UNIT -5 (DECO)

Basic Processing Unit:

A computer in its simplest form comprises five functional units namely input unit, output unit memory unit, arithmetic & logic unit and control unit. Figure 1 depicts the functional units of a computer system.

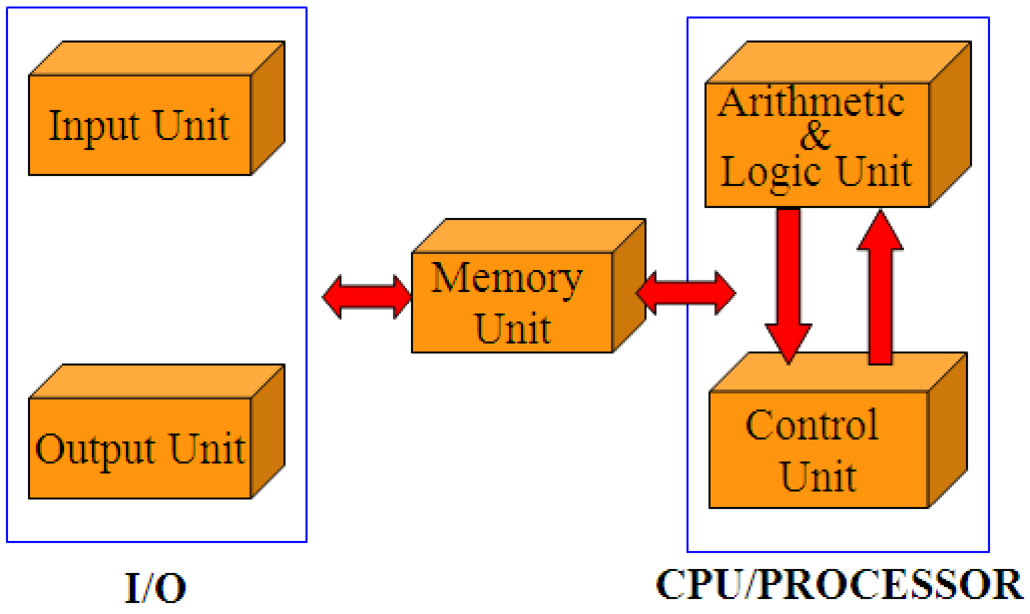


Figure 1: Basic functional units of a computer

1. Input Unit: Computer accepts encoded information through input unit. The standard input device is a keyboard. Whenever a key is pressed, keyboard controller sends the code to CPU/Memory.

Examples include Mouse, Joystick, Tracker ball, Light pen, Digitizer, Scanner etc.

2. Memory Unit: Memory unit stores the program instructions (Code), data and results of computations etc. Memory unit is classified as:

a) Primary /Main Memory

b) Secondary /Auxiliary Memory

Primary memory is a semiconductor memory that provides access at high speed. Run time program instructions and operands are stored in the main memory. Main memory is classified again as ROM and RAM. ROM holds system programs and firmware routines such as BIOS, POST, I/O Drivers that are essential to manage the hardware of a computer. RAM is termed as Read/Write memory or user memory that holds run time program instruction and data. While primary storage is essential, it is volatile in nature and expensive. Additional requirement of memory could be supplied as auxiliary memory at cheaper cost. **Secondary memories** are non-volatile in nature.

3. Arithmetic and logic unit: ALU consist of necessary logic circuits like adder, comparator etc., to perform operations of addition, multiplication, comparison of two numbers etc.

4. Output Unit: Computer after computation returns the computed results, error messages, etc. via output unit. The standard output device is a video monitor, LCD/TFT monitor. Other output devices are printers, plotters etc.

5. Control Unit: Control unit co-ordinates activities of all units by issuing control signals. Control signals issued by control unit govern the data transfers and then appropriate operations take place. Control unit interprets or decides the operation/action to be performed.

The operations of a computer can be summarized as follows:

1. A set of instructions called a program reside in the main memory of computer.

2. The CPU fetches those instructions sequentially one-by-one from the main memory, decodes them and performs the specified operation on associated data operands in ALU.

3. Processed data and results will be displayed on an output unit.

4. All activities pertaining to processing and data movement inside the computer machine are governed by control unit.

Arithmetic Logic Unit

ALU is responsible to perform the operation in the computer.
The basic operations are implemented in hardware level. ALU is having collection of two types of operations:

1.Arthmetic-Operations

2.Logical-operations

Consider an ALU having 4 arithmetic operations and 4 logical operation.

