Biyani's Group of Girls' Colleges

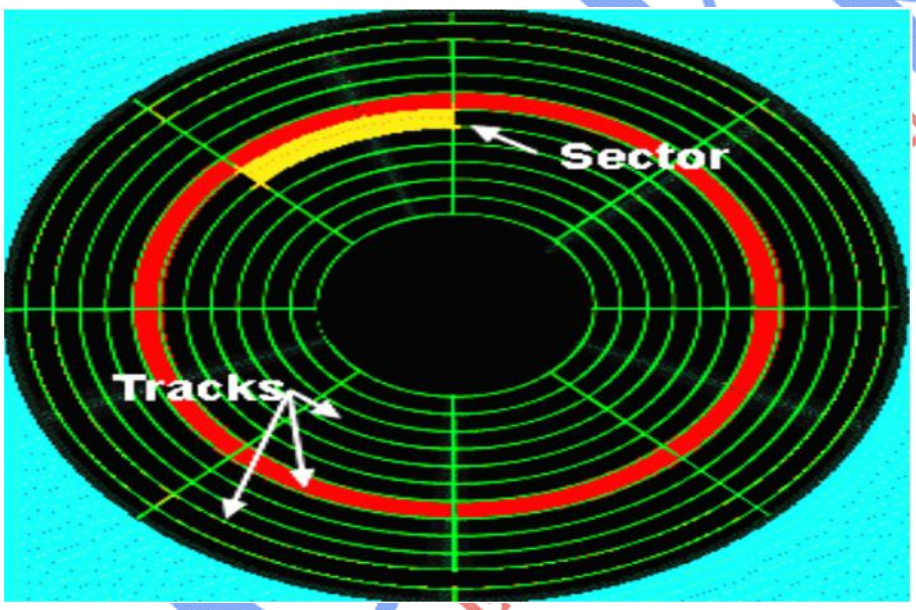
PRE-UNIVERSITY EXAM
Class-BCA-I YEAR
SET-A

Time: 3Hrs **SUBJECT.-Computer Organization** **MM: 100**

1 Very short answer type questions(Maximum word limit-20 words)(2*10)

(a) What is Tracks and Sector?

Ans:-The platters inside a hard drive are structured to facilitate the storage and retrieval of data. Each platter is divided into concentric rings called "Tracks". There are thousands of tracks on each platter. Each track is divided into sectors. A sector, as a rule, holds 512 bytes of data, this is usually the minimum quantity of information which is independently addressable for storage on a hard drive disk. Today, hard drive platters have thousands of sectors in a single track.



(b) Define VGA.

Ans:-Video Graphics Array (**VGA**) is the display hardware first introduced with the IBM PS/2 line of computers in 1987. Through widespread adoption, the term has also come to mean either an analog computer display standard, the 15-pin D-subminiature **VGA** connector, or the 640×480 resolution characteristic of the **VGA** hardware.

(c) What do you mean by program and interrupt I/O?

Ans :-In system **programming**, Program is a set of instruction get in instruction set and **interrupt** is a signal to the processor emitted by hardware or **software** indicating an event that needs immediate attention. An **interrupt** alerts the processor to a high-priority condition requiring the interruption of the current code the processor is executing.

(d) What is RTL?

Ans :-Register transfer language(RTL) is a kind of intermediate representation (IR) that is very close to assembly **language**, such as that which is used in a compiler. It is used to describe data flow at the **register-transfer** level of architecture.

(e) Define accumulator

Ans :-An **accumulator** is a register for short-term, intermediate storage of arithmetic and logic data in a computer's CPU (central processing unit).In a modern computers, any register can function as an accumulator. The most elementary use for an accumulator is adding a sequence of numbers.

(f) What is DMA?

Ans:- Direct memory access (DMA) is a feature of computer systems that allows certain hardware subsystems to access main system memory (Random-access memory), independent of the central processing unit (CPU).

(g) Define Main memory

Ans:-Main memory refers to physical memory that is internal to the computer. The word main is used to distinguish it from external mass storage devices such as disk drives. Other terms used to mean main memory include RAM and primary storage. The computer can manipulate only data that is in main memory.

(h) What is ALU and CU?

Ans:-An **arithmetic logic unit** (ALU) is a digital circuit used to perform **arithmetic and logic** operations. 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).

(i) What is SCSI and Network card?

Ans: The Small Computer System Interface, is a set of American National Standards Institute (ANSI) standard electronic interfaces that allow personal computers (PCs) to communicate with peripheral hardware such as disk drives, tape drives, CD-ROM drives, printers and scanners faster and more flexibly than previous parallel data transfer interfaces A **Network interface card**, NIC, or **Network card** is an electronic device that connects a computer to a computer **network**, usually a **LAN**. It is considered a piece of computer hardware. Today, most computers have **network cards**.

(j) What do you mean by effective address in addressing mode?

Ans:- **Addressing Modes**. The term **addressing modes** refers to the way in which the operand of an instruction is specified. Information contained in the instruction code is the value of the operand or the **address** of the result/operand.

PART-II

Short Answer type questions (5X4=20marks)(Word limit:80 words)

(a) What is Cache and Virtual memory ?

Ans:-Cache memory, also called **CPU memory**, is random access **memory** (RAM) that a **computer** microprocessor can access more quickly than it can access regular RAM. This **memory** is typically integrated directly with the **CPU** chip or placed on a separate chip that has a separate bus interconnect with the **CPU.**

Virtual memory is a valuable concept in **computer architecture** that allows you to run large, sophisticated programs on a **computer** even if it has a relatively small amount of RAM. A **computer** with **virtual memory** artfully juggles the conflicting demands of multiple programs within a fixed amount of physical **memory**.

(b) Define Memory buffer register.

Ans:-A memory buffer register (MBR) is the register in a computer's processor, or central processing unit, CPU, that stores the data being transferred to and from the immediate access store. It contains the copy of designated memory locations specified by the memory address register. It acts as a buffer allowing the processor and memory units to act independently without being affected by minor differences in operation. A data item will be copied to the MBR ready for use at the next clock cycle, when it can be either used by the processor for reading or writing or stored in main memory after being written.

(c)Why auxiliary storage device and I/O card use?

Ans:-Auxiliary storage device:-It is **storage** that's separate from the computer itself, where software and data can be stored on a permanent basis. **Secondary storage** is **necessary** because memory, or primary **storage**, loses its data when a computer is turned off where as **secondary storage** does not. Therefore, it is commonly known as non-volatile **storage**.

I/O Card :-In computing, input/output or I/O (or, informally, **io** or **IO**) is useful for the communication between an information processing system, such as a computer, and the outside world, possibly a human or another information processing system.

(d) Explain the instruction cycles.

Ans:-An instruction cycle (also known as the fetch–decode–execute cycle or the fetch-execute cycle) is the basic operational process of a computer. It is the process by which a computer retrieves a program instruction from its memory, determines what actions the instruction dictates, and carries out those actions. The instruction cycle is how many of these machine cycles are needed to complete an instruction. CPUs the instruction cycle is executed sequentially, each instruction being processed before the next one is started. In most modern CPUs the instruction cycles are instead executed concurrently, and often in parallel, through an instruction pipeline: the next instruction starts being processed before the previous instruction has finished,

(e) What is stack pointers and paging?

Ans:-A stack pointer is a small register that stores the address of the last program request in a stack. When a new data item is entered or "pushed" onto the top of a stack, the stack pointer increments to the next physical memory address, and the new item is copied to that address. paging is a memory management scheme by which a computer stores and retrieves data from secondary storage for use in main memory. In this scheme, the operating system retrieves data from secondary storage in same-size blocks called pages.

Q-3 Long Answer Questions **(12*5)**

(a) What is Register ?Explain the types of register with it's functionality and utility.

Ans:-A processor register (CPU register) is one of a small set of data holding places that are part of the computer processor. A register may hold an instruction, a storage address, or any kind of data (such as a bit sequence or individual characters). Some instructions specify registers as part of the

instruction. For example, an instruction may specify that the contents of two defined registers be added together and then placed in a specified register. A register must be large enough to hold an instruction - for example, in a 64-bit computer, a register must be 64 bits in length. In some computer designs, there are smaller registers - for example, half-registers - for shorter instructions. Depending on the processor design and language rules, registers may be numbered or have arbitrary names.a processor register is a very fast computer memory used to speed the execution of computer programs by providing quick access to commonly used values-typically, the values being in the midst of a calculation at a given point in time. These registers are the top of the memory hierarchy, and are the fastest way for the system to manipulate data. In a very simple microprocessor, it consists of a single memory location, usually called an *accumulator*. **Registers** are built from fast multi-ported memory cell. They must be able to drive its data onto an internal bus in a single clock cycle. The result of ALU operation is stored here and could be re-used in a subsequent operation or saved into memory. Registers are normally measured by the number of bits they can hold, for example, an “8-bit register” or a “32-bit register”. Registers are now usually implemented as a register file, but they have also been implemented using individual flip-flops, high speed core memory, thin film memory, and other ways in various machines.

There are several other classes of registers:

(a) Accumulator: It is most frequently used register used to store data taken from memory. Its number varies from microprocessor to microprocessor.

(b) General Purpose registers: General purpose registers are used to store data and intermediate results during program execution. Its contents can be accessed through assembly programming.

(c) Special purpose Registers: Users do not access these registers. These are used by computer system at the time of program execution. Some types of special purpose registers are given below:

- Memory Address Register (MAR): It stores address of data or instructions to be fetched from memory.
- These Registers are used for performing the various Operations. While we are working on the System then these Registers are used by the **CPU for Performing the Operations**. When We Gives Some Input to the System then the I**nput will be Stored into the Registers** and When the System will gives us the Results after Processing then the Result will also be from the Registers. So that they are used by the **CPU for Processing the Data** which is given by the User. Registers Perform:-
- Memory Buffer Register (MBR): It stores instruction and data received from the memory and sent from the memory.
- Instruction Register (IR): Instructions are stored in instruction register. When one instruction is completed, next instruction is fetched in memory for processing.
- Program Counter (PC): It counts instructions.

  The instruction cycle is completed into two phases: (a) Fetch Cycle and (b) Execute Cycle. There are two parts in instruction- opcode and operand. In fetch cycle opcode of instruction is fetched into CPU. The opcode, at first, is reached to Data Register (DR), then to Instruction Register (IR). Decoder accesses the opcode and it decodes opcode and type of operation is declared to CPU and execution cycle is started. 1)Fetch: The Fetch Operation is used for taking the instructions those are given by the user and the Instructions those are stored into the Main Memory will be fetch by using Registers.

2) **Decode**: The Decode Operation is used for interpreting the Instructions means the Instructions are decoded means the CPU will find out which Operation is to be performed on the Instructions.

3) **Execute:** The Execute Operation is performed by the CPU. And Results those are produced by the CPU are then Stored into the Memory and after that they are displayed on the user Screen.

