UNIT – 1

STRUCTURE OF DESKTOP COMPUTERS

The desktop computers are the computers which are usually found on a home or office desk. They consist of processing unit, storage unit, visual display and audio as output units, and keyboard and mouse as input units. Figure 1 shows these five functional

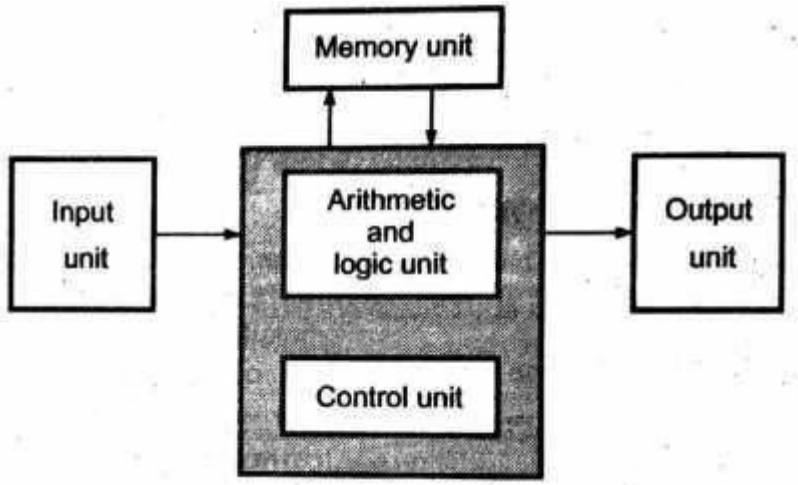

Figure 1. Units of a computer.

Input Unit - The input unit accepts the digital information from user with the help of input devices such as keyboard, mouse, microphone etc. The information received from the input unit is either stored in the memory for later use or immediately used by the arithmetic and logic unit to perform the desired operations.

Memory Unit - The memory unit is used to store programs and data. Usually, two types of memory devices are used to form a memory unit: primary storage memory device and secondary storage memory device. The primary memory, commonly called main memory is a fast memory used for the storage of programs and active data. These memories are fast but they are small in capacities and expensive. Therefore, the computer uses the secondary storage memories such as magnetic tapes, magnetic disks for the storage of large amount of data.

Arithmetic and Logic Unit - The Arithmetic and Logic Unit (ALU) is responsible for performing arithmetic operations such as add, subtract, division and multiplication and logical operations such as AND, OR, Inverting etc.

Output Unit - The output unit sends the processed results to the user using output devices such as video monitor, printer, plotter, etc. The video monitors display the output on the CRT screen whereas printers and plotters give the hard- copy output.

Control Unit - The control unit co—ordinates and controls the- activities amongst the functional. The basic function, of control unit is to fetch the instructions stored in the main memory, identify the operations and the devices involved in it and accordingly generate control signals to execute the desired operations.

CPU (CENTRAL PROCESSING UNIT)

The CPU is the brain of the Computer system. It works as an administrator of a system. All the operations within the system are supervised and controlled by CPU. It interprets and co-ordinates the instructions. The data and instructions are temporarily stored in its memory unit. After performing Operation, the result of operation can be stored in this memory unit.

