Tulsiramji Gaikwad-Patil College of Engineering & Technology,
Nagpur Department of Information Technology

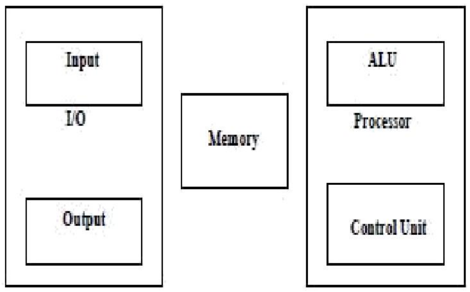
Notes
Academic Session: 2018- 2019

Subject: CAO **Semester: IV**

Unit – I

Syllabus
Basic Structure of Computers: Functional Units, Basic Operational Concepts, Bus Structures, Software, Multiprocessors and Multicomputer. Machine Instructions: Memory Locations and Addresses, Memory Operations, Machine program sequencing, addressing modes and encoding of information, Assembly Language, Stacks, Queues and Subroutine.

Q1)	**What is computer? Explain different types of computer.**	**7M**
Ans:-	**Computer types** 1. A computer can be defined as a fast electronic calculating machine that accepts the (data) digitized input information process it as per the list of internally stored instructions and produces the resulting information. 2. List of instructions are called programs & internal storage is called computer memory. **The different types of computers are** **1. Personal computers:** - This is the most common type found in homes, schools, Business offices etc., It is the most common type of desk top computers with processing and storage units along with various input and output devices. **2. Note book computers:** - These are compact and portable versions of PC 3. Work stations: - These have high resolution input/output (I/O) graphics capability, but with same dimensions as that of desktop computer. These are used in engineering applications of interactive design work. **4. Enterprise systems:** - These are used for business data processing in medium to large corporations that require much more computing power and storage capacity than work stations. Internet associated with servers has become a dominant worldwide source of all types of information. **5. Super computers:** - These are used for large scale numerical calculations required in the applications like weather forecasting etc.	

Q2)	Explain different Functional unit of computer.	7M
Ans:-	**Functional unit** 1. A computer consists of five functionally independent main parts input, memory, arithmetic logic unit (ALU), and output and control unit.  Input I/O Output Memory ALU Processor Control Unit Fig a : Functional units of computer 2. Input device accepts the coded information as source program i.e. high level language. This is either stored in the memory or immediately used by the processor to perform the desired operations. 3. The program stored in the memory determines the processing steps. Basically the computer converts one source program to an object program (i.e. into machine language). 4. Finally the results are sent to the outside world through output device. All of these actions are coordinated by the control unit. **Input unit: -** 1. The source program/high level language program/coded information/simply data is fed to a computer through input devices keyboard is a most common type. 2. Whenever a key is pressed, one corresponding word or number is translated into its equivalent binary code over a cable & fed either to memory or processor. 3. Joysticks, trackballs, mouse, scanners etc are other input devices. **Memory unit: -** 1. Its function into store programs and data. It is basically to two types 1. Primary memory 2. Secondary memory **1. Primary memory: -** 1. Is the one exclusively associated with the processor and operates at the electronics speeds programs must be stored in this memory while they	

are being executed.

2. The memory contains a large number of semiconductors storage cells. Each capable of storing one bit of information. These are processed in a group of fixed site called word.
3. To provide easy access to a word in memory, a distinct address is associated with each word location. Addresses are numbers that identify memory location.
4. Number of bits in each word is called word length of the computer. Programs must reside in the memory during execution. Instructions and data can be written into the memory or read out under the control of processor.
5. Memory in which any location can be reached in a short and fixed amount of time after specifying its address is called random-access memory (RAM).
6. The time required to access one word in called memory access time. Memory which is only readable by the user and contents of which can't be altered is called read only memory (ROM) it contains operating system.
7. Caches are the small fast RAM units, which are coupled with the processor and are often contained on the same IC chip to achieve high performance. Although primary storage is essential it tends to be expensive.

2 Secondary memory: -

1. Is used where large amounts of data & programs have to be stored, particularly information that is accessed infrequently.
2. Examples: - Magnetic disks & tapes, optical disks (ie CD-ROM's), floppies etc.,

Arithmetic logic unit (ALU):-

1. Most of the computer operators are executed in ALU of the processor like addition, subtraction, division, multiplication, etc. the operands are brought into the ALU from memory and stored in high speed storage elements called register.
2. Then according to the instructions the operation is performed in the required sequence.
3. The control and the ALU are many times faster than other devices connected to a computer system.
4. This enables a single processor to control a number of external devices

such as key boards, displays, magnetic and optical disks, sensors and other mechanical controllers.

Output unit:-

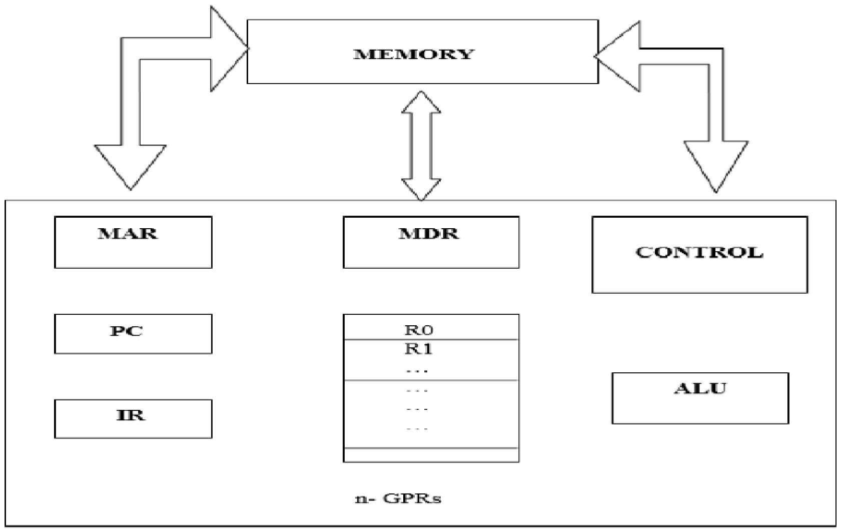
1. These actually are the counterparts of input unit. Its basic function is to send the processed results to the outside world.
2. Examples: - Printer, speakers, monitor etc.

Control unit: - It effectively is the nerve center that sends signals to other units and senses their states. The actual timing signals that govern the transfer of data between input unit, processor, memory and output unit are generated by the control unit.

Q3) **Explain Basic operational concepts of computer.**

Ans:- **Basic operational concepts**

1. To perform a given task an appropriate program consisting of a list of instructions is stored in the memory. Individual instructions are brought from the memory into the processor, which executes the specified operations. Data to be stored are also stored in the memory. Examples: - Add LOCA, R0
2. This instruction adds the operand at memory location LOCA, to operand in register R0 & places the sum into register. This instruction requires the performance of several steps,
   1. First the instruction is fetched from the memory into the processor.
   2. The operand at LOCA is fetched and added to the contents of R0
   3. Finally the resulting sum is stored in the register R0
3. The preceding ads instruction combines a memory access operation with an ALU Operations. In some other type of computers, these two types of operations are performed by separate instructions for performance reasons.
4. Load LOCA, R1 Add R1, R0 Transfers between the memory and the processor are started by sending the address of the memory location to be accessed to the memory unit and issuing the appropriate control signals. The data are then transferred to or from the memory.



1. The fig shows how memory & the processor can be connected. In addition to the ALU & the control circuitry, the processor contains a number of registers used for several different purposes.
2. The instruction register (IR):- Holds the instructions that are currently being executed. Its output is available for the control circuits which generates the timing signals that control the various processing elements in one execution of instruction.
3. The program counter PC: - This is another specialized register that keeps track of execution of a program. It contains the memory address of the next instruction to be fetched and executed.
4. Besides IR and PC, there are n-general purpose registers R0 through Rn-1.

   a. The other two registers which facilitate communication with memory are: MAR – (Memory Address Register):- It holds the address of the location to be accessed.
   b. MDR – (Memory Data Register):- It contains the data to be written into or read out of the address location.