Types of Registers are as Followings

MAR stand for Memory Address Register

This register holds the memory addresses of data and instructions. This register is used to access data and instructions from memory during the execution phase of an instruction. **Suppose CPU wants to store some data in the memory or to read the data from the memory.** It places the address of the-required memory location in the MAR. In a computer, the Memory Address Register (MAR) is the CPU register that either stores the memory address from which data will be fetched to the CPU or the address to which data will be sent and stored. In other words, MAR holds the memory location of data that needs to be accessed

Accumulator Register

This Register is used for storing the Results those are produced by the System. When the CPU will generate Some Results after the Processing then all the Results will be Stored into the **AC Register**. An accumulator is a register for short-term, intermediate storage of arithmetic and logic data in a computer's CPU (central processing unit). The term "accumulator" is rarely used in reference to contemporary CPUs, having been replaced around the turn of the millennium by the term "register." In a modern computers, any register can function as an accumulator. The most elementary use for an accumulator is adding a sequence of numbers. The numerical value in the accumulator increases as each number is added.

Memory Data Register (MDR)

MDR is the register of a computer's control unit that contains the **data to be stored in the computer storage** (e.g. RAM), or the **data after a fetch from the computer storage**. It acts **like a buffer** and holds anything that is copied from the memory ready for the processor to use it. **MDR hold the** information **before it goes to the decoder.**

MDR which contains the data to be written into or readout of the addressed location. For example, to retrieve the contents of cell 123, we would load the value 123 (in binary, of course) into the MAR and perform a fetch operation. When the operation is done, a copy of the contents of cell 123 would be in the MDR. To store the value 98 into cell 4, we load a 4 into the MAR and a 98 into the MDR and perform a store. When the operation is completed the contents of cell 4 will have been set to 98, by discarding whatever was there previously.

The MDR is a two-way register. When data is fetched from memory and placed into the MDR, it is written to in one direction. When there is a write instruction, the data to be written is placed into the MDR from another CPU register, which then puts the data into memory.

The Memory Data Register is half of a minimal interface between a micro program and computer storage, the other half is a memory address register.

Index Register

A hardware element which holds a number that can be added to (or, in some cases, subtracted from) the address portion of a computer instruction to form an effective address. Also known as **base register**. An index register in a computer's CPU is a processor register used for modifying operand addresses during the run of a program. An index register is a circuit that receives, stores, and outputs instruction-changing codes in a computer. This circuit is also called an address register or a register of modifications.

Memory Buffer Register

MBR stand for *Memory Buffer Register*. This register holds the contents of data or instruction read from, or written in memory. It means that this register is used to store data/instruction coming from the memory or going to the memory. A memory buffer register (MBR) or memory data register (MDR) is the register in a computer's processor, or central processing unit, CPU, that stores the data being transferred to and from the immediate access storage. It contains the copy of designated memory locations specified by the memory address register. It acts as a buffer allowing the processor and memory units to act independently without being affected by minor differences in operation. A data item will be copied to the MBR ready for use at the next clock cycle, when it can be either used by the processor for reading or writing or stored in main memory after being written.

Data Register

A register used in microcomputers to temporarily store data being transmitted to or from a peripheral device. When there is a write instruction, the data to be written is placed into the MDR from another CPU register, which then puts the data into memory. The Memory Data Register is half of a minimal interface between a micro program and computer storage, the other half is a memory address register (MAR).

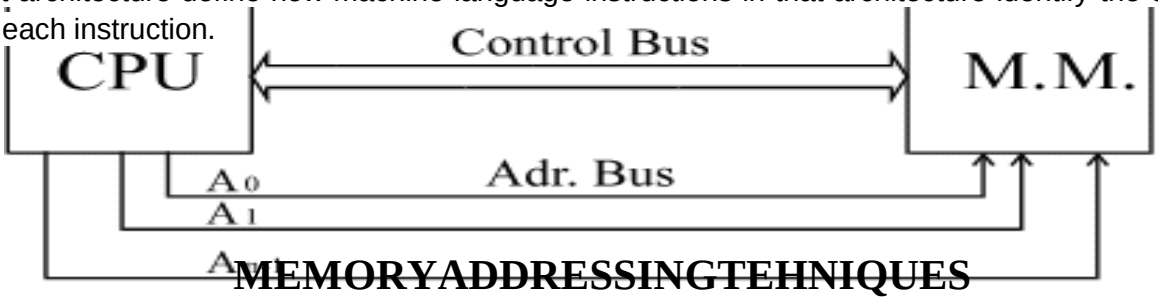
Program Counter

The program counter (PC), commonly called the instruction pointer (IP) in Intel x86 microprocessors, and sometimes called the instruction address register, or just part of the instruction sequencer in some computers, is a processor register

It is a 16 bit special function register in the 8085 microprocessor. It keeps track of the the next memory address of the instruction that is to be executed once the execution of the current instruction is completed. In other words, it holds the address of the memory location of the next instruction when the current instruction is executed by the microprocessor.

(b) What is Addressing techniques? Define different types of addressing techniques.

Ans:-Addressing Technique are an aspect of the instruction set architecture in most central processing unit (CPU) designs. The various addressing modes that are defined in a given instruction set architecture define how machine language instructions in that architecture identify the operand(s) of each instruction.



MEMORYADDRESSINGTEHNIQUES

General considerations concerning the memory addressing

It has been presented the fact that in modern computers the memory unit is implemented with integratedmodules.TheconnectionbetweenCPUandmainmemoryisrealizedbymeansofthree buses:

The address bus ADRBUS;
The data bus DATA BUS;
The control bus CONTROLBUS.

Theaddressbusisunidirectional;thedatabusisbidirectional,whilethecontrolbuscontains command and status lines. The memory is organized on locations:

A memory location has m ranks noted Bm-1, …B1, B0.

Reading and writing the memory is performed in parallel, in other words one reads or writes an m bit word data time. Addressing the memory is done via the address bus (ADRBUS) and it is assumed this bus comprises lines. Each combination of bits of this bus defines a certain address. The connection between the bus content and a specific location is done via a decoding process. If the address bus has n lines then the address set is2n.
A ={0,1,...2n-1} where: A = address space

On the other hand, given a certain memory geometry, it results the set of physical locations where m bit words are stored.

Definition: The set of all physical locations is called the space of the memory M.

The connection between the address space and the memory space is given by a translation function, defined as: fT: A M.

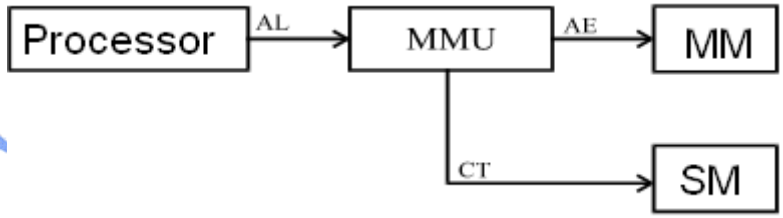
In accordance to the previously mentioned notions, this function corresponds to a decoding function. In case of simple systems(microcomputers)the two sets will be identical M=As of Is a simple function.

When the width of the memory word is not enough to represent the data, then several consecutive locations are used. If for instance, a datum is represent able on two words in memory–AIU, meaning addressable informational unit, then, in order to represent it,the first location is used for the least significant word(from address j),and the next location(address j+1)for the most significant word.

Modern processors present memory management facilities. These memory management operations refer to virtual memory implementation, memory protection, etc.

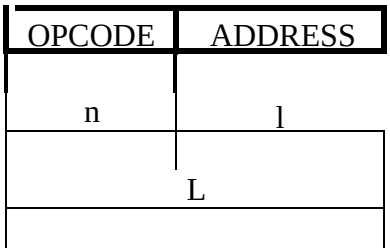
The Memory Management Unit–MMU can be built in the control unit inside the CPU or can be external(with respect to the same control unit).In particular, in case of microprocessors it can be incorporated or sold separately as an additional module – the MMU module.

In order to implement the concept of virtual memory the next simplified scheme is used:



The practical implementation of the virtual memory concept is based on the transfer of blocks of information between secondary and main memories .The most wide spread method is memory segmentation.

It has been already shown that each instruction at machine level comprises two main fields:



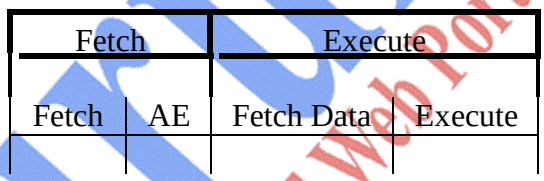
The address field comprises those constitutive elements of the logic address; processing this logic address one finds the effective or physical address. In case of the effective address:

AE =f(AL)
where:
AL– logic address; AE
– effective address

Besides these constitutive elements in the address field forming the logic address, there may exist other elements stored in CPU registers or memory.

In modern computers usually the effective and logic addresses are not the same. Quite the opposite, this function is more and more complex. The purpose is to ensure greater flexibility in programming ,as well as shortening the length of the programs. It is also investigated shortening the instructions length, even If the total volume of addressed memory (the memory space) is constantly increasing.

It has been shown that the instruction cycle comprises two main phases. Each of these phases is divided in sub-phases as in the next figure:



The second phase(AE) is important in the calculation of the operands effective addresses (determination of AE).

I. Assumed operand addressing

There exist situations in which the operand is assumed as known, based on the OPCODE, so it doesn't need to be explicitly emphasized in the instruction, neither as value, nor as address.

For example:

- Increment instruction–the increment1 is assumed as already known,and if this instruction is executed then the value of the accumulator register is increased with 1.
- SHIFT instruction–one assumes the shift value is1,there is non for supplementary specifications.

II. Implicit addressing

Ithasbeenpresentedthatinthecaseofinstructionswithoneortwoaddressesthereisaspecial register in the CPU, the accumulator, which is used when performing the operations. This is because of the way the computer itself is designed and built. It is not necessary to specify it in any other way.

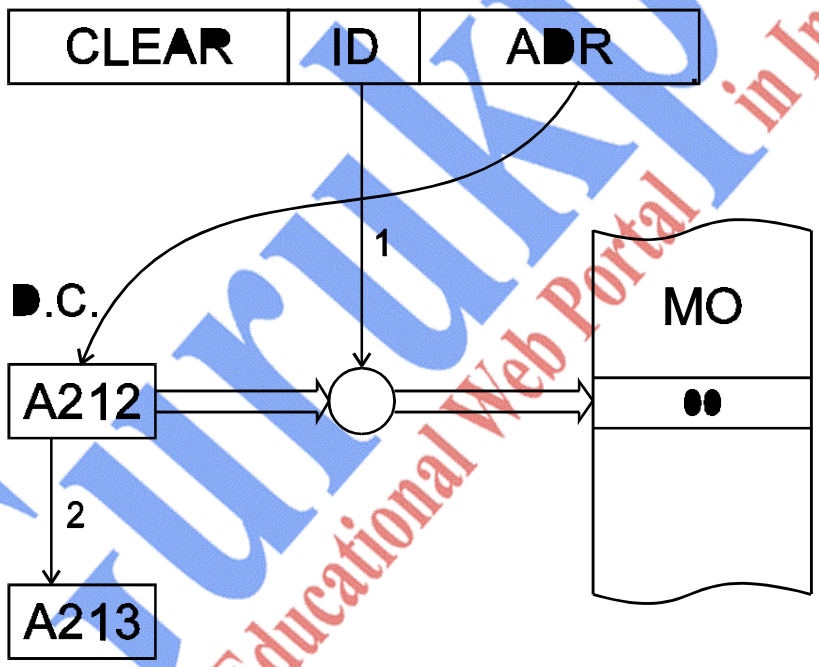
OPCODE	ADR1	ADR2	ACC(ADR1)*(ADR2)

OPCODE	ADR	ACC(ADR)*(ACC)

Moreover, in modern CPUs, one can designate certain general registers to store an operand's address. These general registers are generally referred to as DATACOUNTER. When designing a CPU, the implicit addressing is considered in the following manner: the operand is read from memoryfromtheaddressspecifiedbytheDataCounter,thisregisteractsasacounterwith increment/decrementfacilities.Thisway,itallowsaddressingofawholeblockofdata.Onemay noticeimplicitaddressingthroughself-incrementationandimplicitaddressingthroughself-decrementation.Additionally,aloadoperationmustbeimplemented,inordertobeabletoloadthe start address of the data block.