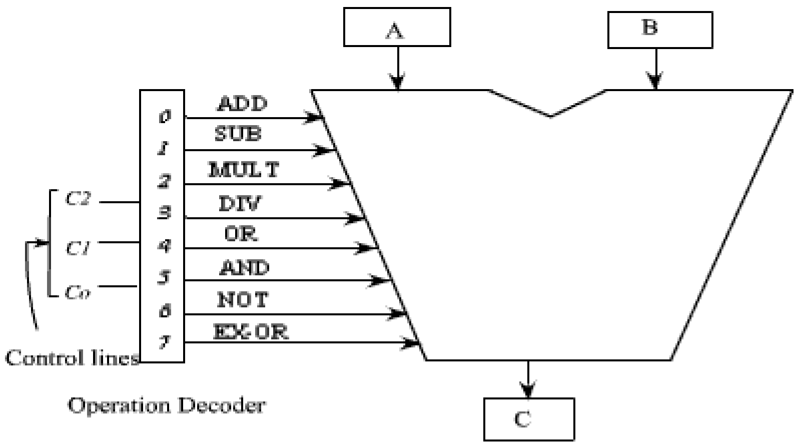
To identify any one of these four logical operations or four arithmetic operations, two control lines are needed. Also to identify the any one of these two groups- arithmetic or logical, another control line is needed. So, with the help of three control lines, any one of these eight operations can be identified.Consider an ALU is having four arithmetic operations. Addition, subtraction, multiplication and division. Also consider that the ALU is having four logical operations: OR, AND, NOT & EX-OR.

We need three control lines to identify any one of these operations. The input combination of these control lines are shown below:

C_1	C_0	Arithmetic $C_2 = 0$	Logical $C_2 = 1$
0	0	Addition	OR
0	1	Subtraction	AND
1	0	Multiplication	NOT
1	1	Division	**EX-OR**

Control line is used to identify the group: logical or arithmetic, i.e: arithmetic operation: logical operation.
Control lines and are used to identify any one of the four operations in a group. One possible combination is given here.

A decode is used to decode the instruction. The block diagram of the ALU is shown in figure below.



The ALU has got two input registers named as A and B and one output storage register, named as C. It performs the operation as:

C = A op B

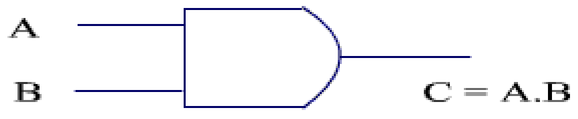
The input data are stored in A and B, and according to the operation specified in the control lines, the ALU perform the operation and put the result in register C.

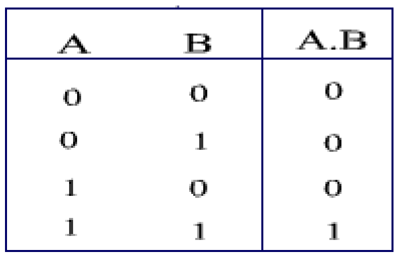
As for example, if the contents of controls lines are, 000, then the decoder enables the addition operation and it activates the adder circuit and the addition operation is performed on the data that are available in storage register A and B. After the completion of the operation, the result is stored in register C.

We should have some hardware implementations for basic operations. These basic operations can be used to implement some complicated operations which are not feasible to implement directly in hardware.

There are several logic gates exists in digital logic circuit. These logic gates can be used to implement the logical operation.

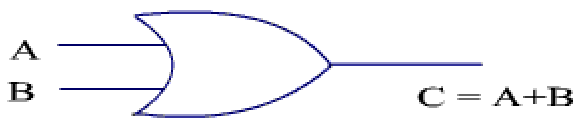
Some of the common logic gates are mentioned here.
AND gate: The output is high if both the inputs are high. The AND gate and its truth table is shown in Figure below

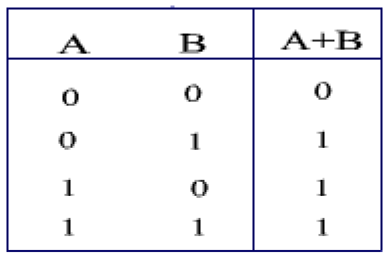




A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

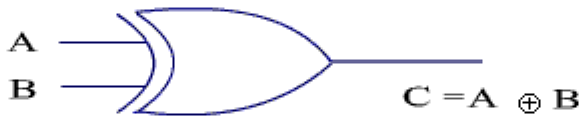
OR gate: The output is high if any one of the input is high. The OR gate and its truth table is shown in Figure below.

