4 Jan 2019
Friday

Computer Organisation

⇒ Two types of computers
- General purpose
- Embedded system

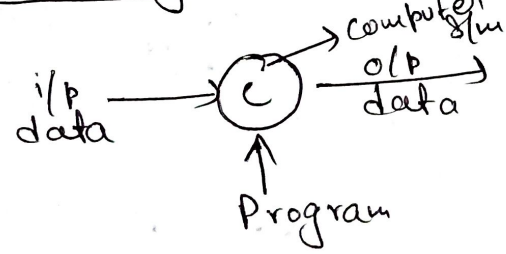
⇒ Memory
⇒ Register
⇒ ALU (Arithmatic Logic Unit)

Notes

Three kinds of Memory:
- ↳ Long term memory ROM
- ↳ Short term memory SRAM → flip flop memory
- ↳ DRAM → middle term memory
  - ↳ when the power is off, data is present until capacitor gets discharged.

Computer System →
[It converts one form of data into another form] under the control of program.

Objective of Computer Sys → Execution of a program



* Analog signal means computer waves.
* Digital signal means discrete waves

Analog

Digital

* Signals understand only 2 values → 0 & 1
* Computer understands only digital signals, it understand sequence of 0's & 1's.
* Language understood by computer is Binary Language.

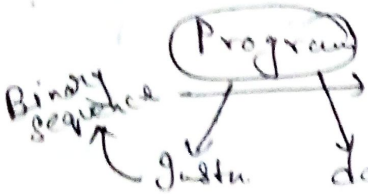
* The digital signals consist of 2 values as 0 & 1 called 0's & 1's

* The no. system whose radix is 2 called binary no. system so ultimately computer understands Binary Language.

CPU understands only Binary Language. As in our computer, ASCII is a translator to convert one form to another such as keyboard & CPU

Program → Sequence of instructions along with data

a+b → Identifying the instruction is not meaningful operation. Processing is a main key for getting meaningful data.



① Instruction
② data

Def① Instruction is a binary sequence / pattern (1's & 0's pattern) which is designed for processor to perform some task.

Data → Binary sequence associated with a value.

→ While operating the instruction, how the processor come to know about task.
☆ Program stored in → memory → address.

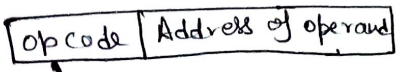
→ How our processor recognise the operations associated with respect to binary pattern.
☆ It is done by instn format

Instruction format → Layout of Instn
↳ Depend upon size of Instn

Internal structure of Instn

Before giving the layout of Instn, we need to give the size of Instn / length of Instn.

Instruction format



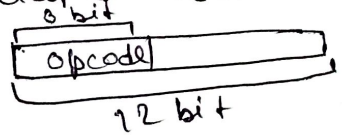
opcode → Type of task

→ How many operations introduced?

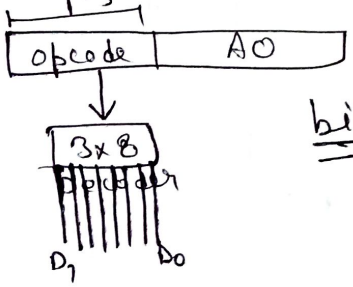
Encoding If there are 2^n combinations, they are delivered by n bits. (o/p)

Decoding If there are n bits, then it is defined by 2^n bits (o/p).

Eg-2 Instructions are encoded binary pattern which is associated with unique operations.



Q) How the length of opcode is restricted?
The length of the opcode field based on no. of operations supported by processor.



binding means meaning associated with symbol.
↳ association between signal and operation

Signal	operation
D_0	+
D_1	−
D_2	*

At the time of designing, the designer responsibility.

Computer Architecture:

It is concerned with the structure and behaviour of the various functional modules of the computer and how they interact to provide processing needs of the user.

Computer Organisation

It is concerned with the way the hardware components are connected together to form a computer system.

Ch- Register Transfer & Microoperations

Register Transfer Language
Register Transfer
Bus & Memory Transfer
Arithmetic microoperations
Logic microoperations
shift microoperations.
Arithmetic logic shift unit.

Register Transfer Language.

o Microoperations

Operations executed on data stored in registers are called microoperations.

A microoperation is an elementary operation performed on information stored in one or more registers. Result of operation may replace the previous binary info of a register or may be transferred to another register.

eg of microoperations are shift, count, clear and load.

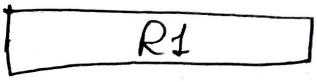
The internal hardware organisation of a digital computer is best defined by specifying:

1) The set of register it contain & their fun.
2) Sequence of microoperations performed on binary info stored in registers.
3) Control that initiates the seq of microoperations.

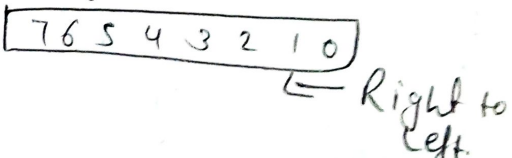
The symbolic notation used to describe the microoperation transfers among registers is called a register tranfer language.
It is convenient tool for describing the internal organisation of digital computers in concise & precise manner. It can also be used to facilitate the design process of digital systems.

Register Transfer

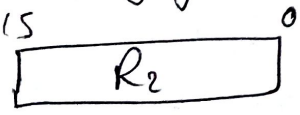
(a) Register R



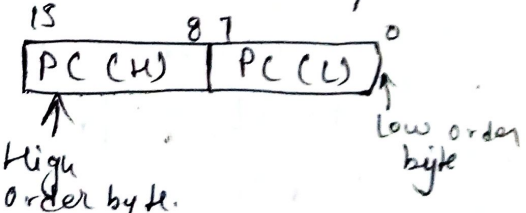
(b) Showing individual bits.



(c) Numbering of bits.



(d) Divided into two parts



Info tranfer from one register to another is designated in symbolic form by means of replacement operator.

R2 ← R1

denotes a transfer of the content of R1 into register R2.

Under a predetermined control condition

eg:- if (P=1) then (R2 ← R1)

Symbol:- P: R2 ← R1

MAR → Memory access register.

Symbol	Description	Examples
Letters (And numerals)	Denotes a register	MAR, R2
Parentheses ()	Denotes a part of register	R2(0-7), R2(L)
Arrow (←)	Denotes transfer of info	R2 ← R1
Comma,	Separates two microoperations	R2 ← R1, R1 ← R2

Bus & Memory Transfers

A digital computer has many registers, and rather than connecting wires between all registers to transfer information between them, a common bus is used.

Bus is a path over which info is transferred from any of several sources to any of several destinations.

From a register to Bus: Bus ← R

BUS ← R

The transfer from bus to register R1 ← BUS

The content of selected register is placed on the BUS and the content of the bus is loaded into register R1 by activating its load control input

Memory Transfer

The transfer of info from memory to outside world i.e. I/O interface is called a read operation. The transfer of new info to be stored in memory is called a write operation. These kind of transfers are achieved via a system bus. It is necessary to supply the address of the memory location for memory transfer operations.

Memory Read

The memory unit receives the address from a register, called the memory address register designated by MAR. The data is transferred to another register, called the data register designated by DR. The read operation can be stated as:

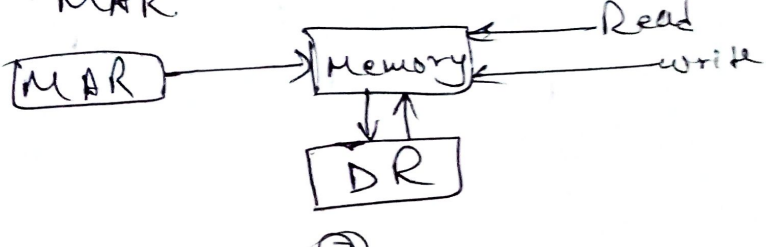
Read: DR ← [MAR]

Memory write

The memory write operation transfers the content of a data register to a memory word M. Selected by the address. Assume that the data of Register R1 is to be written to the memory at the address provided in MAR.

Write: [MAR] ← R1.

Note: Location pointed by MAR is written not MAR



Arithmetic Microoperations

The basic arithmetic operation are addition, subtraction, complement, increment and decrement. Some symbolic representation of arithmetic microoperat... is

Operation	Symbol
Addition	$C \leftarrow A+B$
Subtraction	$C \leftarrow A-B$
1's complement	$C \leftarrow \overline{B}$
2's complement	$C \leftarrow \overline{B}+1$
Increment	$A \leftarrow A+1$
Decrement	$B \leftarrow B-1$

1011 - 11
0100
+1
0101 - 25

Multiplication and division are valid operation but it is not the basic microoperation.

Logic Microoperations

A group of bits called strings is stored in the register, the logic microoperation consider each bit of register as compared to the arithmetic microoperation where we take whole content of register to perform arithmetic operation.

Symbol for AND gate $C \leftarrow A \wedge B$
OR gate $C \leftarrow A \vee B$

Shift microoperation

Shift microoperation transfer binary bits blw the register serially, this shift operation is also used to perform binary multiplication & binary division.

Register can be shift to left or to right. There are no conventional symbol for shift operation. The shift microoperation are represented by shl & shr to shift the content to left & to right.

The one bit shift is represented by symbolic representation & is given in form of a statement.

$A \leftarrow shl\ B$ — (1)
$C \leftarrow shr\ A$ — (2)

The first statement specifies as one bit shift to the left of register B & second statement specifies one bit shift to right of register A.

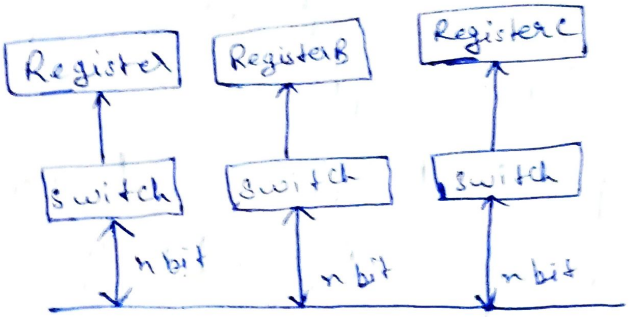
Shift microoperation		Symbol	Comments
Logical	Shift left	$A \leftarrow shl\ A$	One bit shift to left of A
	Shift right	$B \leftarrow shr\ B$	One bit shift to right in B
Circular	~~Cir~~ circular shift left	$C \leftarrow cil\ C$	circular shift left of Register 'C' and result transfer to same register
	Circular shift right	$C \leftarrow cir\ C$	circular shift right of Register 'C' and result transfer to same register
Arithmetic	Arithmetic shift left	$B \leftarrow ashl\ B$	ASL of B
	Arithmetic shift right	$C \leftarrow ashr\ C$	A.S.R of C.