Operating steps are

1. Programs reside in the memory & usually get these through the I/P unit.
2. Execution of the program starts when the PC is set to point at the first instruction of the program.
3. Contents of PC are transferred to MAR and a Read Control Signal is

sent to the memory.

4. After the time required to access the memory elapses, the address word is read out of the memory and loaded into the MDR.
5. Now contents of MDR are transferred to the IR & now the instruction is ready to be decoded and executed.
6. If the instruction involves an operation by the ALU, it is necessary to obtain the required operands.
7. An operand in the memory is fetched by sending its address to MAR & Initiating a read cycle.
8. When the operand has been read from the memory to the MDR, it is transferred from MDR to the ALU.
9. After one or two such repeated cycles, the ALU can perform the desired operation.
10. If the result of this operation is to be stored in the memory, the result is sent to MDR.
11. Address of location where the result is stored is sent to MAR & a write cycle is initiated.
12. The contents of PC are incremented so that PC points to the next instruction that is to be executed.

1. Normal execution of a program may be preempted (temporarily interrupted) if some devices require urgent servicing, to do this one device raises an Interrupt signal.
2. An interrupt is a request signal from an I/O device for service by the processor. The processor provides the requested service by executing an appropriate interrupt service routine.
3. The Diversion may change the internal stage of the processor its state must be saved in the memory location before interruption. When the interrupt-routine service is completed the state of the processor is restored so that the interrupted program may continue.

Q4) Explain Bus structure of computer Ans:-

Bus structure

1. The simplest and most common way of interconnecting various parts of the computer.
2. To achieve a reasonable speed of operation, a computer must be organized so that all its units can handle one full word of data at a

given time. A group of lines that serve as a connecting port for several devices is called a bus.

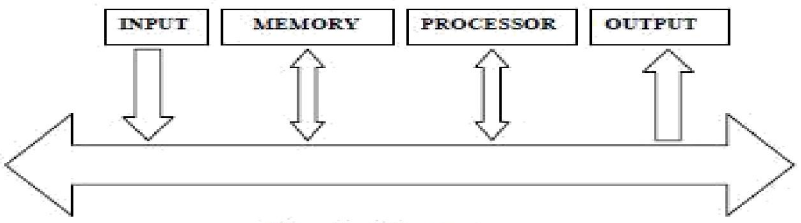


Fig c: Single bus structure

3. In addition to the lines that carry the data, the bus must have lines for address and control purpose.
4. Since the bus can be used for only one transfer at a time, only two units can actively use the bus at any given time. Bus control lines are used to arbitrate multiple requests for use of one bus.
5. Single bus structure is
   - Low cost
   - Very flexible for attaching peripheral devices
6. Multiple bus structure certainly increases the performance but also increases the cost significantly.
7. All the interconnected devices are not of same speed & time, leads to a bit of a problem. This is solved by using cache registers (ie buffer registers). These buffers are electronic registers of small capacity when compared to the main memory but of comparable speed.

The instructions from the processor at once are loaded into these buffers and then the complete transfer of data at a fast rate will take place

Advantages:

1. Low cost
2. Simple to operate

Disadvantages:

1. Difficult to do maintenance operations
2. Failure of bus results in failure of the whole substation
3. Can only be employed where loads can be interrupted or where we have some other supply source

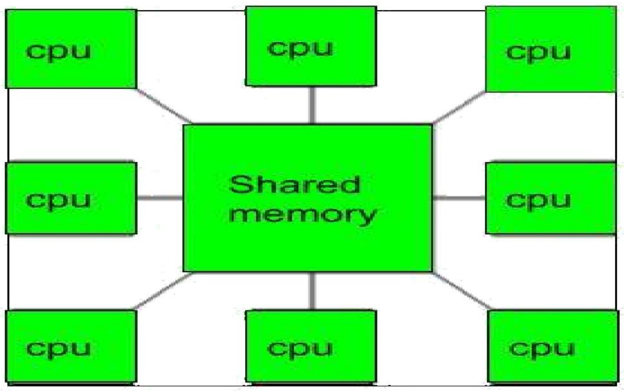
Q5) **Explain the difference between Multiprocessor & Multicomputer**

Ans:- **Multiprocessor & Multicomputer:-**

1. Multiprocessor:

A Multiprocessor is a computer system with two or more central processing units (CPUs) share full access to a common RAM. The main objective of using a multiprocessor is to boost the system's execution speed, with other objectives being fault tolerance and application matching.

There are two types of multiprocessors, one is called shared memory multiprocessor and another is distributed memory multiprocessor. In shared memory multiprocessors, all the CPUs shares the common memory but in a distributed memory multiprocessor, every CPU has its own private memory.



Applications of Multiprocessor –

1. As a uniprocessor, such as single instruction, single data stream (SISD).

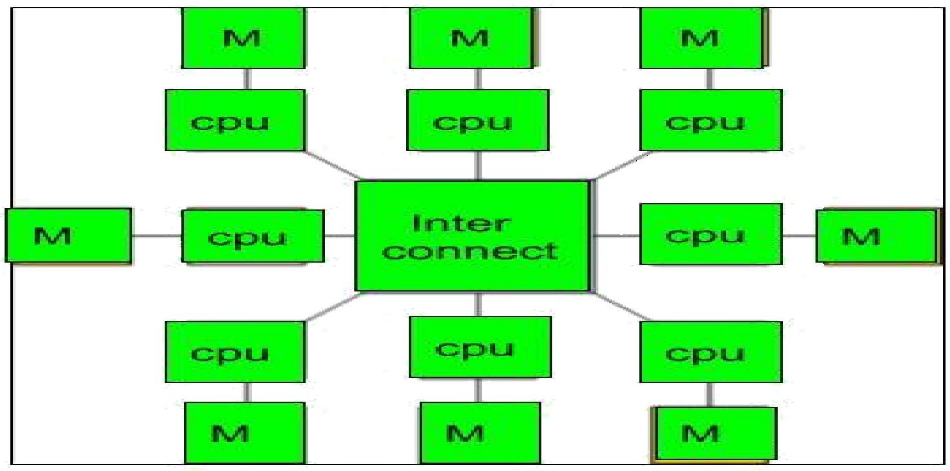
2. As a multiprocessor, such as single instruction, multiple data stream (SIMD), which is
3. usually used for vector processing.
4. Multiple series of instructions in a single perspective, such as multiple instruction, single data stream (MISD), which is used for describing hyper-threading or pipelined processors.
5. Inside a single system for executing multiple, individual series of instructions in multiple perspectives, such as multiple instruction, multiple data stream (MIMD).

Benefits of using a Multiprocessor –

- Enhanced performance.
- Multiple applications.
- Multi-tasking inside an application.
- High throughput and responsiveness.

- Hardware sharing among CPUs.

2. Multicomputer:
A multicomputer system is a computer system with multiple processors that are connected together to solve a problem. Each processor has its own memory and it is accessible by that particular processor and those processors can communicate with each other via an interconnection network.



As the multicomputer is capable of messages passing between the processors, it is possible to divide the task between the processors to complete the task. Hence, a multicomputer can be used for distributed computing. It is cost effective and easier to build a multicomputer than a multiprocessor.

