Q What is ALU.

An arithmetic logic unit (ALU) is a digital circuit used to perform arithmetic and logic operation. it represents the fundamental building block of the central processing unit (CPU). of a computer. Modern CPUs contain very powerful and complex ALUs. In Addition to ALUs, Modern CPUs contain a control unit (CU). Most of the operations of a CPU are performed by one or more ALUs. which load data from input registers. A register is a small amount of storage available as part a CPU. The control unit tell the CPU what operation perform of data and the ALU stores the result in an output register. The control unit moves the data between these registers, the ALU, and memory.

★ it does all processes related to arithmetic and logic operation that need to be done on instruction words. In some microprocessor architecture the ALU is divided into the Arithmetic unit (AU) And the logic unit (LU).

★ An ALU can be designed by engineers to calculate any operation As the operations become more complex, the ALU also becomes more expensive, takes up more space in the CPU and dissipates more heat.

That is why engineers make the ALU powerful enough to ensure that the CPU is also powerful and fast, but not so complex as to become prohibitive in terms of cost and other disadvantages.

★ An arithmetic logic unit is also known as an integer unit (IU).

★ The main functions of the ALU are to do arithmetic and logic operations, including bit shifting operation. These are essential process that need to be done on almost any data that is being processed by the CPU.

★ ALUs routinely perform the following operation

★ Logical operation: These include AND, OR, NOT, XOR, NOR, NAND etc.

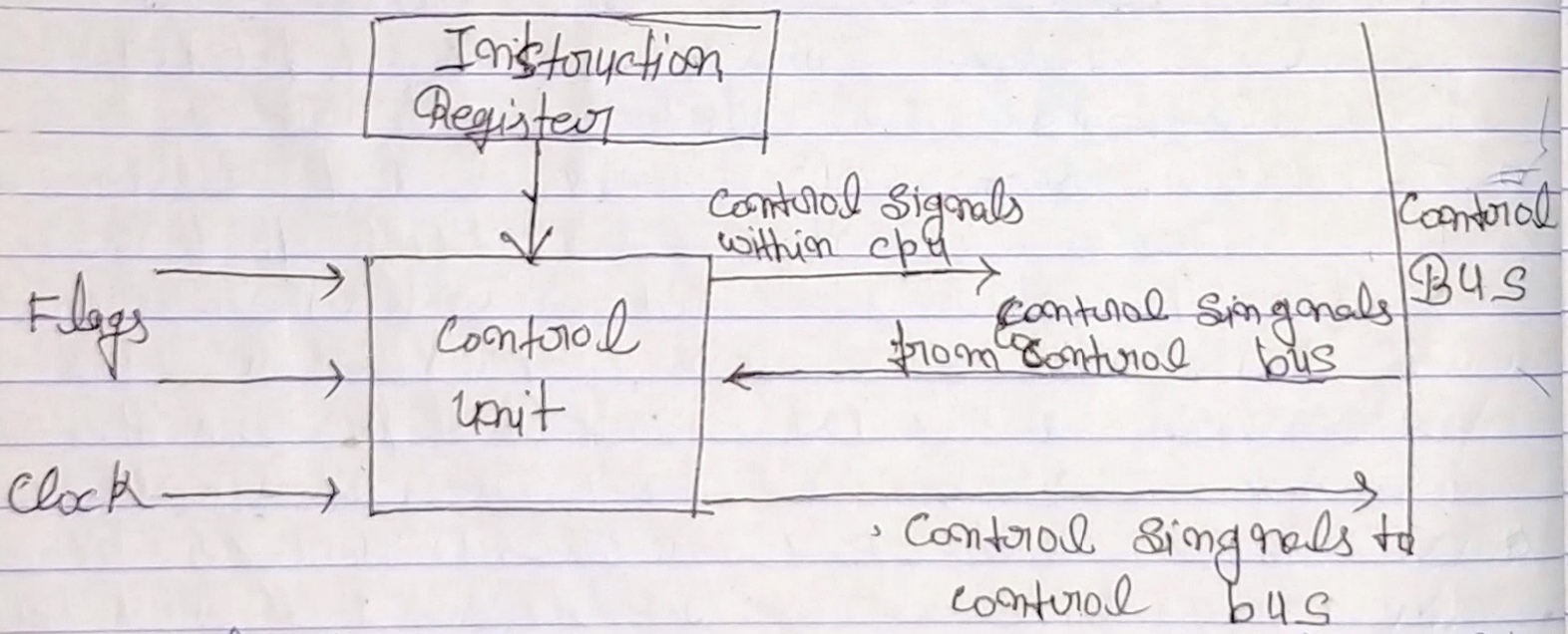
★ Bit-Shifting operations: This pertains to shifting the positions of bits by a certain number of places to the right or left which is considered a multiplication operation

★ Arithmetic operations: This refers to bit addition and subtraction. Although multiplication and division are sometimes used, these operation to substitute for multiplication and subtraction for division.

• Some processors contain more than one ALU - for example, one for fixed point operations and another for floating point operations.

- Typically, the ALU has direct input and output (access) to the processor controller, main memory (random access memory or RAM in a personal computer) and I/O devices
- I/O flow along an electronic path that is called a bus.
- The input consists of and instruction word (sometimes called a machine instruction word) that contains an operation code (sometimes called an "op code") one or more operands and sometimes a format code.
- The operations code tells the ALU what operation to perform and the operants are used for the operation (for example, two operands might be added together or compared logically.) the format may be combined with the op code and tells, for example, whether this is a fixed point or a floating-point instruction.
- The output consists of a result that is placed in a storage register and setting that indicate whether the operation was performed successfully (if it is not, some sort of status will be stored in a permanent place that is sometimes called the machine status word.
- In general, the ALU includes storage places for input operands, operands that are being added the accumulated result (storage in an accumulator), and shifted results. The flow of bits and the operations performed

Control unit :- Control unit is the part of the computer's central processing unit (CPU), which directs the operation of the processor. it was included as part of the Von Neumann Architecture by John von Neumann. it is the responsibility of the control unit to tell the computer's memory, arithmetic/logic unit and input and output devices how to respond to the instructions that have been sent to the processor. it fetches internal instructions of the programs from the main memory to the processor instruction register, and based on this register contents, the control unit generates a control signal that supervises the execution of these instructions.



Block Diagram of the control unit

A control unit works by receiving input information to which it converts into control signals which are then sent to the central processor. The computer's processor then tells the attached hardware what operations to perform the functions that a control unit performs are dependent on the type of CPU because the architecture of CPU varies from manufacturer to manufacturer. Examples of devices that require a control unit are —:

Control processing units (CPUs)
Graphics processing units (GPUs)

Functions of the control unit —:

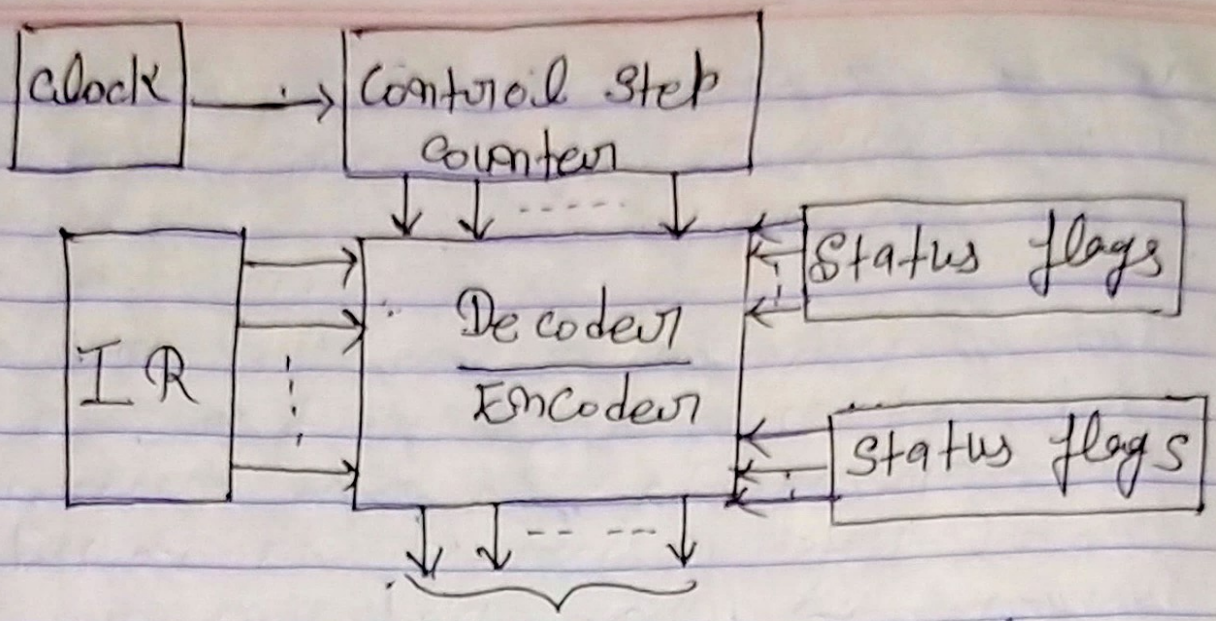
1. It coordinates the sequence of data movement into, out of, and between a processer's many sub-units.
2. It interprets instructions.
3. It controals data flow inside the processer.
4. It recives external instructions or command to which it converts to sequence of control signals.
5. It controals many execution units (i.e ALU, data buffers and registers) contained with in a CPU.
6. It also handles multiple tasks such as fetching, decoding, execution handling and storing results.

Control Design Methods –; The execute instructions; the CPU must have some means of generating the control in the proper sequence. Computer designers have used a wide variety of techniques to solve this problem. Most of these techniques, however, fall into one of two categories.

1. Hardwired control unit –;
2. Micro-programmed control unit –;

Hardwired control unit –; It is a controller as a sequential logic circuit or externally supplied instructions. In a hardwired implementation the following points must be noted:

1. The control unit is essentially a combinational circuit.
2. The control logic is implemented with gates, flip flops, decoders and other digital circuits.
3. A hardwired control as the name implies requires change in the wiring among the various components if the design has to be modified or changed. its input logic signals are transformable into a set of output logic signals that are called control signal.



Block Diagram of Hardwired control

Micro-programmed control unit :- The design of control unit must include logic for sequencing through micro-operations, for executing micro-operations, Interpreting opcodes, and for making decisions based on ALU flags. it is difficult to design and test such a piece of hardware. An alternative, which is quite common contemporary CISC processors, is to implement a micro programmed control unit.

Due to lack of flexibility in hardwired control unit, it is quite difficult to design and implement as in angry computers the numbers of micro programmed control unit.

The sequence of execution of micro-operations through a program will considered instructions with each of the instruction describing.

(i) one or more micro operations to be executed
(ii) The information about the micro instruction to be executed next.

Such as instruction is termed as micro instruction and such a program is termed micro-program of Firmware. In the micro programmed, the control unit information is stored in a sequence of micro-operations.

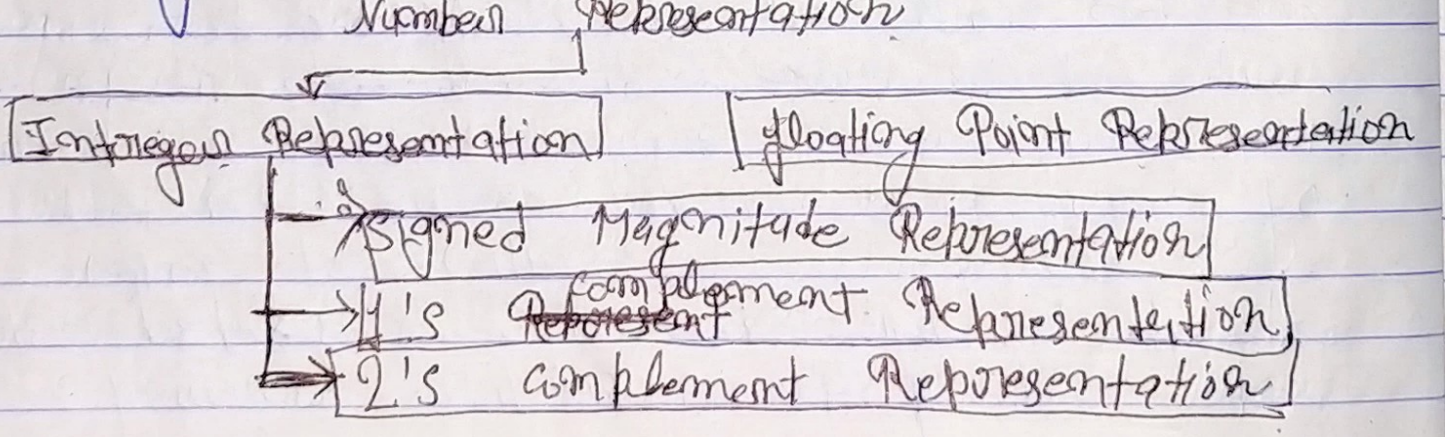
Number representations in ALU (Computer Arithmetic)

The Arithmetic and logic unit (ALU) of a computer performs arithmetic and logic operations over the data.

Basically, number can represent by ALU in two ways

(1) Integer Representations
(i) Signed magnitude Representation
(ii) 1's complement Representation
(iii) 2's complement Representation

2) Floating point Representation



INTEGER REPRESENTATION —;

Arithmetic logic unit (ALU) requires a highly specialized hardware section to uses and distinguish between the signed and unsigned numbers and also needs a special set of algorithms to accomplish the arithmetic operations on signed numbers

Different ways of coding Negative Numbers —;

There are three common ways of coding negative number on computers.

1. Signed Magnitude Representation
2. 1's complement Representation
3. 2's complement Representation

1. Signed Magnitude Representation —; As computer Represent all information by mean of 0s and 1s, the concept of sign must also be represented with these symbols. A straightforward approach is to reserve one bit of the Number such as the most significant bit, as the sign bit. The Sign bit can be either 0 or 1 where 0 denotes positive and 1 denote negative and the remaining bits can then be used to denote the magnitude as a conventional binary Number. This approach is know as the Signed magnitude representation of signed binary numbers.

In other word Signed magnitude representation is the simplest of expressing a Signed number in a digital computer. The Nth bit that is the most Significant Bit (MSB) of the magnitude Representation. if the leftmost bit is 0, the number is positive and if the leftmost bit is 1, then the Number is negative

for example → following Representation shows how to express positive and negative number and its equivalent binary Representation using the signed magnitude technique.

	Sign bit	Magnitude
+18 =	0	0010010
-18 =	1	0010010

In the above example the MSB, Most Significant Bit, is set as 1. The Sign bit having above 1 Indicates that the number is negative.

eg-1 A Signed Number is expressed using a Signed technique.

$(-33)_{10} = (\quad)_2$

binary value of $(33)_{10}$ is.