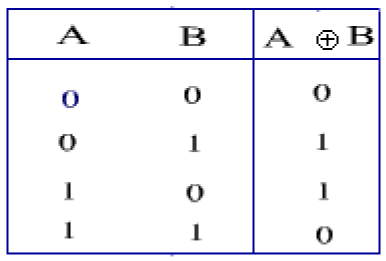




A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

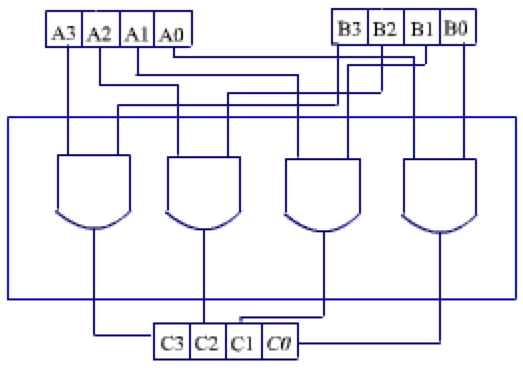
EX-OR gate: The output is high if either of the input is high. The EX-OR gate and its truth table is given in Figure below.





A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

If we want to construct a circuit which will perform the AND operation on two 4-bit number, the implementation of the 4-bit AND operation is shown in the Figure below.



Arithmetic Circuit

Binary Adder :

Binary adder is used to add two binary numbers.

In general, the adder circuit needs two binary inputs and two binary outputs. The input variables designate the augends and addend bits;The output variables produce the sum and carry.

The binary addition operation of single bit is shown in the truth table

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

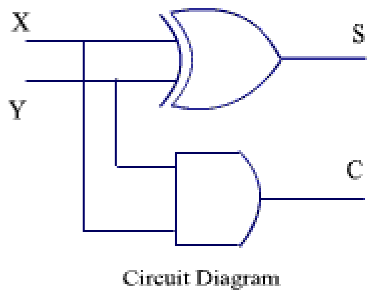
C: Carry Bit

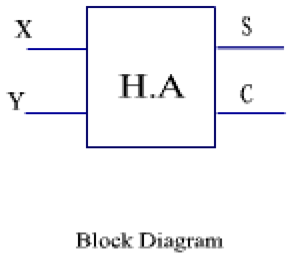
S: Sum Bit

The simplified sum of products expressions are

$S = x'y + xy'$

$C = xy$



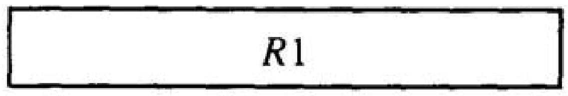


Circuit diagram and Block diagram of Half Adder

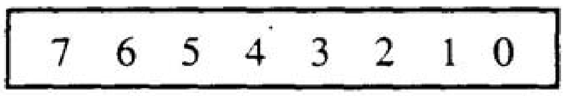
Register Transfer

Digital systems are composed of modules that are constructed from digital components, such as registers, decoders, arithmetic elements, and control logics. The modules are interconnected with common data and control paths to form a digital computer system .The operations executed on data stored in registers are called micro-operations .A micro-operation is an elementary operation performed on the information stored in one or more registers.
Examples are shift, count, clear, and load.

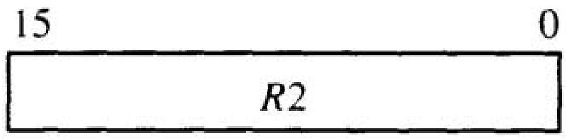
Designate computer registers by capital letters to denote its function. The register that holds an address for the memory unit is called MAR. The program counter register is called PC. IR is the instruction register and R1 is a processor register. The individual flip-flops in an n-bit register are numbered in sequence from 0 to n-1