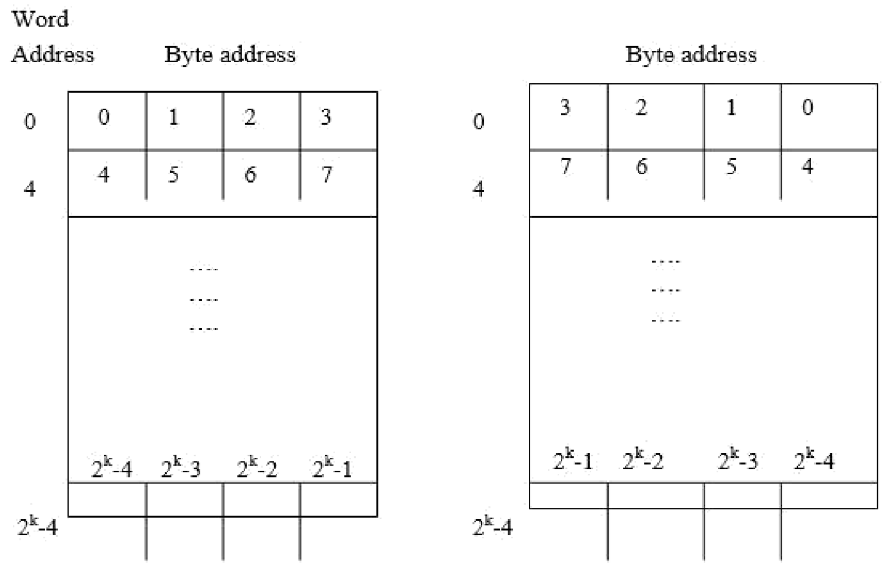
Difference between multiprocessor and Multicomputer:

1. Multiprocessor is a system with two or more central processing units (CPUs) that is capable of performing multiple tasks where as a multicomputer is a system with multiple processors that are attached via an interconnection network to perform a computation task.
2. A multiprocessor system is a single computer that operates with multiple CPUs where as a multicomputer system is a cluster of computers that operate as a singular computer.
3. Construction of multicomputer is easier and cost effective than a multiprocessor.
4. In multiprocessor system, program tends to be easier where as in multicomputer system, program tends to be more difficult.
5. Multiprocessor supports parallel computing, Multicomputer supports distributed computing.

Q6) Explain the difference between big-endian and little-endian assignments

Ans:- **BIG-ENDIAN AND LITTLE-ENDIAN ASIGNMENTS:-**

1. There are two ways that byte addresses can be assigned across words, as shown in fig b.
2. The name big-endian is used when lower byte addresses are used for the more significant bytes (the leftmost bytes) of the word.
3. The name little-endian is used for the opposite ordering, where the lower byte addresses are used for the less significant bytes (the rightmost bytes) of the word.
4. In addition to specifying the address ordering of bytes within a word, it is also necessary to specify the labeling of bits within a byte or a word. The same ordering is also used for labeling bits within a byte, that is, b7, b6… b0, from left to right.



(a) Big-endian assignment (b) Little-endian assignment

Q7) Explain straight-line sequencing in detail. Ans:- INSTRUCTION

EXECUTION AND STRAIGHT-LINE SEQUENCING:-

Instruction Execution: There are 2 phases for executing an instruction. They are,

- Instruction Fetch
- Instruction Execution

Instruction Fetch:

The instruction is fetched from the memory location whose address is in PC. This is then placed in IR.

Instruction Execution:

Instruction in IR is examined and decoded to determine which operation is to be performed.

Program execution Steps:

1. The three instructions of the program are in successive word locations, starting at location i. since each instruction is 4 bytes long, the second and third instructions start at addresses i + 4 and i + 8.

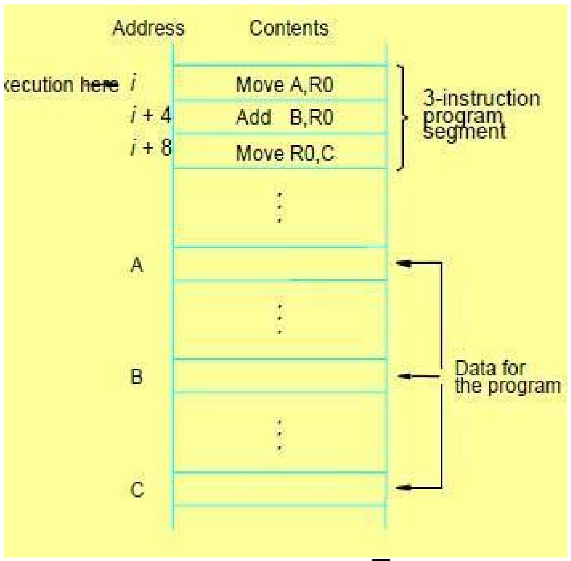


Figure 8:A program for █ [A]+ [B]

2. Let us consider how this program is executed. The processor contains a register called the program counter (PC), which holds the address of the instruction to be executed next. To begin executing a program, the address of its first instruction (I in our example) must be placed into the PC.

3. Then, the processor control circuits use the information in the PC to fetch and execute instructions, one at a time, in the order of increasing addresses. This is called straight-line sequencing.

4. During the execution of each instruction, the PC is incremented by 4 to point to the next instruction. Thus, after the Move instruction at location i + 8 is executed, the PC contains the value i + 12, which is the address of the first instruction of the next program segment.

5. Executing a given instruction is a two-phase procedure. In the first phase, called instruction fetch, the instruction is fetched from the memory location whose address is in the PC.

	6. This instruction is placed in the instruction register (IR) in the processor. The instruction in IR is examined to determine which operation is to be performed. The specified operation is then performed by the processor. This often involves fetching operands from the memory or from processor registers, performing an arithmetic or logic operation, and storing the result in the destination location.	
Q8)	**Explain different Addressing modes with example**	
Ans:-	**Addressing modes:** The term addressing modes refers to the way in which the operand of an instruction is specified. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually executed **Types of Addressing Modes:** 1. **Immediate Addressing Mode –** In immediate addressing mode the source operand is always data. If the data is 8-bit, then the instruction will be of 2 bytes, if the data is of 16-bit then the instruction will be of 3 bytes. **Examples:** MVI B 45 (move the data 45H immediately to register B) LXI H 3050 (load the H-L pair with the operand 3050H immediately) JMP address (jump to the operand address immediately) 2. **Register Addressing Mode –** In register addressing mode, the data to be operated is available inside the register(s) and register(s) is(are) operands. Therefore the operation is performed within various registers of the microprocessor. **Examples:** MOV A, B (move the contents of register B to register A) ADD B (add contents of registers A and B and store the result in register A INR A (increment the contents of register A by one) 3. **Direct Addressing Mode –** In direct addressing mode, the data to be operated is available inside a memory location and that memory location is directly specified as an operand. The operand is directly available in the instruction itself. **Examples:** LDA 2050 (load the contents of memory location into accumulator A) LHLD address (load contents of 16-bit memory location into H-L register pair) IN 35 (read the data from port whose address is 01) 4. **Register Indirect Addressing Mode –**	

IN register indirect addressing mode, the data to be operated is available inside a memory location and that memory location is indirectly specified b a register pair.

Examples:
MOV A, M (move the contents of the memory location pointed by the H-L pair to the accumulator)
LDAX B (move contains of B-C register to the accumulator)
LXIH 9570 (load immediate the H-L pair with the address of the location 9570)

5. **Implied/Implicit Addressing Mode –**
In implied/implicit addressing mode the operand is hidden and the data to be operated is available in the instruction itself.

Examples:
CMA (finds and stores the 1's complement of the contains of accumulator A in A)

RRC (rotate accumulator A right by one bit)
RLC (rotate accumulator A left by one bit)

6. **Auto-increment mode**: Effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are automatically incremented to point to the next consecutive memory location.**(R1)+**.Example:

Add R1, (R2)+ // OR

R1 = R1 +M[R2]

R2 = R2 + d

7. **Auto-decrement mode**: Effective address of the operand is the contents of a register specified in the instruction. Before accessing the operand, the contents of this register are automatically decremented to point to the previous consecutive memory location. –**(R1)**Example:

Add R1,-(R2) //OR

R2 = R2-*d*

R1 = R1 + M[R2].