$(33)_{10} = (100001)_2$

8	7	6	5	4	3	2	1
0	0	1	0	0	0	0	1

for negative $(-33)_{10} = (10100001)_2$

8	7	6	5	4	3	2	1
1	0	1	0	0	0	0	1

in eight bit Representation the 8th bit stands for sign bit. if the 8th bit is 1 then negative number else it is a positive number

Signed Magnitude, ~~the 8th bit stands for sign~~ always yields two form representations for zeros i.e one for positive zero (0) and the other for the negative zero (-0).

for example -;

$(0)_{10}$ = 00000000

$(-0)_{10}$ = 10000000 (Sign magnitude)

which is inconvenient, as it is slightly more difficult to test for 0 (an operation performed very frequently on a computer) - if there is a signed magnitude.

Drawbacks of Signed Magnitude Representation -;

1. Special care must be taken whenever a subtraction operation is to be performed as the same bit position is used for both signed and unsigned numbers.
2. It is considerably difficult and cumbersome to trace a signed magnitud Representation in the hardware circuits.

3- Signed magnitude always yield two forms of Representation for Zeroes ie one for positive Zero (0) and another for negative (-0).

eg -; Express the following numbers in binary using the signed method

(i) -127

(ii) -1

(i) $(-127)_{10} = (\quad)_2$

$(127)_{10} = (01111111)_2$

$(-127)_{10} = (1111\ 1111)_2$

(ii) $(-1)_{10} = (\quad)_2$

$(1)_{10} = (0000\ 0001)_2$

$(-1)_{10} = (1000\ 0001)_2$

ex Subtract 10011000_2 (-24) from $(10101011)_2$ (-43) using signed magnitude arithmetic.

Solution -; one can convert the subtraction to an addition by negative -24 which gives us 24, and then we can add this to -43 giving us a new problem of -43 +24 However we know from the addition rules above that because the sign have differ one must actually subtract the smaller magnitude from the larger magnitude (or Subtract 24 from 43) and make the result negative (since 43 is a larger than 24).

```
  0 1 1
  0 1 0 1 0 1 1        borrows
- 0 0 1 1 1 0 0 0      43
-------------------    -24
  0 0 1 0 0 1 1        (19)
```

Note that we are not concerned with the sign until we have performed the subtraction we know the answer must be negative so we end up with $1010101_2 - 1001100_2 = 0010011_2$ in signed magnitude representation.

for example 3 :

Add 10010011_2 (-19) to 00001101_2 (+13) using signed magnitude arithmetic.

The first number (augend) is negative because its sign bit is set to 1. The second number addend is positive. first determine which of the two number larger in magnitude and use that number for the augend. its sign will be the sign of result.

```
  1 0 0 1 0 0 1 1        borrows
- 0 0 0 0 1 1 0 1        -19
-------------------      +13
  1 0 0 0 0 1 1 0        (-6)
```

with the inclusion of the sign bit, we see that $10010011_2 - 00001101_2 = 10000110$ signed magnitude representation.

Ex 4

Subtract $0100\,1111_2$ from $0110\,0011_2$ using signed magnitude arithmetic

Solution :-

borrow

$$\begin{array}{r} 99 \\ -(79) \\ \hline 20 \end{array} \qquad \begin{array}{r} 0110\,0011 \\ 0100\,1111 \\ \hline 0010\,0100 \end{array}$$

1's complement Representation

To Perform the one's complement operation on a set of binary digits, simply replace all 0's digits with 1's and vice-versa. In other words, subtract all digits of the binary numbers with 1's. The resultant bit pattern is called the one's complement of the given set of binary numbers.

Ex 8 Illustrate the procedure to obtain 1's complement for a given binary set

Let $X_{(10)} = 0001\,1001$, then

1's complement $= 1110\,0110$ (all 1s and 0's are inverted)

$Y = 1111\,0111$

1's complement $= 0000\,1000$

Step by Step Method of 1's complement :-

Step 1: Positive integer is represented in the same way as Signed magnitude representation.

Step 2 :- Negative integer is represented by the 1's complement of the positive integer with the same magnitude.

for ex :- 18 = 00010010
-18 = 1's complement 18 = 11101101 & hence
$(-18)_{10} = (11101101)_2$

Ex 9 :- express the following numbers in binary from using 1's complement method.

(i) (-33) (ii) -127 (iii) -1

(-33) = ( )2
$(33)_{10} = (100001)_2$
(33) 10 = (00100001)
1's complement of 00100001 is (11011110)
Hence as per 1's complement Method
$(-33)_{10} = (11011110)_2$

$(-127)_{10} = (\quad)_2$
127 = (01111111)2
and hence
$(-127)_{10} = (10000000)_2$

(iii) $(-1)_{10} = (\quad)_2$
$(1)_{10} = (00000001)_2$
and hence
$(-1)_{10} = (11111110)_2$

Ex 10 Express 23_{10} and -9_{10} in 8-bit binary assuming a computer is using 1's complement representation.

$(23)_{10} = (00010111)_2$

$(-9)_{10} = (00001001)_2 = 11110110_2$ 1's complement

To carry out a 1's complement, we first express the subtrahend 9 in the 1's complement then add it to the minuend 23; we are effectively now adding -9 to 23. The high order bit will have a 1 or 0 carry which is added to the low order bit of the same.

Ex 11 $(9)_{10} = (\ )_2$

(a)
$$\begin{array}{r} 00001001 \\ 11101000 \\ \hline 11110001 \end{array}$$

$+9$ 00001001

-9 11101001

11110010 1's

+ 1

11110011 2's

$+23$ 00010111

-23 11101000

+ 1

11101001

$+22$ 00010110

$+22$ 11101001 1's

+ 1

11101010 2's

Table : Comparison of Sign magnitude and Two's complement Representations for 4 bit.

Decimal Representation	Sign magnitude Representation	Two's complement Representation
+7	0111	0111
+6	0110	0110
+5	0101	0101
+4	0100	0100
+3	0011	0011
+2	0010	0010
+1	0001	0001
0	0000	0000
-0	1000	—
-1	1001	1111
-2	1010	1110
-3	1011	1101
-4	1100	1100
-5	1101	1011
-6	1110	1010
-7	1111	1001
-8	—	1000

A useful illustration of the 2's complement representation is a value box, in which the value on the right in the box is 1 (2^0) and each succeeding position to the left is double in value, until the left most position, which is negated figure (3-2 (a)) Most Negative 2's complement number that can be Represented is -2^{n-1}

is any of the bits other than the sign bit is 1, it adds a positive amount to the number. Also it is clear that a negative number must have a 1 at its leftmost position and a position number must have 0 in that position. Thus, the largest positive number is a 0 followed by all 1's. which equal $2^{n-1}-1$.

128	64	32	16	8	4	2	1

(a) Eight position two's complement value box

-128	64	32	16	8	4	2	1
1	0	0	0	0	1	1	1

-128 +2 +1 = -125

(b) Convert Binary 10000011 to Decimal

-128	64	32	16	8	4	2	1
1	0	0	0	0	0	0	0

-120 = -128 +8

(c) Convert Decimal -120 to Binary

Use of value box for conversion between 2's complement binary and decimal

Remainder of illustrates the use of the value box to convert from 2's complement to decimal and from decimal to 2's complement.

(13) Add 9_{10} and -23_{10} using 2's complement arithmetic.

```
0 0 0 0 1 0 0 1      9
1 1 1 0 1 0 0 1    (-23)
1 1 1 1 0 0 1 0    (-14)10
```

(14) Find the sum of $(23)_{10}$ and -9_{10} in binary using 2's complement arithmetic

```
   1 <- 1 1 1 1 1 1
Discard 0 0 0 1 0 1 1 1    (23)
carry   1 1 1 1 0 1 1 1    (-9)
        0 0 0 0 1 1 1 0    (14)10
```

Addition and subtraction of Signed numbers :-

Let the magnitude of number be A and B. When Signed numbers are considered there are eight different conditions to consider, depending on the Sign of numbers operations performed as shown in table - 3.2

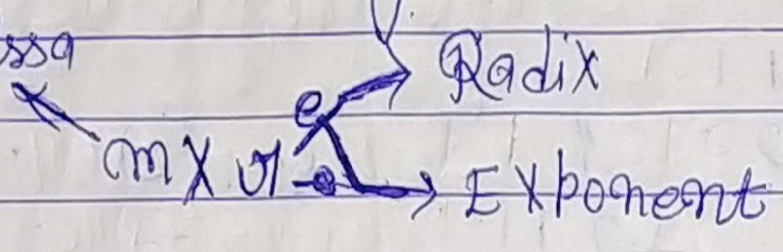
Operation	Add Magnitude	when A>B	when A<B	when A=B
(+A) + (+B)	+ (A+B)			
(+A) + (-B)		+ (A-B)	-(B-A)	-(A+B)
(-A) + (+B)		-(A-B)	+(B-A)	+(A-B)
(-A) + (-B)	-(A+B)			
(+A) - (+B)		+(A-B)	-(B-A)	+(A-B)
(+A) - (-B)	+ (A+B)			
(-A) - (+B)	-(A+B)			
(-A) - (-B)		-(A-B)	+(B-A)	+(A-B)

Floating point representation

Floating point Representation of a number has two Parts

(i) Mantissa : it represents signed, fixed-point number

(ii) Exponent : Position of the decimal (or binary) Point and is called the exponent. The fixed Point mantissa may be a fraction or an integer



Floating point is always interpreted a number in the following form. only the mantissa 'm' and the exponent 'e' are physically represented in the register. The radix 'r' and radix point position of the mantissa are always assumed.

for ex → The decimal number +6132.789 is represented in floating point with a fraction and an exponent as follows.

fraction	Exponent
+0.6132789	+04

10^4

A floating point binary number is represented in a similar manner except that is uses base 2 for the exponent.

Fraction has a 0 in the leftmost Position to denote positive. The binary Point of the fraction follows the sign bit but is not shown in the register. The exponent has the equivalent binary number +4. The floating point number is equivalent to:

$$m \times 2^{4} = +(.1001110)_2 \times 2^{+4}$$

The floating Point number is said to normalize if the most signification digit of the mantissa non-zero. For example → decimal number .350 is normalized but 00035 is not. Regardless of where the Position of the radix Point is assumed to be in the mantissa. the number is normalized only if its leftmost digit is non-zero.

Floating point addition and subtraction —;

In floating point arithmetic, addition, and subtraction are more complex than multiplication and division This is because of the need for subtraction.

1. check for zero
2. Align the significands
3. Add or subtract the significants
4. Normalized the result.

for example —;

Concider the unsigned floating Point addition $(.101 \times 2^3 + .111 \times 2^4)$ which the significant have three significant digits.

Addition and Subtraction using Guard, Round and Stick bits –;

0.101×2^3
$+ 0.111 \times 2^4$

0.0101×2^4
$+ 0.1110 \times 2^4$
1.0011×2^4

$.10011 \times 2^5$ GRS

Ex 22 floating point Arithmetic –;

(i) $.1011 \times 2^7 + .0111 \times 2^4$
(ii) $.1011 \times 2^7 - .0111 \times 2^4$

Addition

(i) $.1011 \times 2^7 + .0111 \times 2^4 = .0.1011000 \times 2^7 + 0.0000111 \times 2^7$
$.1011111 \times 2^7$

Subtraction

$.1011 \times 2^7 - .0111 \times 2^7 = 0.1011000 \times 2^7 - .0000111 \times 2^7$
0.1010001×2^7

Q23 Perform the following operation –;

(i) A+B
(ii) A-B

given $A = 225$
$B = 30$

2	225	R
2	112	1
2	56	0
2	28	0
2	14	0
2	7	0
2	3	1

$A = 225 = 11100001 = 1.1100001 \times 2^7$
$B = 30 = 1110 = 1.110 \times 2^3$

$A = 1.1100001 \times 2^7$
$B = 0.0001110 \times 2^7$
$A+B = 1.1101111 \times 2^7$

$A = 1.1100001 \times 2^7$
$B = 0.0001110 \times 2^7$
$A-B = 1.1001001 \times 2^7$

Computer instructions are a set of machine language instructions that a particular Processor understands and executes. A computer performs tasks on the basis of the instruction provided.

An instrations comprises of groups called fields. These fields include:

- The operation code (opcode) field which specifies the operation to be performed.
- The address field which contains the location of the operand, i.e. register or memory location.
- The mode field which specifies how the operand will be located

Instruction Representation - Within the computer each instruction is represented by a sequence of bits. The instruction is divided into fields, corresponding to the consitituent elements of the instruction. during instruction execution, an instruction is read

into an instruction register (IR) in the cpu. The cpu must be able to extract the data from the various instruction fields to perform the required operation.
it is difficult for the programmer and the reader of textbooks to deal with binary representations of machine instructions. Thus, it has become common practice to use a symbolic representation of machine instructions. opcodes are represented by abbreviations called mnemonics, that indicate the operation common examples include

A set of instructions is said to be complete if the computer includes a sufficient number of instruction in each of the following categories
- Arithmetic, logical and shift instructions
- A set of instructions for moving information to and from memory and processor registers.
- Instruction which controls the program together with instructions that check status conditions
- Input and output instructions.

Arithmetic, logic and shift instructions provide computational capabilities for processing the type of data the user may wish to employ. A huge amount of binary information is stored in the memory unit, but all computations are done in processor registers. Therefore, one must passes the capability of moving information

Program control instructions such as branch instructions are used change the sequence in which the Program is executed.
Input and output instructions act as an interface between the computer and the user. Programs and data must be transferred into memory, and the results of computations must be transferred back to the user

Type of operands —: Machine instructions operate on data. The most important general categories of data are —:
Addresses
Numbers
Characters
Logical data

Number —: All machine language include numeric data types. Even in nonnumeric data Processing, there is a need for number to act as counters, field width, and so forth. An important distinction between numbers stored in a computer is that the latter are limited. Thus, the programmer is faced with understanding the consequences of rounding, overflow and underflow.
Three type of numerical data are common in computers.
Integer or fixed point
Floating point

Logical data :- Normally each word or other addressable unit (byte, half-word, and soon) is treated as a single unit of data. It is sometimes useful, however, to consider an n-bit unit as consisting 1-bit items of data, each item having the value 0 or 1. When data are viewed this way, they are considered to be logic data. There are two advantages to the bit-oriented view.

First, we may sometimes wish to store an array of boolean or binary data items; in which each item can take on only the values 1 (true) and 0 (false) with logical data, memory can be used most efficiently for this storage.

Second there are occasions when we wish to manipulate the bits of a data item.

Type of operations :- The Number of different opcodes varies widely from machine. However, the same general type of operations are found on all machines. A useful and typical categorization is the following.

Data transfer	System control
Arithmetic	Transfer of control
Logical	
Conversion	
I/O	

Data transfer :- The most fundamental type of machine instruction is the data transfer instruction. The data transfer instruction must specify several things. The location of the source and destination operands must be specified. Each location could be memory. A register or the top of the stack. The length of data to be transferred must be indicated. As with all instructions with operands, the mode of addressing for each operand must be specified.

In term of cpu action, data transfer operations are perhaps the simplest type. If both source and destination are registers, then the cpu simply causes data to be transferred from one register to another, this is an operation internal to the cpu. If one or both operands are in memory then the cpu must perform some or all of following actions.

1. Calculate the memory address, based on the address mode.
2. If the address refers to virtual memory translate from virtual to actual memory address
3. Determine whether the addressed item is in cache.
4. If not, issue a command to the memory module.

