C.S.E → 5th Sem UNIT-II 30.

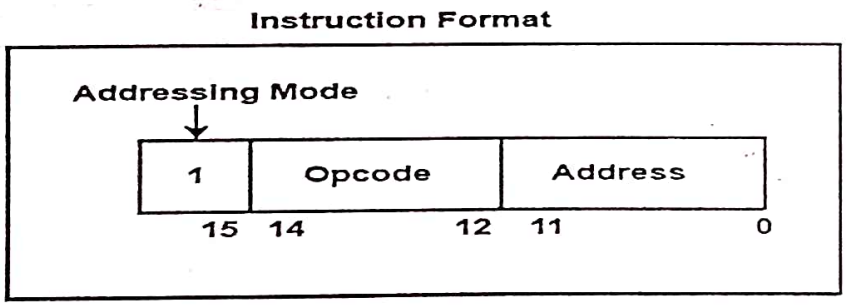
Basic Computer organization and Design: Instruction codes, stored program organization, computer registers and common bus system, computer instructions, timing and control, instruction cycle: Fetch and Decode, Register reference instructions; Memory reference instructions. Input, output and Interrupt: configuration, instructions, Program interrupt, Interrupt cycle, Micro-programmed Control organization, Control Memory, address sequencing, Micro program Example, micro instruction format, Horizontal Vs Vertical micro-programming, design of control Unit, micro program sequencer, Hardwired v/s Micro-Programmed Control Unit.

COMPUTER ARCHITECTURE: INSTRUCTION CODES

An **instruction code** is a group of bits that instruct the computer to perform a specific operation. The operation code of an instruction is a group of bits that define operations such as addition, subtraction, shift, complement, etc. An instruction must also include one or more operands, which indicate the registers and/or memory addresses from which data is taken or to which data is deposited.

Micro-operations-The instructions are stored in computer memory in the same manner that data is stored. The control unit interprets these instructions and uses the operations code to determine the sequences of micro-operations that must be performed to execute the instruction.

An **instruction code** categorized into two elements as **Operation codes** (Opcodes) and **Address**. Opcodes specify the operation for specific instructions. An address determines the registers or the areas that can be used for that operation. Operands are definite elements of computer instruction that show what information is to be operated on.



It consists of 12 bits of memory that are required to define the address as the memory includes 4096 words. The 15th bit of the instruction determines the addressing mode (where direct addressing corresponds to 0, indirect addressing corresponds to 1). Therefore, the instruction format includes 12 bits of address and 1 bit for the addressing mode, 3 bits are left for Opcodes.

There are three parts of the Instruction Format which are as follows –

Addressing Modes

Instructions that define the address of a definite memory location are known as memory reference instructions. The method in which a target address or effective address is recognized within the instruction is known as addressing mode.

The address field for instruction can be represented in two different ways are as follows –

Direct Addressing – It uses the address of the operand.

Indirect Addressing – It facilitates the address as a pointer to the operand.

The address of the operand or the target address is called the effective address.

Effective Address (EA) – It defines the address that can be executed as a target address for a branch type instruction or the address that can be used directly to create an operand for a computation type instruction, without creating any changes.

Opcodes

An **Opcode** is a collection of bits that represents the basic operations including add, subtract, multiply, complement, and shift. The total number of operations provided through the computer determines the number of bits needed for the opcode. The minimum bits accessible to the Opcode should be n for 2n operations. These operations are implemented on information that is saved in processor registers or memory.

Address

The address is represented as the location where a specific instruction is constructed in the memory. The address bits of an instruction code is used as an operand and not as an address. In such methods, the instruction has an immediate operand. If the second part has an address, the instruction is referred to have a direct address.

There is another possibility in the second part including the address of the operand. This is referred to as an indirect address. In the instruction code, one bit can signify if the direct or indirect address is executed.

STORED PROGRAM ORGANIZATION

The simplest way to organize a computer is to have one processor register and an instruction code format with two parts. The first part specifies the operation to be performed and the second

