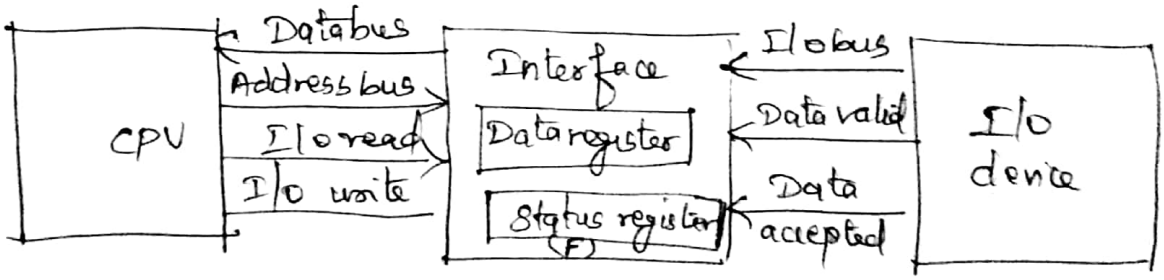
Programmed I/O

Interrupt - Initiated I/O

Direct memory Access.

Programmed I/O

- Programmed I/O operations are the result of instructions written in the computer program.
- Each data item transfer is initiated by an instruction in the program.
- In programmed I/O method, I/O device does not have direct access to memory.
- In programmed I/O method, the CPU stays in the program loop until the I/O indicates that it is ready for data transfer.

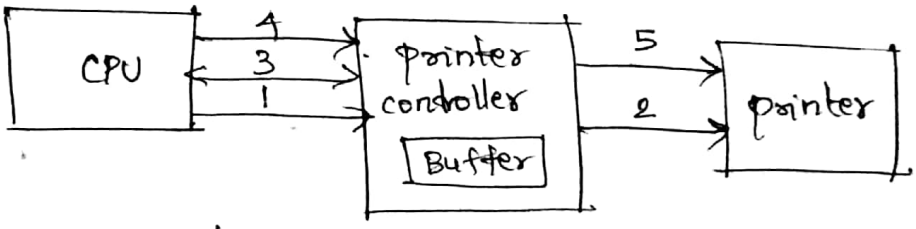


- A program is written to check the flag in the status register to determine if a byte has been placed in to data register by the I/O device.

Scanned by CamScanner

Interrupt - Initiated I/O

* CPU being keep on pointing the status of the flag bit is avoided by using a interrupt facility. This problem being encountered in the programmed I/O method is overcome by this mode.
* Device initiated I/O transfer.
* External device is connected to a pin called interrupt (INT) pin on the processor chip.
* When the device needs an I/O transfer with the computer, it activates the interrupt pin of the processor chip.
* Vectored Interrupt → The source that interrupt supplies the branch address to the computer. This information is called Interrupt vector.
* Non vectored Interrupt → The branch address is assigned to a fixed location in the memory.



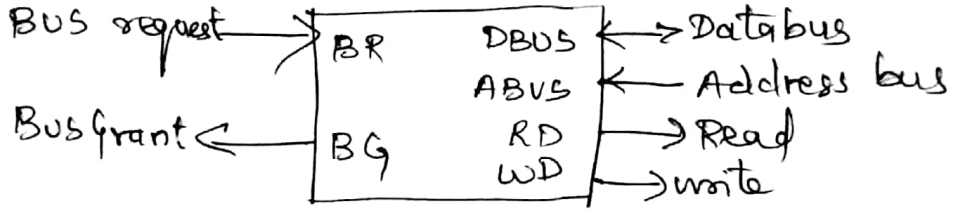
1. CPU Initializes printer controller of the I/O transfer.
2. Printer controller checks status of printer.
3. If printer is ready, printer controller informs CPU by sending an interrupt - signal.
4. CPU transfers data from memory to printer controller buffer.
5. From buffer, data is transferred to the printer.

Scanned by CamScanner

IV Direct Memory Access (DMA)

- In DMA, the interface transfers data into and out of the memory unit through memory bus.
- DMA controller takes over the buses to manage the transfer directly between I/O device & memory.
- Used for fast transfer of information b/w magnetic disk & memory.

CPU bus signals for DMA transfer

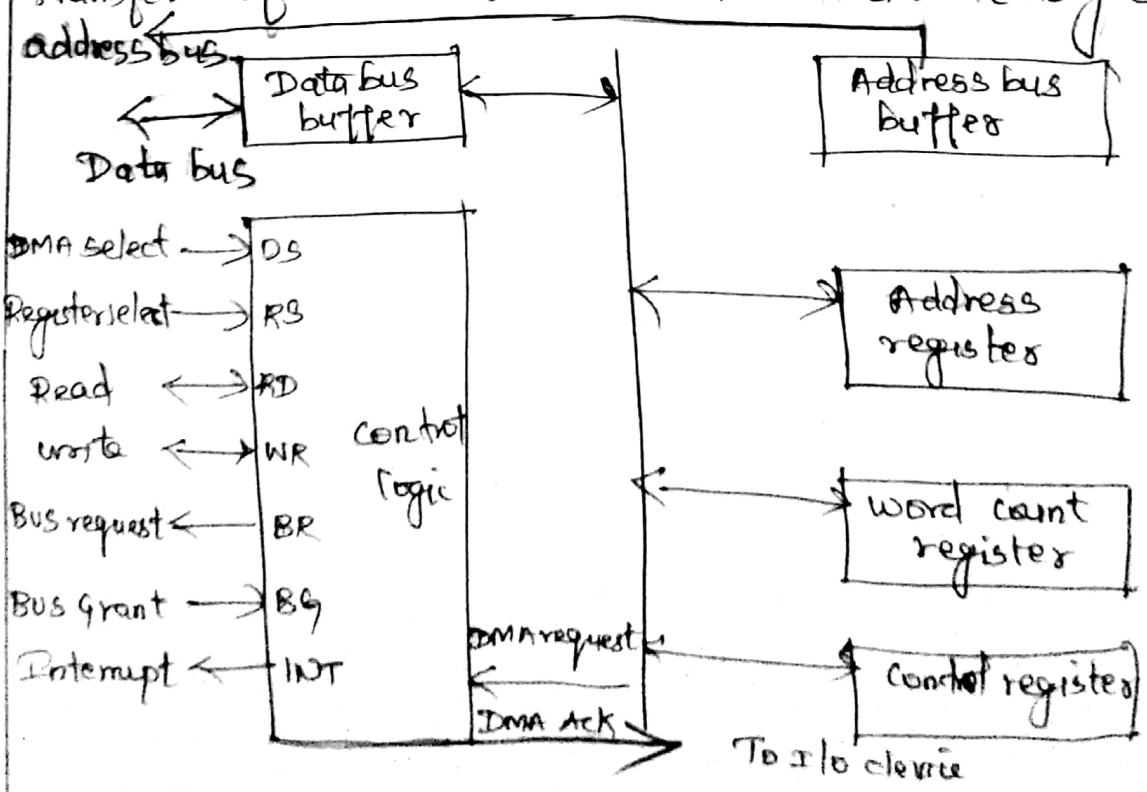


Bus Request (BR)