Ex

operation mnemonic	Name	Number of bits transferred	Description
L	Load	32	Transfer from memory register
LH	Load half-word	16	Transfer from memory to register
ST	Store	32	Transfer from register to memory

Logical :- Most machines also provide a variety of operations for manipulating individual bits of a word or other addressable units, often referred to as bit twiddling : They are based upon boolean operations.
Some of the basic logical operations that can be performed on boolean or binary data are AND, OR, NOT, XOR, ...
These logical operations can be applied bitwise to n-bit logical data units.
Thus, if two registers contain data.
(R_1) - 10100101 (R_2) - 00001111
then
R_1 . AND . R_2 - 00000101
In addition to bitwise logical operations most machines Provide a variety of shifting and rotating functions such as Shift left - Shift right, right rotate, left rotate...

Arithmetic :- Most machines Provide the basic arithmetic operations of add, subtract,

multiply and divide. These are invariably provided for signed integer (fixed point) numbers. Often they are also provided for floating-point and packed decimal numbers.
other possible operations include a variety of single-operand instructions: for example —:
- Absolute: Take the absolute value of the operand.
- Negate: Negate the operand.
- Increment: Add 1 to the operand.
- Decrement: Subtract 1 from the operand.

Conversion: conversion instructions are those that change the format or operate on the format of data. An example is converting from decimal to binary.

Input/output -: As we saw, there are a variety of approaches taken, including isolated programmed I/O, memory-mapped programm I/O, DMA, and the use of an I/O Processor. many implementations provide only a few I/O instructions, with the specific actions specified by parameters, codes, or command words.

System controls -: System control instructions are those that can be executed only while the processor is in a certain privileged state or is executing a program in a specical privileged area of memory typically. these instructions are reserved for the use

Some examples of system control operations are as follows. A system control instruction may read or alter a control register. Another example is an instruction to read or modify a storage protection key, such as is used in the S/390 memory system. Another example is access to process control blocks in a multiprogramming system.

Transfer of control — For all the operation types discussed so far the next instruction to be performed is the one that immediately follows, in memory, the current instruction. However, a significant fraction of the instructions in any program have as their function for these instructions the operation performed by the cpu is to update the program counter to contain the address of some instruction in memory.

Branch Instruction — A branch instruction also called a jump instruction, has as one of its operands the address of the next instruction to be execute Most often, the instruction is a conditional branch instruction. That is, the branch is made (update program counter to equal address specified in operand) only if a certain condition is met. otherwise is executed (increment Program Counter as usual)

Skip instructions :- Another common form of transfer-of-control instruction is the skip instruction. The skip instruction includes an implied address typically, the skip implies that one instruction be skipped, thus the implied address equals the address of the next instruction plus one instruction-length.

Characters :- A common form of data is text or character strings. While textual data are most convenient for human beings, they cannot, in character form, be easily stored or transmitted by data processing and designed for binary data. Thus, A number of codes have been devised by which characters are represented by a sequence of bits. Perhaps the earliest common example of this is the Morse code. Today, the most commonly used character code is the International Reference Alphabet (IRA). Referred to in the United States as the American Stan-dard code for Information Interchange (ASCII) IRA is also widely used outside the United States. Each character in this code is represented by a unique 7-bit pattern, this, 128 different characters can be represented. This is a larger number than is necessary to represented Printable characters, and some of the patterns represented control characters. Some of these represent control characters. Some of these control characters

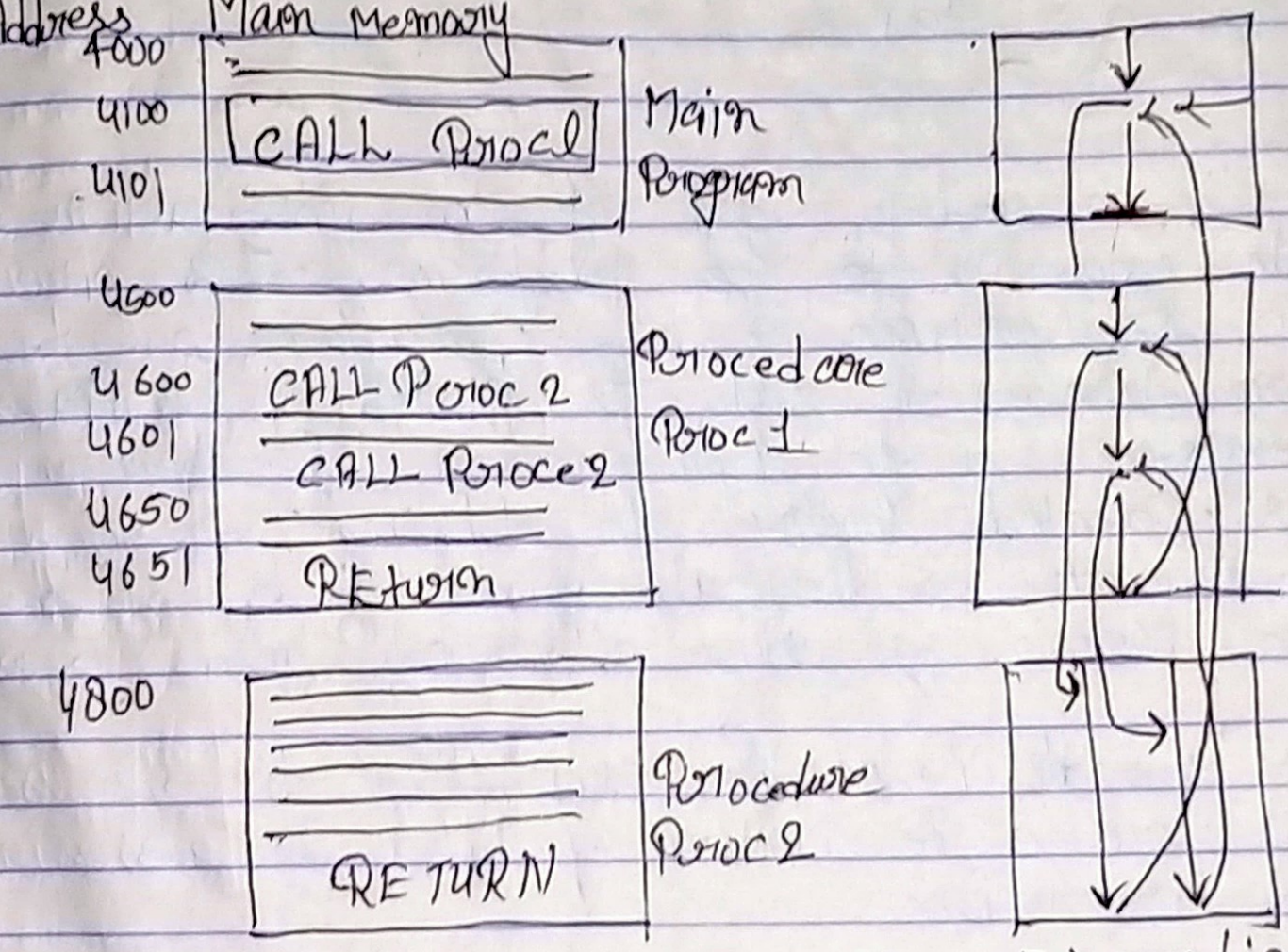
on a page. others are concerned with communications Procedures. IRA-encoded characters are always stored and transmitted using 8 bits per character. The eighth bit may be set to 0 or used as a parity bit for error detection. In the latter case, the bit is set such that the total number of binary 1s in each octet is always odd or always even.

Another code used to encode characters is the Extended binary coded Decimal interchange code (EBCDIC) EBCDIC is used on IBM S/390 machines. It is an 8-bit code. As with IRA, EBCDIC is compatible with packed decimal. In the case of EBCDIC, the code 11110000 through 11111001 represent the digits 0 through 9.

Procedure call Instructions —: Perhaps the most important innovation for the development of programming languages is the procedure, a procedure is a self-contained computer program that is incorporated into a larger program. At any point in the program the procedure may be invoked, or called. The processor is instructed to go and executed the entire procedure and then return to the point from which the call took place.

The two principal reasons for the use of procedures are economy and modularity. A procedure allows the same piece of code to be used many times. This is important for economy in programming

effort and for making the most efficient use of storage space in the system (the program must be stored). Procedures also allow large programming tasks to be subdivided into smaller units. The use of modularity greatly eases the programming task. The procedure mechanism involves two basic instructions: a call instruction that branches from the present location to the procedure, and a return instruction that returns from the procedure to the place from which it was called. Both of these are forms of branching instructions.



1. A Procedure can be called from more than one location.
2. A Procedure call can appear in a procedure. This allows the nesting of procedures to an arbitrary depth.
3. Each Procedure call is matched by a return in the called program.

Because we could like to be able to call a Procedure from a variety of Points. the CPU must somehow save the return address so that the return can take Place appropriately. There are three common Places for storing the return address.

- Register
- Start of called Procedure
- Top of stack

The operation field of an instruction specifies the operation to be performed. This operation to be performed. This operation will be executed on some data which is stored in computer registers or the Main memory. The copy any operand is selected during the program execution is dependent on the addressing mode of the instruction The purpose of using addressing mode is as follows

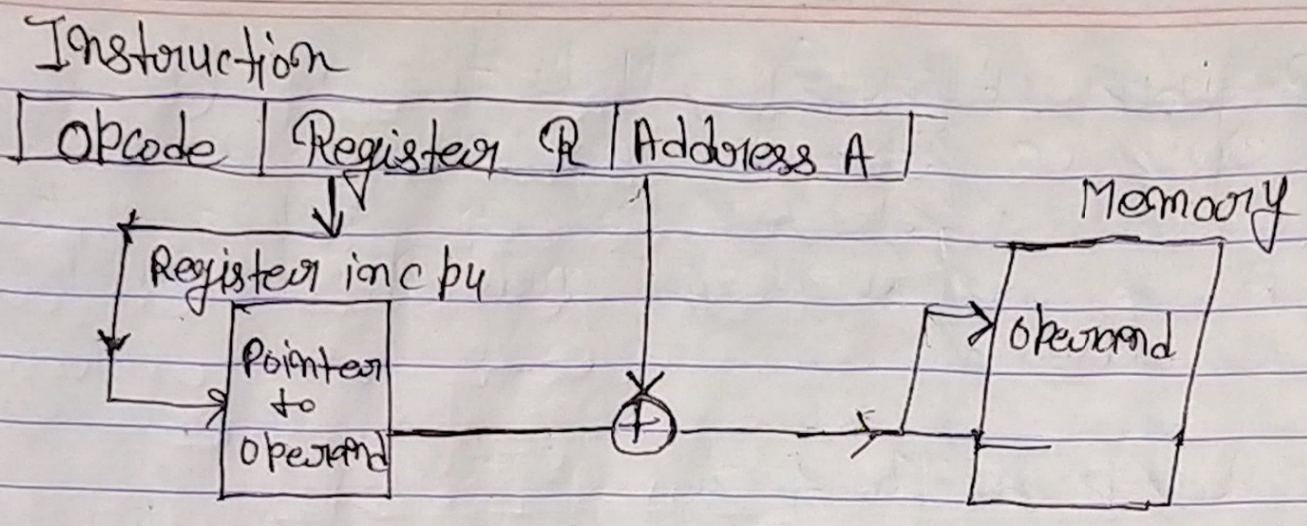
1. To give the programming versatility to the user
2. To reduce the number of bits in addressing field of instruction.

Base Register Addressing Mode :-
It is again a version of Displacement addressing Mode. This can be defined as EA = A + (R), where A is displacement and R holds Pointer to base address.

Stack Addressing Mode :-
In this mode, operand is at the top of the stack. for example : ADD, this instruction will pop top two items from the stack, add them, and will then Push the result to the top of the stack

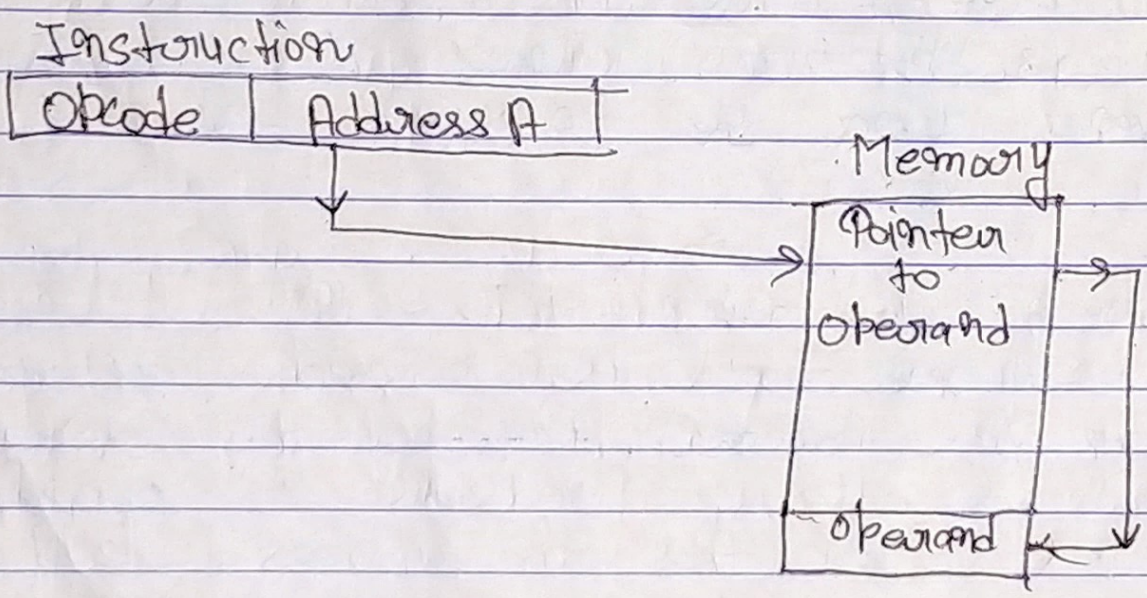
Relative Addressing Mode :-
It is a version of Displacement addressing Mode. In this the contents of Pc (Program counter) is added to address part of instruction to obtain the effective address.
EA = A + (PC), where EA is effective address and Pc is program counter The operand is A cells away from the current cell (the one Pointed to by Pc)

Displacement addressing Mode
In this contents of the indexed register is added to the Address part of the instruction to obtain the effecive address of operand.
EA = A + (R), In this the address field holds two values, A (which is the base value) and R (that holds the displacement) or vice versa



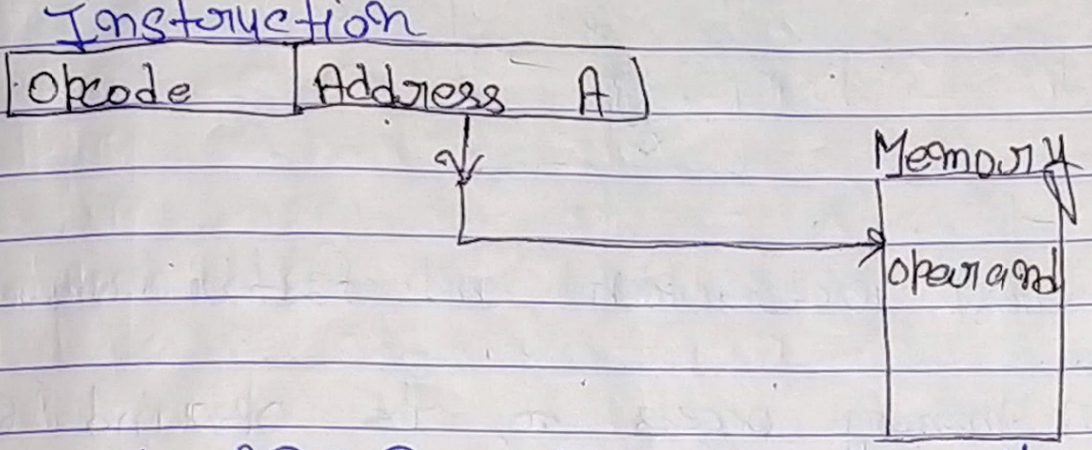
Note ~~Effective Address is the location where operand is present~~.