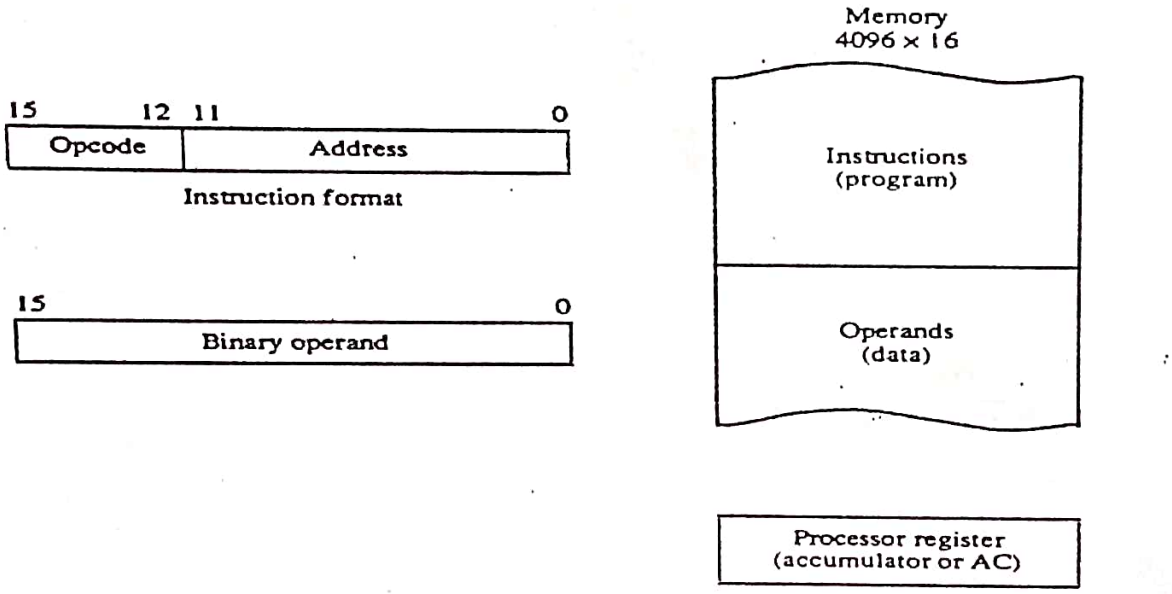
specifies an address. The memory address tells the control where to find an operand in memory. This operand is read from memory and used as the data to be operated on together with the data stored in the processor register.

Figure below depicts this type of organization. Instructions are stored in one section of memory and data in another. For a memory unit with 4096 words we need 12 bits to specify an address since 212 = 4096. If we store each instruction code in one 16-bit memory word, we have available four bits for the operation code (abbreviated opcode) to specify one out of 16 possible operations, and 12 bits to specify the address of an operand.

The control reads a 16-bit instruction from the program portion of memory. It uses the 12-bit address part of the instruction to read a 16-bit operand from the data portion of memory.

It then executes the operation specified by the operation code.

Figure Stored program organization.



- Computers that have a single-processor register usually assign to it the name accumulator and label it AC. The operation is performed with the memory operand and the content of AC.
- If an operation in an instruction code does not need an operand from memory, the rest of the bits in the instruction can be used for other purposes. For example, operations such as clear AC, complement AC, and increment AC operate on data stored in the AC register.
- They do not need an operand from memory. For these types of operations, the second part of the instruction code (bits 0 through 11) is not needed for specifying a memory address and can be used to specify other operations for the computer.

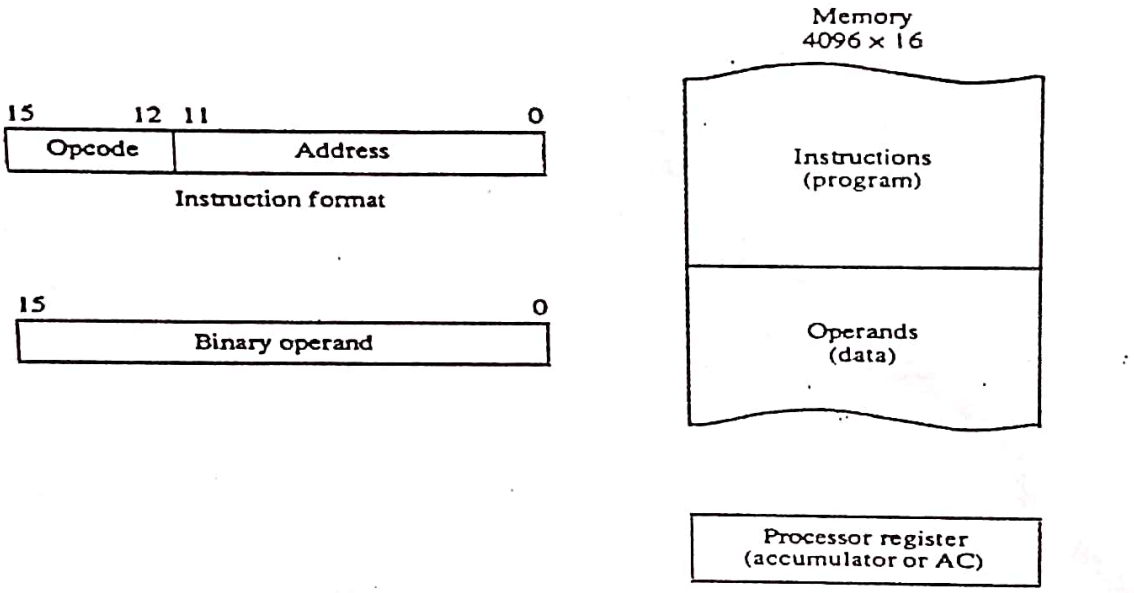
specifies an address. The memory address tells the control where to find an operand in memory. This operand is read from memory and used as the data to be operated on together with the data stored in the processor register.