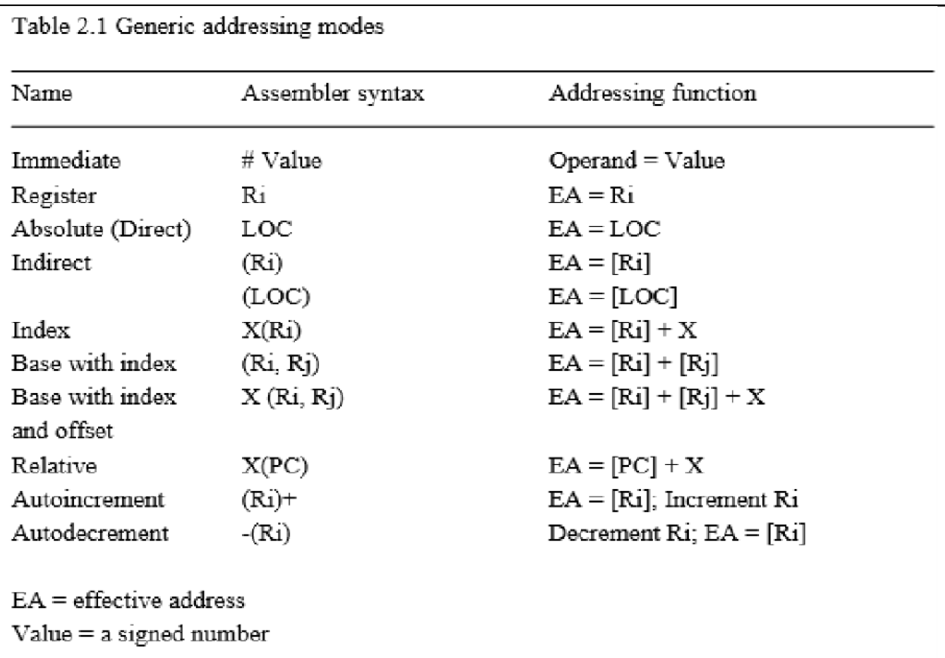


Table 2.1 Generic addressing modes

Name	Assembler syntax	Addressing function
Immediate	# Value	Operand = Value
Register	Ri	EA = Ri
Absolute (Direct)	LOC	EA = LOC
Indirect	(Ri)	EA = [Ri]
	(LOC)	EA = [LOC]
Index	X(Ri)	EA = [Ri] + X
Base with index	(Ri, Rj)	EA = [Ri] + [Rj]
Base with index and offset	X (Ri, Rj)	EA = [Ri] + [Rj] + X
Relative	X(PC)	EA = [PC] + X
Autoincrement	(Ri)+	EA = [Ri]; Increment Ri
Autodecrement	-(Ri)	Decrement Ri; EA = [Ri]

EA = effective address
Value = a signed number

Q9) Difference between Stack and Queue with proper example. **6M**

Stack:

A stack is a list of data elements, usually words or bytes with the accessing restriction that elements can be added or removed at one end of the stack.

End from which elements are added and removed is called the “top” of the stack.

Other end is called the “bottom” of the stack.

Also known as: Pushdown stack. And Last in first out (LIFO) stack.

- *Push* - placing a new item on the stack.
- *Pop* - Removing the top item from the stack.

Data stored in the memory of a computer can be organized as a stack.

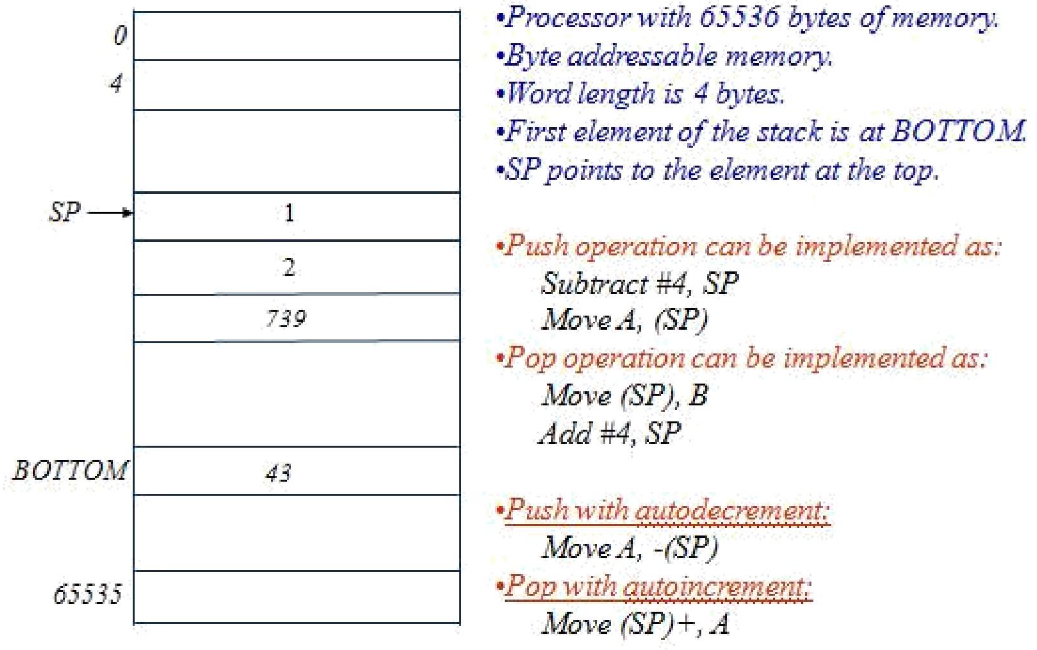
Successive elements occupy successive memory locations.

When new elements are pushed on to the stack they are placed in successively lower address locations.

Stack grows in direction of decreasing memory addresses.

A processor register called as "Stack Pointer (SP)" is used to keep track of the address of the element that is at the top at any given time.

A general purpose register could serve as a stack pointer



0
4
SP
1
2
739
BOTTOM
43
65535
•Processor with 65536 bytes of memory.
•Byte addressable memory.
•Word length is 4 bytes.
•First element of the stack is at BOTTOM.
•SP points to the element at the top.
•Push operation can be implemented as:
Subtract #4, SP
Move A, (SP)
•Pop operation can be implemented as:
Move (SP), B
Add #4, SP
•Push with autodecrement:
Move A, -(SP)
•Pop with autoincrement:
Move (SP)+, A

Sr.No	STACK	QUEUE
1	Objects are inserted and removed at the same end.	Objects are inserted and removed from different ends.
2	In stacks only one pointer is used. It points to the top of the stack.	In queues, two different pointers are used for front and rear ends.
3	In stacks, the last inserted object is first to come out.	In queues, the object inserted first is first deleted.
4	Stacks follow Last In First Out (LIFO) order.	Queues following First In First Out (FIFO) order.
5	Stack operations are called push and pop.	Queue operations are called enqueue and dequeue.
6	Stacks are visualized as vertical collections.	Queues are visualized as horizontal collections.
7	Collection of dinner plates at a wedding reception is an example of stack.	People standing in a file to board a bus is an example of queue.

Q10) **Explain Subroutine linkage & parameter passing method with example.** **7M**

Ans:-

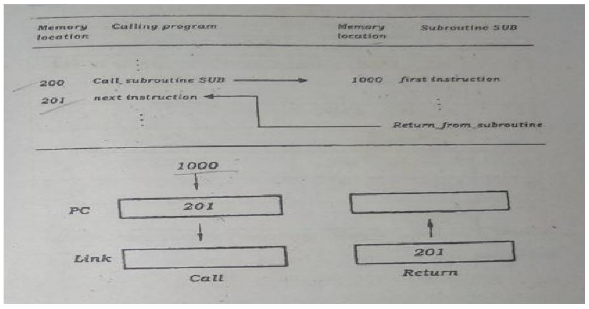
Subroutine Linkage:

- A subroutine is a self-contained sequence of instructions that performs a given computational task.