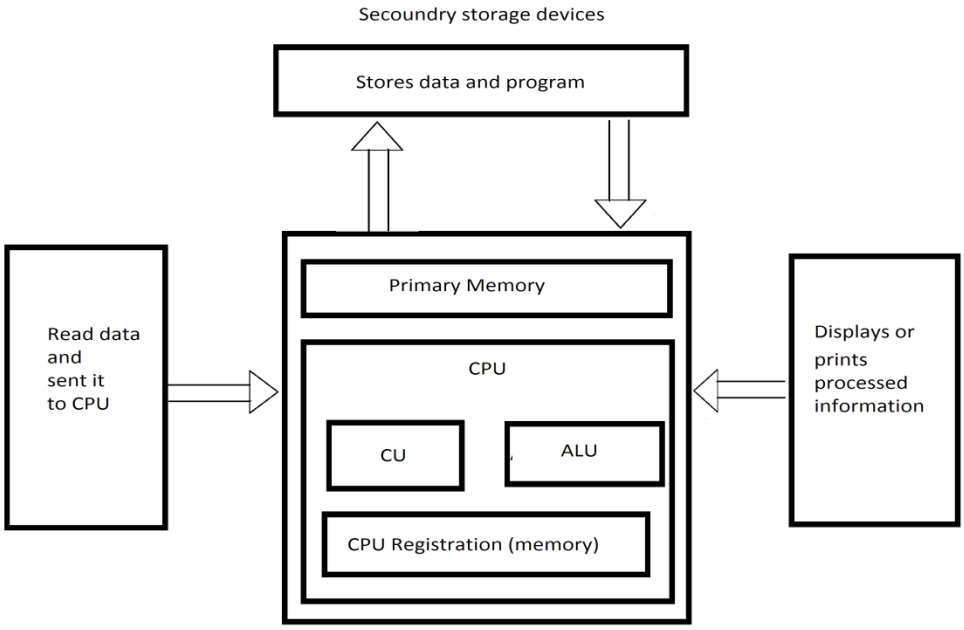

Figure 2: CPU and its interaction with other units

The results of operation are sent towards output unit for the user. Thus, CPU controls all internal and external devices, performs arithmetic and logical operations, and controls the memory usage and control the sequence of operations. For performing all these operations, the CPU has three subunits.

- Arithmetic and Logic Unit (ALU)
- Control Unit
- Memory (CPU registers)

GENERAL REGISTER ORGANIZATION:

CPU Registers - Register is a group of flip-flops which can be used to store a word. It is a high speed temporary storage space for holding data, addresses and instructions during processing the instruction. Registers are not referenced by their addresses; they are directly accessed. To perform execution of instruction, the processor contains a number of registers used for temporary storage of data and some special function registers.

The special purpose registers include Program Counter (PC), Instruction Register (IR), Memory Address Register (MAR) and Memory Data Register (MDR).

Program Counter (PC):- It is used to store the address of next instruction to be executed.

Instruction Register (IR):- It is used to hold the instruction that is currently being executed. The contents of IR are available to the control unit, which generate the timing signals that control the various processing elements involved in executing the instruction.

Memory Address Register [MAR] and Memory Data Register (MDR): - These registers are used to handle the data transfer between the main memory and the processor.

Memory Address Register [MAR] :-The MAR holds the address of the main memory to or from which data is to be transferred.

Memory Data Register [MDR] :-The MDR sometimes also called MBR (Memory Buffer Register) contains the data to be written into or read from the addressed word of the main memory.

Accumulator (AC):- It holds the result generated by ALU.

General purpose registers - These are used to hold the operands for arithmetic and logic operations and/or used to store the result of the operation. Since the access time of these registers is lowest, these are used to store frequently used data.

Figure 3 shows the general Register organization for seven CPU registers. It shows that how registers are selected and how data flow between register and ALU take place. Decoder is used to select a particular

register. The output of each register is connected to two multiplexers (MUX) to form the two buses A and B. The selection lines in each multiplexer select the input data for the particular bus. The A and B buses form the two inputs of an Arithmetic Logic Unit (ALU). The operation select lines decide the micro operation to be performed by ALU. The result of the micro- operation is available at the output bus. The output bus connected to the inputs of all registers, thus by selecting a destination register it is possible to store the result in it.