DMA uses this this to request the CPU to relinquish control of the buses.

DMA controller

Transfer of data between Disk & main memory is DMA.



Scanned by CamScanner

It contains 3 registers:

Address register, Word count register, control register

Address register:

Contains an address to specify the desired location in memory.

word count register

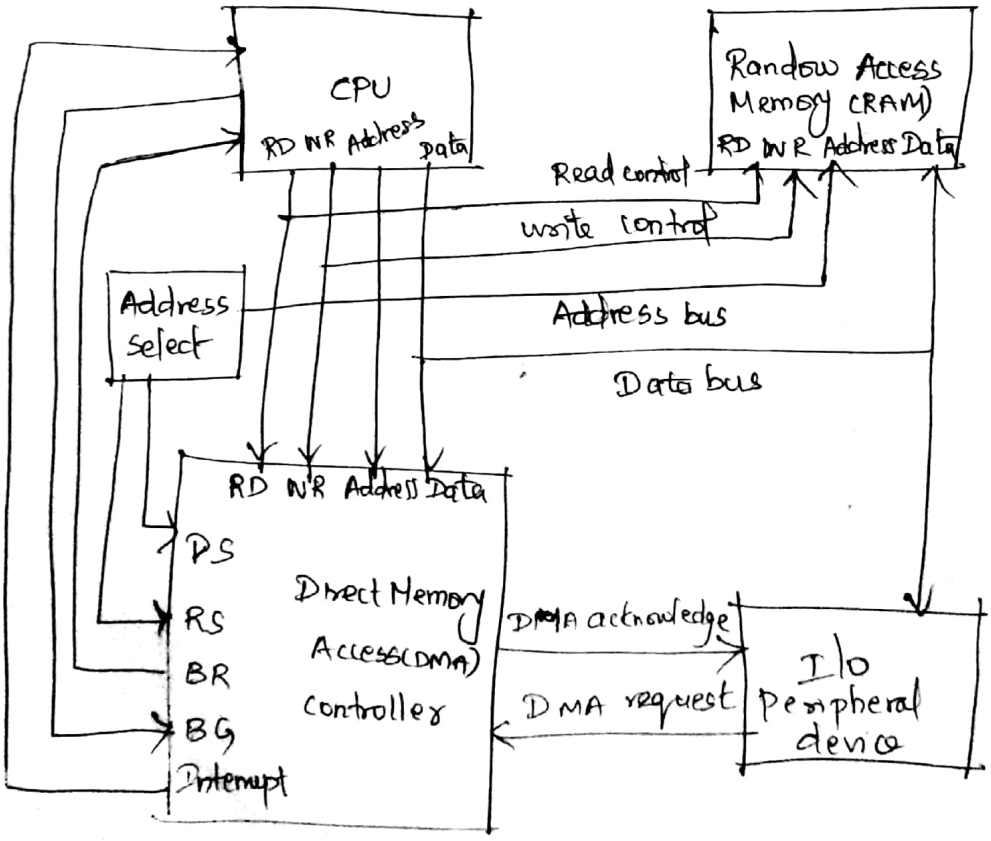
Holds the number of words to be transferred.

Control register

Specifies the modes of transfer.

DMA transfer

The connection b/w the DMA controller and other components



In a computer system for DMA transfer shown above.

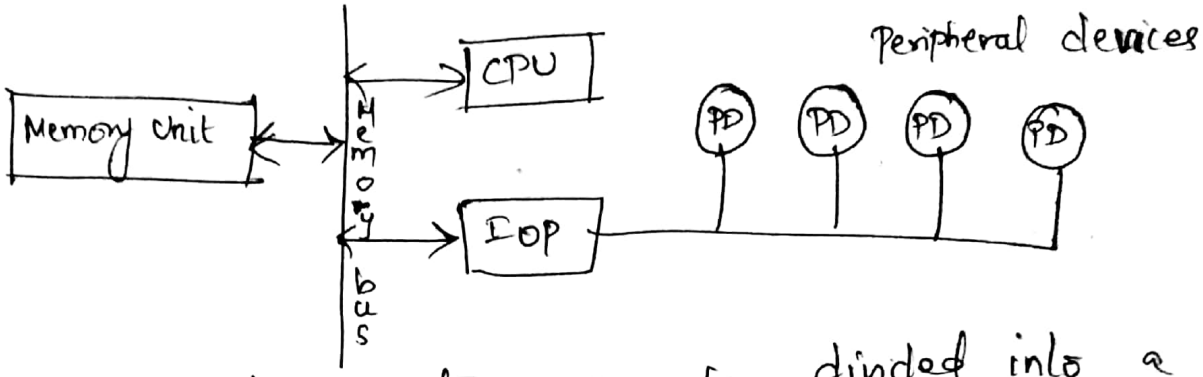
Scanned by CamScanner

- The DMA request line is used to request a DMA transfer
- The bus request (BR) signal is used by the DMA controller to request the CPU to relinquish control of the buses.
- The CPU activates the bus grant (BG) output to inform the external DMA that its buses are in a high-impedence state (so that it can be used in DMA transfer)
- The address bus is used to address the DMA controller and memory at given location.
- The device select (DS) & register select (RS) lines are activated by addressing the DMA controller.
- The RD & WR lines are used to specify either a read or write operation on the given memory location
- The DMA acknowledge line is set when the system is ready to initiate data transfer.
- The data bus is used to transfer data between I/O device & memory.
- When the last word of data in the DMA transfer is transferred, the DMA controller informs the termination of the transfer to the CPU by means of the interrupt time.