Indirect Addressing Mode :-
In this, the address field of instruction gives the address where the effective address is stored in memory. This slows down the execution, as the includes multiple memory lookup to find the operand.



Direct Addressing Mode :-
In this Mode, effective address of operand is present in instruction itself.

- Single memory reference to access data.
- No additional calculations to find the effective address of the operand.

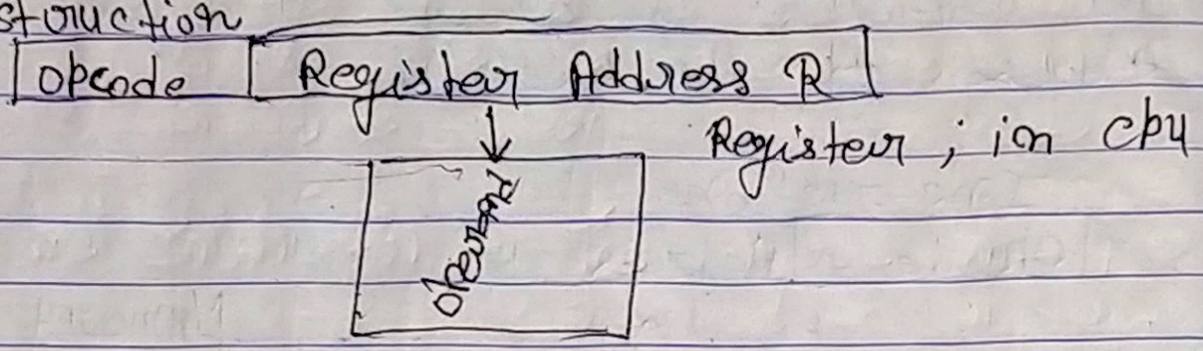


for ex :- ADD R1, 4000 - In this the 4000 is effective address of operand.

Effective Address is the location where operand is present

Immediat Mode :- In this mode, the operand is specified in the instruction itself. An immediate mode instruction has an operand field rather than the address field. for example ADD 7, which says Add 7 to contents of accumulator. 7 is the operand here.

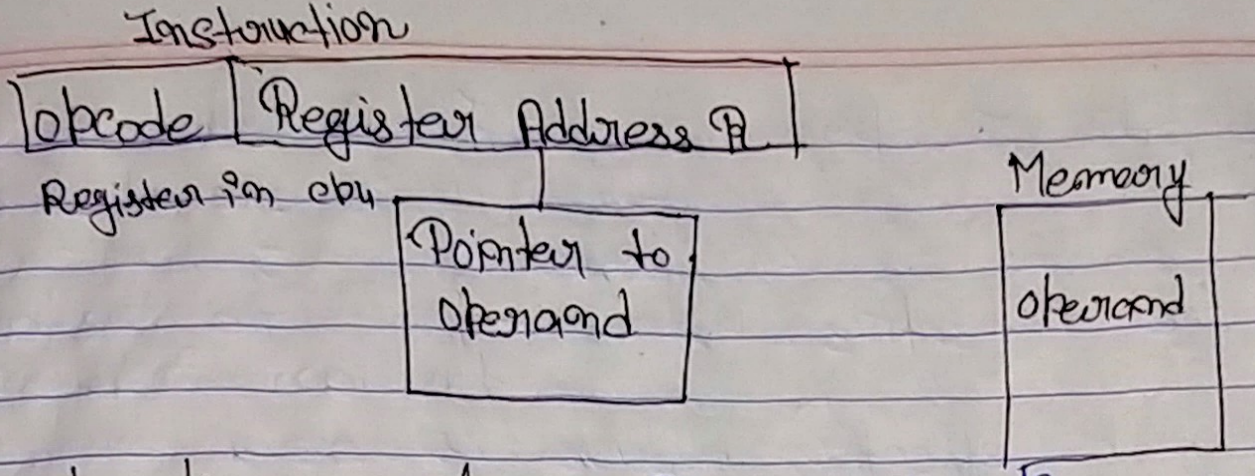
Register Mode :- In this mode the operand is stored in the register and this register is represent in cpu. The instruction has the address of the Register where the operand is stored.



Advantages :-
- Shorter instruction and faster instruction fetch
- Faster memory access to the operand

Disadvantages :-
- very limited address space
- using multiple registers helps performance but it complicates the instructions.

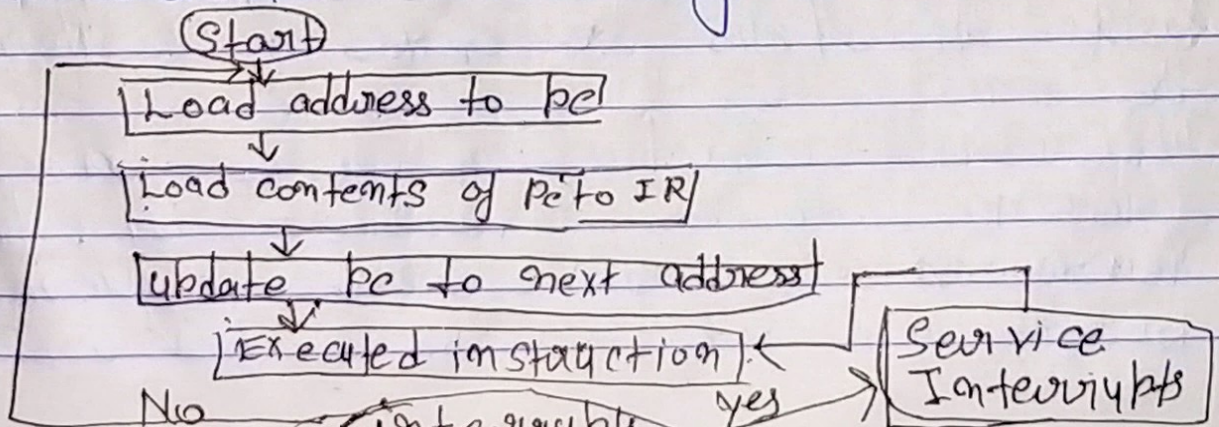
Register Indirect Mode :-
In this mode, The instruction specifies the register whose contents give us the address of operand which is in memory. Thus, the register contains the address of operand rather the operand itself.



Instruction cycle :- An instruction cycle, also known as fetch decode-execute cycle is the basic operational process of a computer. This process is repeated continuously by cpu from boot up to shut down of computer. Following are the steps that occur during an instruction cycle.

Fetch the Instruction :- The Instruction is fetched from memory address that is stored in Program counter (PC) and stored in the Instruction register IR. At the end of the fetch operation PC is incremented by 1 and it then points to the next instruction to be executed.

The cycle is then repeated by fetching the next instruction. Thus in this way the instruction cycle is repeated continuously.



Decode the Instruction –; The instruction in the IR is executed by the decoder.

Read the Effective Address –; if the instruction has an indirect address, the effective address is read from the memory otherwise operands are directly read in case of immediate operand instruction.

Execute the Instruction –; The control unit passes the information in the form of control signals to the functional unit of cpu. The result generated is stored in main memory or sent to an output device.

Computer organization / Von Neumann architecture –;

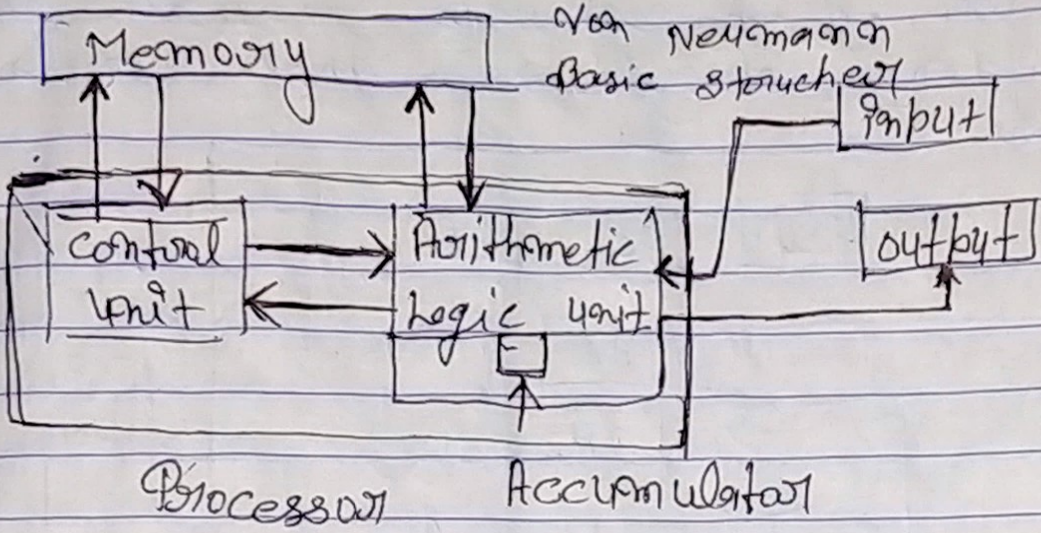
Historically there have been 2 types of computers

1. Fixed Program computer –; Their function is very specific and they could not be programmed, e.g. calculators.

2. Stored Program computers –; These can be programmed to carry out many different tasks, applications are stored on them, hence the name.

The modern computer are based on a stored program concept introduced by John von Neumann. In this stored program concept, program and data are stored in a separate

storage unit called memories and are treated the same. This novel idea meant that a computer built with this architecture would be much easier to reprogram.
The basic structure is like,



It is also known as IAS computer and is having three basic units.

1. The control processing unit (CPU)
2. The main memory unit
3. The Input/output Device

Let's consider them in details.

Control unit :- A control unit (CU) handles all processor control signals. It directs all input and output flow, fetches code for instructions and controlling how data moves around the system.

Arithmetic and logic unit (ALU) :-

The arithmetic logic unit is that part of the cpu that handals all the calculations the cpu may need, e.g. Addition, Subtraction, Comparisons. It performs logical operations, Bit shifting operations, and arithmetic operations.

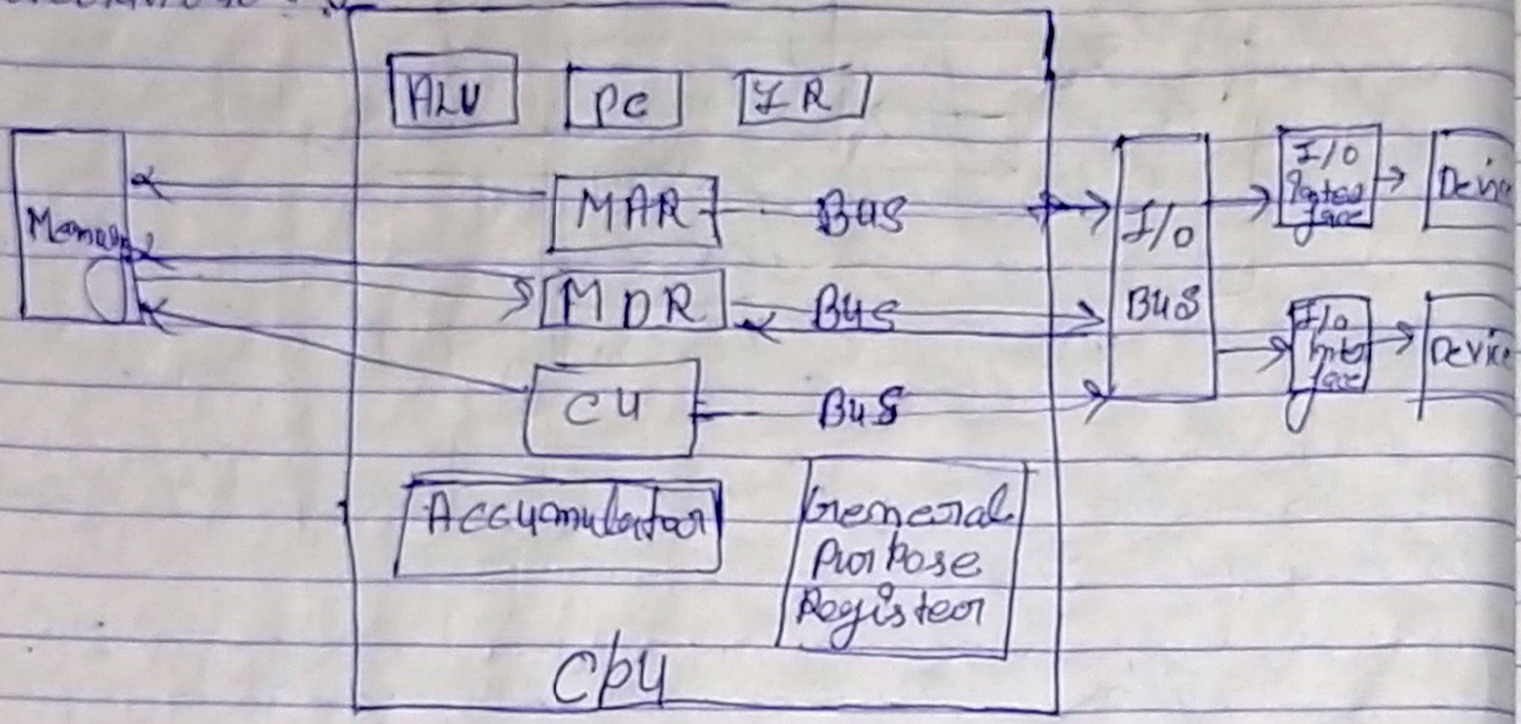


Figure Basic CPU Structure; illustrating ALU.

Main memory unit (Registers) :-

1. Accumulator :- Stores the results of calculations Made by ALU.
2. Program Counter (PC) :- Keep track of the Memory locations of the next instructions to be dealt with The PC then Passes this next address to memory

Memory address Register (MAR) :-

It stores the memory locations of instructions that need to be fetched from memory or stored into memory.

Memory Data Register (MDR) :-

It stores instructions fetched from memory or any data that is to be transferred to, and stored in memory.

5. Current Instruction Register (CIR) :-

It stores the most recently fetched instructions while it is waiting to be coded and executed.