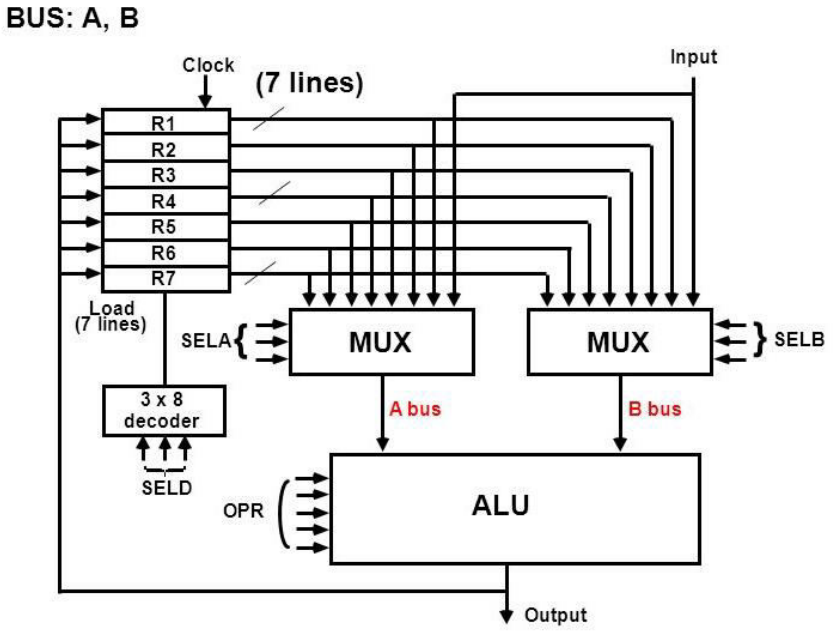

Figure 3: General Register organization for CPU register

CONTROL WORD

The combined value of a binary selection inputs specifies the control word. There are 14 binary selection inputs in the unit and their combined value specifies a Control word. Figure 4 shows the control Word format. It consists of four **fields** SELA, SELB and SELREG contain three bits each and SELOPR field contains four bits. Thus the total bits in the control word are 13-bits.

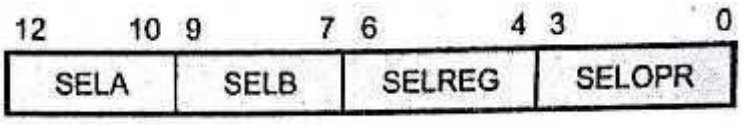

Figure 4: Format of control word

The three bits of SELA select a source registers of the A input of the ALU. The three bits of SELB select a source registers of the B input of the ALU. The three bits of SELREG select a destination register using the decoder. The four bits of SEL OPR select the operation to be performed by ALU.

Table 1: Encoding of Register Selection Fields

Binary Code	SEL A	SEL B	SEL D
000	Input	Input	None
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

The encoding of the ALU operations for the CPU is specified in the table given below. The OPR field has five bits and each operation is designated with a symbolic name.

Table 2: Encoding of ALU Operation

OPR Select	Operation	Symbol
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	Add A+B	ADD
00101	Subtract A-B	SUB
00110	Decrement A	DECA
01000	AND A and B	AND
01010	OR A and B	OR
01100	XOR A and B	XOR
01110	Complement A	COMA
10000	Shift Right A	SHRA
11000	Shift Left A	SHLA

STACK ORGANIZATION

The stack in the digital computer is a part of register unit or memory unit with a register that holds the address for the stack. The part of register array or memory used for stack is called stack area and the register used to hold the address of stack is called stack pointer. The value in the stack pointer always points at the top data element in the stack.

1. **Register Stack**

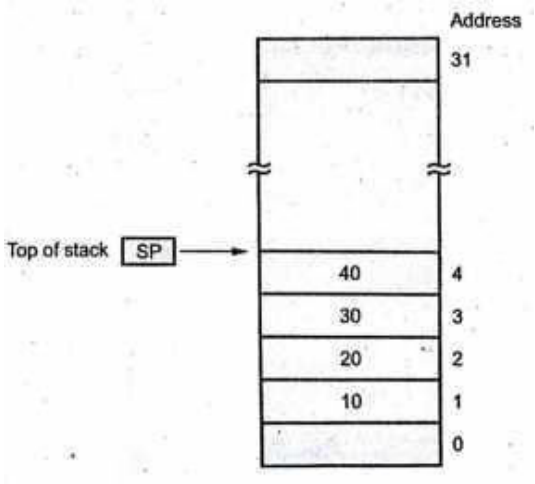

Figure 5 word register stack

A stack can be placed in a portion of a memory unit or it can be organized as a collection of a finite number of CPU registers. The Figure 5 shows the organization of a 32-word register stack. The stack pointer holds the address of the register that is currently the top of stack. As shown in the Figure 5 four data elements 10, 20, 30 and 40 are placed in the stack. The data element 40 is on the top of stack therefore, the content of SP is now 4.