Jan

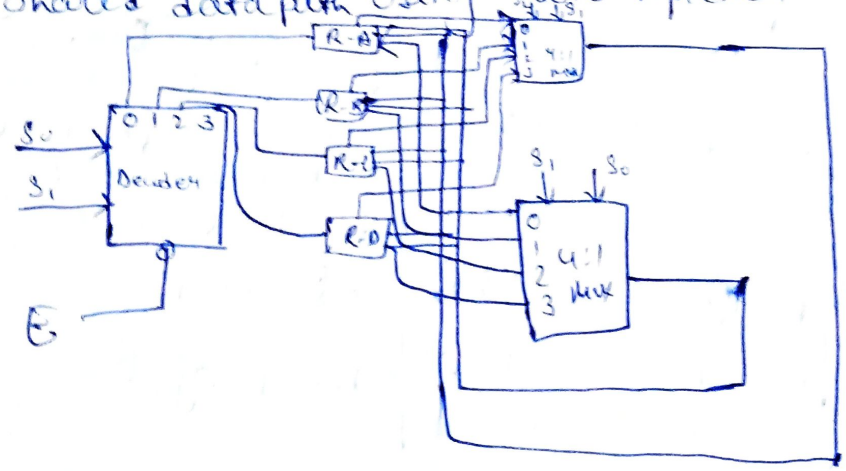
Bus Transfer:

A node efficient for transferring blw registers in a multiple register configuration is a common bus system. It is shared by all the units. Switch are required to enable path to be shared. These switches are implemented by multiplexer or tristate non-inverting buffer.

One common control signal is used to control all the switches, the switches are enable or disable.



Shared data path using multiplexer.

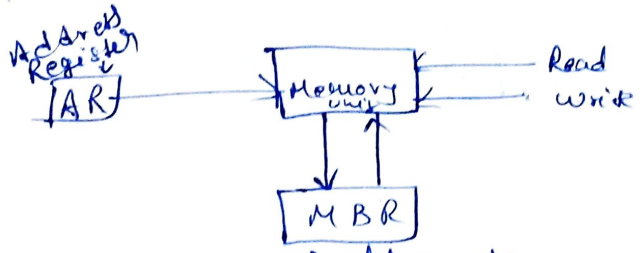


Memory Transfer.

The basic operation of memory Transfer is fetch and store. The fetch operation transfer a copy of contents from memory location to C.P.U.

The word in memory remains unchanged. The Store operation transfer the word information from CPU to a specific memory allocation. Destroying previous content of location.

The memory word is symbolised by letter 'M'. The required information is selected from the memory location by address. If is stored in the address register by AR [Address Register] The data transfer to another register called data register.



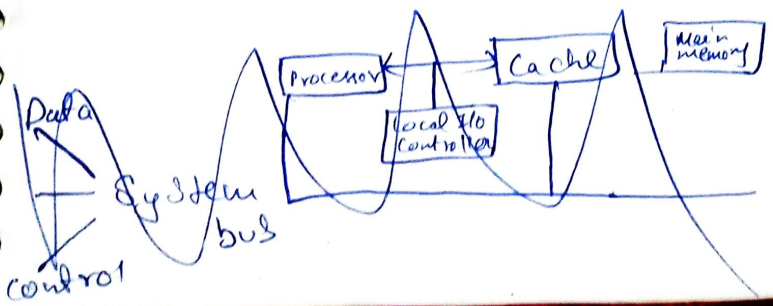
Buffer Memory Register

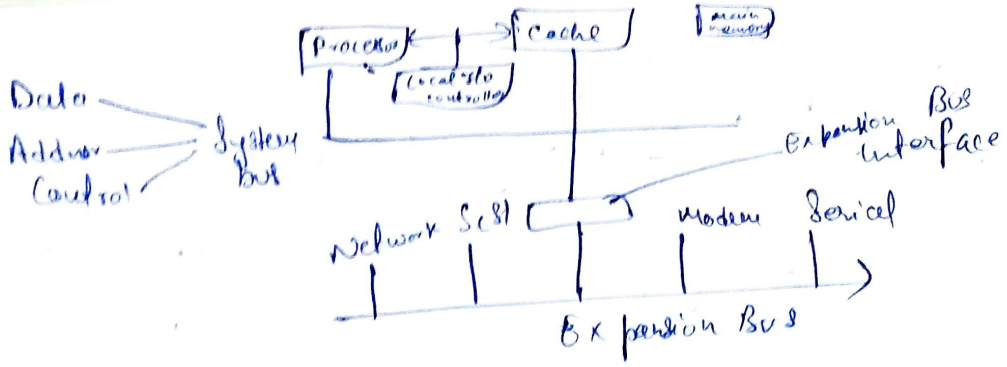
Read DR ← M[AR]
Write M[AR] ← B

Type of Bus

1. Data Bus
2. Address Bus
3. Control Bus

Multiple bus Hierarchy

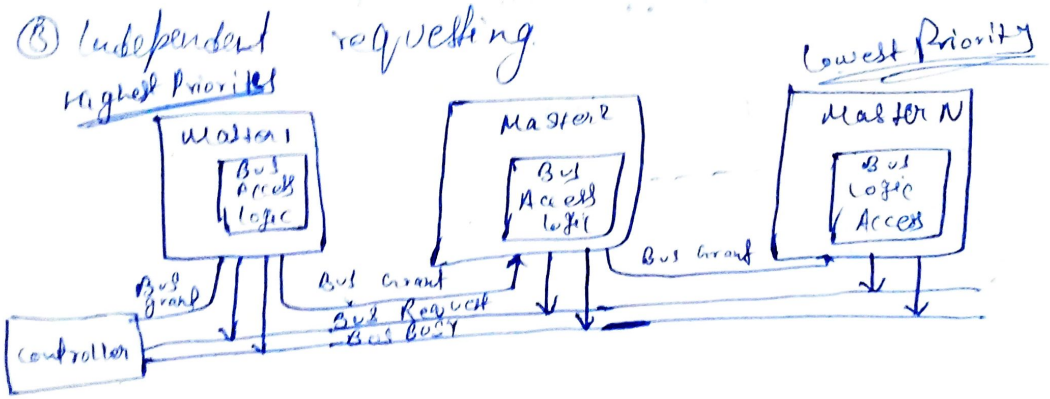




Bus Arbitration:

A mechanism which decides the selection of correct master to access bus is known as bus arbitration

There are three different mechanism commonly used for this
① Daisy chaining.
② Parallel arbitration
③ Independent requesting



Bus

⇒ Specifications:

i) LOGICAL - sequence of actions
ii) ELECTRICAL.
iii) Mechanical

① Data Bus:
The most common bus is the data bus. A data bus carries data.
- It is an electrical path that connects the CPU, memory, Input/output devices & Secondary Storage devices.
- The Bus contain parallel group of lines.
- The number of lines in bus affects the speed at which the data travels blw different components.

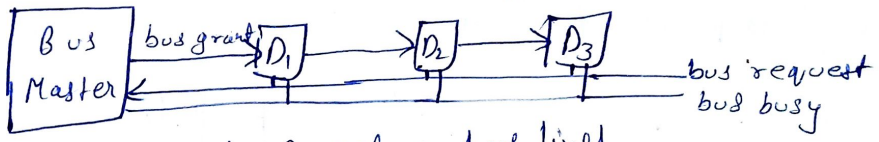
② Address Bus
An address Bus carries address information, It is set of wires similar to the data bus but it only connects CPU & memory.
- Whenever the processor needs data from the memory, it places the address of data on the address bus.
- The address is carried to the memory where the data from the requested address is fetched & placed on the data bus the data bus carries to CPU.

Morris Mano

③ Control Bus.
The control Bus carries control information from the control unit to the other units.
- The control information is used for directing the activities of all units.
- The control unit control the functioning of other units eg: input/output devices, Secondary Storage e.t.c.

o Daisy chain Bus Arbitration.

deciding who gets accessed to the bus for driving the transaction.



simply only 3 extra bus lines

- bus grant
- bus request
- bus busy

Disadvantage

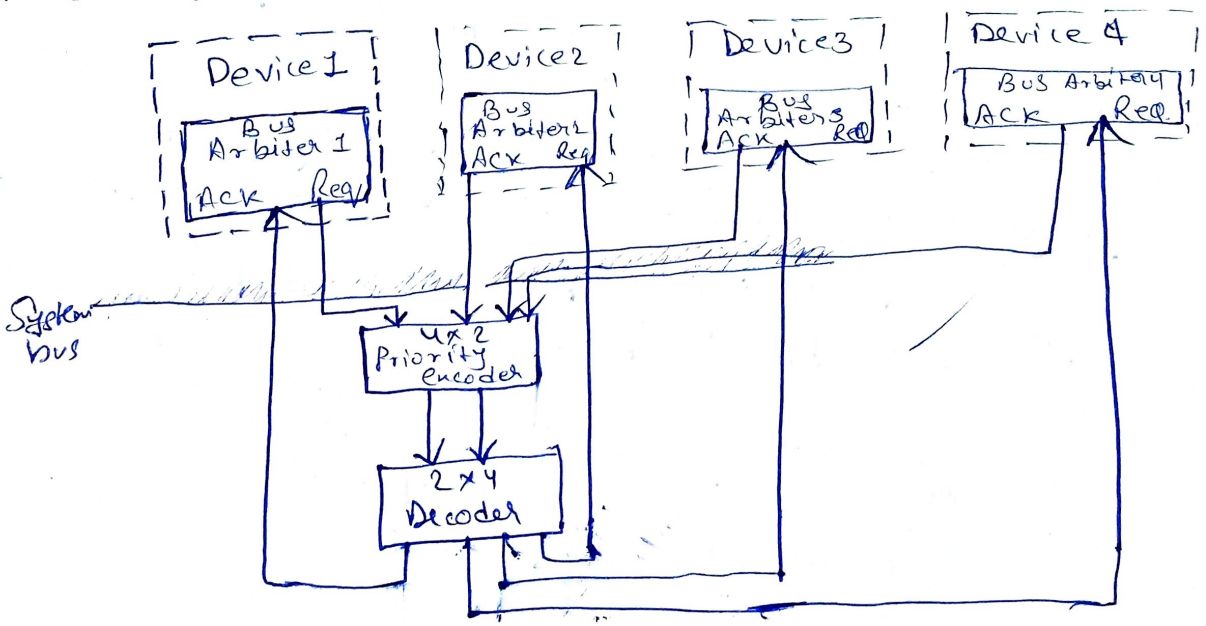
1) Hardware priority
2) Poor fault tolerance