1) A repeated task is implemented as a subroutine.
2) To save space, only one copy of the instruction that constitute the subroutine is placed in the **main** memory and any program that required the use of the subroutine simply branches to its starting location.
3) When the program branches to a subroutine we say that it is calling the subroutine.

4) The instruction that perform this branch operation is called a call subroutine instruction.
5) Since the subroutine may be call from different places in a calling program. Provision must be made for returning to the appropriate location.
6) The content of the PC must be saved by the call subroutine instruction to enable correct return to the calling program.
7) The way in which computer makes it possible to call and return from subroutine is referred to as its subroutine linkage method.
8) The simplest subroutine linkage method in which the return address is saved in specific location such a register, is called link register call subroutine is a special branch instruction that perform following task.

1. Save the contentment of Pc in link register.
2. Then jump to the address specified by the instruction.



Stack is most efficient data structure to store return address. The most efficient way is to store the return address in a memory stack.

The advantage of using a stack for the return address is that when a succession of subroutines is called, the sequential return addresses can be pushed into the stack.

The return from subroutine instruction causes the stack to pop and the contents of the top of the stack are transferred to the program counter.

- A subroutine call is implemented with the following micro operations:
- If another subroutine is called by the current subroutine, the new return address is pushed into The stack and so on.
- The instruction that returns from the last subroutine is implemented by the Micro operations:

PC←M [SP] Pop stack and transfer to PC SP ←SP + 1 Increment stack pointer

Parameter passing:

A parameter is passed to the subroutine by leaving the data in a register, or

memory, and allowing the subroutine to use it.
A parameter is passed back to the main program by allowing the subroutine to change the data.
This is the way parameters are passed in assembly language.
When the parameter being passed to the subroutine is in a register, this is referred to as the call-by-value technique of passing parameters.
If the data is in memory and the address is passed to the subroutine, this is called call-by-reference.
It is important to document all parameter-passing details in subroutines

Q11) **Explain 3-address, 2- address, 1-address and zero- address instruction format with example.**

Ans:- **THREE-ADDRESS INSTRUCTIONS:-**

Computers with three-address instruction formats can use each address field to specify either a processor register or a memory operand. The program in assembly language that evaluates X = (A + B) * (C + D) is shown below, together with comments that explain the register transfer operation of each instruction.

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

It is assumed that the computer has two processor registers, R1 and R2. The symbol M [A] denotes the operand at memory address symbolized by A.
The advantage of the three-address format is that it results in short programs when evaluating arithmetic expressions. The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

TWO-ADDRESS INSTRUCTIONS

Two address instructions are the most common in commercial computers. Here again each address field can specify either a processor register or a memory word. The program to evaluate X = (A + B) * (C + D) is as follows

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

The MOV instruction moves or transfers the operands to and from memory and processor registers. The first symbol listed in an instruction is assumed to be both a source and the destination where the result of the operation is transferred.

ONE-ADDRESS INSTRUCTIONS:

One-address instructions use an implied accumulator (AC) register for all data manipulation. For multiplication and division there is a need for a second register. However, here we will neglect the second and assume that the AC contains the result of tall operations. The program to evaluate X = (A + B) * (C + D) is

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

ZERO-ADDRESS INSTRUCTIONS:-

A stack-organized computer does not use an address field for the instructions ADD and MUL. The PUSH and POP instructions, however, need an address field to specify the operand that communicates with the stack. The following program shows how X = (A + B) * (C + D) will be written for a stack organized computer. (TOS stands for top of stack)

To evaluate arithmetic expressions in a stack computer, it is necessary to convert the expression into reverse Polish notation. The name "zero-address" is given to this type of computer because of the absence of an address field in the computational instructions

```
PUSH    A    TOS ← A
PUSH    B    TOS ← B
ADD          TOS ← (A + B)
PUSH    C    TOS ← C
PUSH    D    TOS ← D
ADD          TOS ← (C + D)
MUL          TOS ← (C + D) * (A + B)
POP     X    M [X] ← TOS
```

Q12) Explain assembly language and execution of programs in detail.

ASSEMBLY LANGUAGE

1. Machine instructions are represented by patterns of 0s and 1s. Such patterns are awkward to deal with when discussing or preparing programs.

2. Therefore, we use symbolic names to represent the pattern. So far, we have used normal words, such as Move, Add, Increment, and Branch, for the instruction operations to represent the corresponding binary code patterns.

3. When writing programs for a specific computer, such words are normally replaced by acronyms called mnemonics, such as MOV, ADD, INC, and BR. Similarly, we use the notation R3 to refer to register 3, and LOC to refer to a memory location.
4. A complete set of such symbolic names and rules for their use constitute a programming language, generally referred to as an assembly language.

5. Programs written in an assembly language can be automatically translated into a sequence of machine instructions by a program called an assembler.

6. When the assembler program is executed, it reads the user program, analyzes it, and then generates the desired machine language program.
7. The latter contains patterns of 0s and 1s specifying instructions that will be executed by the computer.
8. The user program in its original alphanumeric text format is called a source program, and the assembled machine language program is called an object program.

ASSEMBLER DIRECTIVES:-

1. In addition to providing a mechanism for representing instructions in a program, the assembly language allows the programmer to specify other information needed to translate the source program into the object program.

2. We have already mentioned that we need to assign numerical values to any names used in a program. Suppose that the name SUM is used to represent the value 200. This fact may be conveyed to the assembler program through a statement such as

   SUM EQU 200

3. This statement does not denote an instruction that will be executed when the object program is run; in fact, it will not even appear in the

object program.

4. It simply informs the assembler that the name SUM should be replaced by the value 200 wherever it appears in the program. Such statements, called assembler directives (or commands), are used by the assembler while it translates a source program into an object program.

ASSEMBLY AND EXECUTION OF PRGRAMS:-

1. A source program written in an assembly language must be assembled into a machine language object program before it can be executed. This is done by the assembler program, which replaces all symbols denoting operations and addressing modes with the binary codes used in machine instructions, and replaces all names and labels with their actual values.

2. The assembler assigns addresses to instructions and data blocks, starting at the address given in the ORIGIN assembler directives. It also inserts constants that may be given in DATAWORD commands and reserves memory space as requested by RESERVE commands.

3. As the assembler scans through a source programs, it keeps track of all names and the numerical values that correspond to them in a symbol table.

4. Thus, when a name appears a second time, it is replaced with its value from the table. A problem arises when a name appears as an operand before it is given a value. For example, this happens if a forward branch is required.

5. A simple solution to this problem is to have the assembler scan through the source program twice. During the first pass, it creates a complete symbol table. At the end of this pass, all names will have been assigned numerical values.

6. The assembler then goes through the source program a second time and substitutes values for all names from the symbol table. Such an assembler is called a two-pass assembler.

7. The assembler stores the object program on a magnetic disk. The object program must be loaded into the memory of the computer before it is executed. For this to happen, another utility program called a loader must already be in the memory.

8. When the object program begins executing, it proceeds to completion unless there are logical errors in the program. The user must be able to find errors easily.

9. The assembler can detect and report syntax errors. To help the user find other programming errors, the system software usually includes a

debugger program.

10. This program enables the user to stop execution of the object program at some points of interest and to examine the contents of various processor registers and memory locations.

Q13) Write a short note on

1. Memory locations and addresses 2. Byte Addressability

Memory locations and addresses

1. Number and character operands, as well as instructions, are stored in the memory of a computer. The memory consists of many millions of storage cells, each of which can store a bit of information having the value 0 or 1.

2. Because a single bit represents a very small amount of information, bits are seldom handled individually. The usual approach is to deal with them in groups of fixed size.

3. For this purpose, the memory is organized so that a group of n bits can be stored or retrieved in a single, basic operation. Each group of n bits is referred to as a word of information, and n is called the word length. The memory of a computer can be schematically represented as a collection of words as shown in figure (a).