Scanned by CamScanner

V. Input output Processor (IOP)

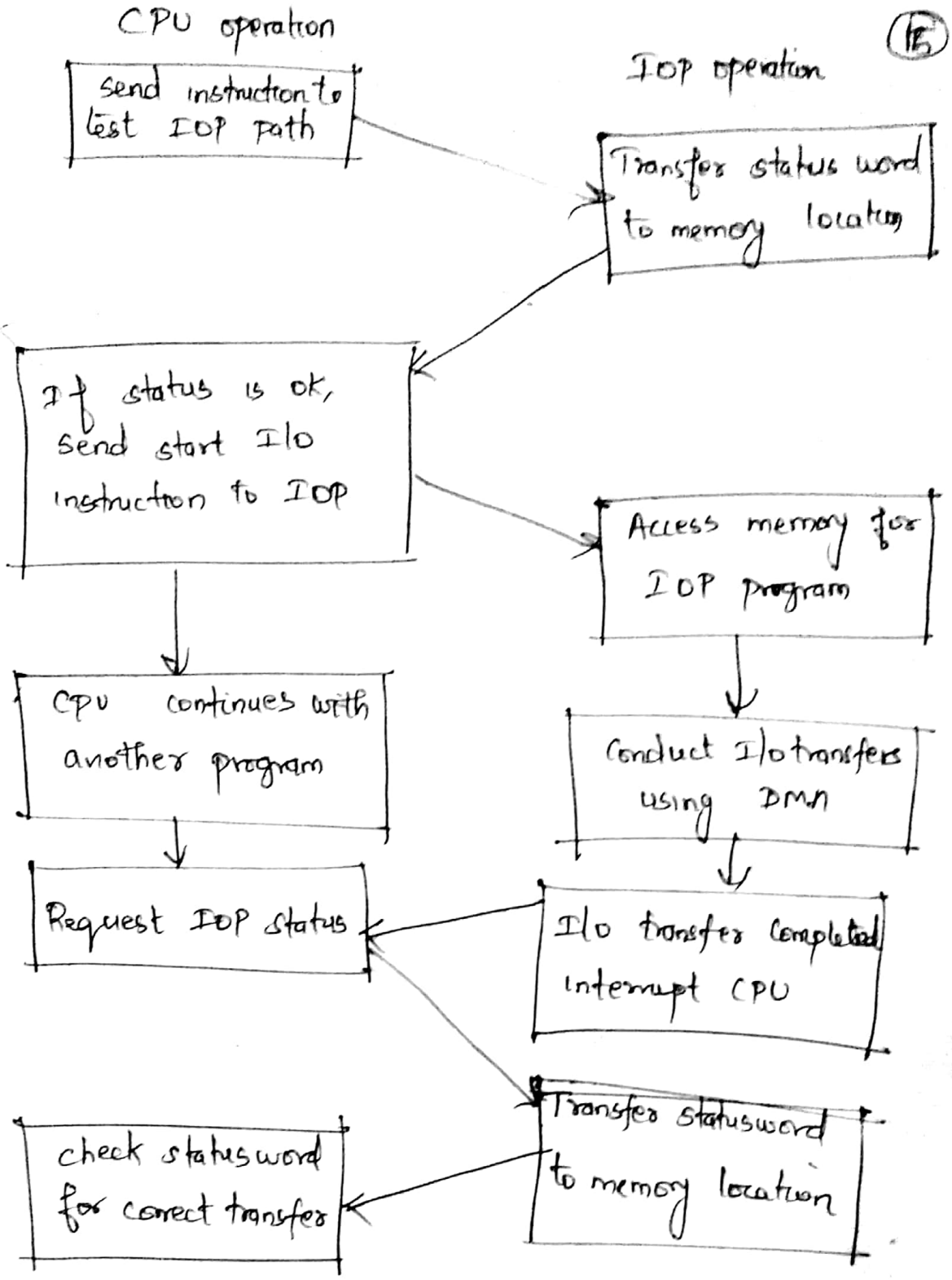
IOP is a set of programming interfaces that allow a programmer to coordinate activities among different program processes that can run concurrently in an OS, used to handle many request.



* A computer system can be divided into a memory unit, and no. of processors comprised of CPU & one or more IOP's.
* IOP takes care of input & output tasks thereby relieving the CPU from I/O transfer.
* IOP can fetch & execute its own instruction.
* IOP instructions are specially designed to facilitate I/O transfer.
* IOP can perform other processing tasks such as arithmetic, logic, branching & code translation.
* Instructions read from memory by an IOP are sometimes called commands.

IPC mechanisms can be classified into pipes, first in, first out & shared memory.

Scanned by CamScanner



VI Interfacing to different types of I/O devices

Keyboard connection to the processor:

A typical keyboard consists of mechanical switches that are normally open. When a key is pressed, its switch closes & establishes a path for an electrical signal. This signal detected by an encoder circuit that generates ASCII code for the corresponding character.

Scanned by CamScanner

- The following figure shows the h/w approach.
- The output from the encoder consists of the bits that represent the encoded character and one control signal, called valid which indicates that a key is being pressed. This information is sent to the input interface. The interface contains a data register, DATAIN and a status flag, SIN. When a key is pressed, the valid signal changes from 0 to 1. This causes the ASCII code to be loaded into DATAIN & SIN to to set to 1. The status flag SIN is cleared to 0 when the processor reads the contents of the DATAIN register. The I/O interface is connected to the asynchronous bus on which transfers are controlled using the handshake signals Ready & Accept, indicated in the following figure. The third control line, R/$\overline{W}$ distinguishes read & write transfers.