6. Instruction Buffer Register (IBR) :-

The instruction that is not to be executed immediately is placed in the instruction buffer register IBR.

BUSES :- Data is transmitted from one part of a computer to another connecting all major internal components to the CPU and memory by the means of Buses Type :-

Data BUS :- it carries data among the memory unit, the I/O devices and the Processor

Address Bus -: It carries the address of data (not the actual data) between Memory and processor.

control Bus -: It carries control commands from the CPU (and status signals from other devices) in order to controll and coordinate all the activities within the compute

Booth's Algorithm -: Booth's algorithem provides step to perform multiplication among binary number these binary numbers are in 2's complement representation. using this algorithem the will be no need to convert the final result into its 2's complement even if the sign the multiplier and the multiplicand are that the same.

Steps booth algorithm multiplication of two 4-Bit binary Number.

Step 1 - The multiplier and the multiplicand are placed in registers Q and respectively Q and M registers are of 4 bits each.

Step 2 - A 1 bit register Q_1 is placed logically to the right of least significant bit Q_0 of register Q. Remember least significant bit is the right most bit

Step 3 -: An 8 bit register A is also used. Both A and Q_1 are initialized to zero (The result of multiplication will be placed in A and Q_1

A counter c is set to a number equal to the number of bits in the multiplier ie. 4

Step 4 :- Bits in the multiplier Q are scanned

Step 5 :- The rightmost bit in Q i.e. Q_0 is compared with the bit in register Q_1. if these bits are the same i.e 1-1 or 0-0 then the bits in the register A, Q and Q_1 are shifted to right once. The shift performed is arithmatic the value of sign bit is taken into consideration.

Step 3 :- If the bits are different then the multiplicand (bit in M) is added to or subtracted from the bits in the register A if the bits are 0-1 then the multiplicand is added to A and if the bits are 1-0 then the multiplicand is subtracted from. After doing addition or subtraction, the bit in A, Q and Q_1, are arithmetically shifted right.

Step 4 :- After this, the counter c is decrement by one (In our case, counter c was initially initiallized to 4.)

Ex 5 Show the step by step multiplication using Booth's algorithem for the following multiplication $(-11) * (-8)$

Solution Booth multibilication Algorithem

$(+8)_{10} = (1000)_2$

$(-8)_{10} = (0111)_2$

1

0111 2's com

1

$(1000)_2$

Similarly $(+11)_{10} = (1011)_2$

$(-11)_{10} = (0100)_2$

0100 , 2's com

1

$(-11)_{10} = (0101)_2$

Multiplicand (B) = (0101)(-11) and Multiplier Q ← (1000)(-8)

Steps	A	Q	Q-1	Operation
	0000	1000	0	Initial
Step 1	0000	0100	0	Right Shift
Step 2	0000	0010	0	"
Step 3	0000	0001	0	"
Step 4	1011	0001	0	A ← A − B
	0101	1000	1	Right Shift
Result 01011000 = +88				

Ex 5 Multiply 85 x 59 using booth's algorithem

Multiplicand = 85 = 01010101, 2's complement = 10101011

Multiplier 59 = 00111011

Iteration	Step/Section	Accumulator	Multiplier	Q_{n-1}
0	Initial values	00000000	00111011	0
1	(a) Subtract M's	10101011	00111011	0
	(b) Shift right	11010101	10011101	1
2	(a) No action	11010101	10011101	1
	(b) Shift right	11101010	11001110	1
3	(a) Add M+ 2's	00111111	11001110	1
	(b) Shift right	11001010	11100111	0
4	(a) Subtract M	11001010	11100111	0
	(b) Shift right	11100101	01110011	1

5	(a)	No action	11100101	01110011	0
	(b)	Shift right	11110010	10111001	1
6	(a)	No action	11110010	10111001	1
	(b)	Shift right	11111001	01011100	1
7	(a)	Add MD	01001110	01011100	1
		Shift right	00100111	00101110	0
8	(a)	No action	00100111	00101110	0
	(b)	Shift right	00010011	10010111	0

Product — ignored

Product = 00010011 (10010111)
= 5015

Table Example of Booth's algorithem (7×3)

M = 7 = 0111

Q = 3 = 0011

Step	A	Q	Q_{-1}	M	Initial values	
	0000	0011	0	0111		
Step1	1001	0011	0	0111	A ← A−M	First cycle
	1100	1001	1	0111	Shift	
Step2	1110	0100	1	0111	Shift	Second cycle
Step3	0101	0100	1	0111	A ← A+M	third cycle
Step4	0010	1010	0	0111	Shift	
	0001	0101	0	0111	Shift	fourth cycle
	Result 00010101 = +21					

II Unit

Parallel computing :- It is the use of multiple Processing elements simultaneously for Solving any Problem. Problems are broken down into instructions and are solved concurrently as each resource which has been applied to work is working at the same time.

Advantages :- of Parallel computing over serial computing are as follows.

1. It saves time and money as many resources working together will reduce the time and cut potential costs.
2. It can be impractical to solve larger Problems on Serial computing.
3. It can take advantage of non-local resources when the local resources are finite.
4. Serial computing 'wastes' the Potential computing Power, thus Parallel computing makes better work of hardware.

* Types of Parallelism :-

1. Bit level Parallelism :- It the form of Parallel computing which is based on the increasing Processor's size. It reduces the number of instructions that the system must execute in order to perform a task on large-sized data. Ex consider a scenario where an 8-bit Processor must

compute the sum of 16 bit integers. It must first sub up the 8 lower-order bits, then add the 8 higher-order bits, thus requiring two instructions to perform the operation. A 16-bit Processor can perform the operation with just one instruction.

Instruction-level Parallelism → A Processor can only address less than one instruction per each clock cycle Phase. These instruction can be re-order and grouped which are later on executed concurrently without affecting the result of the Program This is called instruction-level Parallelism.

Task Parallelism → Task Parallelism employs the decomposition of a task into subtasks and then allocating each of the subtasks for execution. The Processors perform execution of sub tasks concurrently.

Why Parallel computing → (1) The whole real world runs in dynamic nature ie many things happen at a certain time but at different places concurrently. This data is extensively huge to manage.
(2) Real world data needs more dynamic simulation and modeling and for achieving the same Parallel computing is the key.
(3) Parallel computing Provides concurrency and saves time and money.

4) Complex, large datasets, and their management can be organized only and only using parallel computing's approach.
5. ensures the effective utilization of the resources. The hardware is guaranteed to be used effectively whereas in serial computing only some part of hardware was used and the rest rendered idle
6. Also, it is impractical to implement real-time systems using serial computing.

Application of Parallel computing :-

- Data bases and data Mining.
- Real time simulation of systems.
- Scicne and Engineering.
- Advanced graphics, augmented reality and virtual reality.

Limitations of Parallel computing :-

- it address such as communication and synchronization between multiple sub-tasks and processes which is difficult to achieve
- The algorithem must be managed in such a way that they can be handled in the Parallel mechanism.
- The algorithem or Program must have low coupling and high cohesion. But it's difficult to create such Program.
- More technically skilled and expert Programmers can code a Parallelism based

*Architecture of 8086 —; A Microprocessor is an Integrated circuit with all the function of a CPU. however, It cannot be used stand alone since unlike a microcontroller It has no memory or Peripherals.

8086 does not have a RAM or ROM inside It However, it has internal registers for storing intermediate and final results and interfaces with memory located outside it through the system BUS.

In case of 8086, it is a 16 bit Integer Processor in a 40 pin, Dual Inline Packaged I.C. ~~In case of 8086~~,

The size of the internal registers (Present with in the chip) indicate how much information the Processor can operate on at a time (In this case, 16-bit registers) and how it moves data around internally with in the chip, sometimes also referred to as the internal data bus.

8086 Provides the Programmer with 14 internal registers, each 16 bits or 2 bytes wide.

Memory Segmentation —; (1) To increase execution speed and fetching speed 8086 Segments the memory.

2. It's 20 bit address bus can address 1MB of memory, it Segments it into 64 KB Segment

8086 works only with four 64KB segments with in the whole 1MB memory.

The Internal architecture of Intel 8086 is divided into 2 units : The Bus interface unit (BIU), and The Execution unit (EU).

2 The BUS Interface Unit (BIU) :-

It Provide the interface of 8086 to external Memory and I/O devices via the system BUS. It performs various machine cycles such as memory read, I/O read etc. to transfer data between memory and I/O devices.

BIU performs the following functions :-

- It generates the 20 bit physical address for memory access.
- It fetches instructions from the Memory
- It transfers data to and from the Memory and Input output.
- Maintains the 6 byte prefetch instruction queue (Supports pipelining)

3 The Execution unit (EU) :-

The Main components of the EU are general purpose registers, the ALU, special purpose registers, Instruction Register and Instruction Decoder and the Flag/Status Register.

1- Fetches instructions from the queue in BIU, decodes and executes arithmetic and logic operations using the ALU.
2- Sends control signals for internal data transfer operations within the microprocessor
3- Sends request signals to the BIU to access the external module.
4- It operates with respect to T-States (clock cycles) and not Machine cycles.

8086 has four 16 bit general purpose registers AX, BX, CX and DX. Store intermediate values during execution Each of these have two 8 bit parts (higher and lower).

- AX register: In hold operands and results during multiplication and division operations. Also an accumulator during string operations.

BX register: It holds the memory address (offset address) in indirect addressing modes.

CX register: It holds count for instructions like loop, rotate, shift and string operations.

DX register: It is used with AX to hold 32 bit values during multiplication and division.

Arithmetic Logic Unit (16 bit) -; Peerforms 8 and 16 bit arithmetic and logic operations.

Specical Purpose registers (16-bit) -;

Stack Pointer -; Points to stack top. Stack Is In Stack Segment, used during instruction like PUSH, POP, CALL, RET .etc.

Base Pointer -; BP can hold offset address of any location in the Stack Segment. It Is used to access random locations of the Stack.

Source Index -; It holds offset address In data Segment during string operations.

Distination Index -; It hold offset address In Extra Segment during string operations.

Instruction Register and Instruction Decoder -;

The EU fetches an opcode from the queue In to the Instruction register. The Instruction register The Instruction decoder decodes It and sends the Information to the control circuit for execution.

Flag Status register (16 bits) -; It has 9 flags that help change or recognize the State of the Microprocessor.

6 Status flags -; 1. Carry flag (CF)
2. Parity flag (PF)
3. Auxiliary carry flag (AF)
4. Zero flag (Z)
5. Sign flag (S)
6. Overflow flag (O)

Status flags are updated after every arithmetic and logic operation.

3 Control flag -;
1. trap flag (TF)
2. Interrupt flag (IF)
3. Direction flag (DF)

These flags can be set or reset using control instructions like CLC, STC, CLD, STD, CLI, STI etc.
The control flags are used to control certain operations.

* In the 8086 Microprocessor, the registers are categorized into Mainly four types.

1. General Purpose registers.
2. Segment Registers
3. Pointers and index Registers.
4. Flag Status Registers.

1. General Purpose Registers :- The use of general Purpose registers is to store temporary data. While the instructions are executed in the control unit, they may work on some numeric value or some operands. These needs to be stored somewhere so that the processor can operate on them easily. So, these registers are used in these cases. There are 4 genal-Purpose registers of 16-bit length etch. Each of them is future divided into two subparts of 8-bit length each: one high, which stores the higher order bits and another low which stores the lower order bits.

i. AX = [AH : AL]
ii. BX = [BH : BL]
iii. CX = [CH : CL]
DX = [DH : DL]

2. Segment Registers :- There are 4 segment registers in 8086 Microprocessor and each of them is of 16 bit. The code and instructions are stored inside these different segments.

(1). Code Segment (CS) Register :- The user can not modify the content of these registers. only the microprocessor's compiler can do this.

(2) Data Segment (DS) Register :- The user can modify the content of the data segment.

Stack Segment (SS) Registers -; The SS is used to store the information about the memory segment. The operations of the SS are manipulate by PUSH and POP

* Extra Segment (ES) Register -; By default, the control of the compiler remains in DS where the user can add and modify the instructions. if there is less space in that segment, then ES is used. ES is also used for copying purpose.

3 Pointers and Index Registers -; The Pointers will always store some address on memory location - in 8086 Microprocessor they usually stored the offset through which the actual address is calculated.

1. Instruction Pointer (IP) -; The Instruction Pointer usually stores the address of the next instruction that is to be executed. Apart from this, it also acts as an offset for CS register.

2. Base Pointer (BP) -; The base pointer stores the base address of the memory Also, it acts as an offset for stack segment (SS)

(3) Stack Pointer (SP); The stack Pointer Points at the current top value of the stack. I like the BP,

it also acts as an offset to the state segment. The indexes are used with the extra segment and they usually are used for copying the contents of a particular block of memory to a new loca

4. Source Index (SI) :- It stores the offset address of the source.

5. Destination Index (DI) :- It stores the offset address of the destination.

4) Flag or Status Register :- The flag or status register is a 16-bit register which contains 9 flags. and the remaining 7 bits are idle in this register These flages tell about the status of the Processo after any arithmetic or logical operation. If the flag is 1, the flag is set, and if it is '0', it is said to be reset.

* Micro-operations :- The operation executed on data stored in registeres are called Micro-operations. A Micro-operation is an elementa operation performed on the information stored in one or more registers.

Ex :- Shift, count, clear and load

Types of Micro-operations :- The Micro-operations in digital computer are of four types

1. Register transfer micro operations transfer binary information from one register to another
2. Arithmetic Micro-operations perform arithmetic operations on numeric data stored in registeres.
3. Logic Micro-operations perform bit Manipulation operation on non-numeric data stored in registeres.
4. Shift micro-operations performs shift micro-operations performed on data.

Arithmetic Micro-operations :- Some of the basic Micro operations are addition, subtractions, increment and decrement.

Symbolic Designation	Description
$R3 \leftarrow R_1 + R_2$	Contents of R_1+R_2 transferred to R_3
$R_3 \leftarrow R_1 - R_2$	contents of R_1-R_2 transferred to R_3
$R_2 \leftarrow (R_2)'$	Compliment the contents of R_2
$R_2 \leftarrow (R_2)' + 1$	2's compliment the contents R_2
$R_3 \leftarrow R_1 + (R_2)' + 1$	R_1 + the 2's compliment of R_2 (Sub)
$R_1 \leftarrow R_1 + 1$	Increment the contents of R_1 by 1.
$R_1 \leftarrow R_1 - 1$	Decrement the contents of R_1 by 1.

Logic Micro operations :- These are binary Micro-operations performed on the bits stored in the registers. These operations consider each bit separately and treat them as binary variables.

Let us consider the X-OR Micro-operations with the

$P: R_1 \leftarrow R_1$ X-OR R_2

In the above statement we have also included a control function.
Assume that each register has 3 bits. Let the content of R_1 be 010 and R_2 be 100, the X-OR Micro-operation will be:

010 → R1
100 → R2
110 → R_1 after P=1

Shift Micro-operations → These are used for serial transfer of data. That means we can shift the contents of the Register to the left or right. In the shift left operation the serial input transfers a bit to the right most position and in shift right operation the serial input transfers a bit to the left most position. There are three types of shifts as follows.