My notes

Computer Architecture Organization

11 Jan 2019

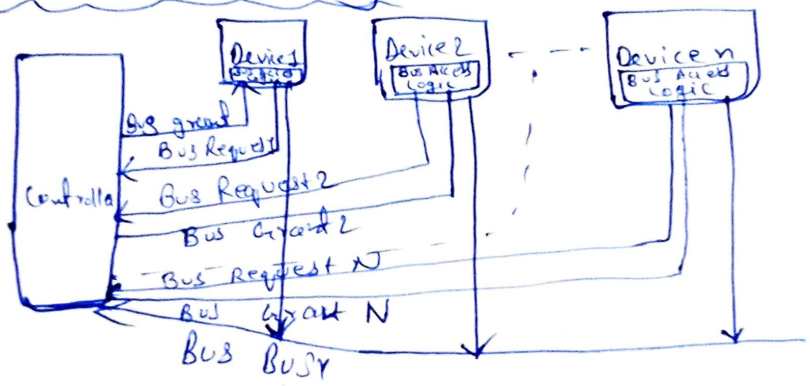
Parallel Arbitration:



Encoder - Priority
Decoder - grant

The output of the encoder generate a 2-bit code which represent the highest priority unit among those requesting the bus.
This 2-bit code is given to the input of 2 X 4 decoder which enables the proper acknowlegdement to grant bus access to the highest priority unit.
A device is utillized in the system only after it recieve the acknowlegded signal.

Independent Priority



Data Representation

⇒ Processor Design.

C.P.U

Register

↳ General Purpose Register (File)
↳ Accumelator Register
↳ Index Register (Address + ~~Register~~ Data)
↳ Valid Area Register
↳ Data Register
↳ Address Register

Prof. Morris Anno

Computer Architecture	Computer Organization
It refers to those attributes of a system that are visible to the programmer and have a direct impact on logical execution of a program.	It refers to the operational units and the interconnection between them that achieve the architectural specifications.
• Low level concepts/logical units	• High level concepts.
• Concepts that the programmer deals with directly.	• transparent from programmer.
• defines the system in an abstract manner.	• Realisation of the abstract model.
• 'what' does the system do/ what functionality is being provided.	• 'How' to implement.
• Types of instruction, addition modes.	• Physical components like circuits with adder, subtractor e.t.c.

Basic components of a computer.

① Central Processing Unit (CPU).

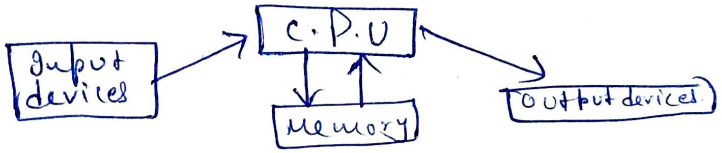
↳ Control unit: has a set of registers and circuits to generate control signals.

↳ Arithmetic logic unit: responsible for executing arithmetic and logic operations.

② Main Memory / Memory
↳ A place to store data and instructions temporarily

③ I/O components.
↳ Input module: consists of some basic components that accept data & instructions in some form and convert them into an internal form of signals usable by system.

↳ Output module:
It serves as a medium for displaying the results.



The von Neumann Architecture:

Almost all computers designs are based on the concepts developed by John Von Neumann. Such designs are reffered to as Von Neumann Architecture.

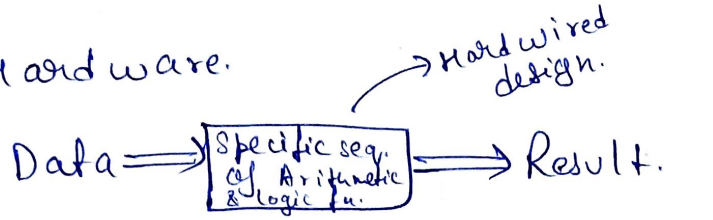
The 3 main concepts:-

① Data and instructions are stored in a single read-write memory.

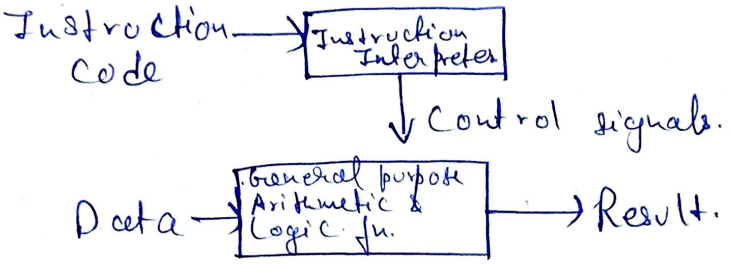
② The contents of this memory are addressable by location irrespective of the type of data or instruction present there.

③ Execution occurs in a sequential manner from one instruction to the next, unless changed explicitly.

(A) Hardware.



(B) Software.



Interaction Between various computer components.

Exchange of data takes place b/w memory & cpu. The CPU uses 2 of its internal registers for this purpose.

1) Memory Address Register (MAR).
   ↳ Specifies the address in the memory for next Read/write.

2) Memory Buffer Register (MBR)
   Contains data to be written in memory/ recieves data read from the memory.

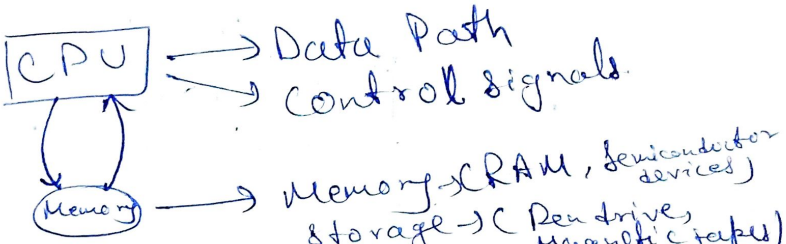
I/O Address Register:
   ↳ Specifies the particular I/O device to interact with CPU.

I/O Buffer Register:
facilitates/enable exchange of data between I/O modules & processor.

Memory
- ↳ set of locations.
- ↳ each location is addressed sequentially.
  - ↳ Data or instruction.

NPTEL LEC-1

I/O → Man machine Interface



Organization : architecture
↓ ↓
users' designers'
eg) Multiplier how implemented?
Instruction set
Programmers' : Designers'
SOFTWARE : HARDWARE

APPLICATION

(Microprocessor)
CONTROL (C.E) : DATA PROCESSING (C.S)
Special purpose : General Purpose.
ROM : RAM
Hardware Oriented : Software

Disadvantages of Direct mapping :

Unit 4
Processor Design

* Stack Organisation:
  i) Stack
  ii) Stack pointer
  iii) Push & POP
* Register Stack
* Memory Stack.

=> Instruction format in processor
-> Instruction: It is a command to processer on specifying data.
Each Instruction has two parts:
① Task to perform
② Data to be operated upon is called operand

The purpose of an instruction is to specify both an operation to be carried out by CPU or also the process of set of data used in operation.

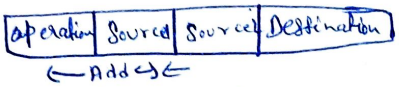
A computer must have instruction performing fours type of operation.
1) Data transfer between the memory and the CPU register.
2) Arithmetic and logic operation on data.
3) Program sequencing
4) Input Output transfer.

Instruction has two parts:
① operation code (Opcode)
② Operand.

~~Instruction~~

Instruction format is categorised:

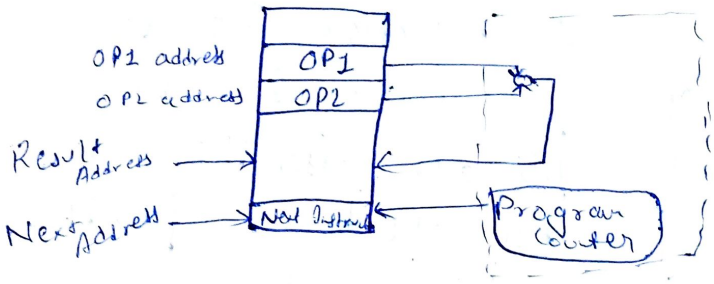
1) Three address
2) Two address
3) One address
4) zero address



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

If n-bits are needed to specify the address of each operand, the imported form of the must contain three n-bit for address existing purpose.

Add	A, B, R1	$R_1 \leftarrow M[A] + M[B]$
Add	C, D, R2	$R_2 \leftarrow M[C] + M[D]$
MUX	R1, R2	$X \leftarrow R_1 + R_2$.