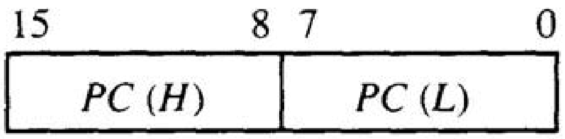
(a) Register *R*



(b) Showing individual bits

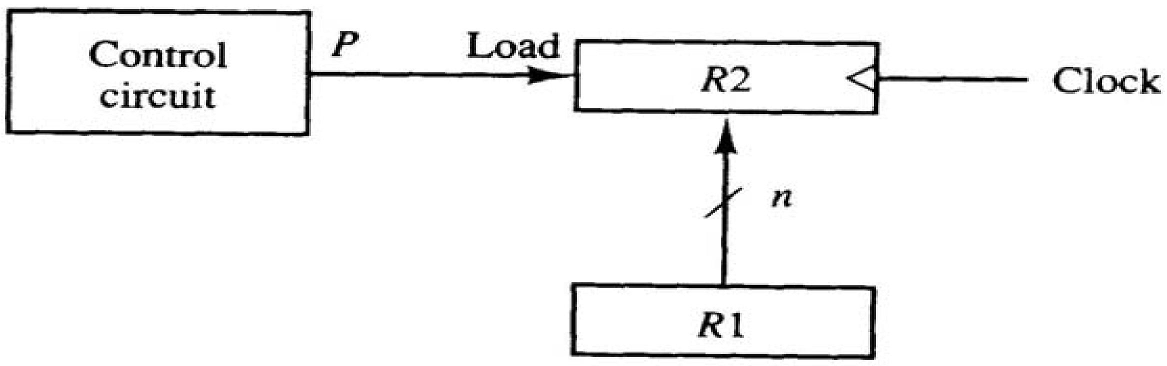


(c) Numbering of bits



(d) Divided into two parts

If the transfer is to occur only under a predetermined control condition, designate it by
If (P = 1) then (R2→ R1) or, P: R2 →R1, where P is a control function that can be either 0 or 1
every statement written in register transfer notation implies the presence of the required hardware construction.



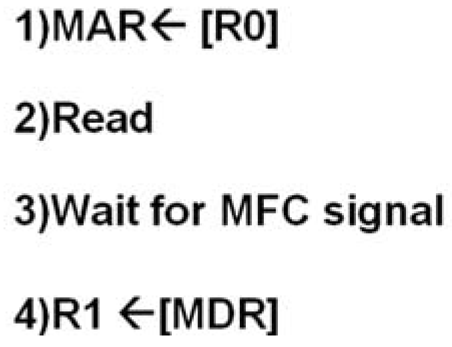
Word:

In computer architecture, a word is a unit of data of a defined bit length that can be addressed and moved between storage and the computer processor. Usually, the defined bit length of a word is equivalent to the width of the computer's data bus so that a word can be moved in a single operation from storage to a processor register. For any computer architecture with an eight-bit byte, the word will be some multiple of eight bits. In IBM's evolutionary System/360 architecture, a word is 32 bits, or four contiguous eight-bit bytes. In Intel's PC processor architecture, a word is 16 bits, or two contiguous eight-bit bytes. A word can contain a computer instruction, a storage address, or application data that is to be manipulated (for example, added to the data in another word space). The number of bits in each word is known as word length. Word length refers to the number of bits processed by the CPU in one go. With modern general purpose computers, word size can be 16 bits to 64 bits. The time required to access one word is called the memory access time. The small, fast, RAM units are called caches. They are tightly coupled with the processor and are often contained on the same IC chip to achieve high performance.

Fetching a word from memory

- CPU transfers the address of the required information to MAR from where it is transferred through Address Bus to Memory
- In the same time CPU uses it's control lines of memory bus to indicate that a read operation is required
- After issuing this request CPU waits until it receives a feedback from the memory indicating that the requested function has been completed
- This is done using another control signal on the memory bus, referred to as Memory Function Completed (MFC)
- The memory sets this signal to 1 to indicate that the contents of the specified location in the memory have been read and are available on the data lines of the memory bus and thus available for use inside the CPU.
- This completes the memory operation

Assume that the address of the memory location to be accessed is in register R0 and data is to be loaded into register R1. Following sequence of operations are used.



1)MAR$\leftarrow$ [R0]

2)Read

3)Wait for MFC signal

4)R1 $\leftarrow$[MDR]

Duration of step depends on Memory access time. During this time CPU can carry out tasks that do not require MAR and MDR.

Storing a word in to memory

That is similar process with fetching a word from memory.

- The required address is loaded into the MAR
- After that data to be written are loaded into MDR, and a write command is issued.
- If we suppose that the data word to be stored in the memory is in R2 and that the memory address is in R1, the Write operation needed the following sequence:
- MAR - [R1]

- MDR -[R2]
- Write
- Wait for the MFC

Move R2, (R1) requires the following sequence (signal):