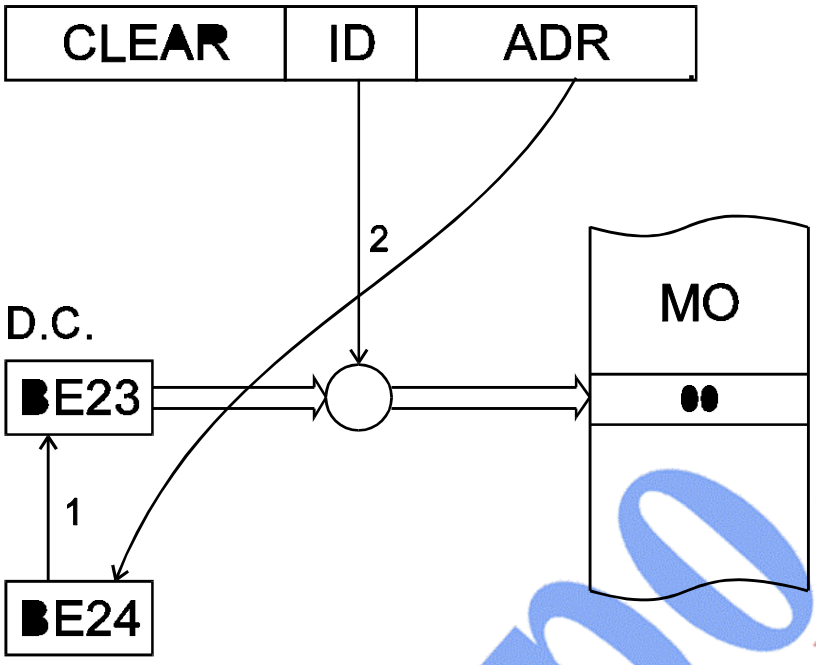
One can notice two variants in which the operand read operation is performed:

- Beforeincrementation/decrementation;incrementation/decrementation post-operation;
- After incrimination/decrementation; incrimination/decrementation pre-operation.



Implicit addressing with post operation incrementation

This is an implicit addressing with post operation incrimination.
Step no 1 – the location A212 is addressed, and a CLEAR is performed
Step no 2 – the Data Counter is incremented, thus having the value A213.



Implicit addressing with pre operation decrementation

This is an implicit addressing with pre operation decrementation.
Step no 1 – the DC content is modified/decremented, to the value BE23
Step no 2 – a CLEAR operation is performed at the address BE23.

Remark: There are CPUs that can define simultaneously several DCs, there fore it is possible to process several data blocks.

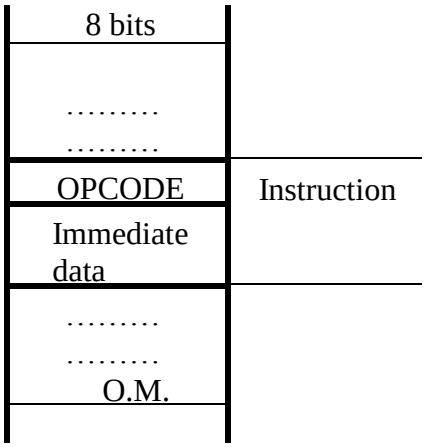
III. Immediate addressing

Thisaddressingmechanismviolatesthegeneraladdressingprinciple;theoperandisnotseparately stored but is an integral part of the instruction. This type of operand is called immediate data.

OPCODE	ID	OPERAND

Considering that there is the OPCODE field in the instruction format, it results that the length of the operand is shorter than the standard operand length, but the main advantage is that this data is immediately available after the **fetch instruction** phase, the **fetch data** phase being no longer necessary. Each **fetch** implies a memory read, a time consuming operation, but the execution is faster when the data is available together with the instruction itself.

In case of minicomputers with16 bit words, the immediate data that comes with the OPCODE is a 16 bit word (first the OPCODE and then the data).



16 bits	
…… ……	
OPCODE	Instruction
Immediate data	
…… …… O.M.	

IV.Direct addressing

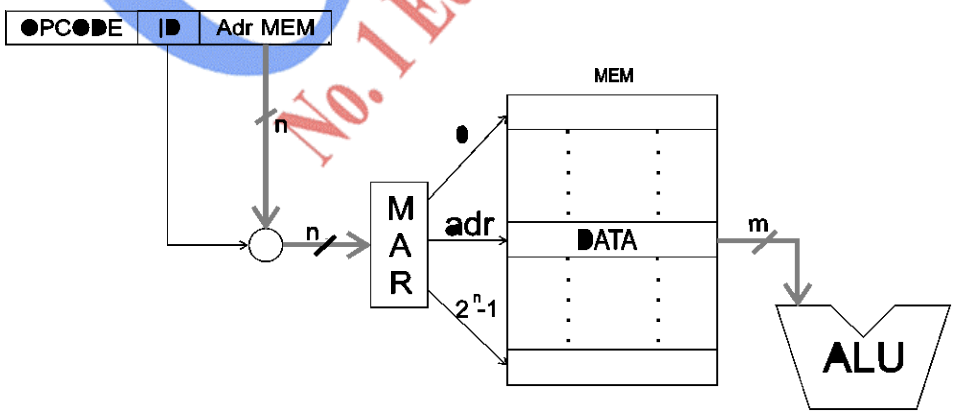
The direct addressing represents the natural memory addressing mode: the logic address coincides with the effective address.

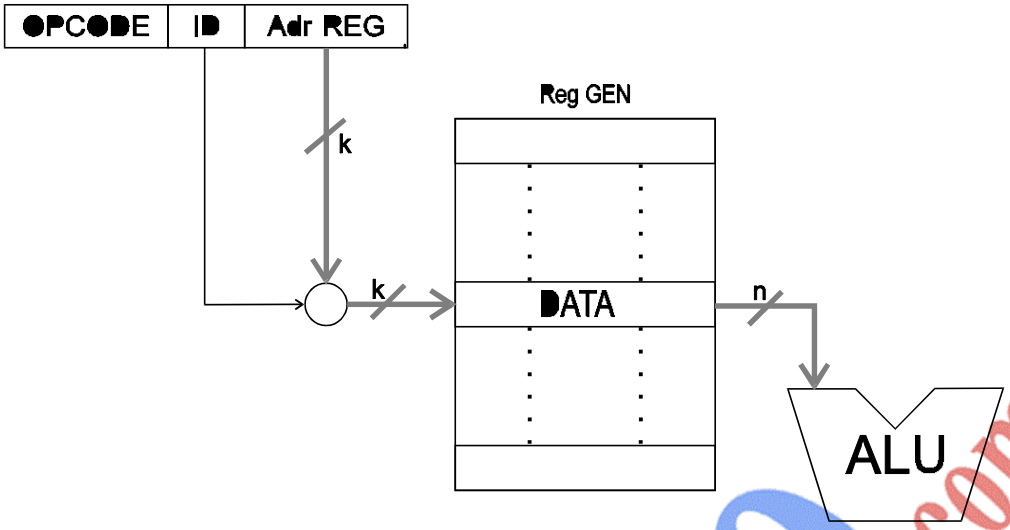
One can notice the following direct addressing sub classes:
1.direct addressing to a register;
2.direct addressing to memory;
3.direct addressing to an input-output device;
4.combined direct addressing.

1.Directaddressingtoregister

All modern processors contain a certain number of general registers with very fast access, whose reunion forms the ***local memory***. The current data to be processed is brought in to these registers. Because the number of these registers is limited (small), the register address is also small (*important advantage*).

On the other hand the access to data is much faster, because the memory read cycle is not started (because data isn't read from main memory).***Reading data from these local registers***/local memory ***is performed extremely fast***, due to the fact that they are CPU components.





Direct addressing to register

2.Directaddressingtomemory

In this case, the operand(data) is located in the main memory, in the data. Addressing the main memory is done by means of ***MAR***(memory address register).Ifthememoryhas2^nlocations then the address field must be on **n** bits.

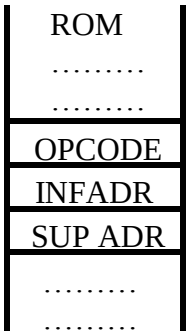
Because the modern memory modules are getting larger with each day, it results that the number of ranks used for address is steadily increasing also, there for the instructions whose operands are in memory and are retrieved in this manner necessitate a large number of bits (*longer length*).

Direct addressing to memory

Executing such an instruction assumes transfer of the logic address from the address field into the MAR register, start of a memory read cycle and extraction of the operand from the addressed location.

In case of the 8 bit microcomputers such an instruction has 3 bytes:

- the first byte is the OPCODE;
- the second byte is the inferior half of the address;
- the third byte is the superior half of the address.



After reading the first byte (OPCODE) the control unit decodes the instruction and thus knows it is a 3 byte instruction with direct addressing to memory.

3. Direct addressing to input-output devices
This category of instructions is very diverse and should contain the following fields:

4.Combined addressing

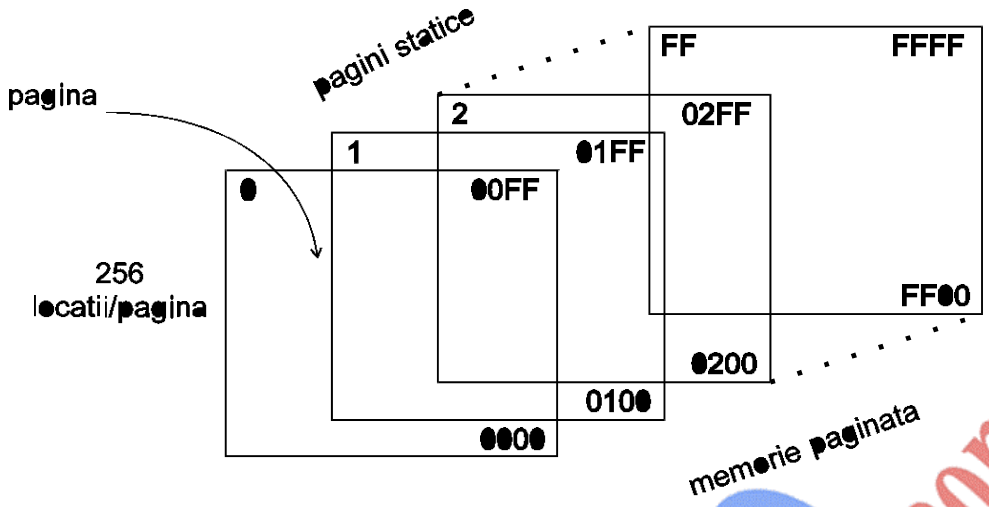
There are situations when one operand is located in the memory and these contain a general register so the instruction must contain both addresses .It is a combination of the first two cases. If both operands would be in the memory the instruction would be very long (*every inconvenient case*).