Two address

move	A, R1	$R_1 \leftarrow M[A]$
Add	B, R1	$R_1 \leftarrow R_1 + M[B]$
Mov	C, R2	$R_2 \leftarrow M[C]$
Add	D, R2	$R_2 \leftarrow R_2 + M[D]$
MUL	R_1, R_2	$R_2 \leftarrow R_1 * R_2$
MOV	R_2, X	$X \leftarrow R_2$

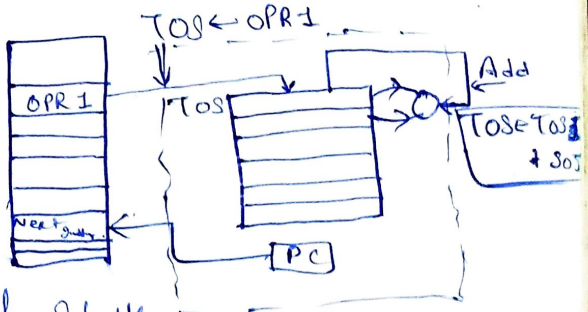
One address Instruction. — Accumulator Register

Load	A	AC ← M[A]
ADD	B	AC ← AC + M[B]
Store	R1	M[R1] ← AC
Load	C	AC ← M[C]
ADD	D	AC ← AC + M[D]
Mull	R1	AC ← AC * M[C]
Store	X	M[X] ← AC

Zero address Instruction:-
The push down stack in the CPU allows A.L.U instruction with no address.
Operands are push on to the stack from memory and ALU operations implicitly operate on the top of the stack.
eg- The simple addition is performed as follows using zero address stack instruction.

Opr 3 ← opr1 + opr2

PUSH	opr1
PUSH	opr2
ADD	
POP	opr3



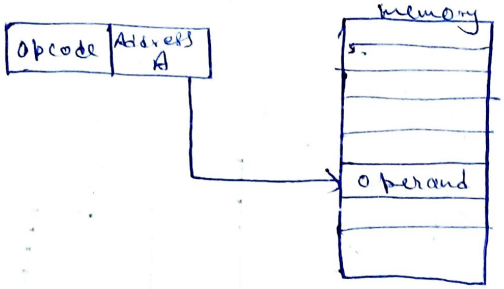
Assignment
1. Explain the concept of stack Organisation?
2. Write the difference b/w RISC and CISC
3. Explain any one method of bus arbitration with an application.

Addressing mode

The operation field of an instruction specify the operation to be performed. This operation must be execute on some data stored in computer register or memory work. The various method of addressing mode:-

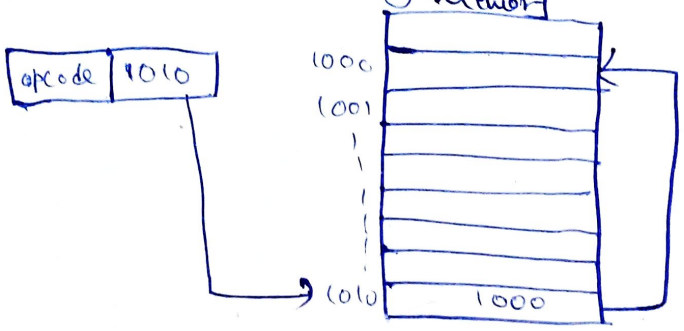
① Register mode:-
In this mode general purpose register store the data temporary. In this mode the operands are kept in register that resides in CPU. If content is loaded to select register the old content replaced by new content.

② Direct mode of addressing:-



In this mode, the operand is in memory location. The address of location is given in instruction. The operand resides in memory location and it's address is given directly by Address field of instruction.

③ Indirect addressing mode:-



In this mode the address field of instruction where the effective address is stored in memory. Control fetch the instruction from memory And use its address part to access memory again to read the effective address.

Effective address = Address part of instruction + Content

= Add[R] + value

(Class)

Auto increment mode.

In this mode the effective address of operand is the content of register specified in instruction.

After accessing the operand the content of this register are incremented to point to the next item in the list.

The auto increment mode by putting the specified register in paranthesis, to show that the content of register are used as effective address. followed by '+' sign do indicate that these content are to be incremented after the operand is access.

Auto Decrement mode.

In this mode the content of register specify in instructions are decrement.

These content are used as effective address of operand.

The auto decrement mode by putting specified register in paranthesis (preceded by '-' sign). to indicate that content of register are to be decrement before we use as the effective address.

Immediate mode.

In this mode required data is directly move to required register or memory location.

The data to be moved to a register or moving location in this mode is provided as part of instruction itself. The immediate operand can only be the source operand.

"Address is available as part of instruction itself."

"Register not hold the address but instructions as a address".

is moved to Register B, when the new data is loaded in the register memory location as mentioned in the instruction.

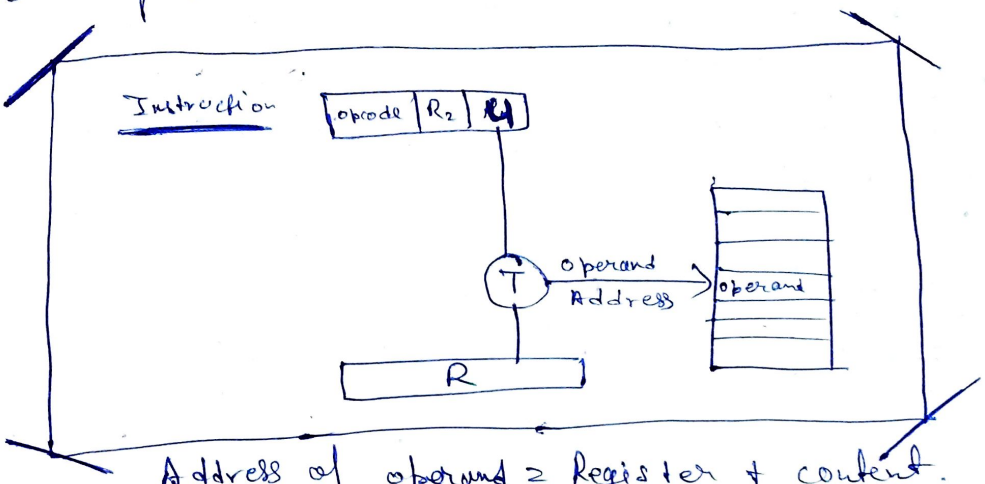
Index addressing mode.

The effective address of operand is generated by a constant value to the content of a register. The register used may be either a special register for this purpose or more commonly, it may be anyone of a set of general purpose register in the CL. It is referred as an index register as X(R). where, 'x' denotes a constant and R is the name of register involved.

The effective address is an operand.

$$A_{eff} = X + (R)$$

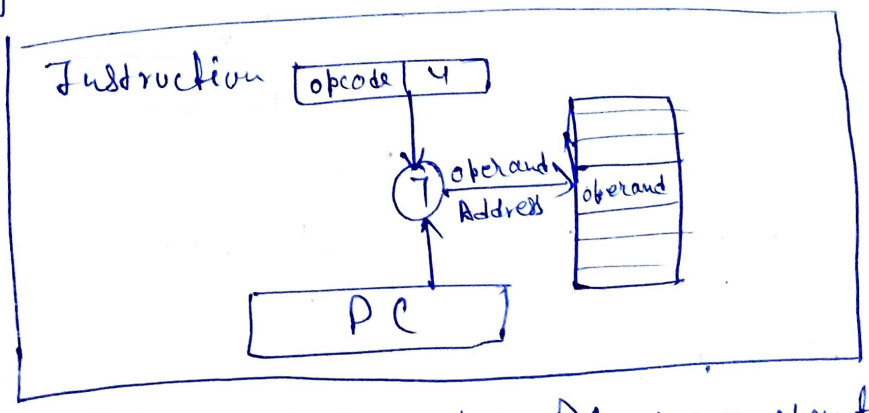
That addressing modes also known as base mode or displacement mode.



Address of operand = Register + content.

#Relative addressing mode.

It is similar to Index but the base address is held in the program counter Rather than in another register. This allow the storage of memory operand at a fix offset from the current instruction.



Address of operand = P.C + constant

PC = Program Counter

Why Addressing modes are used?

① To give programming versatility to the user by providing such facilities as pointers to memory, counters to loop control, indexing of data and program relocation.

② To reduce the no. of bits in addressing field of instruction

Program Counter (PC)

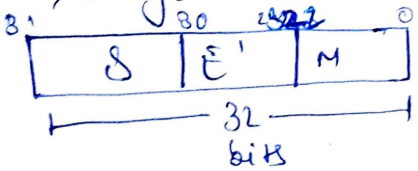
is the register that keeps track of the instructions in program stored in memory.

Effective address = address part of instruction (a signed no. [2's complement]) + content of CPU register

Data Representation.

IEEE standards of floating point representations

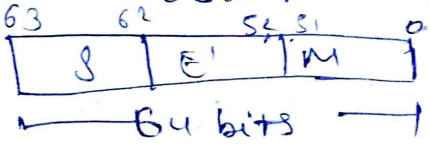
↳ Single Precision



M = mantissa (23 bits)
E' = signed components
S = signed bit
E' = E + 127

$E' = E + Bias$

↳ Double Precision



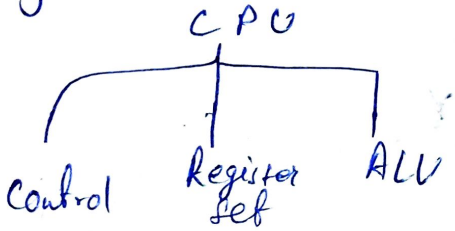
M = mantissa (52 bits)
E' = E + 1023

Ch- Central Processing Unit

* Introduction
* Register Organization
* Stack organization
* Instruction formats
* Addressing modes
* Data transfer & manipulation
* Program Control
* Reduced Instruction Set computer (RISC)

Introduction

CPU: The part of computer that performs bulk of data-processing operations is called central processing unit.