R1out, MARin

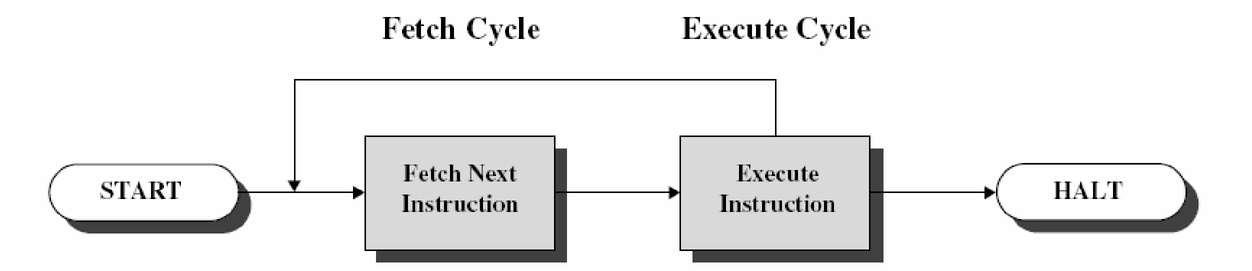
R2out, MDRin. Write

MDRoutE,WMFC

Execution of Complete Instruction

An instruction cycle consists of an instruction fetch, followed by zero or more operand fetches, followed by zero or more operand stores, followed by an interrupt check (if interrupts are enabled) The major computer system components (processor, main memory, I/O modules) need to be interconnected in order to exchange data and control signals. The most popular means on interconnection is the use of a shared system bus consisting on multiple lines.

The basic function performed by a computer is execution of a program, which consists of a set of instructions stored in memory. The processor does the actual work by executing instructions specified in the program. In the simplest form, instruction processing consists of two steps: the processor reads (fetches) instructions from memory one at a time and executes each instruction. The processing required for a single instruction is called an instruction cycle. An instruction cycle is shown in figure:



Program execution halts only if the machine is turned off, some sort of unrecoverable error occurs, or a program instruction that halts the computer is encountered.

Instruction Fetch and Execute

• The processor fetches an instruction from memory – program counter (PC) register holds the address of the instruction to be fetched next

• The processor increments the PC after each instruction fetch so that it will fetch the next instruction in the sequence – unless told otherwise

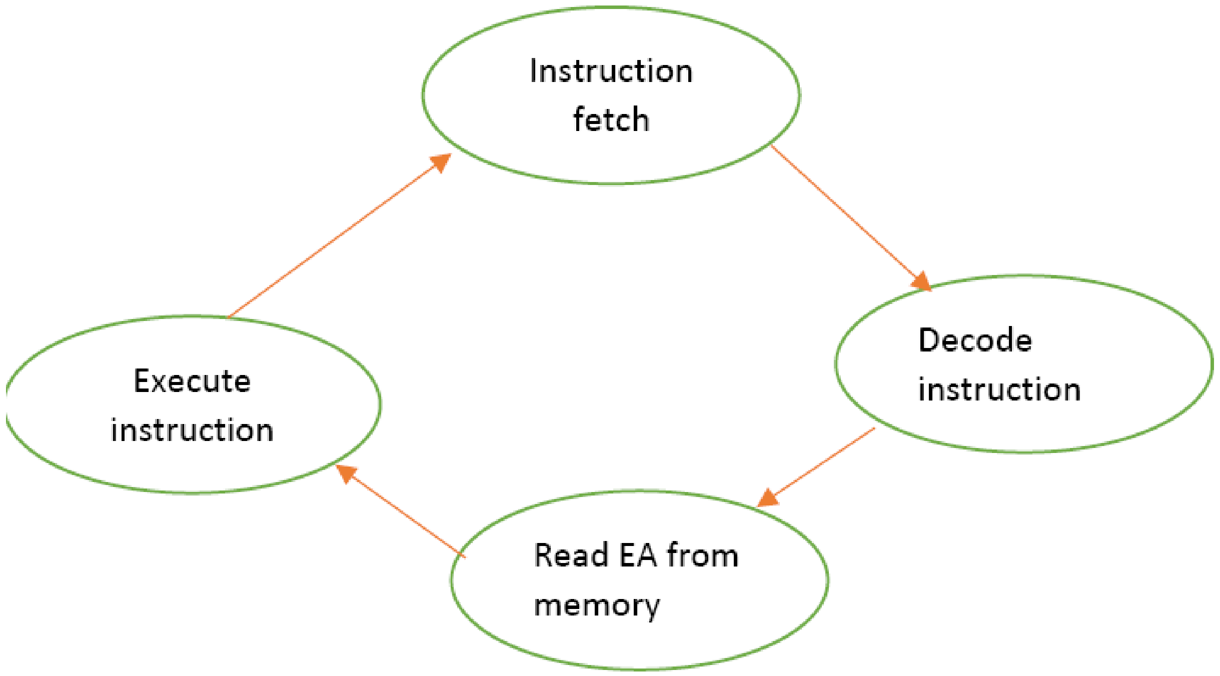
• The fetched instruction is loaded into the instruction register (IR) in the processor – the instruction contains bits that specify the action.