2. Memory Stack

The operation of memory stack is exactly similar to the register stack. It is implemented using computer memory instead of CPU register array. The number of registers in the CPU is limited and it restricts the size of stack in the stack computer. But when stack is implemented using memory its size is extended upto the memory addressing capacity of the CPU.

INSTRUCTION FORMAT:

Computer has a variety of instruction code formats, it is the control unit within the CPU that interprets each instruction code and provides the necessary control functions needed to process the instruction. The format of an instruction is usually depicted in a rectangular box symbolizing the bits of the instruction as they appear in memory words or in a control register given in figure 6. The bits of the instruction are divided into groups called fields. These information fields of instructions are called elements of instruction & most common fields found in instruction formats are:

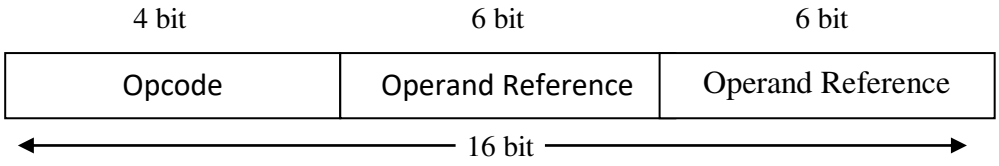

Figure 6: Instruction Format

Operation code: - The operation code field in the instruction specifies the operation to be performed. The operation is specified by binary code hence the name operation code or simply opcode.

Source / Destination operand: - The source/destination operand field directly specifies the source/destination operand for the instruction.

Source operand address: - The operation specified by the instruction may require one or more operands. The source operand may be in the CPU register or in the memory.

Destination operand address: - The operation executed by the CPU may produce result. Most of the time the results are stored in one of the operand. Such operand is known as destination operands.

Next instruction address: - The next instruction address tells the CPU from where to fetch the next instruction after completion of execution of current instruction.

Address Instructions

In these instructions, the locations of all operands are defined implicitly. Such instructions are found in machines that store operands in a structure called a pushdown stack. 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.)

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,

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

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

ONE-ADDRESS INSTRUCTIONS: One-address instructions use an implied accumulator (AC) register for all data manipulation. The program to evaluate X = (A + B) ∗ (C + D) is as follows:

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

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. 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

The name "zero-address" is given to this type of computer because of the absence of an address field in the computational instructions.

I/O SYSTEM

The central processing unit, memory unit and I/O unit are the hardware components/modules of the computer. They work together with communicating each other and have paths for connecting the modules together.

The terminal sends and receives serial information. Each quantity of information has eight bits of an alphanumeric code. The serial information from the keyboard is shifted into the input register INPR. The serial information for the printer is stored in the output register OUTR. These two register communicate with a communication interface serially and with the AC in parallel. The flags are needed to synchronize the timing difference between I/O device and the Computer.

The input-output configuration is shown in Figure 7.

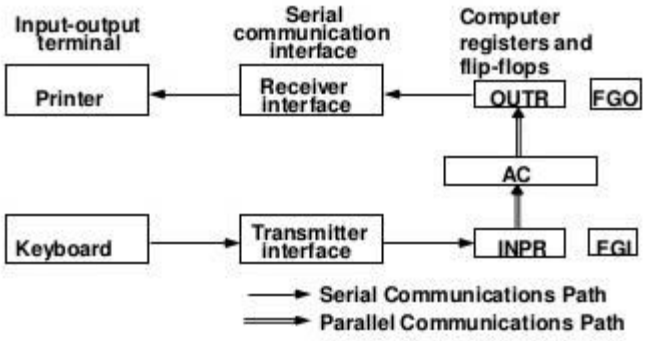

Figure 7: Input-Output Configuration

Bus:

The collection of paths connecting the various modules is called the interconnection structure or Bus. A group of wires called bus is used to provide necessary signals for communication between modules. A bus is a shared transmission medium, it must only be used by one device at a time and when used to connect major computer components (CPU, memory, I/O) is 'called a system bus.