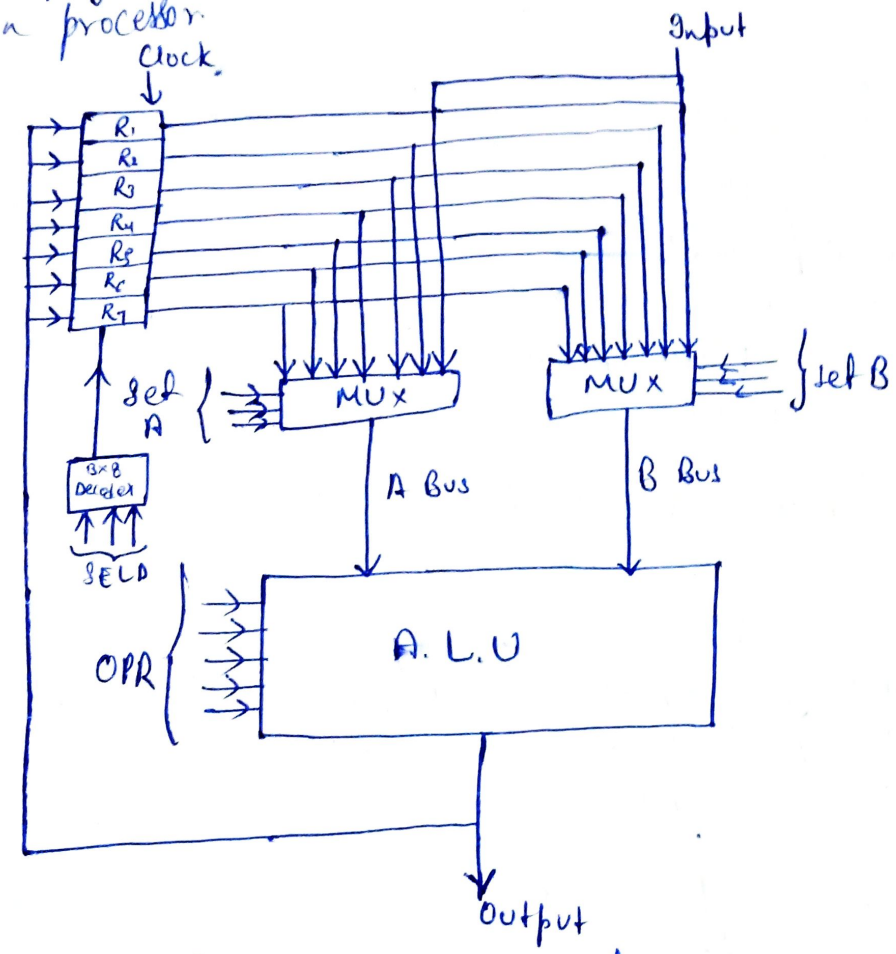


* Register set stores intermediate data used during the execution of instruction.
* ALU performs required microoperation for executing instruction.
* Control unit supervises transfer of info among registers & instructs ALU to which operation to perform.

General Register Organization.

For storing pointers, counters, return addresses etc. memory locations are not preferred as their process in time consuming, so it is convenient & more efficient to store these in processor registers. Large numbers of registers are included in CPU and are connected by a common bus system. Registers communicate with each other in direct data transfers and microoperations so a common unit is provided

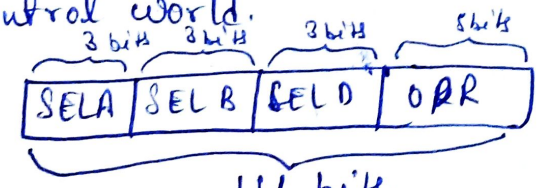
to perform all arithmetic, logic, and shift microoperations in processor.



Bus Organization for Seven CPU registers

→ The decoder activates one of the register load inputs of the selected destination register

Control word.



control world.

SELA → selection of source register for A input of ALU

SELB → " " " " " B " " "

SELD → " " Destination " using decoder.

OPR → operations in ALU

When, SELA or SELB = 000
external input is selected

When, SELD = 000, no destination register is selected but contents of output bus are available in external output

Note: A register can be cleared to 0 with an xor operation, as $x \oplus x = 0$

~~# Stack Organization~~

Class Input/Output Organization.

RISC & CISC

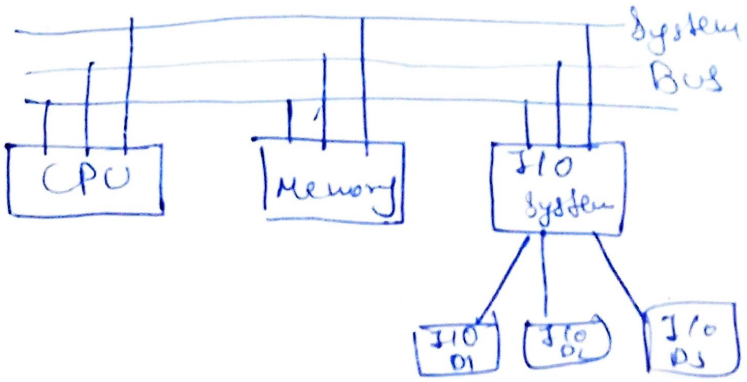
Input-Output Interface

Designing logic circuit and writing instruction to enable the processor to communicate with peripheral devices called interfacing and logic ckts are called I/O ports

Input-Output module.

(I) Interface to CPU and memory ~~Bus~~ By system bus

(II) Interface to one or more input devices by tailored ~~data name~~ database

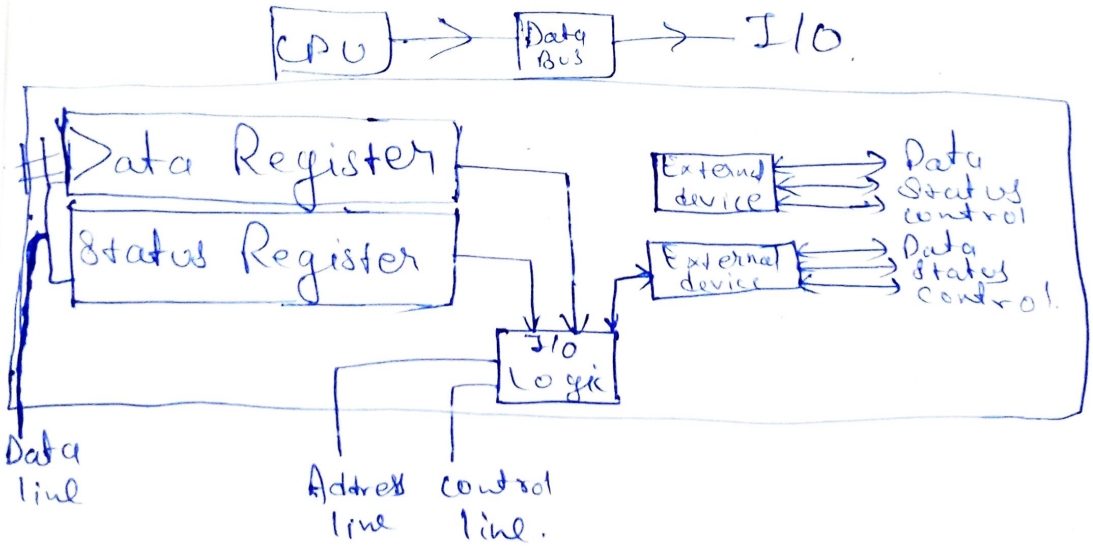


Major Requirement of I/O.

Assignment
A Computer has 64 mb of byte addressed memory. how many bit are needed in memory address.

The major requirement for I/O
1. Control & timing
2. C.P.U communication — Process & I/O., memory
3. Device Communication — status, control signal
4. Data Buffering.
5. Error Detection

1. It will control flow of signals.



Q. How Processor select which device to use to communicate I/O?

R.B.K cache

CO- Crate 15, 17, 01, 07

Q main memory has "2cm" blocks, cache has "2e" blocks "2-way" set associative. Then kth block of main memory maps to k' mode set

Q more than one word are put in one block to exploit spatial local in a program [spatial local & temporal local]

Q Cache size = 4K words
block size = 64 words
Set size = 4 blocks
The no. of bits in "set" & "word" field of MM address are

Tag	S.no	B. off
	4	6
		word

lines = $\frac{CS}{BS} = \frac{4K}{64} = 2^6$

Sets = $\frac{2^6}{4} = 2^4$

Q 4-way set associative
Cache lines = 128
line size = 64 words
PA = 20 bits
Tag, set & word fields are

7	5	6
tag	S.no	word
20		

lines = ~~128~~ 64 = 2^6
Sets = ~~128~~ $\frac{64}{128}$ = $\frac{128}{4}$ = 32 = 2^5

Q ~~Blocks in~~

Q Blocks in cache = 128
4 way set associative MM contains 2^{14} blocks
Block size is 256 eight bit words.

i) How many bits are required for addressing MM?
ii) How many bits are needed to represent the Tag, Set and word fields?

MM = Blocks * size of Block
$= 2^{14} \times 256$
$= 2^{22}$ w

i) PA = 22 bits

ii) | 9 | 5 | 8 |
Tag Set words

$256 = 2^8$

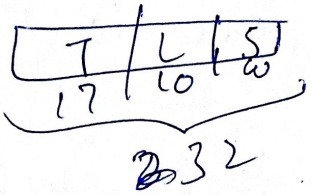
22
Lines = 128
$sets = \frac{128}{4} = 32 = 2^5$

Q. Direct mapped cache.
Cache size = 32 kB
Block size = 32 B
PA = 32 bits.
The no. of bits needed for cache indexing & tag bits are respectively.

Sol | 17 | line 10 | 5 |
Tag. w

32
$line = \frac{32\ kB}{32B} = 2^{10}$ →

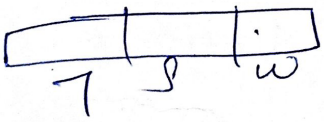
⑥ Consider two cache organisations. The first one is 32 kB 2-way set associated with 32-byte block size. The second one is of the same size but direct mapped. The size of an address is 32 bits in both cases. A 2-to-1 multiplexer has latency of 0.6ns while a k-bit comparator has a latency of (k/10)ns. The "hit latency" of the set associative organization is h1, while that of direct mapped is "h2"



$L = \frac{32kB}{32} = 2^{10}$

$(2^{10}$ to $1)$ mux

$\frac{17}{10}$ ns = 1.7 ns.



class I/O Module

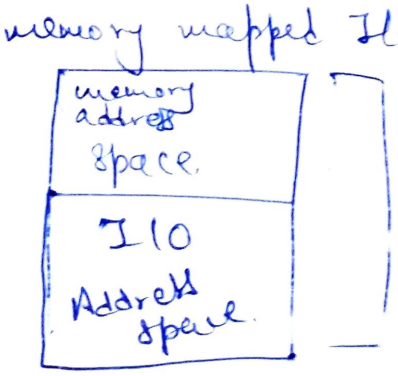
Programmed I/O: communication between I/O & processor is controlled by a program called as Programmed I/O.