V. Paged addressing

Paged addressing is a variant of the direct. It was created out of then shorten the instruction length. It was shown that in case of addressing the memory the address field is long(as the memory volume increases the address field is longer).

There were searched solutions to shorten the address field. One efficient solution is ***dividing the whole memory space in memory pages***. This division is only conceptual and simplifies memory addressing.

Each page has a fixed number of locations, and the number of pages depends on the global volume of the memory and size of a page. The size of a page is linked to the length of the address field in the instruction. If the address field has k bites then the recommended size of a page is 2^klocations.



Example of memory organization on pages

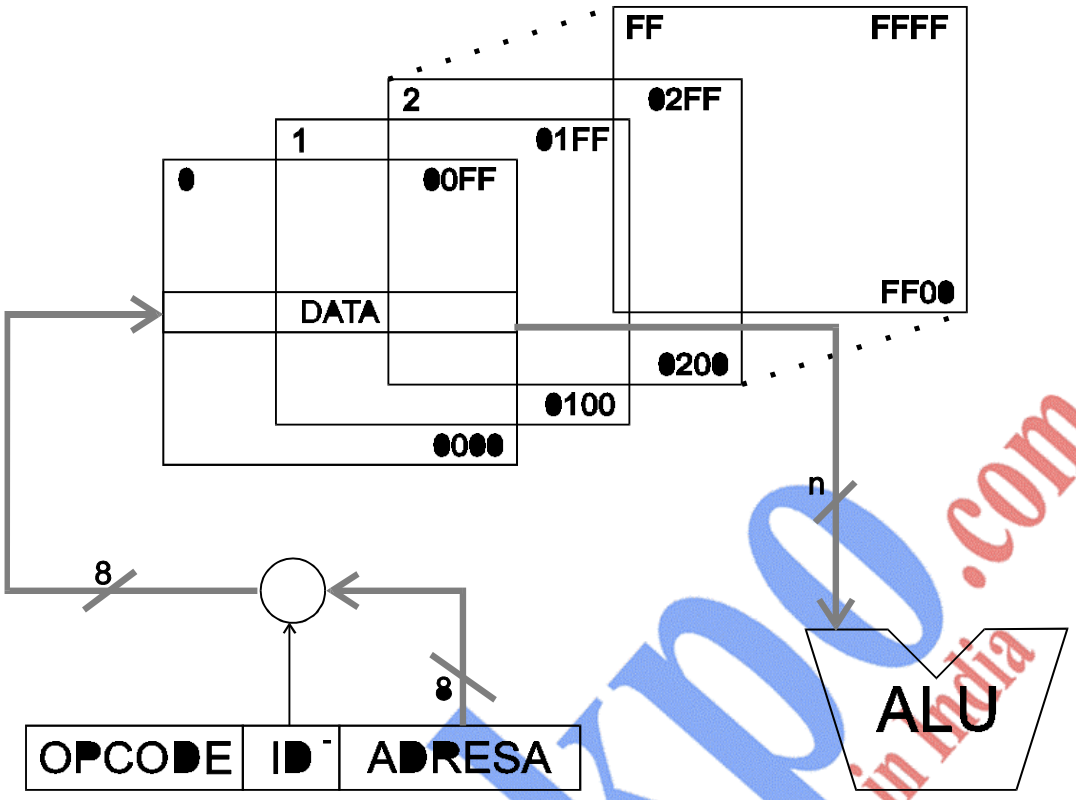
Paged organization of the memory issued in virtual memory as well.The pages can be located at fixed addresses in case of a fixed page or at mobile addresses in case of a dynamic page.

Example: it is considered an instruction with an 8 bit address field; then a page willcontain256 locations.

1.Direct addressing to page zero

It is used the address field on **8** bits (in general **k** bits) of the instruction to address the **256** locations, but only in page zero. The selected location is read and the operand transferred into the ALU.

The advantage of addressing into the page zero is a small length of the. Instead of using 16 bits for the address (in general **n** bits) one uses only8 bits (in general **k** bits where ***n>k***).

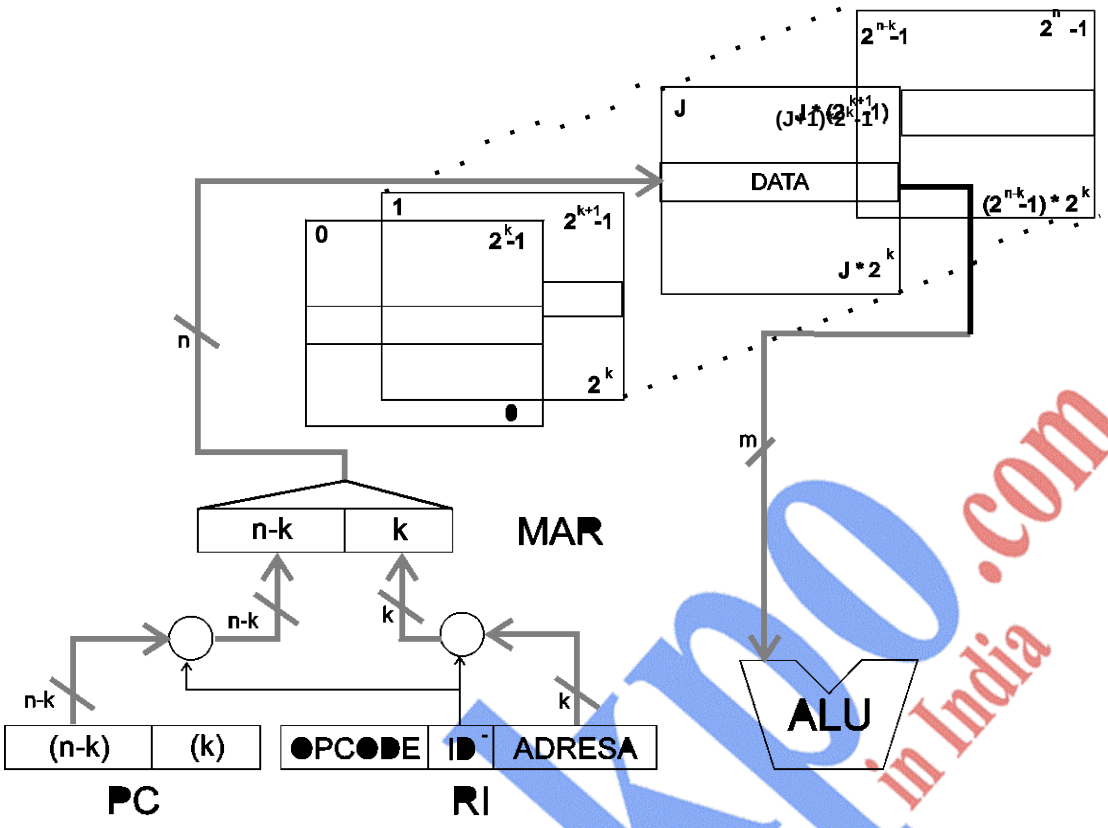


Direct addressing topage zero

The main disadvantage is the access only to this page, with no possibility to access another page.

2.Addressing the current page

In order to eliminate the disadvantages presented earlier, and to be able to address all pages ,a special mechanism will be used, combining a component of the instruction, the **logic address** and a component from a CPU register, namely the **PC**. It is assumed that the address has **n** bits for a total Volume of addressable locations of 2^n. Then the memory is organized in pages of 2^klocations, for a total of 2^{n-k} pages. The least significant part of the address formed of k bits allows addressing a location inside a page and the most significant part of the address on n-k bits allows determination of the right page. Then- k bits are taken from them significant n-k bits of the program counter. Combining the two parts is done through concatenation. The newly obtained address is placed in the MAR which is on n bits and then the desired location is read (a certain page, and a certain location).



Addressing the current page

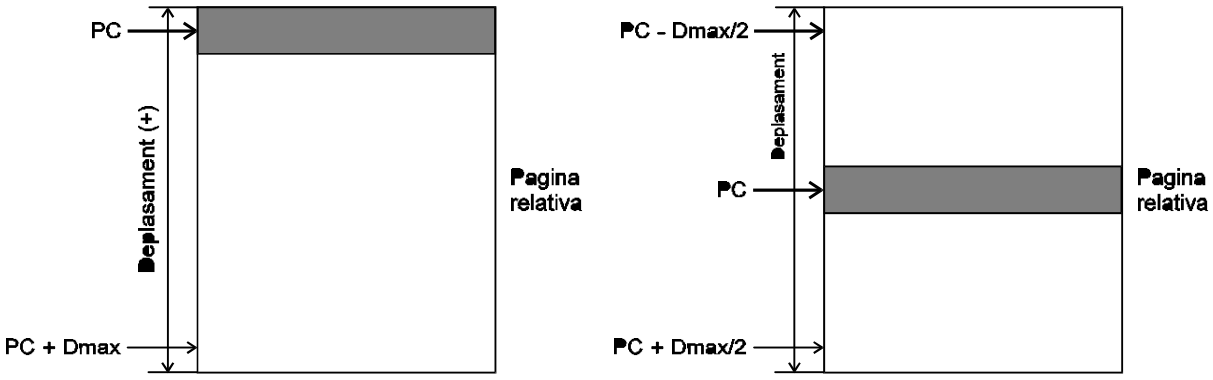
From the way the address is formed it results that ***the data required by an instruction has to be located in the vicinity of that instruction***. This vicinity is materialized by conserving the same page. So, when writing the program, and if using the same addressing technique, the programmer will place the data in the same memory page, because the page address is taken from the PC. When executing the current instruction, the PC has a certain value, out of which them significant(n-k) bits are taken to form the address for the page containing the operand.

There can result critical situations at the frontier between. It is known that after fetch instruction itis issuedthecommandtoincrementthePC.Ifforinstancethecurrentinstructionislocatedatthe address01FF(k=2,n - k = 2),thenafterincrementation01FF+1=0200andthepageaddress changes from 01 to 02.