The system bus is separated into three functional groups: data bus, address bus and control bus

Data lines (data bus) - Move data between system modules. The data bus lines are bidirectional. CPU can read data on these lines from a port, as well as send data out on these lines to a memory location or to a port. Width is a key factor, It determines number of bytes that can be transferred in one cycle and hence the overall system performance.

Address lines (address bus) - Designate source or destination of data on the data bus. It is a unidirectional bus. Width determines the maximum possible memory capacity of the system. It also used to address I/O ports. Typically: High -order bits select a particular module. Lower order bits select a memory location or I/0 ports within the module.

Control lines (Control bus) - Control access to use the data and address lines. Typical control lines include:

1. Memory Read and Memory write
2. I/O Read and [/0 Write
3. Transfer ACK
4. Bus Request and Bus Grant
5. Interrupt Request and Interrupt ACK
6. Clock & Reset

If one module wishes to send data to another, it must:

1. Obtain use of the bus
2. Transfer data via the bus

If one module wishes to request data from another, it must :

1. Obtain use of the bus
2. Transfer a request to the other module over control and address lines
3. Wait for second module to send data

Connecting I/O Devices to CPU and Memory

It shows that how I/0 devices are connected to CPU and memory. I/0 devices are interfaced to CPU through I/O interface or I/O module. The I/O interface consists of circuit, which connect an I/O device to a CPU bus. On one side of the interface we have the bus signals for address, data and control. On the other side we have a data path with its associated controls to transfer data between the interface and the I/O device. Usually I/O interface or I/O module is capable of interfacing more than one external device.

Since data, address and control buses are connected in parallel to CPU, memory and I/O the I/O module is allowed to exchange data directly with memory without going through the processor, using Direct Memory-Access (DMA). The bus interconnection scheme shown in Figure 7 supports following types of data transfers:

1. Memory to processor - Memory read operation
2. Process to memory – Memory write operation
3. Processor to I/O – I/O write operation
4. I/O to processor - I/O read operation
5. I/O to or from memory- DMA operation

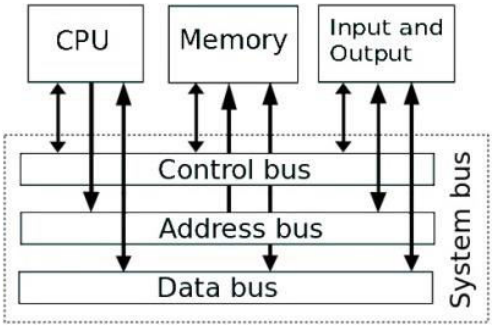

Figure 8: Architecture of System Bus

BUS STRUCTURE

- A more efficient scheme for transferring information between registers in a multiple- register configuration is a common bus system.
- A bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one at a time.
- Control signals determine which register is selected by the bus during each particular register transfer. A common bus system can be constructed using multiplexers.
- These multiplexers select the source register whose binary information is then placed on the bus.
- The system bus is a cable which carries data communication between the major components of the computer, including the microprocessor.
- The system bus consists of three different groups of wiring, called the data bus, control bus and address bus.

REGISTER TRANSFER LANGUAGE

The symbolic notation used to describe the micro operation transfer among registers is called a *register transfer language*. The term "register transfer" implies the availability of hardware logic circuits that can perform stated micro operation and transfer the results to the operation to the same or another register.

Example:

MOV R1, R2

Here MOV is an Opcode, R1 is destination Register and R2 is source Register. In this Instruction Content of R2 is being transferred into Register R1.