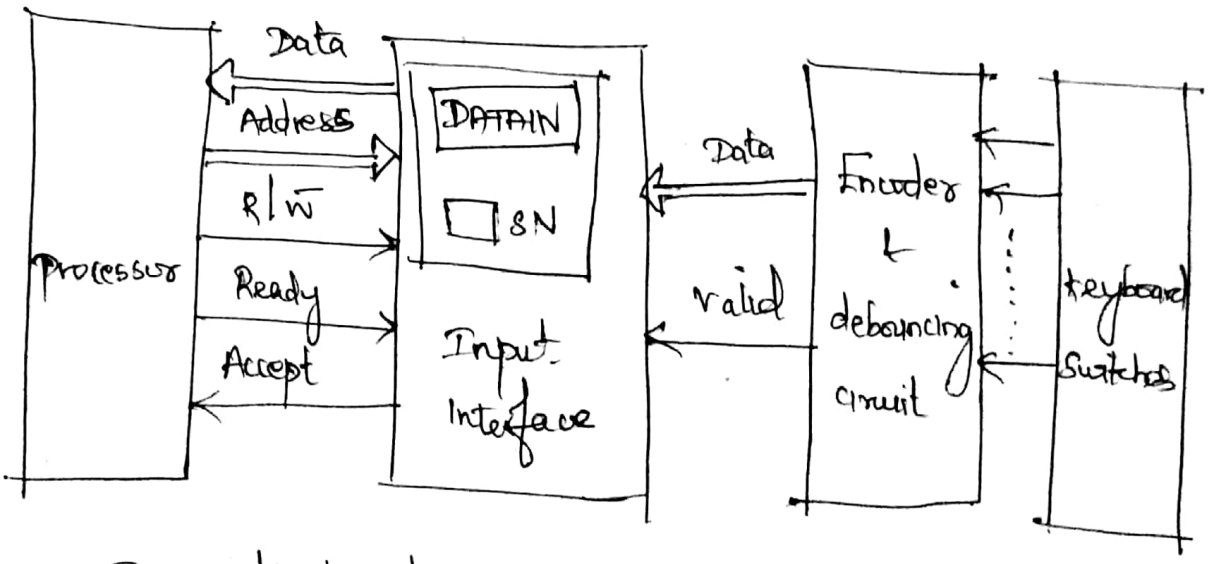


Figure: Keyboard connected to processor.

C. GEETHA
AP/CSE
BHARATH UNIVERSITY

Scanned by CamScanner

UNIT V

PARALLEL ARCHITECTURE

Data flow – Static Dataflow – Dynamic data flow – Dataflow graph
Vector processors – CRAY-I Vector processor – EPIC

---

1. DATA FLOW

Data flow is the movement of data through a system comprised of software & software.

Dataflow computers defined a model or diagram in which the entire process of data movement is mapped, taking into consideration how it changes during process ie, the path taken by a message from origination to destination or the path of data from its source document to final document.

Features of data flow model

- Final results are passed directly as data token between instructions, data will not be shared.
- Information items in a data flow computer appear as operation packets and data tokens.

Types of data flow computers:

* Static model
* Dynamic model.

In data flow arch, performance of an operation on data is triggered by the presence of data items.

Here, data flow done by packets and tokens. packet is a unit of data and token may be any single value such as keyword or operator.

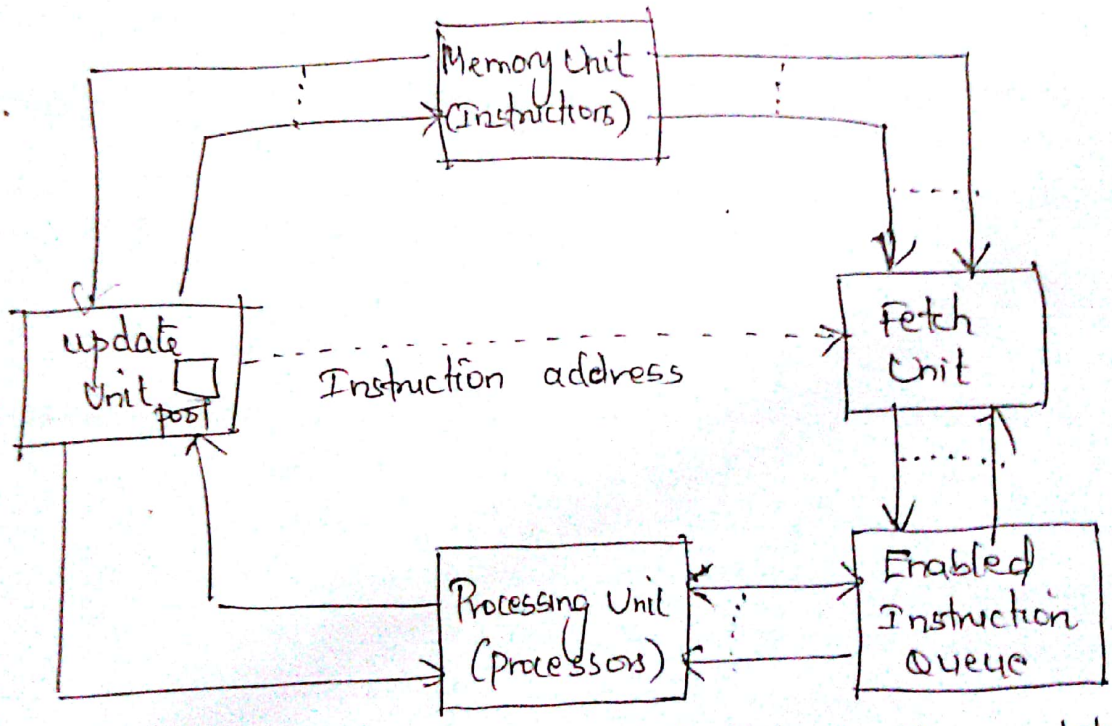
Scanned by CamScanner

STATIC DATAFLOW

In a von neuman machine, The instruction execution is based on the program flow control, whereas in Data flow it is based on the data availability.

* Data tokens are assumed to move along the arc of the dataflow program ~~prog~~ graph to the operator nodes
* Tokens not labelled
* Control tokens must be used to acknowledge the proper timing in transferring data tokens from node to node.

Static dataflow computer organization



* Data tokens are in the input pool of the update Unit
* Update Unit passes data tokens to their destination instruction in the memory unit.