(a) Logical shift → It transfers 0 through the serial input. The symbol "shl" is used for logical shift left and "shr" is used for logical shift right.

$R_1 \leftarrow shl\ R_1$
$R_1 \leftarrow shr\ R_1$

The register symbol must be same on both sides of arrows.

(b) Circular Shift → This circulates or rotates the bits of register around the two ends without any loss of data or contents. In this,

The serial output of the shift register is connected to its serial input. "cil" and "cir" is used for circular shift left and right respectively.

(c) Arithmetic Shift :- The shifts a signed binary number to left or right. An arithmetic shift left multiplies a signed binary number by 2 and shift left devides the number by 2. Arithmetic shift micro-operation leaves the sign bit unchanged because the signed number remains same when it is multiplied or divided by 2.

* Register Transfer :- Information transferred from one register to another is designated in symbolic form by means of replacement operator.

$$R_2 \leftarrow R_1$$

It denotes the transfer of the data from register R_1 into R_2.

Normally we wants the transfer to occur only in predetermined control condition. This can be shown by following if - then statement. if $(P=1)$ then $(R_2 \leftarrow R_1)$.

* Here P is a control signal generated in the control section.

System bus :- This consists of data bus address bus and control bus.

Address bus - This is used to carry the address of data for the memory and its width is equal to the number of bits in the MAR of the memory ; if comp. memory of 64K has 32 bi words then the computer will have a data bus of 32 bits wide and the address bus of 16 bit wide.

Control Bus - ; carries the control signals between the various units of the computer

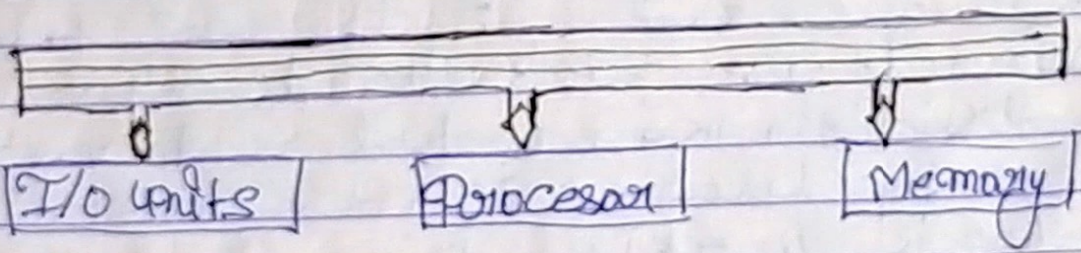
Ex - ; Memory Read/write , I/O Read/write

Ex - ;

Two Types of Bus organizations -;

- Single Bus organization
- Two Bus organization

Single Bus Architecture -;



1- Single Bus organization -;

- Three units share the single bus. At any given point of time, information can be transferred between any two units.
- Here I/O units use the same memory address space (Memory Mapped I/O).

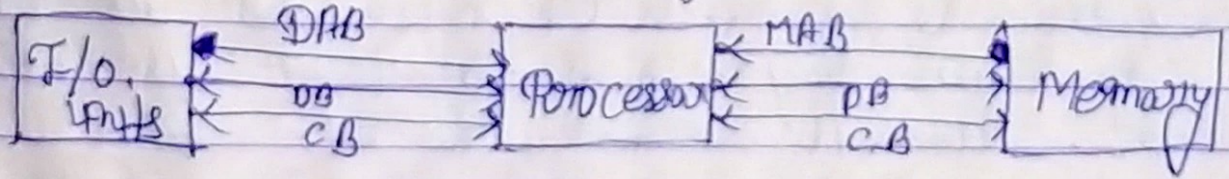
- So no special instructions are required to address the I/O, it can be accessed like a Memory location.
- Since all the devices do not operate at the same speed, it is nessary to smooth out the differences in timings among all the devices A common aproch used is to include buffer registers with the devices to hold the information during transfers

Ex:- Communication between the processor and printer.

:- Two BUS Architecture -:

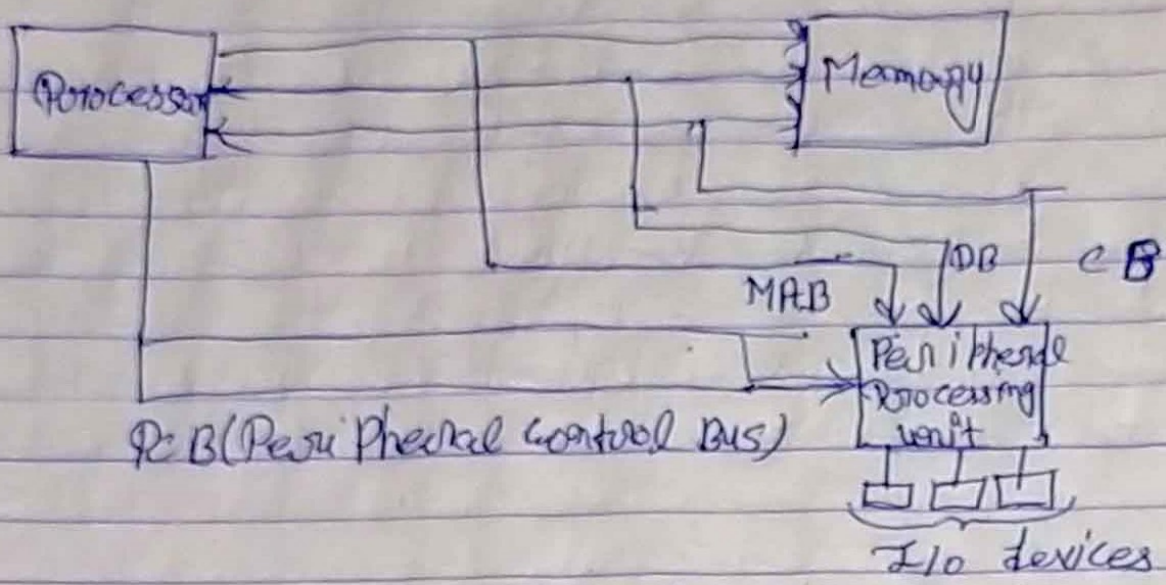
:- Two BUS orgination -:

- Various units are connected through two independent buses.
- I/O units are connected to the Processor through and I/O bus and Memory is connected to the Processor through the memory bus.
- I/O bus consists of address, data and control bus Memory bus also consists of address, data and control bus In this type of arrangements Processor completly supervises the transfer of information to and from I/O units. All the information is first taken to processor and from there to the memory. Such kind of transfers are called as program controlled transfer.



Two BUS orgination

Alternative Two BUS Architecture :-

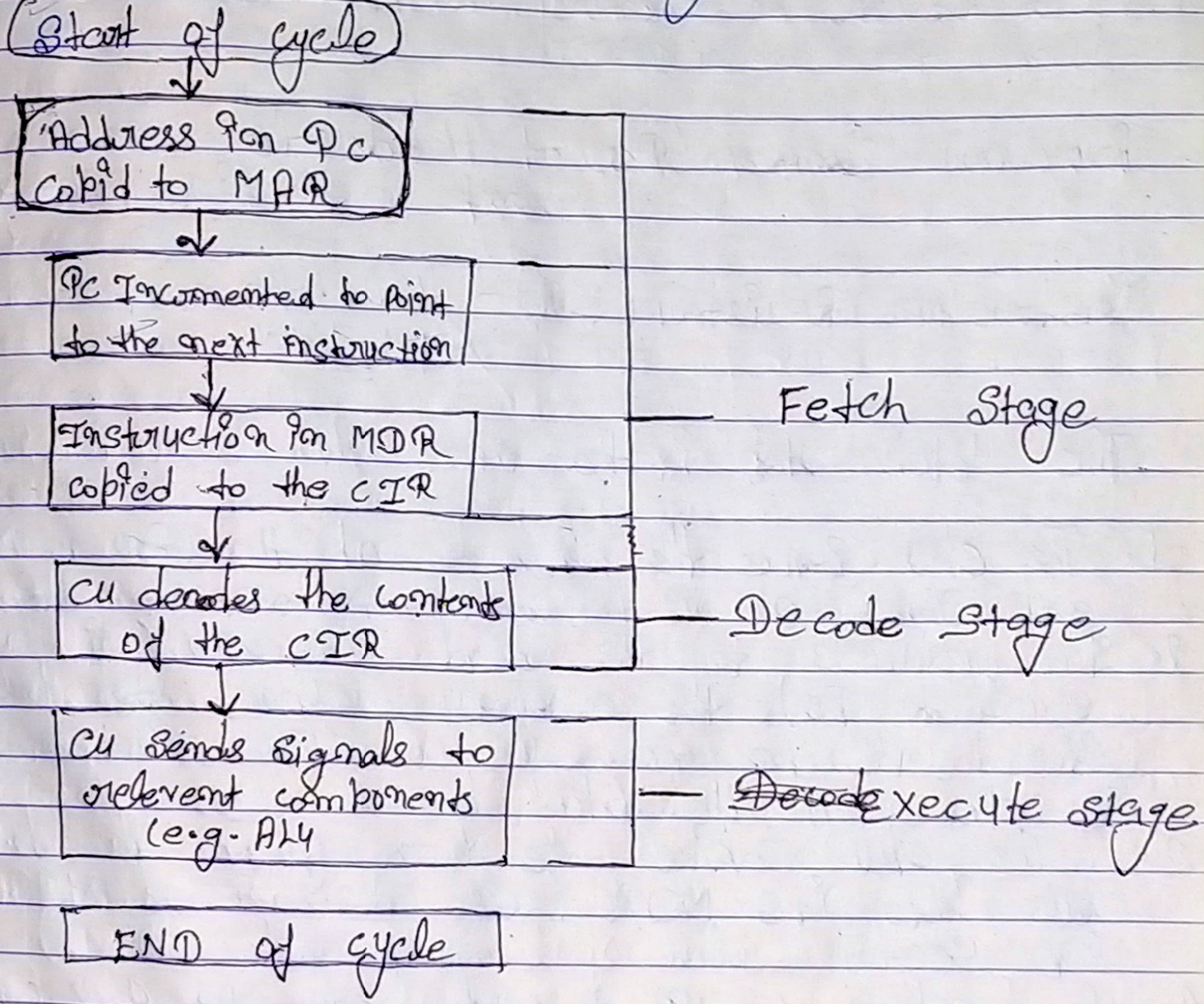


Alternative Two BUS organization

In this I/O units are directly connected to the memory and not the Processor.
This I/O units are connected to special interface logic known as direct Memory Access (DMA) or an I/O channel. This is also called as Peripheral Processor unit (PPU) In this data from the I/O devices is directly send to memory bypassing the Processor.

BUS Structure :- If different functional units of a computer are connected in a organization way than only the computer will carry out the tasks. There are different ways in connecting. When a coard of data is to transferred between units, all its bits are transferred in Parallel, so this requires many number of wires. A collection of lines that connects several devices is called a bus.

The instruction cycle (also known as the fetch-decode-execute cycle, or simple the fetch-execute cycle) is the cycle that the central Processing Unit (CPU) follows from boot-up until the computer has shut down in order to process instructions. It is composed of three main stages: the fetch stage, the decode stage, and the execute stage.



Memory address register (MAR) -; It is connected to the address lines of the system bus. It specifies the address in memory for a read or write operation.

Memory Buffer Register (MBR) -; It is connected to the data lines of the system bus. It contains the value to be stored in memory or the last value read from the memory.

Program Counter (PC) -; Holds the address of the next instruction to be fetched

Instruction Register (IR) -; Holds the last Instruction fetched.

The fetch step is the same for each instruction

1. The CPU sends the contents of the PC to MAR and sends a read command on the address bus
2. In response to the read command (with address equal to PC), the memory returns the data stored at the memory location indicated by PC on the data bus.
3. The CPU copies the data from the data bus into its MDR (also known as MBR; See Section Role of components above)
4. A fraction of a second later, the CPU copies the data from the MDR to the Instruction

register for instruction decoding.

5. The PC is incremented so that it points to the next instruction. This step prepares the CPU for the next cycle.

The control unit fetches the instruction's address from the memory unit.

The decoding process allows the CPU to determine what instruction is to be performed so that the CPU can tell how many operands it needs to fetch in order to perform the instruction. The opcode fetched from the memory is decoded for the next steps and moved to ~~decoded~~ ~~for~~ the appropriate registers. The decoding is done by the CPU's control unit.

Reading the effective address :- The step evaluates which type of operation is to be performed. if it is a memory operation, the computer checks whether it's a direct or indirect memory operation:

- Direct memory operation :- Nothing is done.
- Indirect memory operation :- The effective address is read from memory.

if it is an I/O or register instruction, the computer checks its type and executes the instruction.

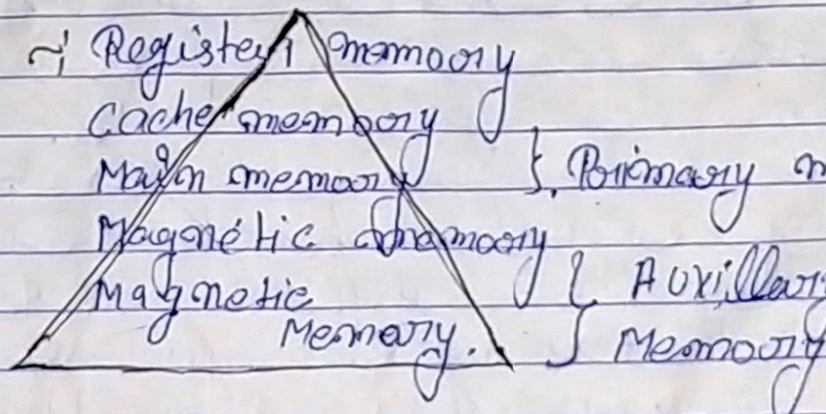
Memory organization in computer Architecture :-

A Memory unit is the collection of storage units or devices together. The memory unit stores the binary information in the form of bit sequally & memory / storage is classified into 2 categories.

Volatile memory :- This loses its data, when Power is switched off.

Non volatile memory :- This is a Permanent Storage and does not lose any data when power is switched off.

Memory Hierarchy :-

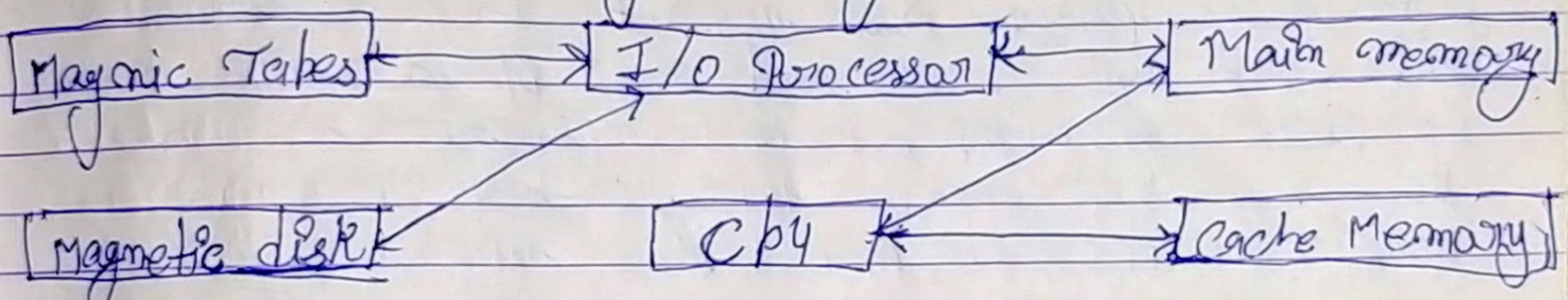


The total memory capacity of a computer can be visualized by hierarchy of components. The memory hierarchy system consists of all storage devices contained in a computer system from the slow Auxiliary memory to fast Main memory and to smaller cache memory.