3.Relative paged addressing

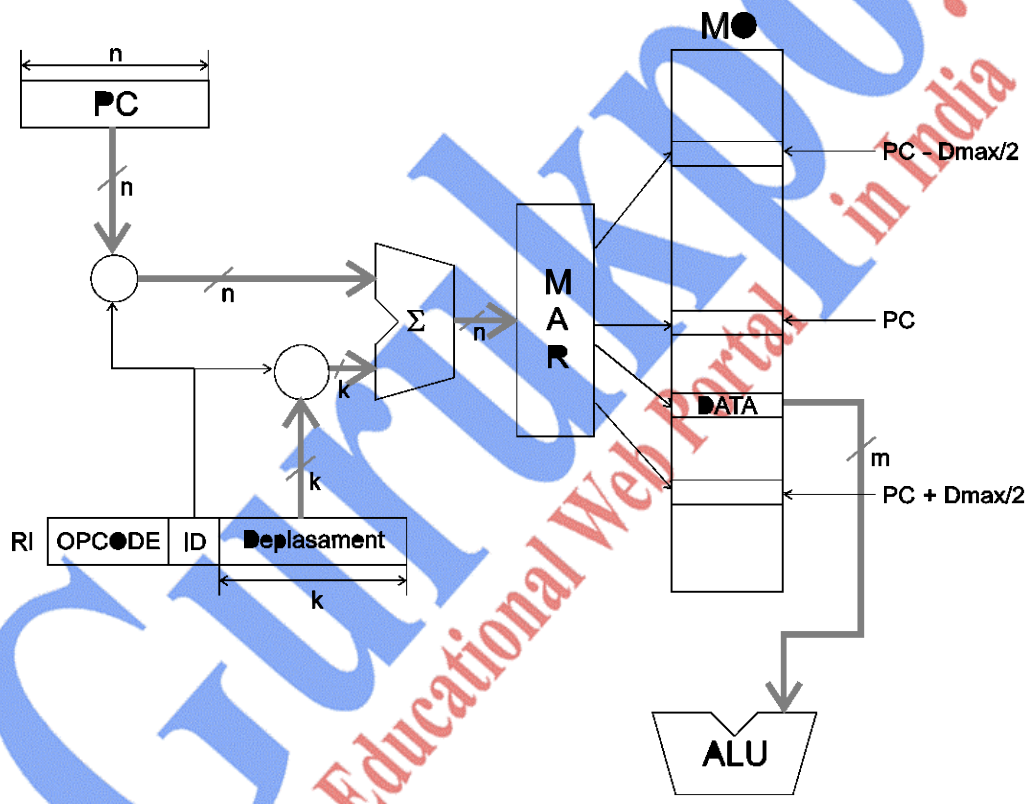
The notion of current page presented above is replaced in this case with the notion of relative page. This relative page is dynamic in the memory space. It is called relative because it is centered on the current value of the PC.So, in this addressing technique the PC is used with all its ranks. In the instruction the previously presented address field (the logic address) is now called **offset**.

This offset represents the value with which the PC is added to find the operand. The offset can be a positive binary number or a binary number with sign. The effective address is the sum of the PC an theoffsetandtheresultisplacedinMAR.Iftheoffsetispositivethenthebaseaddressofthe relative page is the address contained in the PC and the top is PC+D_{max}, where D_{max} is the maximum possible value of the offset. In case the offset is a binary number with sign then the relative page is centered around the location with the address given by the PC.



Page types

The PC has n ranks, while the offset has k ranks (k<n); the addition is performed on n ranks.



Relative page addressing

Example :It is considered an instruction already brought in the RI(instruction register)having an OPCODE, an identifier and an offset=2A,the address is written is hexadecimal digits, PC= 1A00. It is performed a sum between the PC and the offset, thus resulting the effective address.

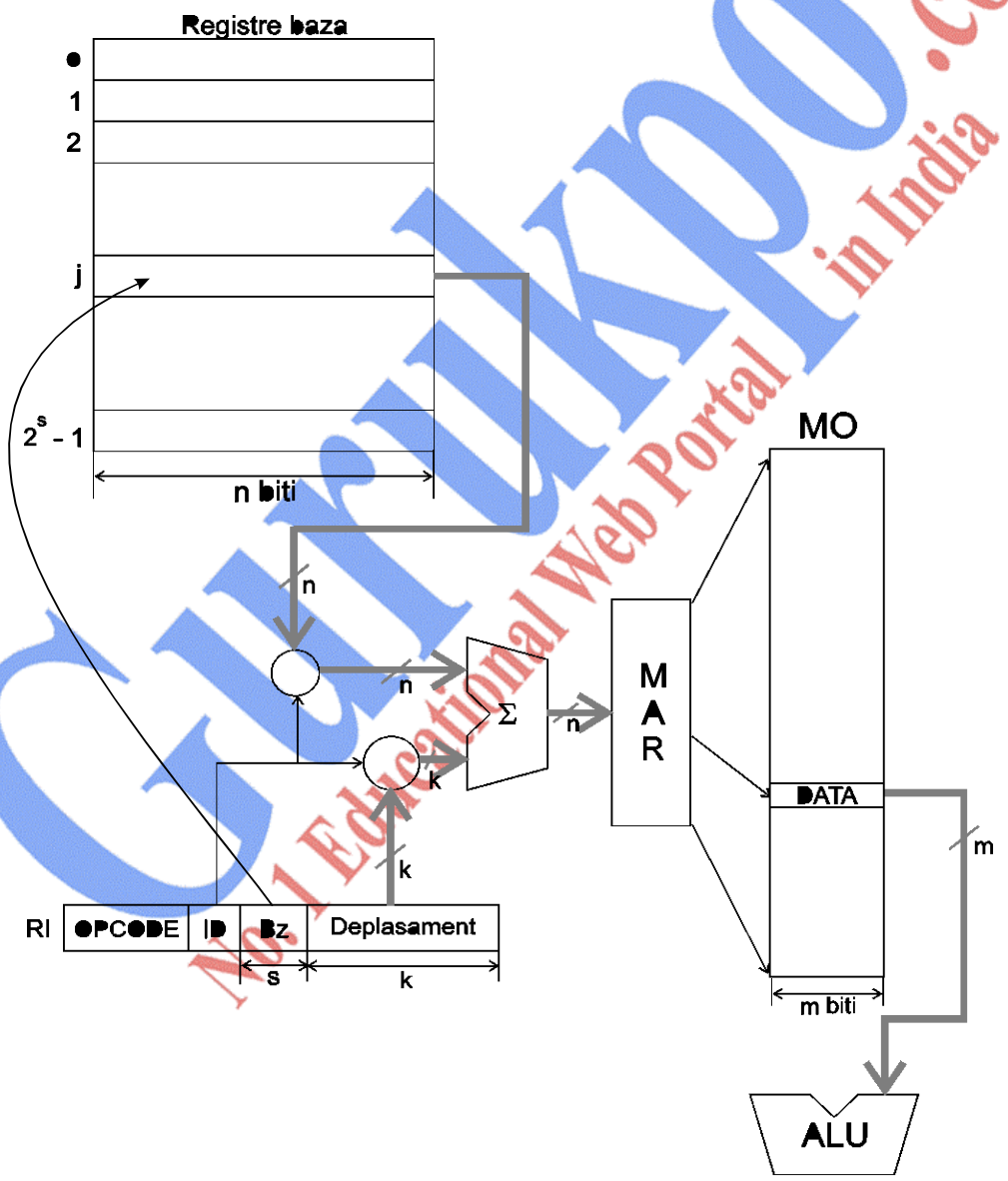
VI. Based addressing

It resembles as mechanism the relative page addressing ,except that instead of the PC,it uses the content of another register or several registers called base registers; these registers are part of the processor general registers group.

In this way it is eliminated the dependency between the data location and the program. The base register has the same length as the MAR, and in the instruction, in the address field it is placed the off set, which can be a positive binary number or a signed binary number. The length of the offset is smaller than that of the MAR. The length of the offset gives the width of the window, so a larger number of ranks in the address field yields a larger window (a larger page).

Calculating the effective address is done by summing the content of the base register with the offset. But because usually there are several base registers, a special field must appear in the instruction, called field B, whose value points to the desired base register.

If for instance there are 8 base registers numbered 0...7, then the B field must have the width 3.In some computer architectures it is adopted the convention that if B=0 then the effective address is given solely by the offset.



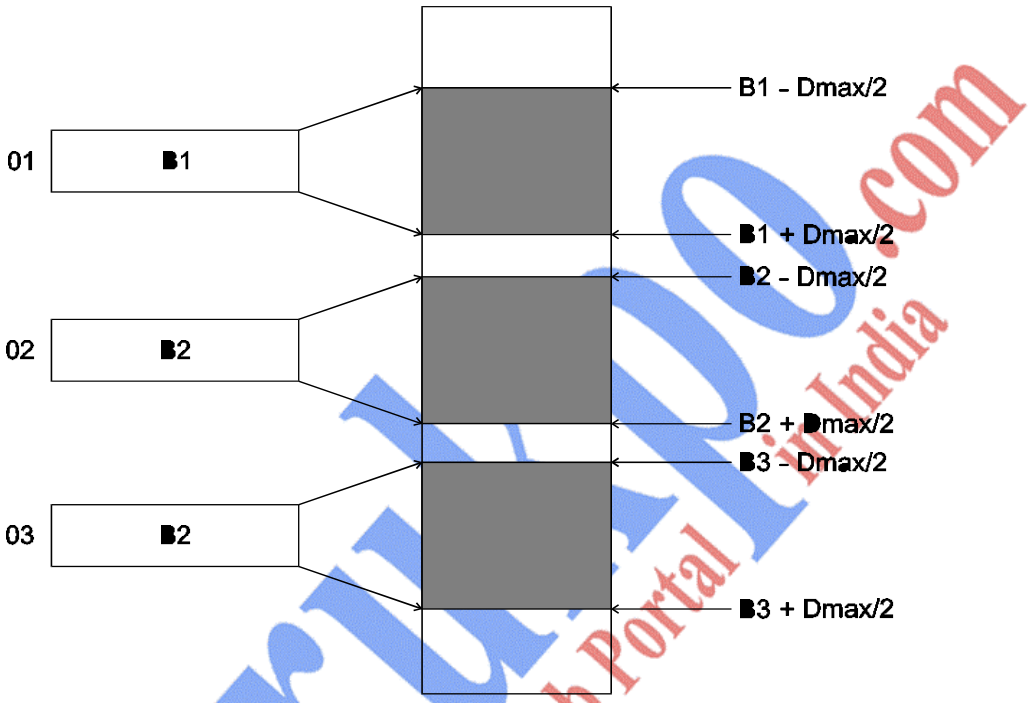
Based addressing

Loading initial values in the base register and modifying them during the execution of the program is done by means of software tools.

Definition: This addressing technique is called addressing through base and offset.

The content of the designated base register is either the centre of the page containing the data or the beginning of this page.

Because several base registers can be used it results that there can be simultaneously defined several current pages for the data. When defining these pages special attention must be paid to avoid address overlapping.