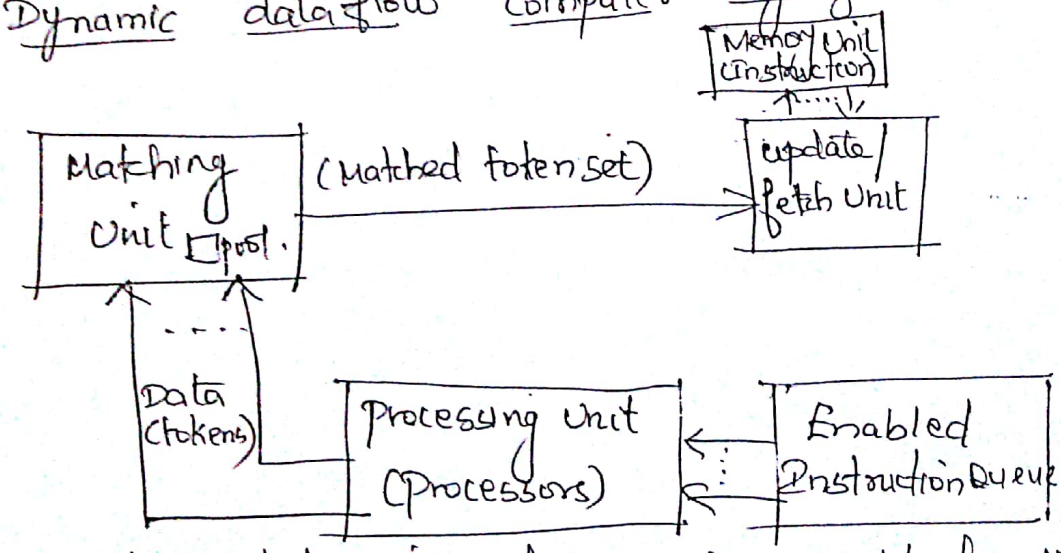
Scanned by CamScanner

* When an instruction receives all its required operand tokens, it is enabled & forwarded to the enabled queue.
* Fetch unit fetches these ~~the~~ instructions when they become enabled.

Dynamic Data flow

* Data tokens from the input pool of the matching unit
* The matching unit arranges data tokens into pairs or sets.
* Matched token sets are released to fetch-update unit.
* Each set of matched token is needed for one instruction execution

Dynamic dataflow computer organization



* Fetch-update unit forms the enabled instructions by merging the token sets with copies sent to their consumer instructions.
* Uses tagged tokens. More than one token can exist in an arc
* Data flow computers include the manchester machine developed by watson at the university of machester.

Scanned by CamScanner

Major design issues

1. Developing intelligent data driven mechanisms for either static or dynamic data flow machines
2. Efficient handling of complex data structures, such as arrays in a data flow environment.
3. Developing a memory hierarchy and memory allocation schemes for supporting data flow computations.
4. Decomposition of programs & assignment of program modules to data flow processors.

DATA FLOW GRAPH

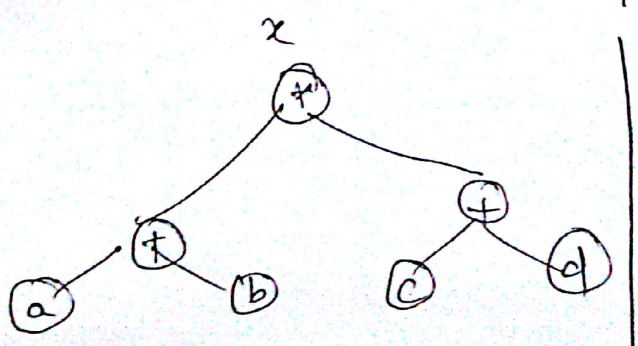
A data flow graph is a graph which represents a data dependencies between number of operations.

It is defined as a directed graph whose nodes correspond to operators and arcs are pointers for forwarding data tokens.

Data flow language

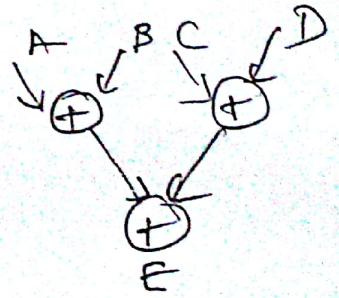
- Irvine dataflow (ID) language
- Value algorithmic language (VAL)

$x = (a+b) * (c+d)$ shown as expression tree.



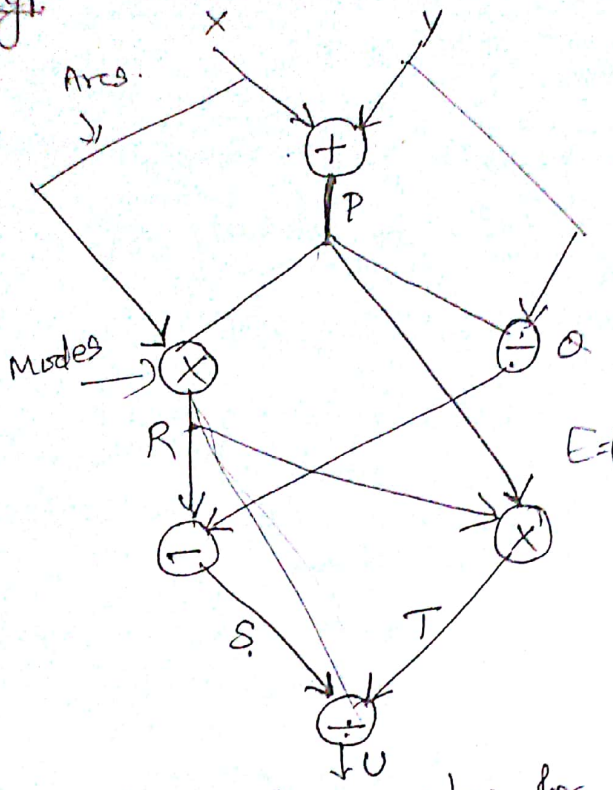
DFG

$E = (A+B) + (C+D)$



Scanned by CamScanner

Eg1:



$P = x + y$

$Q = P \div y$

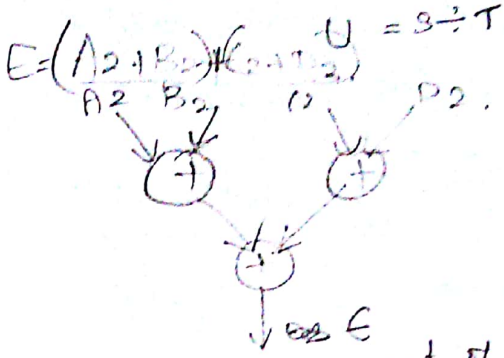
$R = x * P$

$S = P - Q$

$T = R * P$

$U = S \div T$

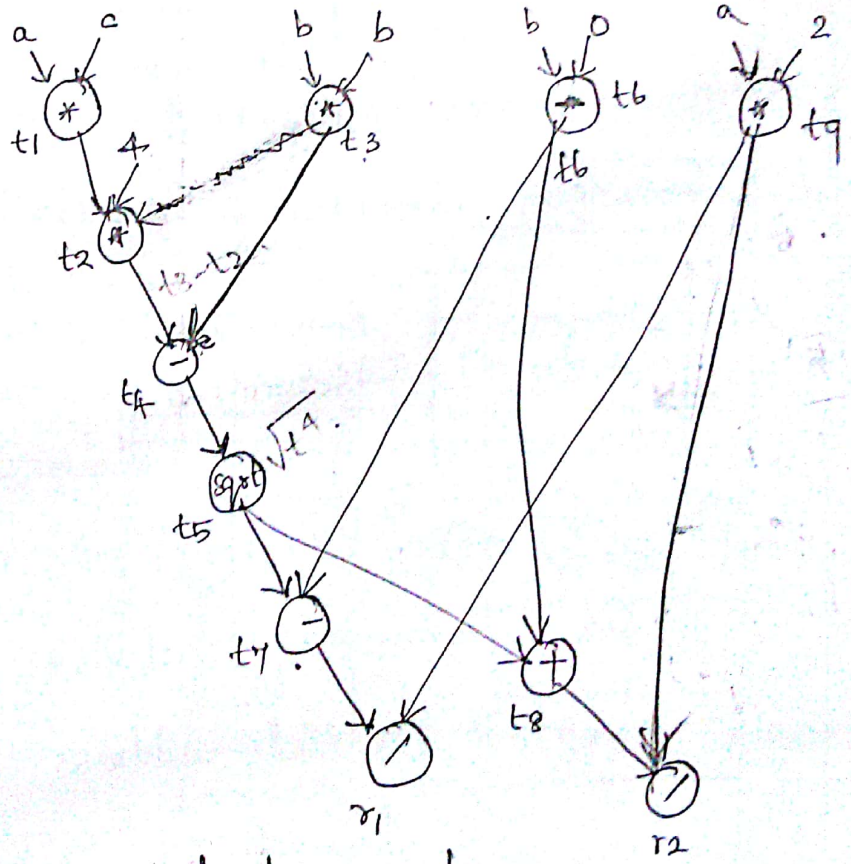
$E = (A2 + B2) * (C2 + D2)$



Eg2: consider the procedure for finding the root of a quadratic equation $r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

quad (a, b, c)

```
t1 = a*c;
t2 = 4*t1;
t3 = b*b;
t4 = t3-t2;
t5 = sqrt(t4);
t6 = -b;
t7 = t6-t5;
t8 = t6+t5;
t9 = 2*a;
r1 = t7/t9;
r2 = t8/t9;
```



Operator represented by node & operands represented as arc in graph.

Scanned by CamScanner

VECTOR PROCESSORS

* Vector processors provide high-level operations that work on vector linear array of numbers.
* It's consist of an ordinary pipelined scalar unit & a vector unit
* vector processor allow vector to deal with as floating point number, as integer.
* The vector processors are machines built primarily to handle large scientific & engineering calculations.

Applications of vector processor

- Long range weather forecasting
- Medical diagnosis
- Image processing

Instruction format for vector processor

operation code	Base address Source 1	Base address source 2	Base address destination	Vector length

Eg: C(1:100) = A(1:100) + B(1:100)

Vector instruction includes

* Initial address of the operands
* Length of vectors
* Operation to be performed.

Scanned by CamScanner

Basic Vector Architecture

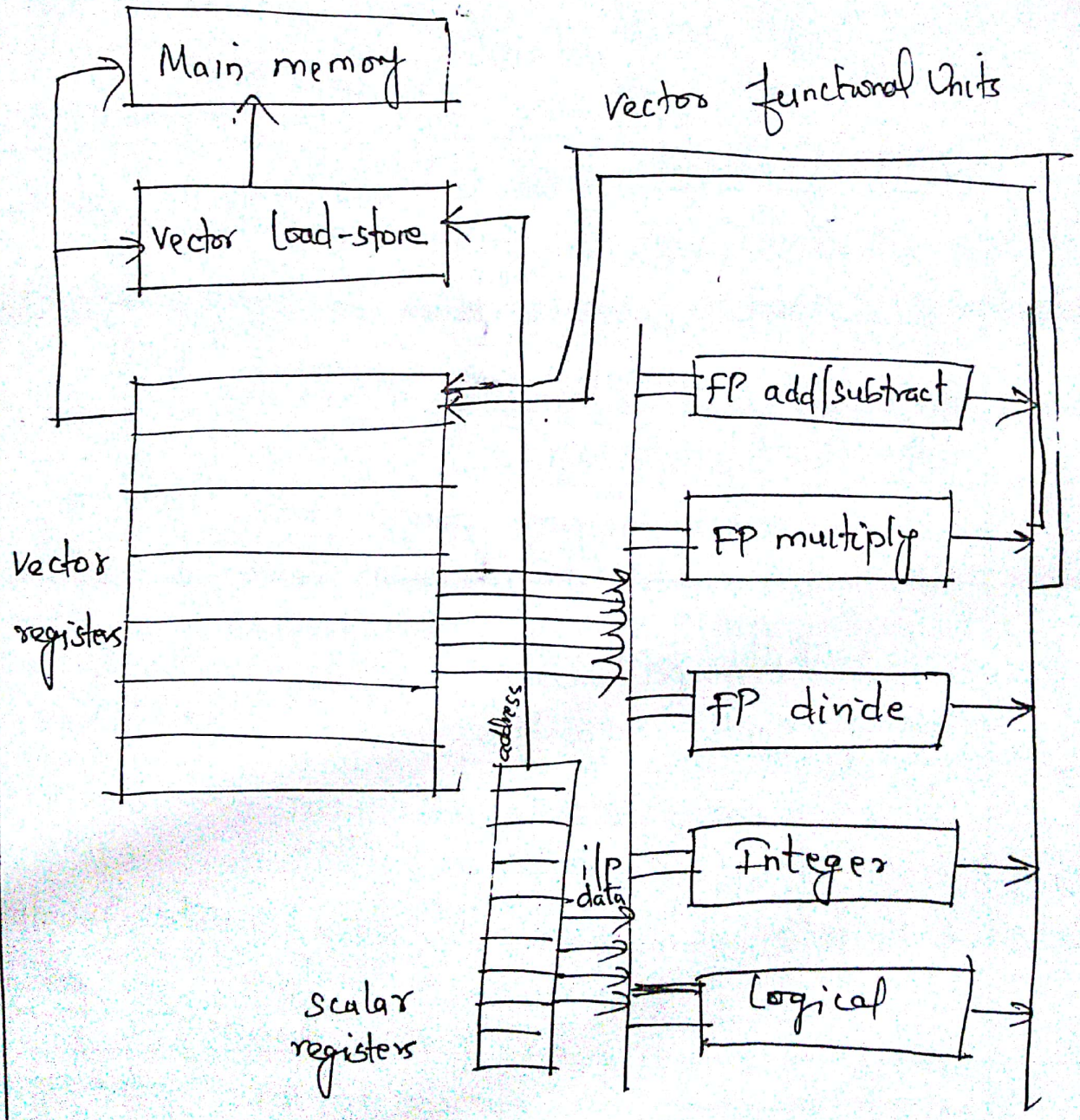
Two primary types of architecture.