- The main memory occupies the central position because it is equipped to communicate directly

with the cpu and with auxiliary memory devices through Input/output Processor (I/O) when the Program not residing in main memory is needed by the cpu, they are brought in from auxiliary memory. programs that currently needed in main memory are transferred into auxiliary memory to provide space in main memory for other programs that are currently in use.

- The cache memory is used to store program data which is currently being executed in the



Memory Access Methods :- Each Memory type is a collection of number memory locations. To access data from any memory, first it must be located and then the data is read from the memory location. following are the methodes to access information from memory locations.

1. Random access :- Main memory are random access memories, in which each memory location has a unique address. Using this unique address any memory location can be reached in the same amount of time in any order.

2. Sequential Access :- This methods allows memory access in sequence or in order.

3. Direct Access :- In this mode information is stored in tracks, with each track having a separate read/write head.

Main memory :- The memory unit communicates directly within the cpu. Auxiliary memory and cache memory, is called main memory. it is the central storage unit of the computer system. It is a large and fast memory used to store data during computer operations. Main memory is made up of RAM and ROM, with RAM integrated circuit chips holing the major share

RAM Random Access Memory :-
DRAM :- Dynamic RAM, is made of capacitor and must be refreshed every 10 ~ 100 ms. It is slower and cheaper than SRAM.
SRAM :- Static RAM, has a six transistor circuit in each cell and retains data, until powered off
NvRAM :- Non-volatile RAM, retains its data, even when turned off. Ex flash memory

- ROM : Read only memory, is non-volatile and is more like a permanent storage for information. It also stores the bootstrap loader program, to load

PROM :- (Programmable ROM)
EPROM :- (Erasable Programmable ROM)
EEPROM :- (Electically Erasable PROM) are some commonly used ROMS.

Auxiliary memory :- Devices that provide backup storage are called auxillary memory for example :- Magnetic disks and tapes are commonly used auxiliary devices. other devices used as auxiliary memory are magnetic drums. it is not directly accessible to the CPU, and is accessed using the Input/output channels.

Cache Memory :- The data or contents of the main memory that are used again and by CPU, are stored in the cache memory. if the data is not found in cache memory then the CPU moves on to the main memory then the CPU transfers block of recent data into the cache and keeps on deleting the old data in cache to accomodate the new one.

Different characteristics of Memory System :-
The different characteristics of memory system include :-
1. location
2. Capacity
3. Access Method
4. unit of transfer
5. Performance
6. Physical types
7. organization
8. Physical characteristics

Location -; This characteristic represents whether the location is external or internal to the computer. External memory is the seperate storage device such as disk and tape. It is accessed by the Processor through I/O controllers. Internal memory is mostly embedded within the computer. It is in various forms such as registers, cache and main memory.

Capacity -; It is the most important whether the location is external or internal to the computer. External memory is the seperate storage device such as disk and tape. It is accessed by the Processor through I/O controllers. Internal memory is mostly embedded within the computer. It is in various forms such as registers, cache and main memory.

Capacity -; It is the most important characteristic of the memory system. it is different for external and internal memory. Capacity of the external memory is measured in terms of bytes. whereas the capacity of Internal memory wherein word length may be 8, 16, or 32 bits.

Access Method -; The characteristic has four type of Method
- Direct
- Sequential
- Random

Direct
*Access Method :- In this method every block has its own unique address. Here, the access is made directly by first reaching to vicinity and then following sequential search. Access time of this method is concerned with current and previous location of data. An example of this method is hard disk.

Sequential Access Method :- In this method access to data is made sequentially. Like direct access Method, it also makes use of shared read-write mechanism. Tape is an example of this method. Access time is different with that of direct access but depends on same Parameters.

Random Access Method :- In this method each block of memory has its unique and Physically wired-in addressing Mechanism. Unlike direct and sequential access the access time of this method does not depend on location. Ram is an example of this Method

Associative Access Method :- In this Method unlike other methods, access to data is made by comparing the required bit location within a word based on the specified data element. Therefore the access time of this Method also depend on location. Cache memory is the example of this Method.

Unit of transfer -: This characteristic measures the transfer rate of data in and out the memory. it is different for both the internal and external memory.
(a) For internal memory, it is the number of electrical lines going into and coming out of the memory. these lines carry data of word length.
(b) For external memory it is the number of bits read out of written into the memory at a time.

Performance -; it is also the most important characteristic of the memory system. The following parameters need to be considered for performance.

(a) Transfer Rate -: It can be defined as the rate at which data is ~~different~~ transmitted into or out of the memory. It is different for different type of memory

(b) Memory Cycle time -: it refer to the time required for accessing a block as well as the Period Prior to start of the second access. it is mostly required for random access memory. it is related with the system bus but not the Processor.

(c) Access time :- It is different types of memory, In random access memory, it is time required to carry out read or write operation. In non random access Memory, it is time required to Place the read-write Mechanism at the desired Place.

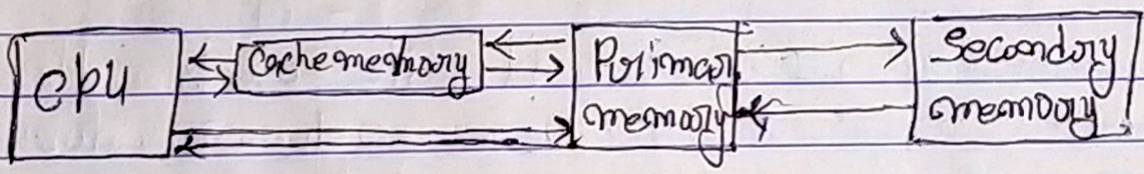
Physical type :- There are various physical types of memory such as,
(a) Magnetic
(b) Semiconductor
(c) Magneto-optical
(d) optical

organization :- It describes the Physical arrangement of the bits. it is very important for random access Memory.

Physical characteristics :- it describes the nature of memory that whether the memory is volatile, non-volatile erasable or non-erasable. Non-volatile memory does not ensure Erasable memory allows one to change or modify data unlike non erasable memory.

Cache memory :- Cache memory is a special very high-speed memory. It is used to speed up and synchronizing with high speed CPU. Cache memory is costlier than main memory or disk memory but economical than CPU registers. Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU. It holds frequently requested data and instructions so that they are immediately available to the CPU when needed.

Cache memory is used to reduce the average time to access data from the main memory. The cache is a smaller and faster memory which stores copies of data from frequently used main memory locations. There are various different independent caches in a CPU, which store instructions and data.



Cache performance :- When the processor needs to read or write a location in main memory, it first checks for a corresponding entry in the cache.

The basic operation of a cache memory is as follows.

- When the cpu needs to access memory, the cache is examined. If the word is found in the cache, it is read from the fast memory.
- If the word addressed by the cpu is not found in the cache, the main memory is accessed to read the word.
- A block of words one just accessed is then transferred from main memory to cache memory. The block size may vary from one word (the one just accessed) to about 16 words adjacent to the one just accessed.
- The performance of the cache memory is frequently measured in terms of a quantity called hit ratio
- When the cpu refers to memory and finds the word in cache, it is said to produce a hit
- If the word is not found in the cache, it is in main memory and it counts as miss.
- The ratio of the number of hits divided by the total cpu references to memory (hits plus misses) is the hit ratio.

We can improve cache performance using higher cache block size, higher associativity, reduce miss rate, reduce miss penalty, and reduce the time to hit in the cache.

Cache Mapping :- There are three different types of mapping used for the purpose of cache memory which are as follows: Direct

Mapping

Mapping Associative mapping, and set Associative mapping. These are -

Direct Mapping -: The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line. or

In direct mapping assigne each memory block to a specific line in the cache. if a line is previously taken up by a memory block when a new block needs to a loaded, the old block is trashed. An address space is split into two parts index field and a tag field. The cache is used to stored the tag field whereas the rest is stored in the main memory. Direct mapping's performance is directly Proportional to the Hit ratio.

$i = j$ modulo m

where

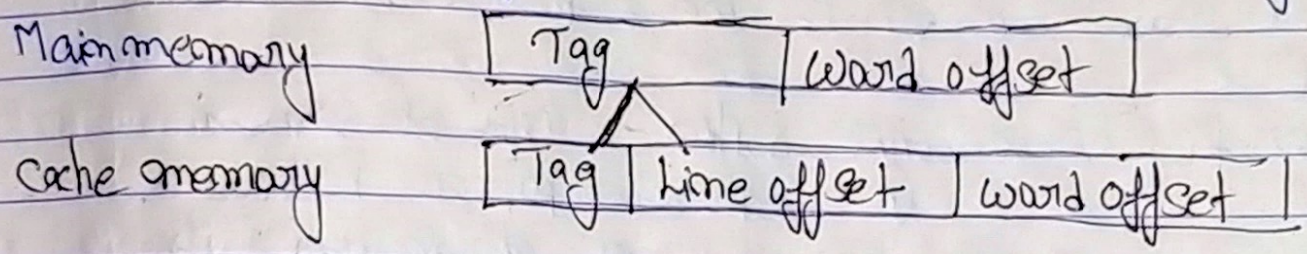
i = cache line number

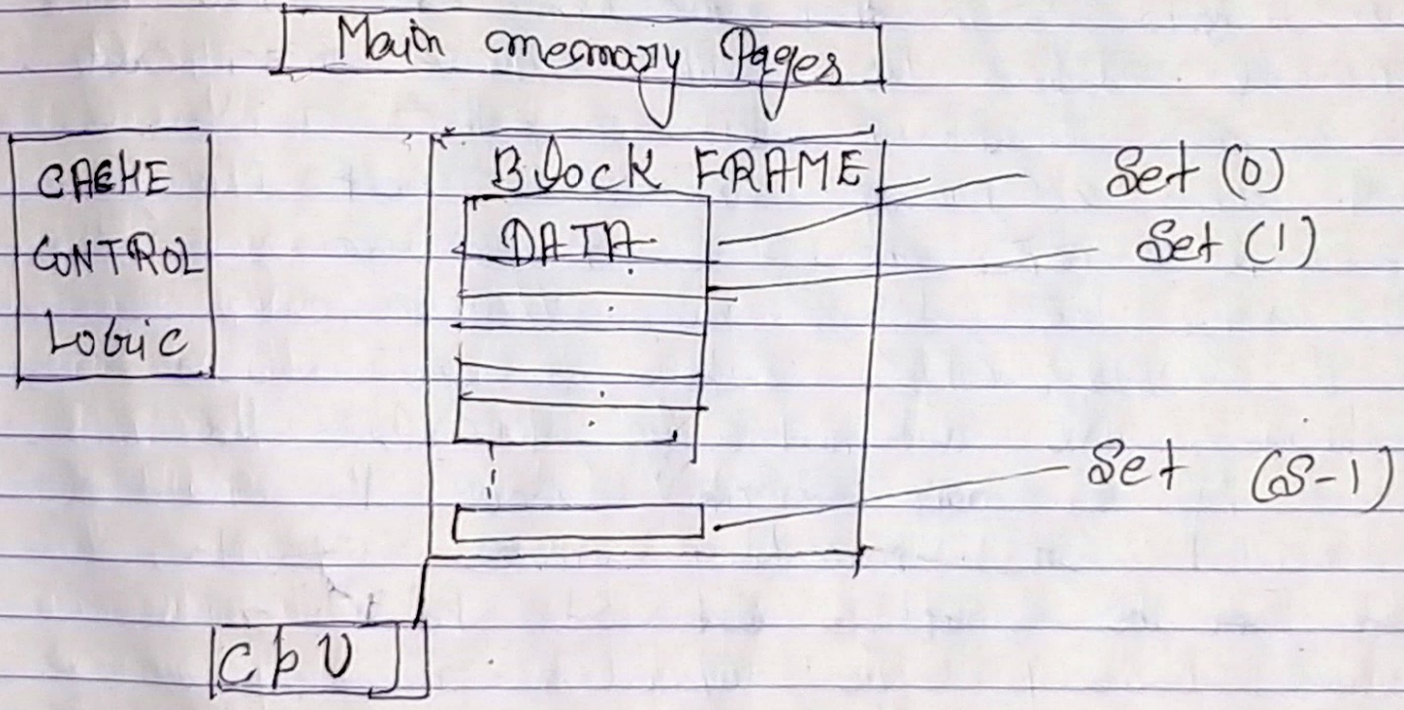
j = main memory block number

m = number of lines in the cache

for purpose of cache access, each main memory address can be viewed as consisting of three fields. The least significant w bits identify a unique word or byte within a block of main memory. In most contemporary machines,

The address is at the byte level. The remainings bits specify one of the 2^s record blocks of main memory. The cache logic interprets theses bits as a tag of $s-r$ bits (most significant portaion) and a line field of r bits. This latter field identifies one of the $m = 2^r$ lines of the cache.





Verchual Memory :- Virtual memory is a memory management technique where secondary memory can be used as if it were a part of the main memory. can be used. Virchual memory is a very common techniq

Virtual memory uses hardware and
allow a computer to compensate for physical
memory shortages, by temporarily transferring
data from random access memory (RAM)
to disk storage. In essence, virtual memory
allows a computer to treat secondary
memory as though it were the main memory.

How virtual memory works:- virtual memory uses
both computer hardware and
software to work. When an application is in
use, data from that program is stored in a
physical address using RAM. More specifically,
virtual memory will map that address to
RAM using a memory management unit (MMU). The
OS will make and manage memory mapping
by using page tables and other data structures.
The MMU, which acts as an address translation
hardware, will automatically translate the address.
If at any point later the RAM space
is needed for something more urgent, the
data can be swapped out of RAM and into
virtual memory. The computer's memory manager
is in charge of keeping track of the shifts
between physical and virtual memory.

use swap files with a fixed number of addresses Each page is stored on a disk, and when the page is needed the OS copies it from the disk to main memory and translates the virtual addresses into real addresses.

However, the process of swapping virtual memory to physical is rather slow. This means that using virtual memory generally causes a noticeable reduction in performance. Because of swapping, computers with more RAM are seen to have better performance.

Types of virtual memory :- A computer MMU handles memory operation including managing virtual memory. In most computers the MMU hardware is integrated into the CPU. There are two ways in which virtual memory is handled: Paged and Segmented.