* Input Output operation means a data transfer b/w I/O & memory or I/O and CPU.
If any I/O operation are completely controlled by the CPU is said to be programmed I/O.

CPU executes program that initiate direct & terminate I/O operation including sensing device status, sending a read or write command & transferring the data.

~~The~~ A no. of i/o device can be used in a system, for each device address is different these device are access by following methods.

1) ~~I/O Addressing~~ Memory mapped I/O
2) I/O Mapped



Complex Instruction Set Computer (CISC)	Reduced Instruction set computer (RISC)
① A large number of instructions.	① Relatively few instructions
② Some instructions that perform specialised tasks and are used infrequently	② Relatively few addressing modes.
3) Large number of addressing modes	③ memory access is limited to load & store instrn
4) Variable length instr^n formats	4) All operations are done within the registers of CPU.
5) Instr^{n}s that manipulate Operands in memory.	5) Single cycle instrn execution
	6) Fixed length, easily decoded instrn format.
	7) Hardwired rather than microprogrammed control

Central Processing Unit

Stack Organization.

A Stack is a storage device that stores info in such a manner that the item stored last is the first item retrieved.

Stack in digital computers is essentially a memory unit with an address register that can count only (after an initial value is loaded into it.)

Stack Pointer (SP)

- Register that holds the address for the stack is called a stack pointer (SP).
- It's value always points at top item in stack.

Two operations of stacks:
① PUSH: Operation of insertion.
② POP: Operation of deletion.

- Register Stack

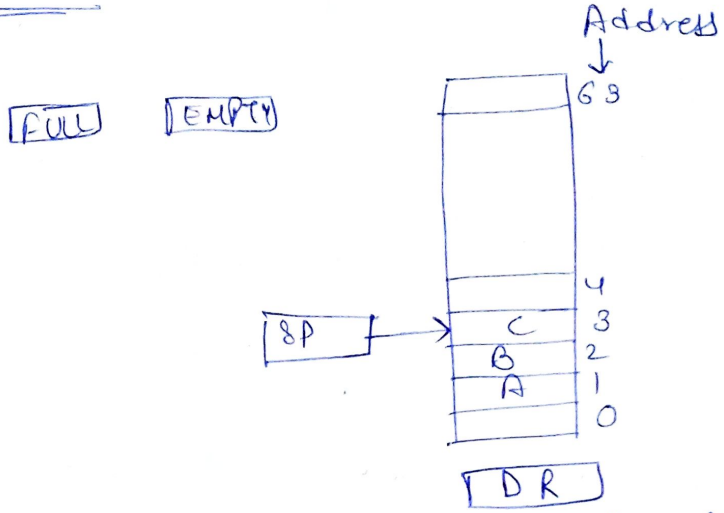


Fig: Block diagram of a 64-word stack.

* The stack pointer contains 6 bits because $2^6 = 64$.
* The one-bit register FULL is set to 1 when the stack is full.
* One bit register EMTY is set to 1 when stack is empty of items.
* DR is data register that holds the binary data to be written into or read out of the stack.

Initially SP is cleared to 0, EMTY is set to 1, and FULL is cleared to 0, so that SP points to the word at address 0 and stack is marked empty and not full.

PUSH Operation.

SP ← SP+1 Increment stack pointer.
M[SP] ← DR Write item on top of stack.
if (SP=0) then (FULL ← 1) Check if stack is full
EMTY ← 0 Mark stack not empty

POP operation.

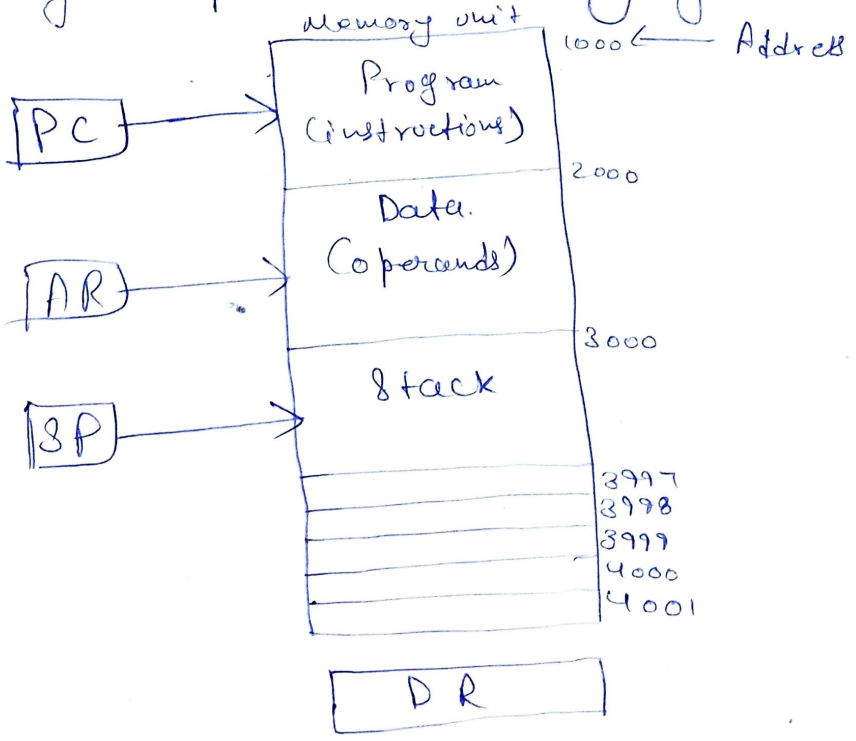
DR ← M[SP]	Read item from top of stack
SP ← SP - 1	Decrement stack pointer
If (SP = 0) then (EMTY ← 1)	Check if stack is empty.
FULL ← 0	Mark the stack not full.

⇒ Memory Stack.

Computer memory partitioned into three segments.
- Program
- Stack
- data.

PC → Program Counter points at the address of next instruction in program.

AR → Address register points at an array of data.



PUSH Operation:

SP ← SP-1
M[SP] ← DR

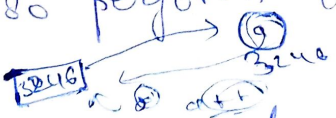
POP operation

DR ← M[SP]
SP ← SP+1

Instruction formats.

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

Computer perform task on the basis of instruction provided. An instruction in computer comprises of groups called fields. These field contains different info, as for computers everything is in 0 & 1 so each field has different significance on basis of which a CPU decide what so perform, the most common fields are:

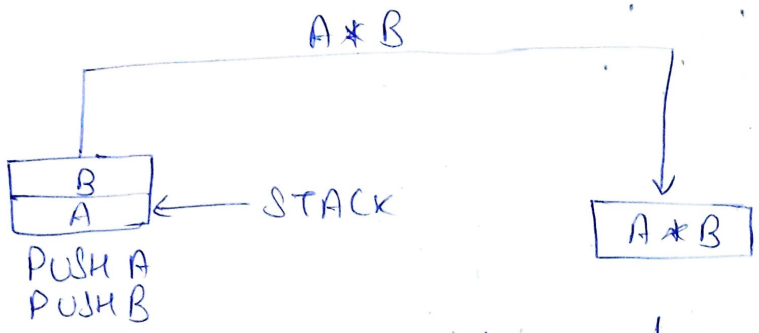


* Operation field which specifies the operation to be performed like addition.
* Address field which contain the location of operand, i.e., register or memory location.
* Mode field which specifies how operand is to be founded.

On basis of no. of address, instrn are classified as:

Note that we will use $X = (A+B)*(C+D)$ expression to showcase the procedure.

- Zero Address Instrn:



A stack based computer do not use address field in instrn. To evaluate an expression first it is converted to reverse Polish Notation i.e. Post Fix Notation.

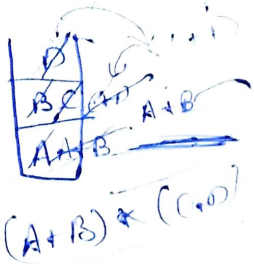
Expression: $X = (A+B)*(C+D)$

Post fixed: $X = AB+CD+*$

TOP means top of stack

M[X] is any memory location.

PUSH A	TOP = A
PUSH B	TOP = B
ADD	TOP = A + B
PUSH C	TOP = C
PUSH D	TOP = D
ADD	TOP = C + D
MUL	TOP = (C+D) * (A+B)
POP X	M[X] = TOP

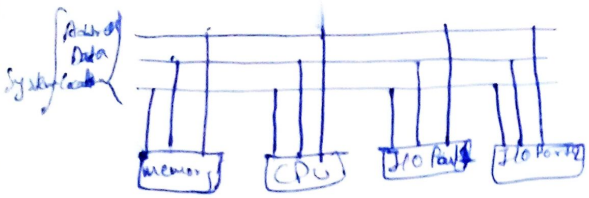


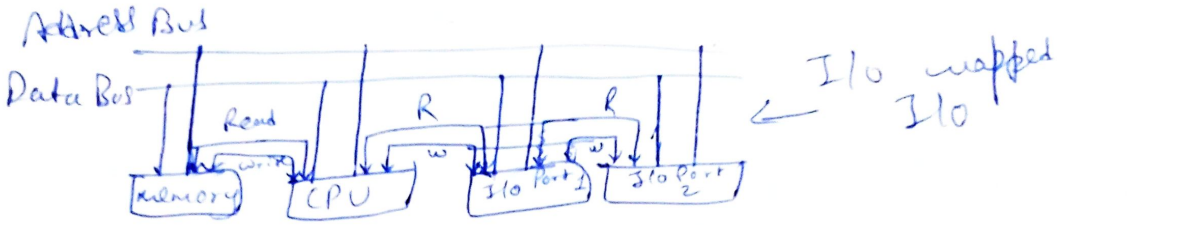
#kernal

Class

Programmed I/O

① Memory mapped I/O ② Mapped I/O Mapped

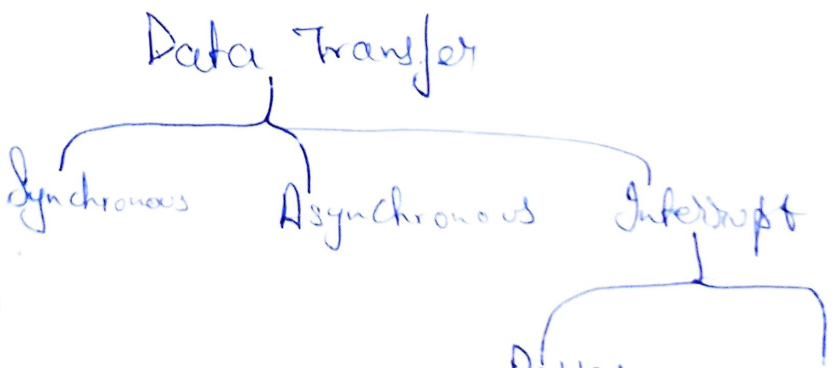




Data Transfer in I/O module.