VII. Indexed addressing

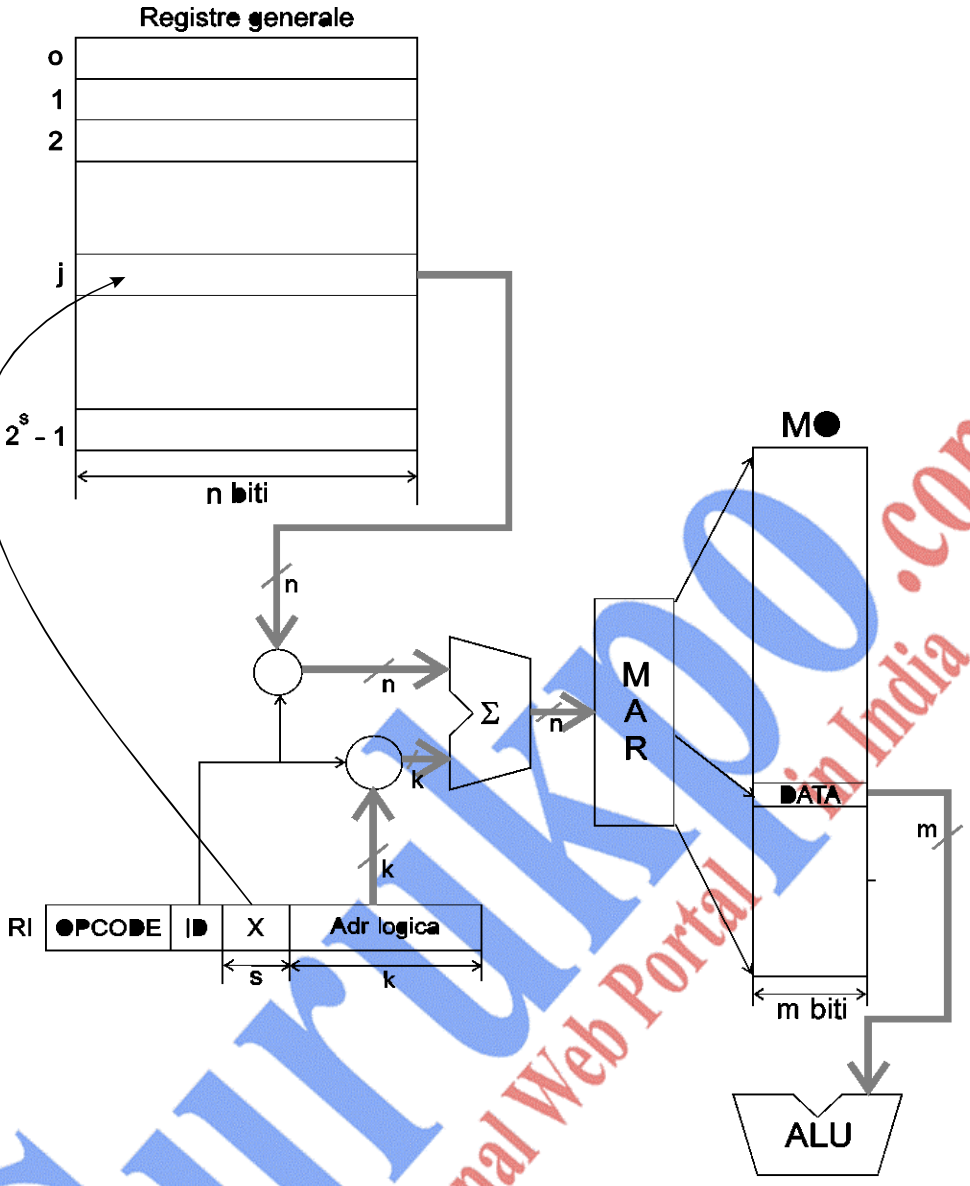
Indexed addressing is a variant of the based addressing. The base registers are replaced with index registers. These registers are part of the processor general registers group.The effective address is obtained in a similar manner as in based addressing: a summing is performed between the content of the index registers and the logic address from the instruction.

Definition. The content of the index register is called index.

What is particular to these index registers is the ability to perform a series of direct operations on them such as: increment, decrement, parallel load, addition, subtraction.

The index registers are frequently used in computer programming, especially in defining cycles. One can simultaneously define several indexes, allowing thus declaration of several cycles and thus nested cycles are possible.

IntheXfielditisintroducedtheaddressoftheindexregister.Iftherearesrankstherecanbe 2^sindexregisters.IntheIDfield(identification)thereappearcombinationsspecifyingacertain addressing technique.



Indexed addressing

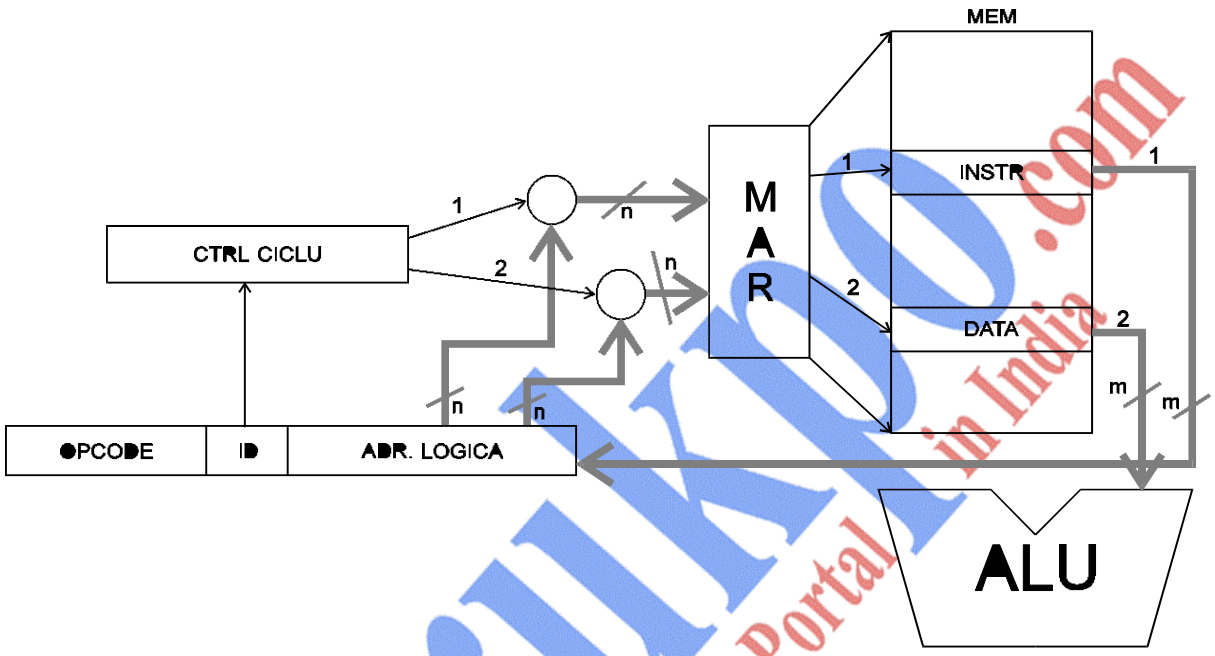
Example of indexed addressing usage: it is required the transfer of a data block with K components from the address area $M_1...M_k$ to $R_1...R_k$. Usually, there should be written k transfer instructions like MOV. Programming is simplified if indexed addressing is used.

In the instructions in the address field the constants R_1 and M_1 appear, and one can perform increment operations on the index register. In the $R_x$index register the K quantity is introduced. On $R_x$one can perform decrementations. When the value of the index becomes zero then the data block transfer is complete.

VIII .Indirect addressing

Indirect addressing is a modern addressing technique that allows shortening the programmes length and introduces flexibility in writing complex programmes.

In indirect addressing the logic address doesn't lead of the effective address of the operand the address of a new instruction. The memory location pointed by the address will be read and the data obtained will be interpreted as a new instruction.



Indirect addressing

In case of the microcomputers the indirect addressing is frequently performed as implicit addressing by means of general registers of the CPU. In case of INTELI 8080 the register pair HL, each on 8 bits, is used as logic address for accessing the operative(main) memory from where the operand is extracted. For instance there is the instruction ADD M(which performs an addition between ACC and the content of the memory location whose address is specified implicitly by the register pair HL content.

ACC(ACC) + Mem ((H)(L))

IX. Addressing for a block of data

It has been shown that a computer may operate with variable length fields as well(data blocks).To address them several options are possible:

1.It is given the start and end addresses for the block;
2.It is given the start address and the length of the field;
3.It is given the start address and it is used a blocked limited term earning a particular combination of bits used only for identifying the block end. This is not recommended as part of the memory is used to store the delimiter.

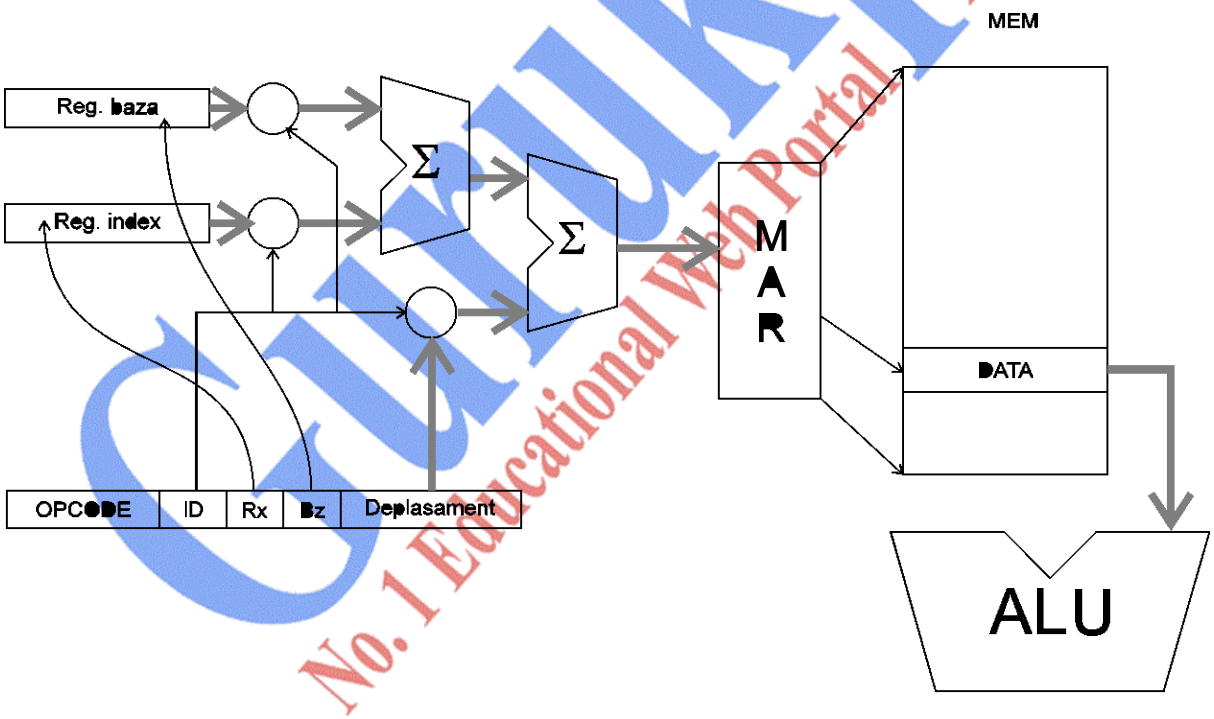
X. Combined addressing techniques

Most of the micro computers and computers use various combinations of the previously presented addressing techniques, in order to take advantage of their benefits.