The features of register transfer logic are:

1. Uses registers as a primitive component in the digital system instead of flip-flops and gates.
2. The information flow and processing tasks among the data stored in the registers is described in a concise and precise manner.
3. Uses a set of expressions and statements which resemble the statements used in programming languages.

4. The presentation of digital functions in register transfer logic is very user friendly.

The micro-operations used in the digital system can be classified as:

1. **Register transfer micro-operations:** The micro-operations that transfer information from one register to another.
2. **Arithmetic micro-operations:** The micro-operations that perform arithmetic operations on numeric data stored in registers.
3. **Logic micro-operation:** The micro-operations that perform bit manipulation operations on non-numeric data stored in registers.
4. **Shift micro-operations:** The micro-operations that perform shift operations on data stored in registers.

Table 3: Examples of Micro operations for the CPU

Micro operations	Symbolic Designation				Control Word
	SEL A	SEL B	SEL D	OPR	
R1<-R2 - R3	R2	R3	R1	SUB	010 011 001 00101
R4<-R4 V R5	R4	R5	R4	OR	100 101 100 01010
R6<-R6+1	R6	-	R6	INCA	110 000 110 00001
R7<-R1	R1	-	R7	TSFA	001 000 111 00000
Output<-R2	R2	-	None	TSFA	010 000 000 00000
Output<-Input	Input	-	None	TSFA	000 000 000 00000
R4<-sh1 R4	R4	-	R4	SHLA	100 000 100 11000
R5<-0	R5	R5	R5	XOR	101 101 101 01100

BUS AND MEMORY TRANSFER

Bus Transfer

A digital computer has many registers and it is necessary to provide data path between them to transfer information from one register to another. If separate lines are used between each register there will be excess number of wires and controlling of those wires make circuit complex. Therefore, in multiple-register configuration a common bus system is used to transfer information between two registers.

Implementation of common bus using multiplexers

The Figure 8 shows the implementation of common bus system for four registers using multiplexers. Each register has four bits, numbered 0 through 3 and they are routed through multiplexers to the common bus. Here, four multiplexers are used to select four bits of the source register. Each multiplexer has four input lines, two select lines and one output line. The four input lines of multiplexer 0 (MUX 0) are connected to the bit 0 outputs of four registers such that bit 0 of register is connected to input 0, bit 0 of register 1 is connected to input 1, bit 0 of register 2 is connected to input 2 and bit 0 of register 3 is connected to input 3. Similarly, inputs for MUX 1 are connected to bit 1 outputs, inputs for MUX 2 are connected to bit 2, and inputs for MUX 3 are connected to bit 3 outputs of registers 0 through 3. To avoid the complexity of the diagram, only input connections for MUX 3 is physically shown.

The two selection lines S1 and S0 are connected to the selection inputs of all four multiplexers. These lines choose the four bits of one register and transfer them into the four-line common bus through OUT lines. When S1S0 = 00, the input 0 of all four multiplexers are selected and applied to the outputs to transfer them on the common bus. As a result a common bus receives the contents of register 0. Since the outputs of register 0 are connected to the input 0 of the multiplexers. Similarly, the content of register 1 are transferred on the common bus when S1S1. The Table below shows the register selection according to the status of S1 and S0 lines.

Table 4: Selection table

S1	S0	Selected Register
0	0	Register 0
0	1	Register 1
1	0	Register 2
1	1	Register 3