① Programmed data transfer → Synchronous, Asynchronous, Interrupt Driven.

② Direct memory access.



#kernal & its types

Direct memory access

The data transfer from memory to input Output device vice-versa can be achieved only through the processor.

When data has to be transferred from memory to I/O device, first the processor sends

address & control signal to memory to read data from memory then processor sends address & control signal to I/O device to write data.

~~It means~~

In the data transfer method, data cannot be directly transferred b/w memory & I/O device, even though they are connected to common bus.

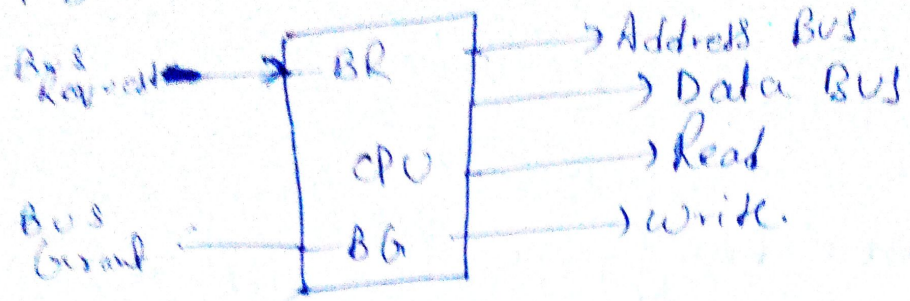
The process in invitable because processor cannot simultaneously select two devices, to overcome this problem, we use direct memory access parameter.

In DMA, data transfer will be useful to transfer large amount of data b/w memory & I/O device.

For direct data transfer b/w I/O device & memory a dedicated hardware device called direct memory access controller.

A DMA controller temporarily borrow the address bus, data bus & control bus from processor & transfer data bytes directly from I/O codes to a series of memory location or vice versa. Some DMA controller can also perform memory to memory transfer.

DMA Controller.



BUS ARBITRATION

BUS master: The device that is allowed to initiate data transfers on the bus at any given time is called Bus master. eg-> processor, DMA controller etc.

Bus arbitration is the process by which the next device to become the bus master is selected and bus mastership is transferred to it. Selection of bus master is usually done on priority basis

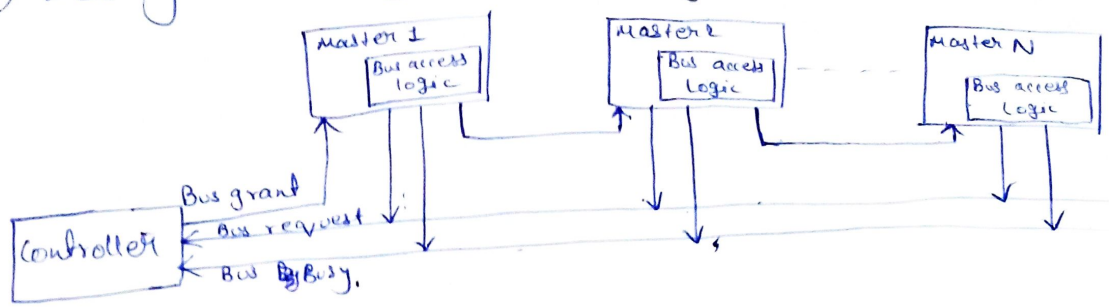
There are two approaches to bus arbitration.
1. Centralized Arbitration.
2. Distributed Arbitration.

Centralized Arbitration.

In Centralized bus Arbitration, a single bus arbiter performs the required arbitration. Bus arbiter may be the processor or separate controller connected to bus

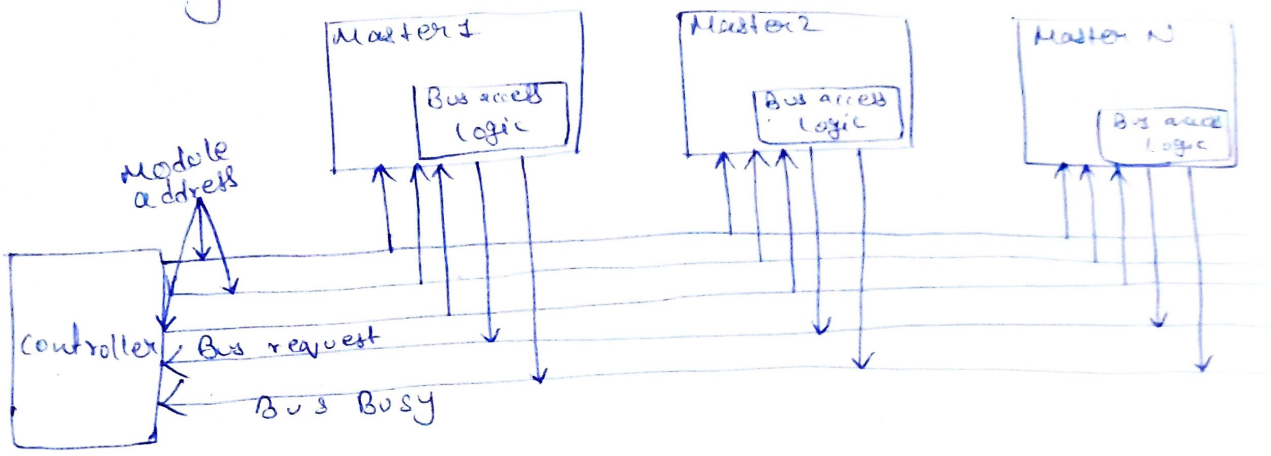
Three different arbitration schemes that use centralized bus arbitration approach are:
1. Daisy chaining
2. Polling method
3. Independent Request.

① Daisy Chaining.



- It is simple and cheaper method. All masters make use of same line for bus request.
- In response to bus request the controller sends a bus grant if bus is free.
- The bus grant signal serially propagates through each master until it encounters the first one that is requesting access to the bus. This master blocks the propagation of the bus grant ~~line~~ signal, activates the busy line and gains control of the bus.
- Therefore any other requesting module will not recieve the grant signal and hence cannot get the bus access.

(b) Polling method.

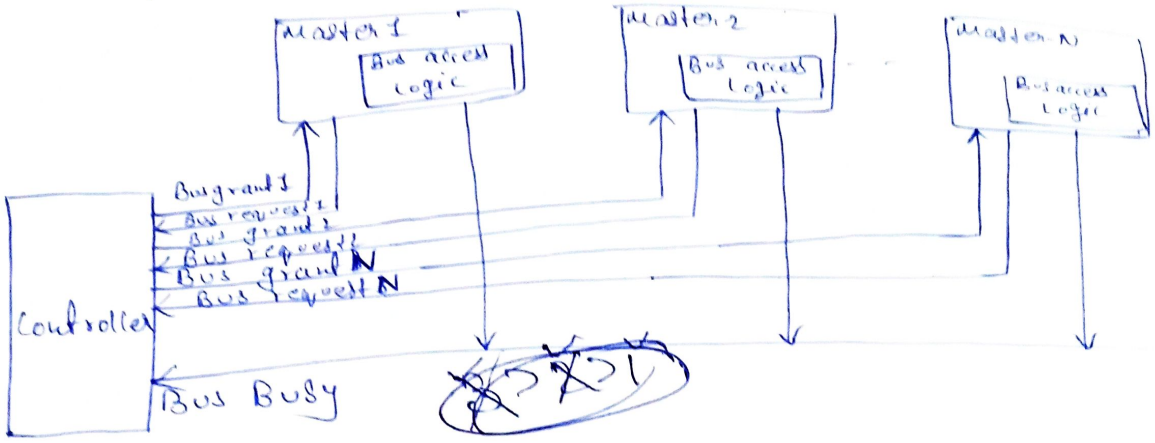


- In this controller is used to generate the addresses for the master. Number of address line required depends on the number of master connected in the system.
- For eg., If there are 8 masters connected in the system, at least three address lines are required. $8 = 2^3$ $64 = 2^6$

  $n = \log_2 n$
- In response to the bus request controller generates a sequence of master address when the requesting master recognizes its address, it activated the busy line and begins to use the bus.

0 0 1
0 1 0
0 0 0
0 1 1
1 0 1
1 1 0
1 0 0
1 1 1

© Independent Request



- In this scheme each master has a separate pair of bus request & bus grant lines and each pair has a priority assigned to it.
- The built in priority decoder within the controller selects the highest priority request and asserts the corresponding bus grant signal.

Distributed BUS arbitration:
All devices participate in the selection of the next bus master.

▷ Advantages of Daisy chaining
- Simplicity & scalability.
- The user can add more devices anywhere along the chain, up to a certain maximum value.

▷ Disadvantages of Daisy chaining
- value of priority depends on position of master bus.
- Propagation delay arises in this method.
- If one device fails entire system will stop working.

▷ Advantages of Polling method
- This method does not favor any particular device and processor.
- This method is quite simple.
- If one device fails entire system will not stop working.