Instruction Cycle

A program that exists inside a computer's memory unit consists of a series of instructions. The processor executes these instructions through a cycle for each instruction.

In a basic computer, each instruction cycle consists of the following phases:

Instruction fetch: fetch instruction from memory
Decode the instruction: what operation to be performed
Read the effective address from memory
Execute the instruction



Registers Involved In Each Instruction Cycle:
Memory address registers (MAR): It is connected to System Bus address lines. It specifies the address of a read or write operation in memory.
Memory Buffer Register (MBR): It is connected to the data lines of the system bus. : It is connected to the system bus Data Lines. It holds the memory value to be stored, 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.

Fetch cycle

The address of the next instruction to execute is in the Program Counter (PC) at the beginning of the fetch cycle.

Step 1: The address in the program counter is transferred to the Memory Address Register (MAR), as this is the only register that is connected to the system bus address lines.

Step 2: The address in MAR is put on the address bus, now a Read order is provided by the control unit on the control bus, and the result appears on the data bus and is then copied into the memory buffer register. Program counter is incremented by one, to get ready for the next instruction. These two acts can be carried out concurrently to save time.

Step 3: The content of the MBR is moved to the instruction register (IR).

Instruction fetch cycle consist of four micro operation:

T1: MAR← PC

T2: MBR ← memory

PC ← PC + step size or length of instruction

T3: IR ← MBR

Decode instruction cycle

The next move is to fetch source operands once an instruction is fetched. Indirect addressing (it can be obtained by any addressing mode, here it is done by indirect addressing) is obtained by Source Operand. You don't need to fetch register-based operands. If the opcode is executed, it will require a similar process to store the result in main memory. Micro-operations take place:

T1: MAR ←IR(address)

T2: MBR ←Memory

T3: IR(address) ← (MBR(Address))

Step 1: The instruction address field is passed to the MAR. This is used to fetch the operand's address.

Step 2: The address field of the IR is updated from the MBR.

Step 3: The IR is now in the state. Now IR is ready for the execute cycle.

Execute instruction Cycle

The initial three cycles (Fetch, Indirect, and Interrupt) are predictable and quick. Each requires simple , small, and fixed micro-operation sequences. The same micro-operation is repeated every time around in each event. Execute instruction cycle is different from them. Like, there is N different series of micro-operations that can occur for a computer with different N opcodes.

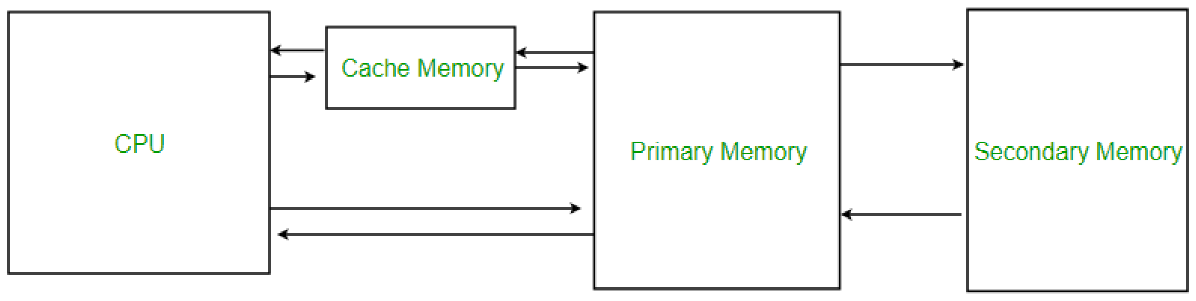
Execute instruction Cycle
The initial three cycles (Fetch, Indirect, and Interrupt) are predictable and quick. Each requires simple , small, and fixed micro-operation sequences. The same micro-operation is repeated every time around in each event. Execute instruction cycle is different from them. Like, there is N different series of micro-operations that can occur for a computer with different N opcodes.
Example
ADD R , X
T1: MAR← (IR(address))
T2: MBR← Memory
T3: R ← (R) + (MBR)
Step 1: The address portion of IR is loaded into the MAR. **Step 2:** The address field of the IR is updated from the MBR, so the reference memory location is read. **Step 3:** Now, the contents of R and MBR are added by the ALU

Cache Memory

Cache Memory is a special very high-speed memory. The cache is a smaller and faster memory that stores copies of the data from frequently used main memory locations. There are various different independent caches in a CPU, which store instructions and data. The most important use of cache memory is that it is used to reduce the average time to access data from the main memory.