4. Modern computers have word lengths that typically range from 16 to 64 bits.

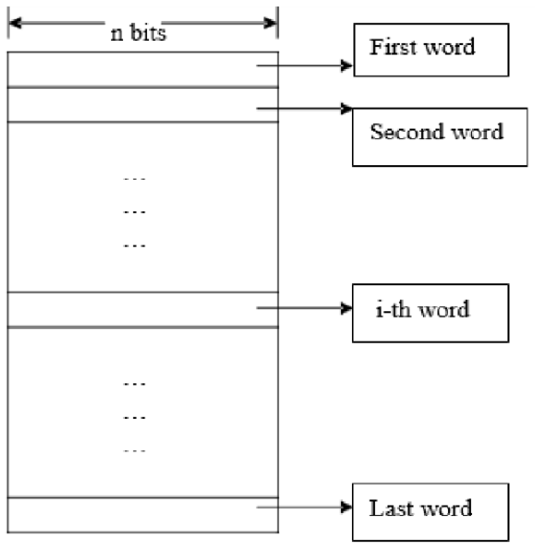
5. If the word length of a computer is 32 bits, a single word can store a 32-bit 2's complement number or four ASCII characters, each occupying 8 bits. A unit of 8 bits is called a byte.

6. Accessing the memory to store or retrieve a single item of information, either a word or a byte, requires distinct names or addresses for each item location. It is customary to use numbers from 0 through 2K-1, for some suitable values of k, as the addresses of successive locations in the memory.

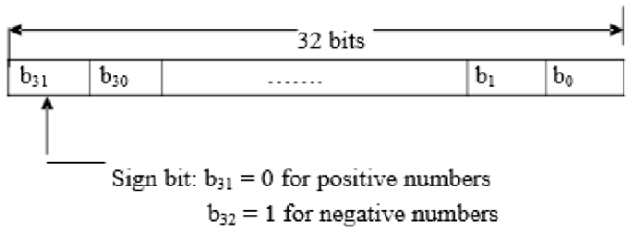
7. The 2k addresses constitute the address space of the computer, and the memory can have up to 2k addressable locations. 24-bit address generates an address space of 224 (16,777,216) locations. A 32-bit address creates an address space of 232 or 4G (4 giga) locations.

BYTE ADDRESSABILITY:-

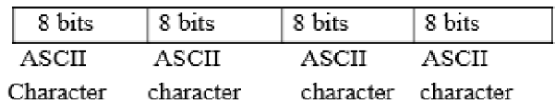
1. We now have three basic information quantities to deal with: the bit, byte and word. A byte is always 8 bits, but the word length typically ranges from 16 to 64 bits. The most practical assignment is to have successive addresses refer to successive byte



(a) A signed integer


32 bits
b_{31}
b_{30}
b_1
b_0
Sign bit: $b_{31} = 0$ for positive numbers
$b_{32} = 1$ for negative numbers

(b) Four characters



2. Locations in the memory. This is the assignment used in most modern computers, and is the one we will normally use in this book. The term byte-addressable memory is use for this assignment.

Byte locations have addresses 0, 1, 2… Thus, if the word length of the machine is 32 bits, successive words are located at addresses 0,4,8,…., with
each word consisting of four bytes.

What are types of software and give application of software

Software: Software, is a collection of data or computer instructions that tell the computer how to work.

Types of software:- computer software can be divided into:

Application software

which is software that uses the computer system to perform special functions or provide entertainment functions beyond the basic operation of the computer itself. There are many different types of application software, because the range of tasks that can be performed with a modern computer is so large—see list of software.

System software

which is software that directly operates the computer hardware, to provide basic functionality needed by users and other software, and to provide a platform for running application software.[5] System software includes:

Operating systems

which are essential collections of software that manage resources and provides common services for other software that runs "on top" of them. Supervisory programs, boot loaders, shells and window systems are core parts of operating systems. In practice, an operating system comes bundled with additional software (including application software) so that a user can potentially do some work with a computer that only has one operating system.

Application of software:

1) Scientific Research.
2) Business application.
3) Banking.
4) Education.
5) Communication.
6) Medicine.
7) Space Technology.

Unit – II

Syllabus

Instruction Sets: Instruction Format, limitations of Short word- length machines, High level language Considerations, Motorola 68000 architecture. Processing Unit: Some fundamental concepts, Execution of a complete instruction, Single, two, three bus organization, Sequencing of control Signals.

Q1). Explain the instruction set of 68000. (7)

Ans:

1. The 6800 has 72 documented instruction commands and a number of additional undocumented instructions.
2. When all the available valid addressing modes are considered there are a total of 197 valid op-codes for the 6800. Motorola refers to the MS byte of the instruction as the op-code.
3. In general instruction execution time is in the range of 2-12 microseconds, depending on instruction and addressing mode used. The 6800 uses internal execution stages to complete instructions in less cycle.
4. The instruction can be 8, 16, or 24 bits in length depending on the op-code and type of addressing used/needed. There are no I/O instructions in the 6800 as all I/O devices are mapped to the memory and so reside in memory space.\
5. The instructions can be grouped into several areas:

- **Memory reference instructions**: These are used for data transfers between either of the two GP registers and the memory. Instructions between the general purpose registers and memory are used for arithmetic and logical instructions. There are some single address instructions which are used to operate on data in memory only.

- **Arithmetic and logic operations:-**These instructions can be carried out between memory and a general purpose register, or between two general purpose registers. The arithmetic operations supported are added, subtract, increment and decrement. Logical operations which are supported are AND, OR, NOT and XOR.

- **Jumps**

  Instructions are provided for fourteen conditional jumps and two unconditional jumps. These are single address memory reference instructions, almost all use relative addressing i.e. the jump is to an address given relative to a pointer, usually the index pointer.

  - **Subroutine Entry and Exit**

  1. The subroutine entry instruction is a single address, memory reference instruction. When used the subroutine entry instruction causes the content of the program counter to be pushed onto the stack.

  2. On return from the sub-routine the program counter is popped from the stack back to the SP register.

  3. As the stack is held in memory it can be as large as the memory itself. Then using the STS (store stack pointer) and LDS (load stack pointer) instructions a program can have multiple stacks.

  4. As the stack is used to store the program counter during sub routine calls, this allows almost unlimited subroutine nesting. Push and Pull instructions are provided for both A and B accumulators.

Register-Register instructions

1. There are a limited number of instructions that operate on the contents of the general purpose registers. These allow for movement of the contents of the stack pointer and index pointer registers, and incrementing and decrementing of these as well.

Shift instructions

1. Right, left and circular shifts are provided. These are of one position only.

Flag instructions

1. There a series of instructions provided to set or clear flags. Flag values can be moved to the GP registers for program analysis or alteration. The branch instructions can test four of the flags; carry, sign, overflow and zero.

Other Instructions

The 6800 also has a 'no-operation' instruction which advances the program counter and a wait instruction which works by extending the length of the clock pulses.

Q2) Explain condition code flag of 68000. (3)

Ans:

This final register contains six flags which are set or cleared in response to how the program executes. These flags are:

- C - Carry, for arithmetic operations which result in a carry.
- V - Overflow, set to 1 when a 2's complement overflow results from an arithmetic operation.
- Z - Zero, set to 1 if result of an operation is 0, otherwise is set to 0.
- N - Negative, is set to indicate a negative number
- I - Interrupt mask, when this bit is set then interrupts are inhibited. Otherwise set to 0 and the processor may be interrupted by IRQ (the Interrupt Request Pin) being in a low state.
- H - Half carry auxiliary flag, set when there is a carry from bit 3 to 4 in some of the arithmetic operations.

- The two remaining bits 7 and 8 are permanently set to 1.

Q3) What is meant by system buses? Explain different bus architecture in computer system.