→ vector – register processors.

→ memory – memory vector processors

Vector – Register processor

Basic structure of vector – register architecture



Scanned by CamScanner

* The above diagram shows the vector register and the multiple floating point ALU's.
* Data is read into the vector registers, which are FIFO queues capable of holding 50-100 floating point values.
* A machine will be provided with several vector registers.
* ~~The~~ Vector registers.
  - Each vector register is a fixed length bank holding a single vector
  - Each vector register must have two read ports & one write port.

Vector functional Units

- Each unit is fully pipelined and can start a new operation on every clock cycle.
- It contains five functional units.

Vector load/store Unit.

- Vector memory unit that loads or stores a vector to or from memory.
- Fully pipelined.

Set of Scalar registers

- provides data as input to the vector functional units.
- Compute address to pass to the vector load store unit.

. Load a vector register from a location in memory.
. perform operations on elements in the vector registers.
. Store data back into memory from the vector registers.

Scanned by CamScanner

Matrix multiplication

Matrix multiplication is the most computational operation performed in computers with vector processors.

Example:

multiplication of two 3x3 matrices, A & B.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}$$

$$C_{ij} = \sum_{k=1}^{3} a_{ik} \times b_{kj}$$

$$C_{11} = a_{11} b_{11} + a_{12} b_{21} + a_{13} b_{31}.$$

Characteristics of vector processing

Vector instructions classified into 4 types.

$f_1 : V \rightarrow V$ — V → denotes vector operand

$f_2 : V \rightarrow S$ — S → denotes scalar operand.

$f_3 : V \times V \rightarrow V$ — f_1, f_2 → unary operations.

$f_4 : V \times S \rightarrow V$ — f_3, f_4 → Binary operations.

Some operations with code.

Type	Mnemonic	Description
f_1	VSQR	Vector Square Root, $B(I) \leftarrow \sqrt{A(I)}$
f_2	VSUM	Vector Summation, $S = \sum_{i=1}^{N} A(I)$.
f_3	VADD	Vector Add $C(I) = A(I) + B(I)$
f_4	S ADD	Vector Scalar Add, $B(I) = S + A(I)$.

Scanned by CamScanner

CRAY-I VECTOR PROCESSOR

* The Cray-I was a super computer designed, manufactured & marketed by Cray Research. It was the first supercomputer to successfully implement the vector processor design.
* These system improve the performance of math operations by arranging memory & registers to quickly perform a single operation on large set of data.

Need of Supercomputer:

* When a computer application requires a very large amount of computation to be completed in a reasonable amount of time, we must use machines with large computing capacity. Such machines are called supercomputers, designed by Seymous Cray.
* In computational applications, supercomputers are needed to handle vectors of data, where vector is a linear array of numbers.
* For eg, an add operation may generate a vector that is the element by element sum of 2 64-element vectors.
* A single memory access operation can cause an entire vector to be transferred b/w main memory & processor register.
* Supercomputers have been marketed by companies such as Cray Research (Cray-I), (Cray-2).
* The main drawback of such machines has been their high cost.
* In the foll. diagram, vectors are contained in a set of eight V registers, each capable of holding 64-elements & a typical vector instruction causes set of operands to be taken from 2 V registers & results returned to third.

Scanned by CamScanner

* It's a most well known vector processor, released in 1976. (Used to operations in parallel)
* Fastest super computer in the late 1970's.
* It has 32 bit instruction length.
* Architecture consists of 3 sections:
  * Main memory
  * The scalar subsystem
  * The vector subsystem.

Scalar: Instruction buffer, 2 file scalar registers, 2 address function register, Scalar floating point functional Units.

Vector: 8 Vector registers, set of 3 vector functional units, Shared set of 3 floating point functional Units.