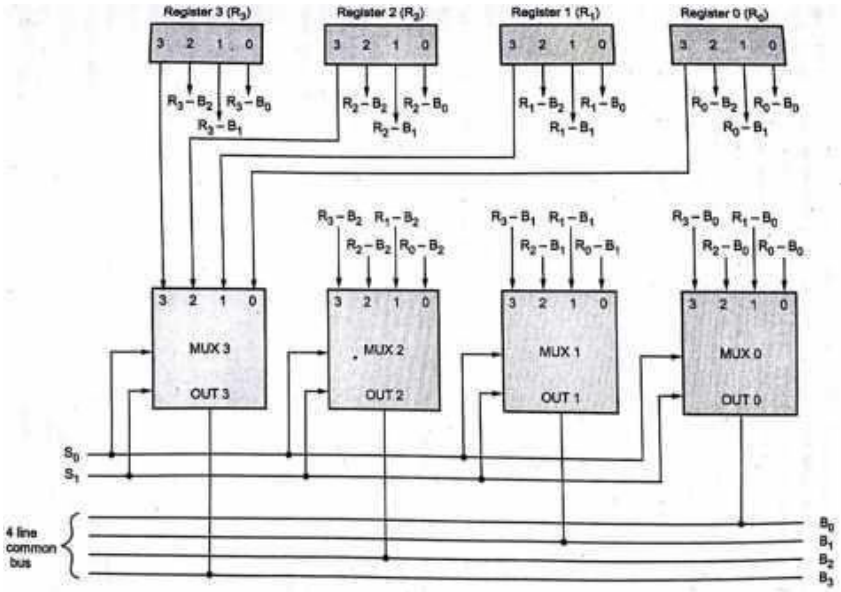

Figure 9. Computer bus System using multiplexer

ADDRESSING MODES

Every instruction of a program has to operate on some data. An addressing mode is a way in which an operand is specified in an instruction.

There are different ways in which an operand may be specified in an instruction.

1. **Implied Mode:** In implied addressing mode, the instruction itself specifies the data to be operated. the implied addressing mode is also called implicit addressing mode, because there is no need to explicitly specify an effective address for either the source or the destination.
   For example -''complement accumulator''
2. **Immediate Mode:** In this mode the operand is specified in the instruction itself. The actual operand to be used in conjunction with the operation specified in the instruction is contained in the operand field.
   Example: MOVE A, #20
3. **Register Mode:** In this mode the operands are in registers that reside within the CPU. The register required is chosen from a register field in the instruction.
   Example: MOV R1, R2
4. **Register Indirect Mode:** In this mode the instruction specifies a register that contains the address of the operand and not the operand itself.
   Effective Address=R
   Example: MOVE A, (R0)
5. **Auto increment or Auto decrement Mode:** After execution of every instruction from the data in memory it is necessary to increment or decrement the register. This is done by using the increment or decrement instruction.
   Example: MOVE R2), + R0
   MOVE (R2), - R0
6. **Direct Address Mode:** In this mode the operand resides in memory and its address is given directly by the address field of the instruction such that the effective address is equal to the address part of the instruction. Example: MOVE A, 2000
7. **Indirect Address Mode:** The effective address of the operand is the contents of a register or main memory location, location whose address appears in the instruction. Indirection is noted by placing the name of the register or the memory address given in the instruction in parentheses.
   Effective address = address part of instruction + context of CPU register
8. **Relative Address Mode:** In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address.
   EA = PC + Address part of instruction
9. **Indexed Addressing Mode:** In this mode the effective address is obtained by adding the content of an index register to the address part of the instruction.
   EA= offset + R
   Example: MOVE 20 [R1], R2
10. **Base Register Addressing Mode:** In this mode the effective address is obtained by adding the content of a base register to the part of the instruction. A base register is assumed to hold a base address and the address field of the instruction, and gives a displacement relative to this base address. The base register addressing mode is handy for relocation of programs in memory to another as required in multi programming systems
    Example: ADD AX, [BX+SI]

Table 5: Eight Addressing Modes for load Instruction

Mode	Assembly Convention	Register Transfer
Direct Address	LD ADR	AC<-M[ADR]
Indirect Address	LD @ADR	AC<-M[M[ADR]]
Relative Address	LD $ADR	AC<-M[PC+ADR]
Immediate Operand	LD #NBR	AC<-NBR
Index Addressing	LD ADR(X)	AC<-M[ADR+XR]
Register	LD R1	AC<-R1
Register Indirect	LD(R1)	AC<-M[R1]
Auto increment	LD (R1)	AC<-M[R1],R1<-R1+1