Ans:

1. The simplest and most common way of interconnecting various parts of the computer.

2. To achieve a reasonable speed of operation, a computer must be organized so that all its units can handle one full word of data at a given time.

3. A group of lines that serve as a connecting port for several devices is called a bus.

4. In addition to the lines that carry the data, the bus must have lines for address and control purpose.

5. Simplest way to interconnect is to use the single bus as shown Since the bus can be used for only one transfer at a time, only two units can actively use the bus at any given time. Bus control lines are used to arbitrate multiple requests for use of one bus.

6. Single bus structure is Low cost Very flexible for attaching peripheral devices.

7. Multiple bus structure certainly increases the performance but also increases the cost significantly.

8. All the interconnected devices are not of same speed & time leads to a bit of a problem. This is solved by using cache registers (i.e. buffer registers). These buffers are electronic registers of small capacity when compared to the main memory but of comparable speed.

9. The instructions from the processor at once are loaded into these buffers and then the complete transfer of data at a fast rate will take place.

Q4). Explain Straight line sequencing in detail

Ans:

1. In the preceding discussion of instruction formats, we used to task $C \leftarrow [A] + [B]$. Shows a possible program segment for this task as it appears in the memory of a computer.

2. We have assumed that the computer allows one memory operand per instruction and has a number of processor registers.

3. The three instructions of the program are in successive word locations, starting at location i. since each instruction is 4 bytes long, the second and third instructions start at addresses i + 4 and i + 8.

4. Let us consider how this program is executed. The processor contains a register called the program counter (PC), which holds the address of the instruction to be executed next.

5. To begin executing a program, the address of its first instruction (I in our example) must be placed into the PC. Then, the processor control circuits use the information in the PC to fetch and execute instructions, one at a time, in the order of increasing addresses. This is called straight-line sequencing.

6. During the execution of each instruction, the PC is incremented by 4 to point to the next instruction. Thus, after the Move instruction at location i + 8 is executed, the PC contains the value i + 12, which is the address of the first instruction of the next program segment.

7. Executing a given instruction is a two-phase procedure. In the first phase, called instruction fetch, the instruction is fetched from the memory location whose address is in the PC.

8. This instruction is placed in the instruction register (IR) in the processor. The instruction in IR is examined to determine which operation is to be performed. The specified operation is then performed by the processor.

9. This often involves fetching operands from the memory or from processor registers, performing an arithmetic or logic operation, and storing the result in the destination location.

Q5). Draw and explain register structure of 68000.

Ans:

Register Set and Programmers Model

1. The 6800 has six internaly accessible registers. These are two 8-bit accumulators or general purpose register (A and B), three 16-bit registers PC, SP, and Index register - X) and an 8-bit condition code or status register which has 6 flags in total. These can be viewed as follows:

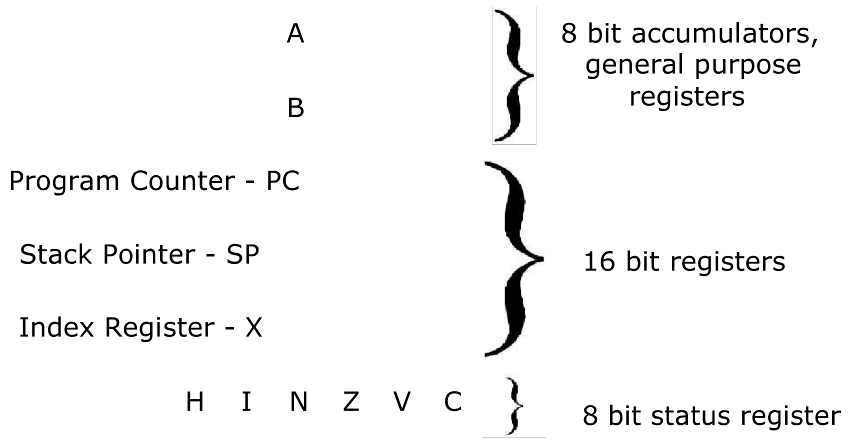
The function of the registers is as follows.

- **The Accumulators A and B**:Each stores and manipulates one 8-bit word under program control.

- **The Index register - X** is a 2-byte register. It holds memory addresses when using indexed-addressing mode instructions.

- **The Program Counter - PC**Is a 2-byte register which contains the address of the next byte of the instruction to be fetched from memory (instructions can be from one to 3 bytes in length). When the current value of the program counter is placed on the address bus, the PC is updated to the value of the next instruction for execution.

- **Stack pointer - SP**A 2-byte register which holds the starting address of sequential memory locations in RAM where the contents of the CPU registers may be stored and retrieved. The 6800 uses RAM for its stack, this has some advantages that are outlined in the section of the 6800



Instruction set.

Status Register or Condition Codes Register

This final register contains six flags which are set or cleared in response to how the program executes. These flags are:

- C - Carry, for arithmetic operations which result in a carry.
- V - Overflow, set to 1 when a 2's complement overflow results from an arithmetic operation.
- Z - Zero, set to 1 if result of an operation is 0, otherwise is set to 0.
- N - Negative, is set to indicate a negative number
- I - Interrupt mask, when this bit is set then interrupts are inhibited. Otherwise set to 0 and the processor may be interrupted by IRQ (the Interrupt Request Pin) being in a low state.
- H - Half carry auxiliary flag, set when there is a carry from bit 3 to 4 in some of the arithmetic operations.
- The two remaining bits 7 and 8 are permanently set to 1.

Q6). Explain branching instruction in 68000 with respect to program flow control.

Ans

1. The 6800 has 72 documented instruction commands and a number of additional undocumented instructions.
2. When all the available valid addressing modes are considered there are a total of 197 valid op-codes for the 6800.
3. Motorola refers to the MS byte of the instruction as the op-code. In general instruction execution time is in the range of 2-12 microseconds, depending on instruction and addressing mode used. The 6800 uses internal execution stages to complete instructions in less cycle.
4. The instruction can be 8, 16, or 24 bits in length depending on the op-code and type of addressing used/needed.
5. There are no I/O instructions in the 6800 as all I/O devices are mapped to the memory and so reside in memory space.

Q. Explain storing a word in memory:

1. In particular, when we fetch the operands (i.e., the registers) we want to send the source and destination registers bits to a device called the **register file**.]
2. For example, if **IR_{25-21}** has value 00111, this means we want register **$r7** from the register file. We sent in 00111 to this circuit, and it returns the contents back to us.
3. If we are executing an I-type instruction, then typically, we'll sign-extend (or zero-extend, depending on the instruction) the immediate part (i.e., **IR_{15-0}**) to 32 bits.
4. Fetching a word from memory:

   At this point, the output of the ALU is written back to the register file. For example, if the instruction was: **add $r2, $r3, $r4** then the *result* of adding the contents of **$r3** to the contents of **$r4** would be stored back into **$r2**.

   The result could also be due to a **load** from memory.

   Some instructions don't have results to store. For example, branch and jump instructions do not have any results to store.

Q7). Define and Explain 4

1. Control memory.

Ans: The 6800 has six internaly accessible registers. These are two 8-bit accumulators or general purpose register (A and B), three 16-bit registers PC, SP, and Index register - X) and an 8-bit condition code or status register which has 6 flags in total. These can be viewed as follows:

The function of the registers is as follows.

The Accumulators A and B: Each stores and manipulates one 8-bit word under program control.