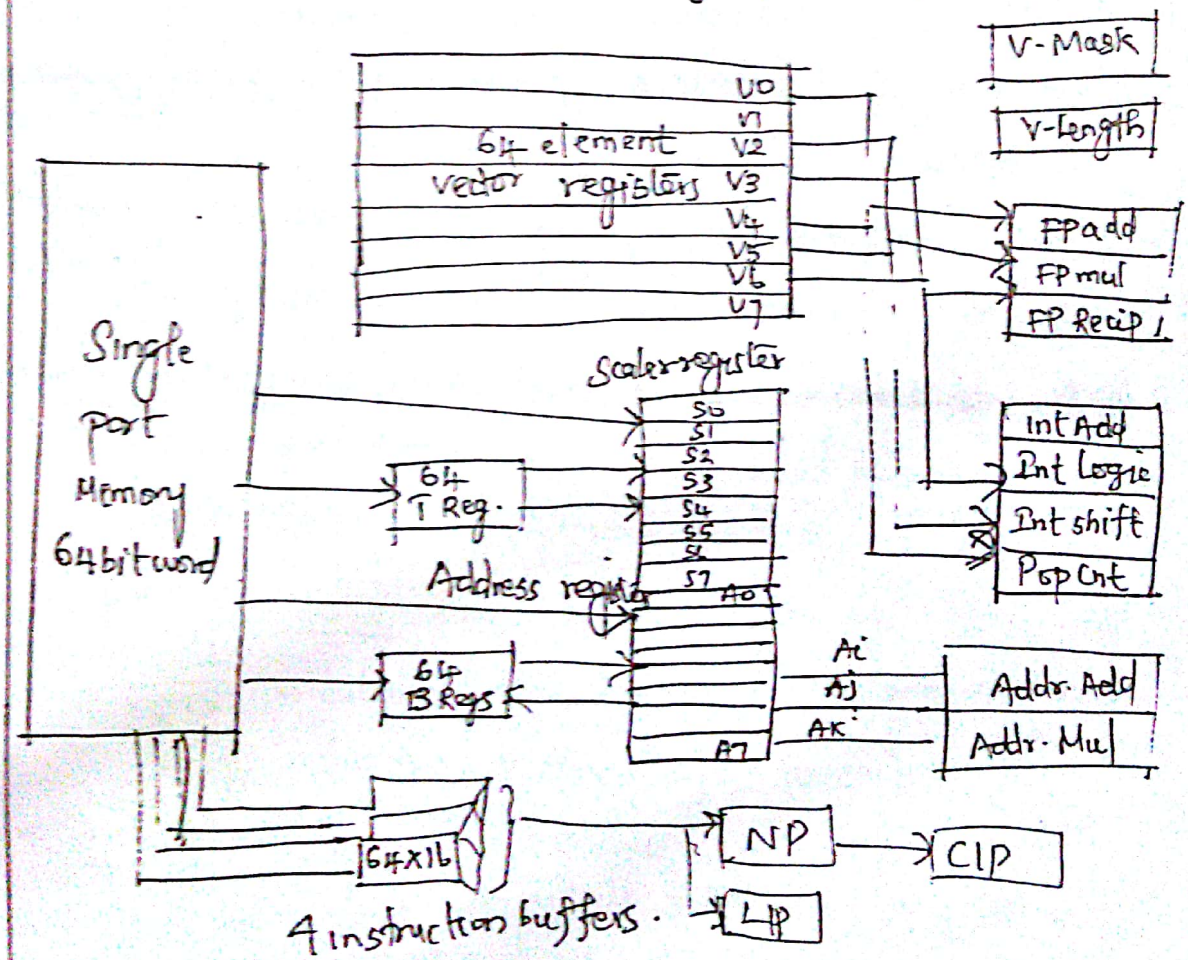


Figure: Cray-1 Vector Computers.

Memory bank cycle 50 ns, Processor cycle 12.5 ns (80 MHz)

Scanned by CamScanner

EPIC (IA64).

Explicitly Parallel Instruction Computing) is a 64-bit microprocessor instruction set, jointly defined & designed by Hewlett packard & Intel, that provides upto 128 general & floating point unit registers & uses loading, predication & explicit parallelism to accomplish its computing tasks. Current 32-bit CISC & RISC microprocessor architecture depend on 32-bit registers, branch prediction, memory latency.

IA64 Architecture

* Developed jointly by Intel & HP.
* First IA64 Processor is the Itanium
* IA64 is a Load/Store arch. with 64-bit memory addresses & registers.

- EPIC permits microprocessor to execute several software instructions in parallel by using the compiler.

In 1989, researches recognized that RISC reaching a limit at one instruction/cycle. $-b+sqrt(b*b-4*a*c)/2*a$

- So new architecture invented, it is called EPIC.
- The basis for the research was VLIW (Very long Instruction word), in which multiple operations are encoded in every instruction & then processed by multiple execution Units.
- EPIC refers to architecture in which features are provided to facilitate compiler enhancements of ILP in all programs, while keeping hardware complexity relatively low.

Scanned by CamScanner

In the EPIC processor, the following features are enabled.

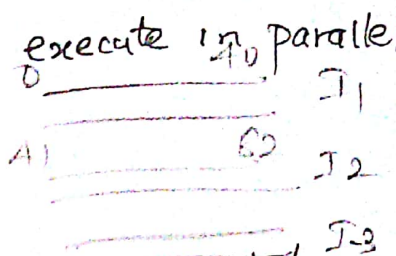
- Groups & Bundles

Instructions are divided into groups & bundles.

Bundle

* Group of three instructions packed into a 128-bit word
* Includes Three 41-bit fields for instructions and one 5 bit template slot.
* Instructions of the bundle are execute in parallel by different functional Units.

41
3
128 − 123
123
5



Group

* Sequence of instructions that are executed in parallel provided the following 2 conditions are fulfilled.

Template

Indicates the types of instructions in a bundle.

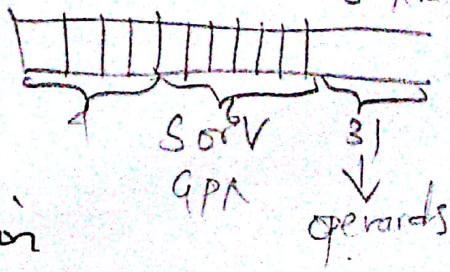
Options

PUSH, POP, Loop

I → ALU & non ALU Integer
M → ALU & memory (load/store)
F → Floating point
B → Branches. (Conds)
L+X → Extended.

Instructions

41 - 10:31 (4 operands & operator)



→ 41 bits in the instruction.
→ 1st 4 bits – major opcode (odd
→ Some bits – Indicate operation
→ Final 6 bits – predicate register
→ Remaining bits specifying the operands.

Scanned by CamScanner

Predication:

→ Way of dealing with the conditional branch.
→ Each instruction is conditional.
→ Executed only when a specific condition or predicate is true.

Features of EPIC architecture

→ Increase the speculative execution of instruction.
→ Delayed exception
→ Multiway branch instructions.
→ Very large architecture register file to avoid the need for register renaming.
→ Explicitly encode multiple operation per instruction
→ Instruction level parallelism
→ Evolved from VLIW architecture.

C. GEETHA
AP/CSE
BHARATH UNIVE

Scanned by CamScanner