Paging divides memory into sections or paging files usually approximately 4 KB in size. When a computer uses up its RAM pages not in use are transferred to the section of the hard drive designated for virtual memory using a swape file. A swap file is a space set aside on the hard drive as the virtual memory extensions of the computer's RAM. When the swap file is needed it's sent back to RAM using a process called page swapping. This system ensures that the computer's OS and applications don't run out of real memory.

The paging process include the use of page tables which translate the virtual addresses that the OS and applications use into the physical addresses that the MMU uses. Entries in the page table indicate whether the page is in real memory. If the OS or a program does not find what it needs in RAM, then the MMU responds to the missing memory reference with a page fault exception to get the OS to move the page back to memory when it's needed. Once the page is in RAM its virtual address appears in the page table

Segmentation is also used to manage virtual memory. This approach divides virtual memory into segments of different lengths. Segments not in use in memory can be moved to virtual memory space on the hard drive. Segmented information on processes are tracked in a segment table, which shows if a segment is present in memory, whether it's been modified and what it's physical address is. In addition file systems in segmentation are only made up of a list of segments mapped into a process potential address space.

Segmentation and paging differ as a memory model in terms of how memory is divied however, it can also be combined. Some virtual memory systems combine segmentation and paging. In this case,

Memory gets divided into frames or pages. The segments take up multiple peges, and the virtual address includes both the segments number and the Page number.

Benefits of using virtual memory -;

Benefits of virtual include -;

- its ability to handle twice as many address as main memory.
- frees applications from managing shared Memory and saves users from having to add memory modules when RAM space runs out;
- increased security because of memory isolation;
- Multible larger applications can be run simultaneously;
- allocating memory is relatively cheap;
- doesn't need external fragmentation;
- effective CPU use;
- data can be moved automatically; and
- Pages in the original Process can be shared during a fork system call.

Limitations -;

- The use of virtual memory has its tradeoffs, Particularly with speed. it's generally better to have as much physical memory as possible, So Programs work directly from RAM or physical memory.

- The use of virtual memory slows a computer because data must be mapped between virtual and physical memory, which requires extra hardware support for address translations.
- The size of virtual storage is limited by the amount of secondary storage as well as by the addressing scheme with the computer system.
- Thrashing can happen if the amount of RAM is too small, which will make the computer perform slower.
- it may take time to switch between applications using virtual memory.

External Memory — Primary memory has limited storage capacity and is volatile. Secondary memory overcomes this limitation by providing permanent storage of data and in bulk quantity. Secondary memory is also termed as external memory and refers to the various storage media on which a computer can store data and programs. The secondary storage media can be fixed or removable. fixed storage media is an internal storage medium like hard disk that is fixed inside the computer storage medium that are portable and can be taken outside the computer are termed as removable storage media.

*uses of Secondary media *

- Permanent Storage :- Primary memory (RAM) is volatile, i.e it loses all information when the electricity is turned off. So, in order to secure the data permanently in the device, Secondary storage devices are needed.
- Portability :- Storage medium, like the CDs, flash drives can be used to transfer the data from one device to another.

Fixed Storage :- A fixed storage is an internal media device that is used by a computer system to store data, and usually these are referred to as the fixed Disks drives or the Hard Drives.

fixed storage devices are literally not fixed, obviously these can be removed from the system for repairing work, maintenance Purpose and also for upgrade etc. But in general, that can't be done without a proper toolkit to open up the computer system to provide Physical access, and that needs to be done by an engineer.

Technically, almost all of the data i.e being Processed on a computer system is stored on some type of built-in fixed storage device

Types of fixed Storage :-

- Internal flash memory (more)
- SSD (Solid-State disk) units
- Hard disk drives (HDD)

Removable Storage :- A Removable Storage is an external media device that is used by a computer system to store data, and usually these are referred to as the Removale Disks drives or the External Drives. Removable Storage is any type of storage device that can be removed / ejected from a Computer system while the system is running. Examples of external devices include CDs, DVDs and Blu-Ray disk drives, as well as diskettes and USB drives. Removable Storage makes it easier for a user to transfer data from one Computer System to another.

In a Storage factors, the main benefit of removable disks is that they can Provide the fast data transfer rates associated with Storage area networks (SANs).

Types of Removable Storage :-

- Optical discs (CDs, DVDs, Blu-ray discs)
- Memory Cards
- Floppy disks
- Magnetic tapes

Secondary Storage Media :- There are the following main type of Storage media.

Magnetic Storage media :- Magnetic media is coated with a magnetic layer which is magnetized in clockwise or anticlockwise direction

when the disk moves, the head interprets the data stored at a specific location in binary 1s and 0s at reading.
Example :- hard disks, floppy disks and magnetic tapes.

Hard disk :- A hard disk consists of one or more circular disks called Platters which are mounted on a common spindle. Each surface of a Platter is coated with a magnetic material. Both surfaces of each disk are capable of storing data except the top and bottom disk where only the inner surface is used. The information is recorded on the surface of the rotating disk by magnetic read/write heads. These heads are joined to a common arm known as access arm.

Optical Storage media :- In optical Storage Media information is stored and read using a laser beam. The data is stored as a spiral pattern of pits and ridges denoting binary 0 and binary 1.
Example :- CDs and DVDs.

Compact Disk :- A compact Disc drive (CDD) is a device that a computer uses to read data that is encoded digitally on a compact disc (CD). A CD drive can be installed inside a computer's compartment

access or it can be used by a peripheral device connected to one of the ports provided on the computer system. A compact disk or CD can store approximately 650 to 700 megabytes of data. A computer should possess a CD Drive to read the CDs There are three types of CDs.

DVD -; In stands for digital versatile Disk or digital video disk. It looks just like a CD and use a similar technology as that of the CDs but allows tracks to be spaced closer enough to store data that is more than six times the CD's capacity. It is a significant advancement in portable storage technology. A DVD holds 4.7 GB to 17 GB of data.

Blue Ray Disk -; This is the latest optical storage media to store high definition audio and video. It is similar to a CD or DVD but can store up to 27 GB of data on a single layer disk and up to 54 GB of data on a dual layer disk. While CDs or DVDs use red laser beam, the blue ray disk uses a blue laser to read/write data on a disk.

Solid state memories -; Solid-state storage devices are based on electronic circuits with no moving parts like the reels of tape, spinning discs etc. Solid State storage devices use special memories called flash memory

to share data. Solid State drive (or flash memory) is used mainly in digital cameras, Pen Drives or USB flash drives.

Pen Drives :- Pen Drives or thumb drives or flash drives are the recently emerged Portable Storage media. It is an EEPROM based flash memory which can be repeatedly erased and written using electric signals. This memory is accompained with a USB connect which enables the Pendrive to connect to the computer. They have a capacity smaller than a hard disk but greater than a CD. Pendrive has following advantages.

Transfer files :- A Pen drive being Plugged into a USB Port of the system can be used as a device to transfer files, documents and Photos to a PC and also vice versa Similarly, selected files can be transferred between a pen drive and any type of workstation.

Portability :- The high weight nature and smaller size of a pen drive make it possible to carry it from Place to Place which makes data transportation an easier task.

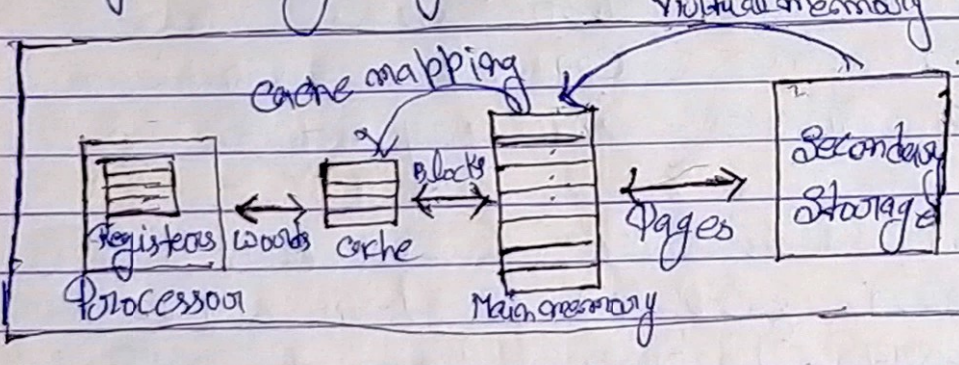
Backup Storage :- Most of the Pen drives now come with a feature of having Password encryption important information related to family, medical records and Photos can be

TransPort Data :- Professionals / students can now easily transport large data files and vide/audio lectures on a Pen drive and gain access to them from anywhere. Independent PC technicians can store work related utility tools, various Programs and files on a high speed 64 GB Pen drive and move from one site to another.

Cache Mapping :- • Cache mapping defines how a block from the main memory is mapped to the cache memory in case of cache Miss.

• Chache Mapping is a technique by which the conten of main memory are brought into the cache memory.

The following diagram illustrates the mapping Process



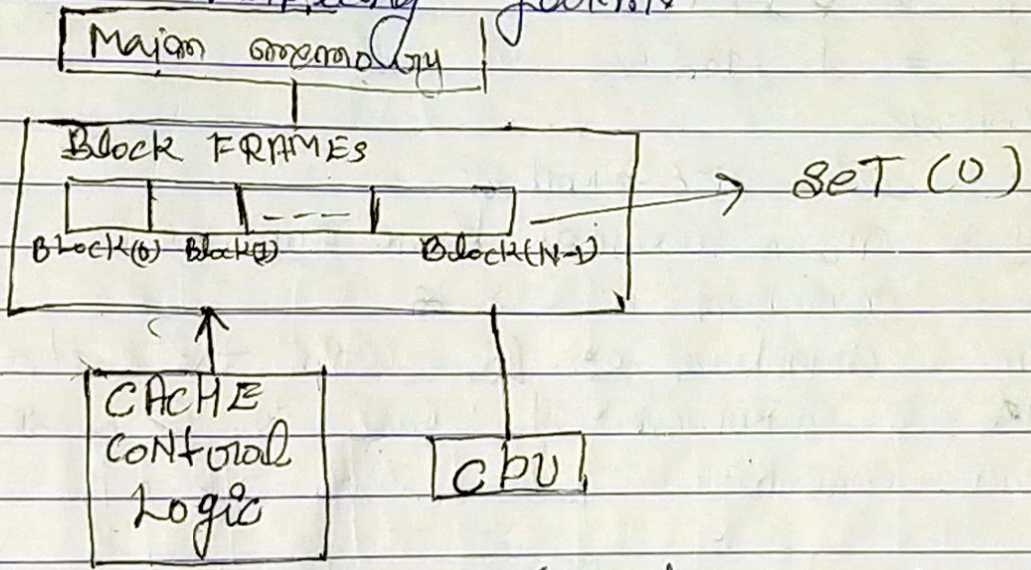
We can Improve Cache Performance using higher cache block size, higher associativity, reduce Miss rates, reduce miss Penalty and reduce the time to hit in the cache.

(91)

Cache Mapping :- There are three different types of mapping used for the Purpose of cache Memory which are as follows Direct Mapping, Associative Mapping, and Set-Associative Mapping. These are -

(1) Direct Mapping —

(2) Associative Mapping :- In this type of Mapping, the associative Memory is used to store content and address of the memory word. any block can go into any line of the cache. This Means that the word id bits are used to identify which word in the block is needed but the tag becomes the Placement of any word at any Place in the cache memory. It is considered to be the fastest and the most flexible mapping form.



Set associative Mapping :- This form of mapping is an enhanced form of direct Mapping are removed. Set accociative address the Problem of Possible thrashing in

the direct Mapping Method. It does this by saying that instead of having exactly one line that a block can map to in the cache, we will group a few lines together creating a set. Then a block in memory can map to any one of the lines of a specific set. Set-associative Mapping allows that each word that is present in the cache can have two or more words in the main memory for the same index address. Set associative cache Mapping combines the best of direct and associative cache mapping techniques.

In this case, the cache consists of a number of sets, each of which consists of a number of sets, each of which consists of a number of lines. The relationships are —

$m = v * k$

$i = j \bmod v$

where

i = cache set number

j = main memory block number

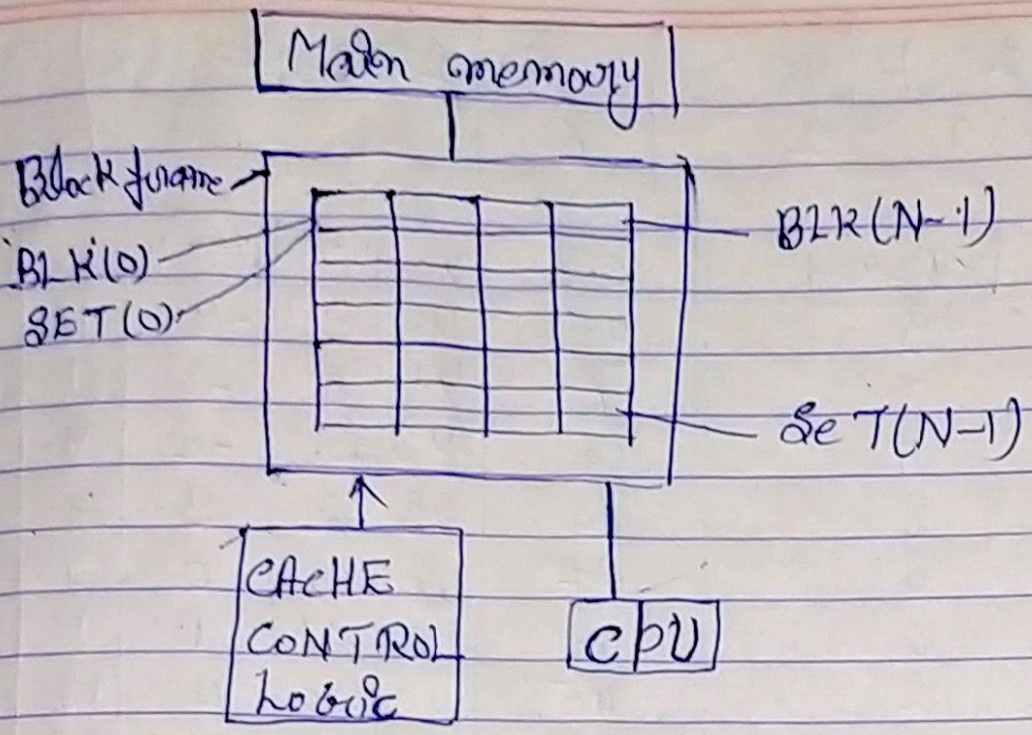
v = number of sets

m = number of lines in the cache number

k = number of lines in each set

Main memory | Tag | word-offset |

Cache memory | Tag | Set offset | word-offset |



CPU (Processor), RAM (Main memory) I/O devices (Mouse, Keyboard, Printer), BUS (data transfer Component).

How could you like these computer parts to communicate and transfer data? you definitely need a fixed size of bits to be considered a single unit of data.

For that, computer scientists agreed to standardize this unit to be 32 bits or 64 bits (depending on the manufacturer Choice).

They gave this unit a name and called it ~~word~~ word. So a word is nothing but a unit of data (bunch of bits (Signal changes of zeros and ones) that moves around from a computer Component to another.