Characteristics of Cache Memory

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



Cache Memory

Levels of Memory

- **Level 1 or Register:** It is a type of memory in which data is stored and accepted that are immediately stored in the CPU. The most commonly used register is Accumulator, Program counter, Address Register, etc.
- **Level 2 or Cache memory:** It is the fastest memory that has faster access time where data is temporarily stored for faster access.
- **Level 3 or Main Memory:** It is the memory on which the computer works currently. It is small in size and once power is off data no longer stays in this memory.
- **Level 4 or Secondary Memory:** It is external memory that is not as fast as the main memory but data stays permanently in this memory.

Cache Performance

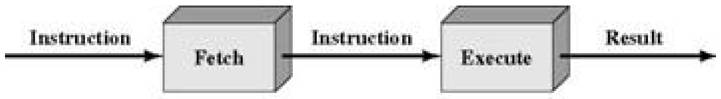
When the processor needs to read or write a location in the main memory, it first checks for a corresponding entry in the cache.

- If the processor finds that the memory location is in the cache, a Cache Hit has occurred and data is read from the cache.
- If the processor does not find the memory location in the cache, a **cache miss** has occurred. For a cache miss, the cache allocates a new entry and copies in data from the main memory, then the request is fulfilled from the contents of the cache.

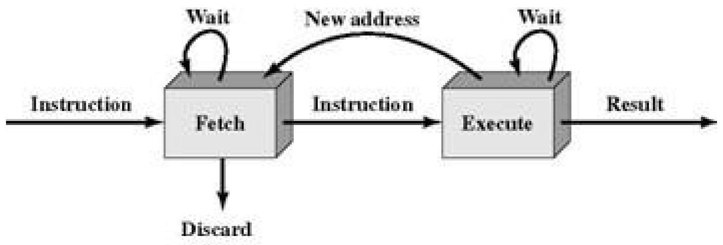
The performance of cache memory is frequently measured in terms of a quantity called **Hit ratio.**

Pipe lining

Performance of a computer can be increased by increasing the performance of the CPU. This can be done by executing more than one task at a time. This procedure is referred to as pipelining. The concept of pipelining is to allow the processing of a new task even though the processing of previous task has not ended. As computer systems evolve, greater performance can be achieved by taking advantage of improvements in technology, such as faster circuitry, use of multiple registers rather than a single accumulator, and the use of a cache memory. Another organizational approach is instruction pipelining in which new inputs are accepted at one end before previously accepted inputs appear as outputs at the other end.