▷ Disadvantages of Polling method.
- Adding bus masters is difficult as increases the no. of address lines of CPU

▷ Advantages of Independent request
- This's method generates fast response.

▷ Disadvantages of Independent request.
- Hardware cost is high as large no. of control lines are required.

Registers

① Accomulator Register.
Accomulator is a register in which intermediate arithmetic and logic results are stored.

② General purpose Register:
Used to store temporary data within the microprocessor.

③ Index or Address Register
is that which receives, stores and outputs instruction - changing codes in a computer.

④ Data Register or Memory Buffer register. that stores the data being transferred to and from the immediate access storage.

② One Address Instrn

This use an implied Accumulator register for data manipulation. One operand is in accumulator and other is in register or memory location. Implied means that the CPU already know that one operand is in accumulator so there is no need to specify it.

opcode	Operand/ Address of operand	mode.

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

AC is accumulator

M[ ] is any memory location.

M[T] is temporary location.

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

T = A+B

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

3) Two address Instruction
This is common in commercial computers. Here two address can be specified in instrn. Unlike earlier in one address Instrn the result was stored in accumulator here result can be stored at different location rather than just accumulator, but require more no. of bits to represent address.

opcode	Destination Address	Source Address	mode

Expression: $X = (A+B)*(C+D)$
R1, R2 are registers
M[ ] is any memory location.

R1 ← A+B
R2 ← C+D
X = (A+B)*(C+D)

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

4) Three address Instrn
This has three address field to specify a register or a memory location. Program created are much short in size but no of bits per instruction increase
These instruction make creation of program much easier but it does not mean that program will run much faster because now instruction only contain more info but each microoperation will be performed in one cycle only.

opcode	Destination addr.	Source addr.	Source addr.	mode

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

R1, R2 are registers

M[] is any memory location

R1 = A + B
R2 = C + D
X = (A+B) * (C+D)

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

Addressing modes:

The term addressing mode refers to the way in which the operand of an instruction is specified. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before operand is actually executed.

"The general subject of specifying where the operands are" is called addressing.

Types

(1) Immediate mode.
Operands are specified implicitly in definition of instruction.

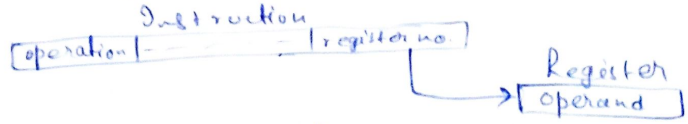
(1) Immediate mode. Or absolute
The actual operand is specified in instruction.

Instruction

| operation | - - - - - | operand. |

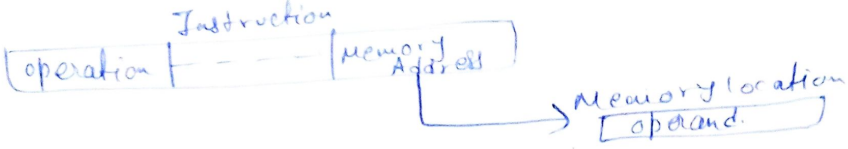
Example: MOV, R1, 123

2) Register mode.
The instruction specified a register that has operand



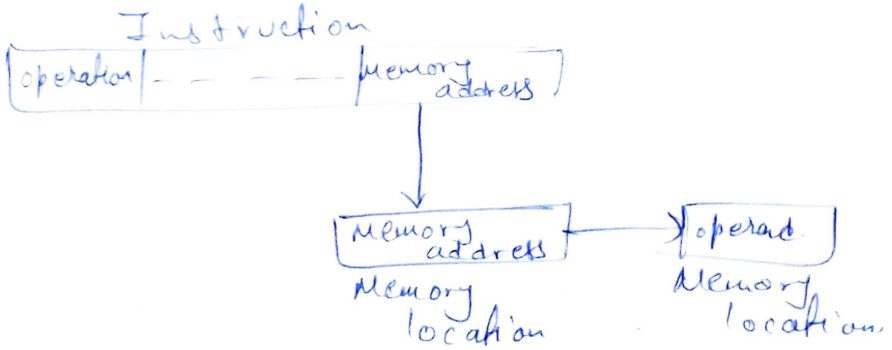
example: MOV R1, R2

3) Direct Address mode.
The instruction has memory address where the operand is located.



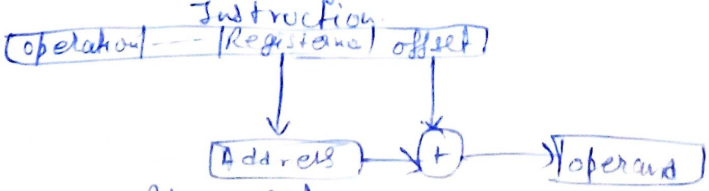
eg. MOV R1, [1000]

4) Indirect Address mode.
The instruction has the address that specifies the effective address of the operand.



eg. ADD R1, (1000)

5) Indexed Addressing mode Or Based Add. mode.
The effective address is sum of an index register and the offset field (constant).



eg -> MOV R1, 2 (R2)

6) Relative address mode.
The effective address is the sum of the address field (offset) and content of program counter (PC).

eg-> ADD R1, [R2+PC]

7) Auto Increment & Auto Decrement modes.

① Post Auto Increment mode:
The effective address of operand is contents of a register specified in the instruction. After accessing the operand, the contents of this register are automatically incremented to point to next item in a list. It is denoted by $(R_i)^+$.
ex. MOV R1,(R2)+.

② Pre Auto Increment mode.
Before accessing the operand, the contents of this register are automatically incremented to point to next item in a list. Then effective address of operand is the incremented contents of register. It is denoted by $+(R_i)$.
eg. MOV R1, +(R1).

③ Post Auto Decrement mode.
The effective address of the operand is the contents of a register specified in instrn. After accessing the operand, the contents of this register are automatically decremented to point to next item in a list. It is denoted as $(R_i)^-$
eg-> MOV R1,(R2)-

⑤ Pre Auto Decrement mode.
Before accessing the operand, the contents of this register are automatically decremented to the next item in a list. Then effective address of the operand is the decremented contents of a register. It is denoted as (-Ri)

eg- MOV R1, -(R1)

Types of machine Instructions

Based on operations performed, machine instructions can be divided into the following:

1) Data transfer Instructions
2) Data manipulation Instructions
3) Program control Instructions

Data transfer Instructions

Instructions that transfer data from one location (Register/Memory) to another location (Register/memory) without changing the data.

eg.
- LOAD: Data transfer from memory to register
- STORE: Data transfer from register to memory.
- MOVE: Data transfer from register to register.
- IN: Transfer data from input device to register
- OUT: Transfer data from register to output device
- PUSH: Gets data from memory or register on top of stack.
- POP: Gets data from top of stack to memory or register.
- XCHG: Exchanges data b/w memory & registers

Data manipulation Instructions

- Arithmetic Instructions:- Performs an arithmetic operation as addition, subtraction, multiplication, division, increment, Decrement e.t.c.
  eg. ADD, SUB, MUL, DIV, INC, DEC, etc.

- Logical Instructions:- Performs bitwise logical operation such as AND, OR, exclusive-OR, NOT, Shift, rotate, e.t.c.
  eg → AND, OR, NOT, XOR, SHL, SHR, ROT etc.

- Arithmetic & Logical Instructions:
  Performs operations such as arithmetic shift left, arithmetic shift right, et.c.
  eg: SAL, SAR, etc.

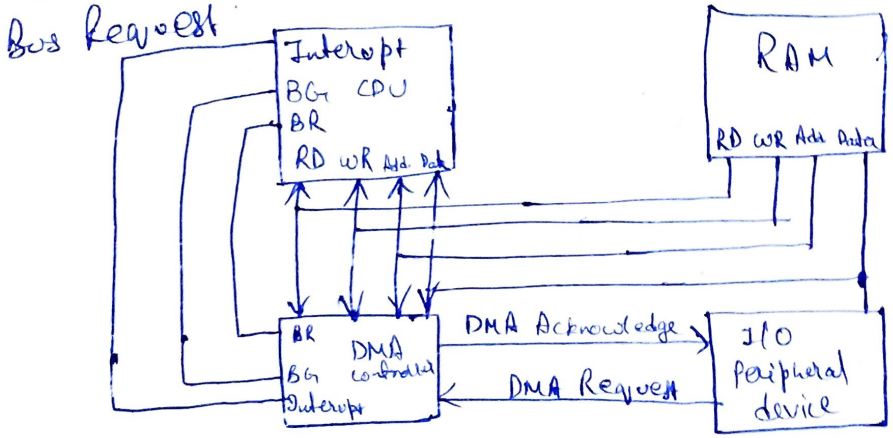
Bus Request –
It is the input signal used by DMA controller to request C.P.U to relinquish control of buses. when this input is active, the CPU terminates execution of correct instruction and place the address bus, data bus and control signal into a high impedence state.

Bus Grant-
It is activated by CPU to inform external DMA that the buses are in high impedence state. The DMA that orginated the bus request can now take control of buses.

Mode of transfer.

When the DMA takes control of Bus System, it communicate directly with memory. The transfer can be in several ways. In DMA bus transfer, A block sequence consisting of a number of memory word is transfers in a continuous burst while the DMA controller is master of memory bus. This mode of transfer is for fast devices an alternative technique called cycle staring allow to transfer one data word at a time after which it must return control of Bus to CPU.

System with DMA Controller.



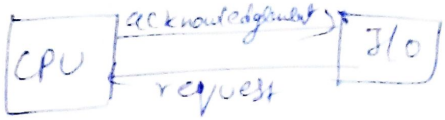
Control signal
→ Read
→ write.

Slave
Load Address.

Master
Acknowledgement.

Interrupt: It allows the processor to respond to external request for attention or service on demand basis. and leave the processor free to perform other task.

- Interrupt
  - By external (Peripheral Device)
  - By Internal.



=> Categories of Interrupt:

1) H/w & S/w interrupt
2) Vector & non-vector.
3) Maskable & non-maskable.

Interrupt handling

1) Single Interrupt
2) Multi Interrupt handling