1.*Multi level indirect addressing*. The indirect addressing is performed in several cycles; each time the information extracted from memory is not an operand but another instruction, except for the last read cycle . Usually there is a limit to levels. For instance in case of the FELIX C family of computers the maximum number of levels is 5.

2.*Indirect and indexed addressing*. In accordance with its name, this technique combines the two already presented techniques. The problem is when to perform the indexing. There are 2cases:indirect with post-indexing and with pre- indexing. In the first situation, the indirect addressing is performed (it can be a multilevel in direct addressing) and at the end it is added the value of the index (from the index register specified in the instruction) thus resulting the effective address. In these situation it is performed the indexed addressing mechanism at first, adding the logic address to the content of the index registe r resulting the address where the pointer is located. The pointer is extracted, anew addressing is performed and the operand is finally read.

3.*Based and indexed addressing*. The effective address calculation mechanism is the following:



Based-indexed addressing

(c) What is CPU design and implementation? Explain the process how a instruction execute in CPU.

Ans:- he operation or task that must perform by CPU are:

- **Fetch Instruction:** The CPU reads an instruction from memory.
- **Interpret Instruction:** The instruction is decoded to determine what action is required.
- **Fetch Data:** The execution of an instruction may require reading data from memory or I/O module.
- **Process data:** The execution of an instruction may require performing some arithmatic or logical operation on data.
- **Write data:** The result of an execution may require writing data to memory or an I/O module.

To do these tasks, it should be clear that the CPU needs to store some data temporarily. It must remember the location of the last instruction so that it can know where to get the next instruction. It needs to store instructions and data temporarily while an instruction is beign executed. In other words, the CPU needs a small internal memory. These storage location are generally referred as registers.

The major components of the CPU are an arithmetic and logic unit (ALU) and a control unit (CU). The ALU does the actual computation or processing of data. The CU controls the movement of data and instruction into and out of the CPU and controls the operation of the ALU.

The CPU is connected to the rest of the system through system bus. Through system bus, data or information gets transferred between the CPU and the other component of the system. The system bus may have three components:

DataBus:
Data bus is used to transfer the data between main memory and CPU.

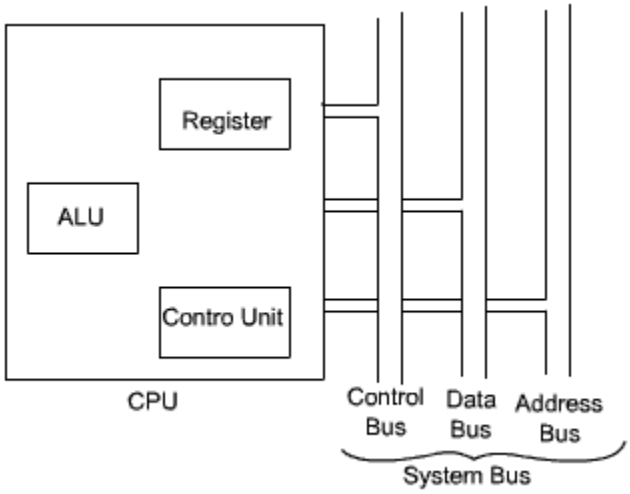
AddressBus:
Address bus is used to access a particular memory location by putting the address of the memory location.

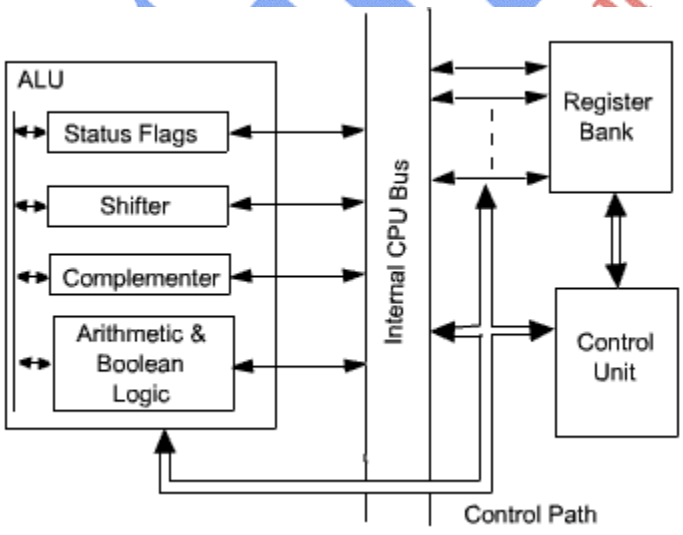
ControlBus:
Control bus is used to provide the different control signal generated by CPU to different part of the system. As for example, memory read is a signal generated by CPU to indicate that a memory read

operation has to be performed. Through control bus this signal is transferred to memory module to indicate the required operation.

There are three basic components of CPU: register bank, ALU and Control Unit. There are several data movements between these units and for that an internal CPU bus is used. Internal CPU bus is needed to transfer data between the various registers and the ALU.

The internal organization of CPU in more abstract level is shown in the Figure 5.1 and Figure 5.2.





Execution of a Complete Instructions:

We have discussed about four different types of basic operations:

- Fetch information from memory to CPU
- Store information to CPU register to memory
- Transfer of data between CPU registers.
- Perform arithmetic or logic operation and store the result in CPU registers.

To execute a complete instruction we need to take help of these basic operations and we need to execute these operation in some particular order.

As for example, consider the instruction : "Add contents of memory location NUM to the contents of register R1 and store the result in register R1." For simplicity, assume that the address NUM is given explicitly in the address field of the instruction .That is, in this instruction, direct addressing mode is used.
Execution of this instruction requires the following action :

1. Fetch instruction
2. Fetch first operand (Contents of memory location pointed at by the address field of the instruction)
3. Perform addition
4. Load the result into R1.

2. Following sequence of control steps are required to implement the above operation for the single-bus architecture that we have discussed in earlier section.

Steps	Actions
1.	PC_{out}, MAR_{in}, Read, Clear Y, Set carry -in to ALU, Add, Z_{in}
2.	Z_{out}, PC_{in}, Wait For MFC
3.	MDR_{out}, Ir_{in}
4.	Address-field- of-IR_{out}, MAR_{in}, Read
5.	$R1_{out}$, Y_{in}, Wait for MFC
6.	MDR_{out}, Add, Z_{in}
7.	Z_{out}, $R1_{in}$
8.	END

3. Instruction execution proceeds as follows:
4. **In Step1:**

The instruction fetch operation is initiated by loading the contents of the PC into the MAR and sending a read request to memory.

To perform this task first of all the contents of PC have to be brought to internal bus and then it is loaded to MAR.To perform this task control circuit has to generate the PC_{out} signal and MAR_{in} signal.

After issuing the read signal, CPU has to wait for some time to get the MFC signal. During that time PC is updated by 1 through the use of the ALU. This is accomplished by setting one of the inputs to the ALU (Register Y) to 0 and the other input is available in bus which is current value of PC.

At the same time, the carry-in to the ALU is set to 1 and an add operation is specified.

In Step 2:

The updated value is moved from register Z back into the PC. Step 2 is initiated immediately after issuing the memory Read request without waiting for completion of memory function. This is possible, because step 2 does not use the memory bus and its execution does not depend on the memory read operation.

In Step 3:

Step3 has been delayed until the MFC is received. Once MFC is received, the word fetched from the memory is transfered to IR (Instruction Register), Because it is an instruction. Step 1 through 3 constitute the instruction fetch phase of the control sequence.

The instruction fetch portion is same for all instructions. Next step onwards, instruction execution phase takes place.

As soon as the IR is loaded with instruction, the instruction decoding circuits interprets its contents. This enables the control circuitry to choose the appropriate signals for the remainder of the control sequence, step 4 to 8, which we referred to as the execution phase. To design the control sequence of execution phase, it is needed to have the knowledge of the internal structure and instruction format of the PU. Secondly , the length of instruction phase is different for different instruction.

In this example , we have assumed the following instruction format :

i.e., **opcode:** Operation Code
M: Memory address for source
R: Register address for source/destination

In Step 5 :

The destination field of IR, which contains the address of the register R1, is used to transfer the contents of register R1 to register Y and wait for Memory function Complete. When the read operation is completed, the memory operand is available in MDR.

In Step 6 :

The result of addition operation is performed in this step.

In Step 7:

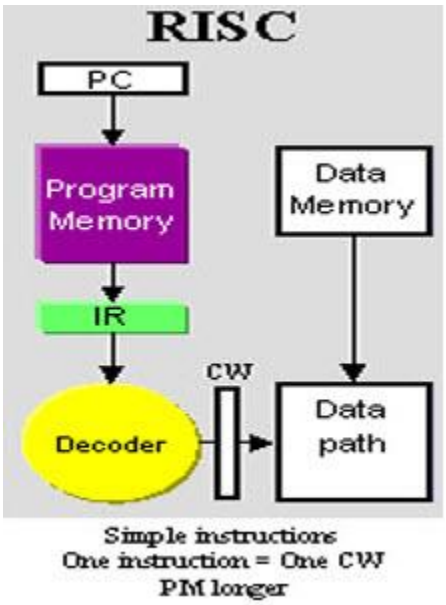
The result of addition operation is transferred from temporary register **Z** to the destination register **R1** in this step.

In step 8 :

It indicates the end of the execution of the instruction by generating End signal. This indicates completion of execution of the current instruction and causes a new fetch cycle to be started by going back to step 1.

(d)What is RISC and CISC ? Explain each with example and differentiate between RISC and CISC.

Ans:-RISC:-it is known as Reduced Instruction Set Computer. It is a type of microprocessor that has a limited number of instructions. They can execute their instructions very fast because instructions are very small and simple.



RISC chips require fewer transistors which make them cheaper to design and produce. In RISC, the instruction set contains simple and basic instructions from which more complex instruction can be produced. Most instructions complete in one cycle, which allows the processor to handle many instructions at same time.

Although RISC system have been defined and designed in a variety of ways by different groups, the key element shared by most design are these:

- A large number of general purpose registers, or the use of compiler technology to optimize register usage.

- A limited and simple instruction set.
- An emphasis on optimizing the instruction pipeline

An analysis of the RSIC architecture begins into focus many of the important issues in computer organization and architecture.

In this instructions are register based and data transfer takes place from register to register.

- It is known as Complex Instruction Set Computer.
- It was first developed by Intel.
- It contains large number of complex instructions.
- In this instructions are not register based.
- Instructions cannot be completed in one machine cycle.
- Data transfer is from memory to memory.
- Micro programmed control unit is found in CISC.
- Also they have variable instruction formats.

Characteristics of Reduced Instruction Set Architecture :

Although a variety of different approaches to reduce Instruction set architecture have been taken, certain characteristics are common to all of them:

1. One instruction per cycle.
2. Register–to–register operations.
3. Simple addressing modes.
4. Simple instruction formats.