Figure below depicts this type of organization. Instructions are stored in one section of memory and data in another. For a memory unit with 4096 words we need 12 bits to specify an address since 212 = 4096. If we store each instruction code in one 16-bit memory word, we have available four bits for the operation code (abbreviated opcode) to specify one out of 16 possible operations, and 12 bits to specify the address of an operand.

The control reads a 16-bit instruction from the program portion of memory. It uses the 12-bit address part of the instruction to read a 16-bit operand from the data portion of memory.

It then executes the operation specified by the operation code.

Figure Stored program organization.



- Computers that have a single-processor register usually assign to it the name accumulator and label it AC. The operation is performed with the memory operand and the content of AC.
- If an operation in an instruction code does not need an operand from memory, the rest of the bits in the instruction can be used for other purposes. For example, operations such as clear AC, complement AC, and increment AC operate on data stored in the AC register.
- They do not need an operand from memory. For these types of operations, the second part of the instruction code (bits 0 through 11) is not needed for specifying a memory address and can be used to specify other operations for the computer.

COMPUTER REGISTERS AND COMMON BUS SYSTEM

Registers are a type of computer memory used to quickly accept, store, and transfer data and instructions that are being used immediately by the CPU. The registers used by the CPU are often termed as Processor registers.

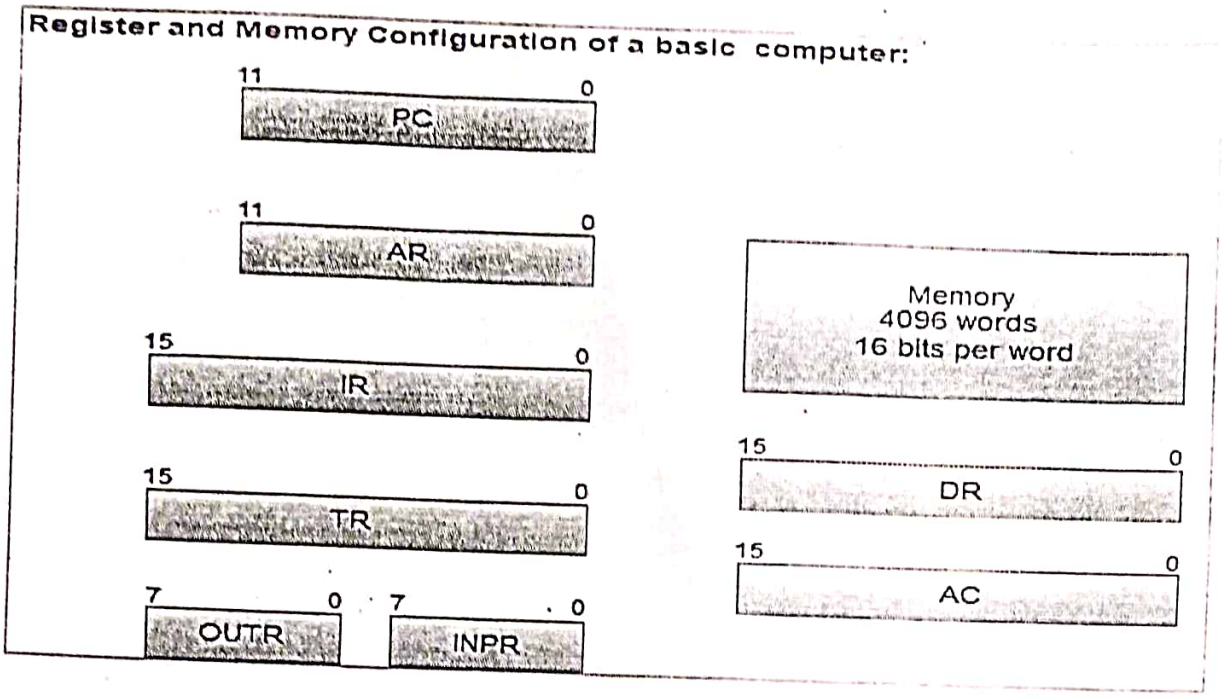
A processor register may hold an instruction, a storage address, or any data (such as bit sequence or individual characters).