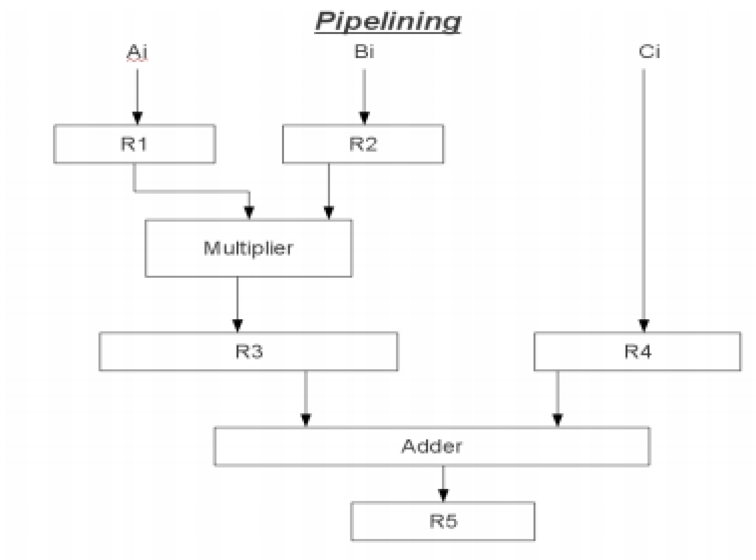
3.1(a) Simplified View



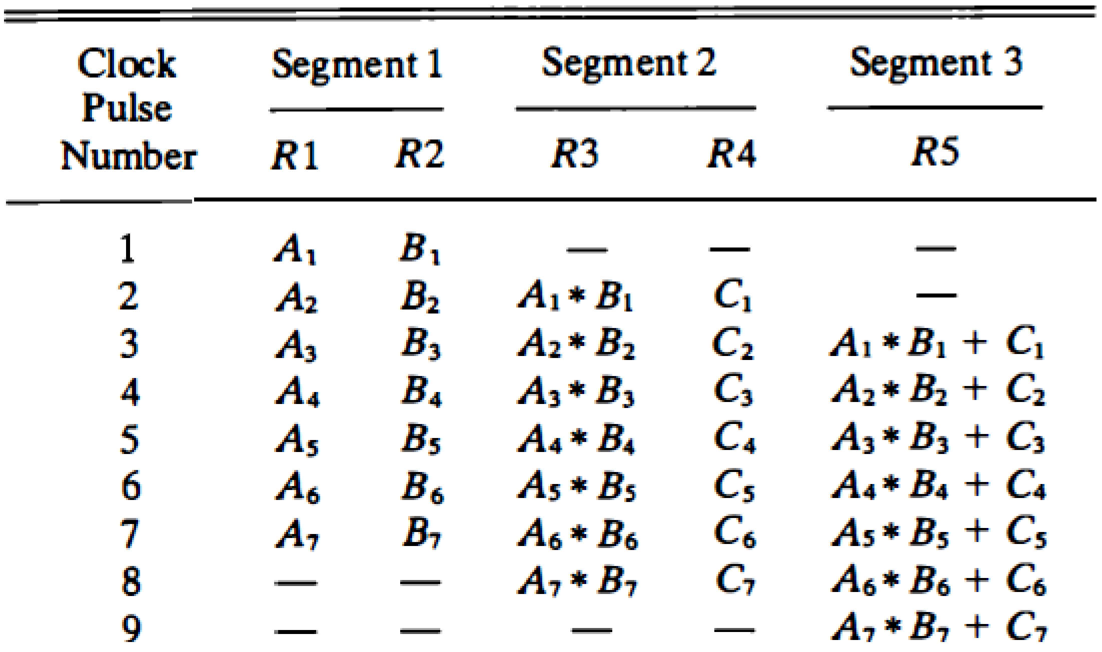
3.1(b) Expanded View

3.1 Two-Stage Instruction Pipeline

Pipelining is a technique of decomposing a sequential process into sub operations, with each sub process being executed in a special dedicated segment that operates concurrently with all other segments.A pipeline can be visualized as a collection of processing segments through which binary information flows. Each segment performs partial processing dictated by the way the task is partitioned. The result obtained from the computation in each segment is transferred to the next segment in the pipeline. The final result is obtained after the data have passed through all segments. Consider the following operation: Result=(A+B)*C First the A and B values are Fetched which is nothing but a "Fetch Operation". The result of the Fetch operations is given as input to the Addition operation, which is an Arithmetic operation. The result of the Arithmetic operation is again given to the Data operand C which is fetched from the memory and using another arithmetic operation which is Multiplication in this scenario is executed. Finally the Result is again stored in the "Result" variable. In this process we are using up-to 5 pipelines which are Fetch Operation (A), Fetch Operation(B) Addition of (A & B), Fetch Operation(C) Multiplication of ((A+B), C) Load ((A+B)*C)



Now consider the case where a k-segment pipeline with a clock cycle time t, is used to execute n tasks. The first task T1 requires a time equal to k t, to complete its operation since there are k segments in the pipe. The remaining n - 1 tasks emerge from the pipe at the rate of one task per clock cycle and they will be completed after a time equal to (n - 1)t, . Therefore, to complete n tasks using a k-segment pipeline requires k + (n - 1) clock cycles. For example, the diagram of Fig. shows four segments and six tasks. The time required to complete all the operations is 4 + (6 - 1) = 9 clock cycles, as indicated in the diagram



Clock Pulse Number	Segment 1		Segment 2		Segment 3
	$R1$	$R2$	$R3$	$R4$	$R5$
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 * B_1$	C_1	—
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_7	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
8	—	—	$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$
9	—	—	—	—	$A_7 * B_7 + C_7$

Pipeline Performance:

Throughput and Speedup Parallel processing is a term used to denote a large class of techniques that are used to provide simultaneous data-processing tasks for the purpose of inaeasing the computational speed of a computer system. The purpose of parallel processing is to speed up the computer processing capability and increase its throughput.

Throughput: Is the amount of processing that can be accomplished during a given interval of time. The amount of hardware increases with parallel processing and with it, the cost of the system increases. However, technological developments have reduced hardware costs to the point where parallel processing techniques a.re economically feasible.

Speedup of a pipeline processing: The speedup of a pipeline processing over an equivalent nonpipeline processing is defined by the ratio

$$\boxed{S = Tseq / Tpipe = n*m / (m+n-1)}$$

the maximum speed up, also called ideal speedup, of a pipeline processor with m stages over an equivalent nonpipelined processor is m. In other words, the ideal speedup is equal to the number of pipeline stages. That is, when n is very large, a pipelined processor can produce output approximately m times faster than a nonpipelined processor. When n is small, the speedup decreases.