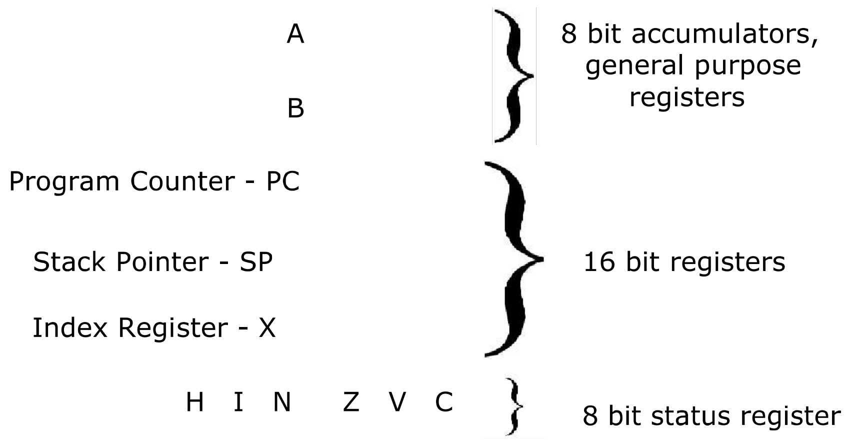
The Index register - X

It is a 2-byte register. It holds memory addresses when using indexed-addressing mode instructions.

The Program Counter - PC Is a 2-byte register which contains the address of the next byte of the instruction to be fetched from memory (instructions can be from one to 3 bytes in length). When the current value of the program counter is placed on the address bus, the PC is updated to the value of the next instruction for execution.

Stack pointer - SP

A 2-byte register which holds the starting address of sequential memory locations in RAM where the contents of the CPU registers may be stored and retrieved. The 6800 uses RAM for its stack, this has some advantages that are outlined in the



section of the 6800 Instruction set.

2. Status Flag:

Ans: Status Register or Condition Codes Register

This final register contains six flags which are set or cleared in response to how the program executes. These flags are:

- C - Carry, for arithmetic operations which result in a carry.
- V - Overflow, set to 1 when a 2's complement overflow results from an arithmetic operation.
- Z - Zero, set to 1 if result of an operation is 0, otherwise is set to 0.
- N - Negative, is set to indicate a negative number
- I - Interrupt mask, when this bit is set then interrupts are inhibited. Otherwise set to 0 and the processor may be interrupted by IRQ (the Interrupt Request Pin) being in a low state.
- H - Half carry auxiliary flag, set when there is a carry from bit 3 to 4 in some of the arithmetic operations.
- The two remaining bits 7 and 8 are permanently set to 1.

Or

Q8). Explain the basic organization of power PC

Ans: PowerPC is a microprocessor architecture that was developed jointly by Apple, IBM, and Motorola. The PowerPC employs reduced instruction-set computing (RISC). The three developing companies have made the PowerPC architecture an open standard, inviting other companies to build on it.
Developed at IBM, reduced instruction-set computing (RISC) is based on studies showing that the simplest computer instructions are the ones most frequently performed. Traditionally, processors have been designed to accommodate the more complex instructions as well. RISC performs the more complex instructions using combinations of simple instructions. The timing for the processor can then be based on simpler and faster operations, enabling the microprocessor to perform more instructions for a given clock speed.
The PowerPC architecture provides an alternative to the popular processor architectures from Intel, including the Pentium. (Microsoft builds its Windows operating system offerings to run on Intel processors, and this widely-sold combination is sometimes called "Wintel".) The PowerPC was first used in IBM's RS/6000 workstation with its UNIX-based operating system, AIX, and in Apple Computer's Macintosh personal computers. Today, PowerPC chips are also used in diverse applications including internetworking equipment, routers, telecom switches, interactive multimedia, automotive control, and industrial robotics.

Q9). Explain the role of multiple condition register in power PC

Ans: PowerPC Registers:

PowerPC's application-level registers are broken into three categories:

General purpose, floating point and special purpose registers.

- **General-purpose registers (GPRs) - r0 to r31**
  - Flat-scheme of 32 general purpose registers.
  - Source and destination for all integer operations
  - Address source for all load/store operations.
  - They also provide access to SPRs.
  - All GPRs are available for use with one exception: in certain instructions, GPR0 simply means the value 0, and no lookup is done for GPR0's contents.
- Some of these registers have special tasks assigned to them:
  - r0 Volatile register which may be modified during function linkage
  - r1 Stack frame pointer, always valid
  - r2 System-reserved register
  - r3-r4 Volatile registers used for parameter passing and return values
  - r5-r10 Volatile registers used for parameter passing
  - r11-r12 Volatile registers which may be modified during function linkage
  - r13 Small data area pointer register
  - r14-r30 Registers used for local variables
  - r31 Used for local variables or "environment pointers“

Floating point registers

- **Floating-point registers (FPRs)- fr0 to fr31**
  - 32 floating-point registers with 64-bit precision.
  - source and destination operands of all floating-point operations
  - can contain 32-bit and 64-bit signed and unsigned integer values, as well as single-precision and double-precision floating-point values.
  - FPR’s also provide access to the FPSCR(Floating-Point Status and Control Register)
  - FPSCR captures status and exceptions resulting from floating-point operations, and also provides control bits for enabling specific exception types.
  - Instructions to load and store double precision floating point numbers transfers 64-bit of data without conversion.
  - Instructions to load from memory single precision floating point numbers convert to double precision format before storing them in the register.
  - f0 Volatile register
  - f1 Volatile register used for parameter passing and return values
  - f2-f8 Volatile registers used for parameter passing

- f9-f13 Volatile registers
- f14-f31 Registers used for local variables

Special-purpose registers (SPRs)

- **The Fixed-Point Exception Register (XER)**- used for indicating conditions for integer operations, such as carries and overflows.
- **The Floating-Point Status and Control Register (FPSCR)-** 32-bit register used to store the status and control of the floating-point operations.
- **The Count Register (CTR)-** used to hold a loop count that can be decremented during the execution of branch instructions.
- **The Condition Register (CR)**-32-bit register grouped into eight fields, where each field is 4 bits that signify the result of an instruction's operation: Equal (EQ), Greater Than (GT), Less Than (LT), and Summary Overflow (SO).
- **The Link Register (LR) contains** the address to return to at the end of a function call.

Q10). Compare hardwired and micro-controlled control unit

Ans: **Hardwird Control**

The control units use fixed logic circuits to interpret instructions and generate control signals from them. The fixed logic circuit block includes combinational circuit that generates the required control outputs for decoding and encoding functions.

Every instruction in a processor is implemented by a sequence of one or more sets of concurrent micro operations. Each micro operation is associated with a specific set of control lines which, when activated, causes that micro operation to take place. Since the number of instructions and control lines is often in the hundreds, the complexity of hardwired control unit is very high. Thus, it is costly and difficult to design. The hardwired control unit is relatively inflexible because it is difficult to change the design, if one wishes to correct design error or modify the instruction set
Microprogramming is a method of control unit design in which the control signal memory CM.
The control signals to be activated at any time are specified by a microinstruction, which is fetched from CM.
A sequence of one or more micro operations designed to control specific operation, such as addition, multiplication is called a micro program.

The address where these microinstructions are stored in CM is generated by micro program sequencer/micro program controller.
The micro program sequencer generates the address for microinstruction according to the instruction stored in the IR.
The micro programmed control unit,

- control memory
- control address register
- Micro instruction register
- Micro program sequencer

Q11). Explain micro programmed control unit

Ans: Every instruction in a processor is implemented by a sequence of one or more sets of concurrent micro operations. Each micro operation is associated with a specific set of control lines which, when activated, causes that micro operation to take place. Since the number of instructions and control lines is often in the hundreds, the complexity of hardwired control unit is very high. Thus, it is costly and difficult to design. The hardwired control unit is relatively inflexible because it is difficult to change the design, if one wishes to correct design error or modify the instruction set

Microprogramming is a method of control unit design in which the control signals memory CM.

The control signals to be activated at any time are specified by a microinstruction, which is fetched from CM.

A sequence of one or more micro operations designed to control specific operation, such as addition, multiplication is called a micro program.

The address where these microinstructions are stored in CM is generated by micro program sequencer/micro program controller.
The micro program sequencer generates the address for microinstruction according to the instruction stored in the IR.
The micro programmed control unit,
- control memory
- control address register
- Micro instruction register
- Micro program sequencer