1. One machine instruction per machine cycle :

A machine cycle is defined to be the time it takes to fetch two operands from registers, perform an ALU operation, and store the result in a register.
With simple, one-cycle instructions there is little or no need of microcode, the machine instructions can be hardwired. Hardware implementation of control unit executes faster than the micro programmed control, because it is not necessary to access a microprogram control store during instruction execution.

2. Register –to– register operations

With register–to–register operation, a simple LOAD and STORE operation is required to access the memory, because most of the operation are register–to-register. Generally we do not have memory–

Simple Addressing Modes

Almost all RISC instructions use simple register addressing. For memory access only, we may include some other addressing, such as displacement and PC-relative. Once the data are fetched inside the CPU, all instruction can be performed with simple register addressing.

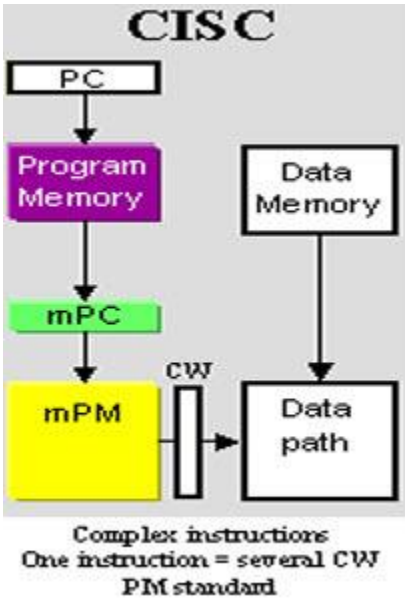
Simple Instruction Format

Generally in most of the RISC machine, only one or few formats are used. Instruction length is fixed and aligned on word boundaries. Field locations, especially the opcode, are fixed. With fixed fields, opcode decoding and register operand accessing can occur simultaneously. Simplified formats simplify the control unit.

CISC(Complex Instruction Set Computer):-

he term CISC stands for ''Complex Instruction Set Computer''. It is a CPU design plan based on single commands, which are skilled in executing multi-step operations.

CISC computers have small programs. It has a huge number of compound instructions, which takes a long time to perform. Here, a single set of instruction is protected in several steps; each instruction set has additional than 300 separate instructions. Maximum instructions are finished in two to ten machine cycles. In CISC, instruction pipelining is not easily implemented.



The computer designers intend to reduce this gap and include large instruction set, more addressing mode and various HLL statements implemented in hardware. As a result the instruction set becomes complex. Such complex instruction sets are intended to-

- Ease the task of the compiler writer.
- Improve execution efficiency, because complex sequences of operations can be implemented in microcode.
- Provide support for even more complex and sophisticated HLLs.

To reduce the gap between HLL and the instruction set of computer architecture, the system becomes more and more complex and the resulted system is termed as **Complex Instruction Set Computer (CISC)**.

The instruction execution characteristics involves the following aspects of computation:

- **Operation Performed:** These determine the functions to be performed by the processor and its interaction with memory.
- **Operand Used:** The types of operands and the frequency of their use determine the memory organization for storing them and the addressing modes for accessing them.
- **Execution sequencing:** This determines the control and pipeline organization.

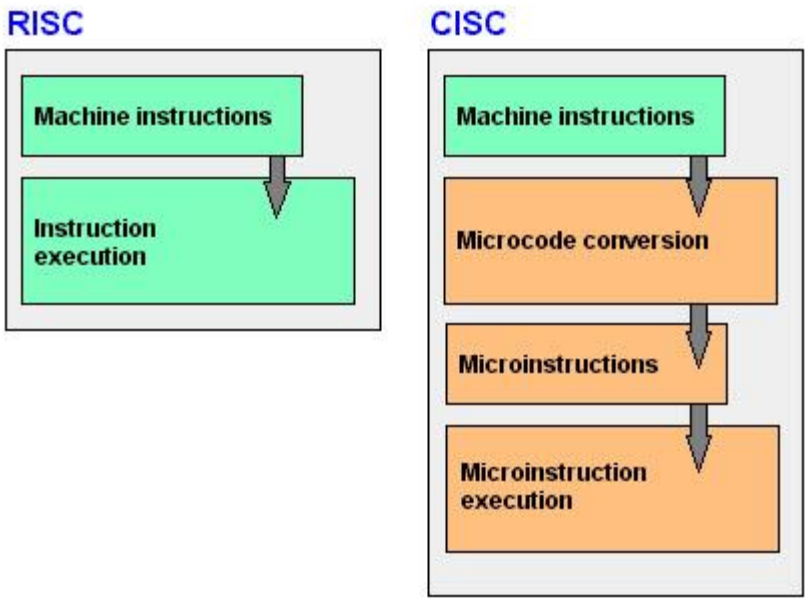
A variety of studies have been made to analyze the behavior of HLL programs. It is observed that

- Assignment statements predominates, suggesting that the simple movement of data is of high importance.
- There is also a presence of conditional statements (IF, Loop, etc.). These statements are implemented in machine language with some sort of compare and branch instruction. This suggest that the sequence control mechanism of the instruction set is important.
-

Difference Between CISC and RISC

1. RISC stands for Reduced Instruction Set Computer.	1. CISC stands for Complex Instruction Set Computer.
2. RISC processors have simple instructions taking about one clock cycle. The average clock cycle per instruction (CPI) is 1.5	2. CSIC processor has complex instructions that take up multiple clocks for execution. The average clock cycle per instruction (CPI) is in the range of 2 and 15.
3. Performance is optimized with more focus on software	3. Performance is optimized with more focus on hardware.
4. It has no memory unit and uses a separate hardware to implement instructions..	4. It has a memory unit to implement complex instructions.
5. It has a hard-wired unit of programming.	5. It has a microprogramming unit.
6. The instruction set is reduced i.e. it has	**6.** The instruction set has a variety of different

only a few instructions in the instruction set. Many of these instructions are very primitive.	instructions that can be used for complex operations.
7. The instruction set has a variety of different instructions that can be used for complex operations.	**7.** CISC has many different addressing modes and can thus be used to represent higher-level programming language statements more efficiently.
8. Complex addressing modes are synthesized using the software.	8. CISC already supports complex addressing modes
9. Multiple register sets are present	9. Only has a single register set
10. RISC processors are highly pipelined	10. They are normally not pipelined or less pipelined
11. The complexity of RISC lies with the compiler that executes the program	11. The complexity lies in the microprogram
12. Execution time is very less	12. Execution time is very high
13. Code expansion can be a problem	13. Code expansion is not a problem
14. Decoding of instructions is simple.	14. Decoding of instructions is complex
15. It does not require external memory for calculations	15. It requires external memory for calculations
16. The most common RISC microprocessors are Alpha, ARC, ARM, AVR, MIPS, PA-RISC, PIC, Power Architecture, and SPARC.	16. Examples of CISC processors are the System/360, VAX, PDP-11, Motorola 68000 family, AMD and Intel x86 CPUs.
17. RISC architecture is used in high-end applications such as video processing, telecommunications and image processing.	17. CISC architecture is used in low-end applications such as security systems, home automation, etc.



(e) Write Short notes on

(i) Differentiate between Microsequencer and Microcontroller.

Ans:-a **sequencer** or **micro sequencer** generates the addresses used to step through the microprogram of a control store. It is used as a part of the control unit of a CPU or as a stand-alone generator for address ranges.the addresses are generated by some combination of a counter, a field from a microinstruction, and some subset of the instruction register. A counter is used for the typical case, that the next microinstruction is the one to execute. A field from the microinstruction is used for jumps, or other logic.Since CPUs implement an instruction set, it's very useful to be able to decode the instruction's bits directly into the sequencer, to select a set of microinstructions to perform a CPU's instructions.

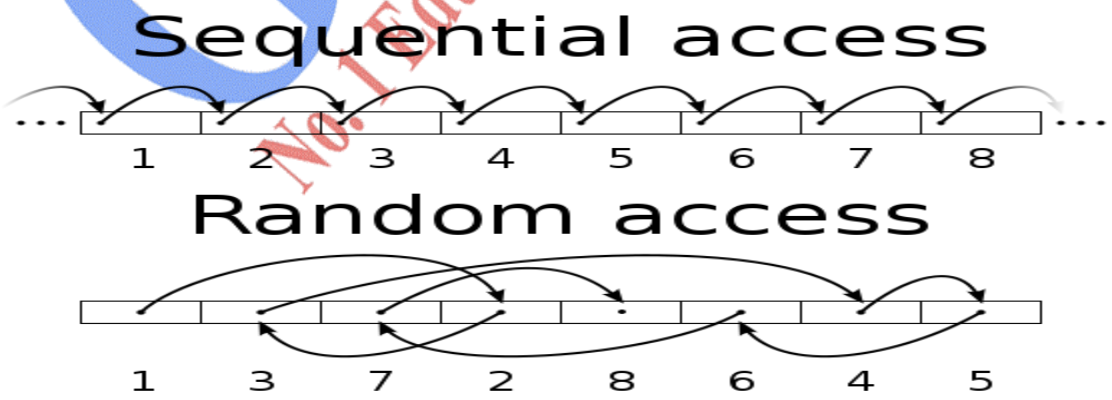
A **microcontroller** contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals. Program memory in the form of Ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a small amount of RAM.

A **microcontroller** can be considered a self-contained system with a processor, memory and peripherals and can be **used** as an embedded system. The majority of**microcontrollers** in **use** today are embedded in other machinery, such as automobiles,

telephones, appliances, and peripherals for computer systems.

(ii) Random V/S sequential Access in Storage Device

Comparing random versus sequential operations is one way of assessing application efficiency in terms of disk use. Accessing data sequentially is much faster than accessing it randomly because of the way in which the disk hardware works. The seek operation, which occurs when the disk head positions itself at the right disk cylinder to access data requested, takes more time than any other part of the I/O process. Because reading randomly involves a higher number of seek operations than does sequential reading, random reads deliver a lower rate of throughput. The same is true for random writing.



(iii) Static v/s Dynamic RAM

SRAM (static RAM) is random access memory (RAM) that retains data bits in its memory as long as power is being supplied. Unlike dynamic RAM (DRAM), which stores bits in cells consisting of a capacitor and a transistor, SRAM does not have to be periodically refreshed. Static RAM provides faster access to data and is more expensive than DRAM. SRAM is used for a computer's cache memory and as part of the random access memory digital-to-analog converter on a video card.A dynamic RAM chip holds millions of memory cells, each made up of a transistor and a capacitor. The chip constantly needs to be refreshed. Static RAM differs as it holds information in a flip flop manner, which means it does not require to constantly refresh and do not use capacitors.

(iv) Pipelining:

Pipelining is an implementation technique where multiple instructions are overlapped in execution. The**computer pipeline** is divided in stages. Each stage completes a part of an instruction in parallel. ... We call the time required to move an instruction one step further in the **pipeline** a machine cycle .A pipe is a message queue. A message can be anything. A filter is a process, thread, or other component that perpetually reads messages from an input pipe, one at a time, processes each message, then writes the result to an output pipe.Without a **pipeline**, a **computer** processor gets the first instruction from memory, performs the operation it calls for, and then goes to get the next instruction from memory, and so forth. While fetching (getting) the instruction, the arithmetic part of the processor is idle.