The computer needs processor registers for manipulating data and a register for holding a memory address. The register holding the memory location is used to calculate the address of the next instruction after the execution of the current instruction is completed.

Following is the list of some of the most common registers used in a basic computer:

Register	Symbol	Number of bits	Function
Data register	DR	16	Holds memory operand
Address register	AR	12	Holds address for the memory
Accumulator	AC	16	Processor register
Instruction register	IR	16	Holds instruction code
Program counter	PC	12	Holds address of the instruction
Temporary register	TR	16	Holds temporary data
Input register	INPR	8	Carries input character
Output register	OUTR	8	Carries output character

The following image shows the register and memory configuration for a basic computer.



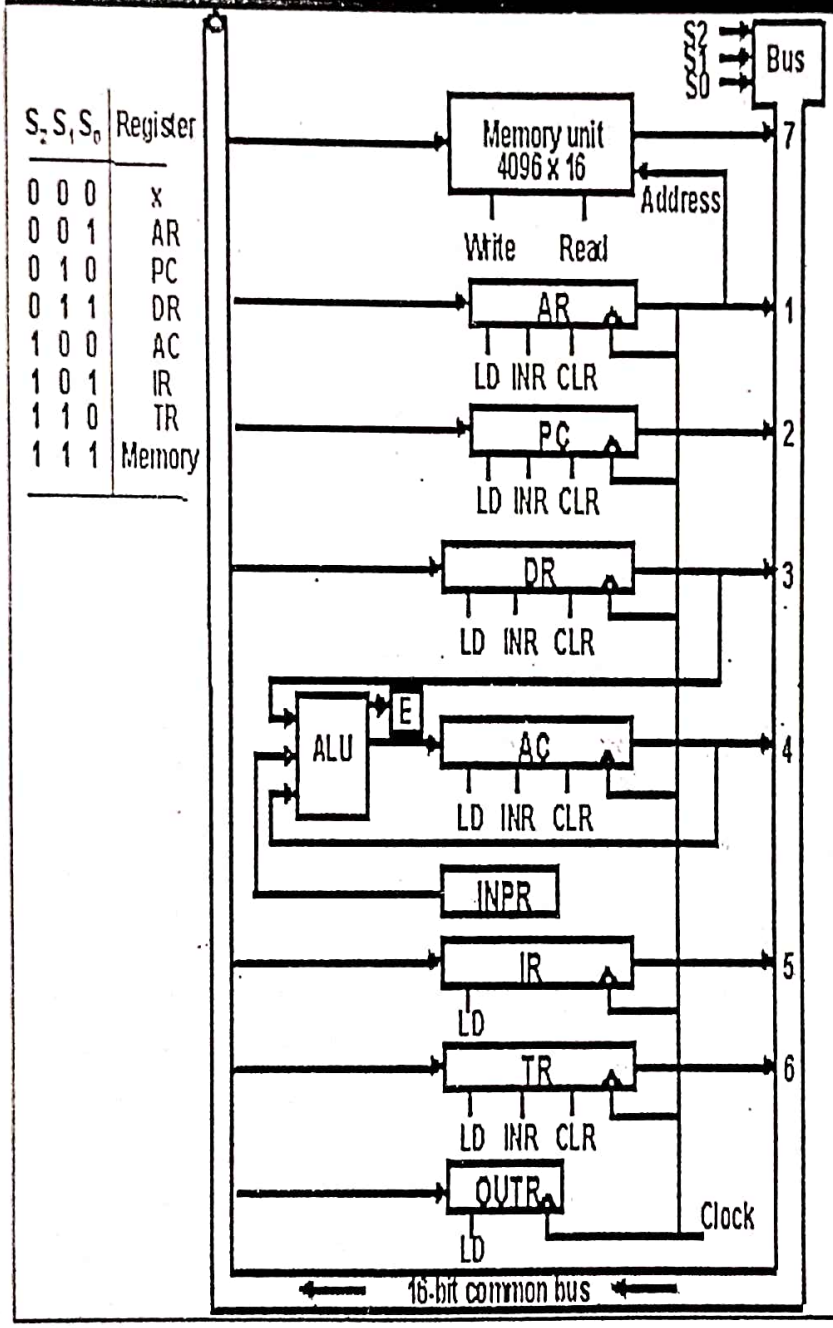
- The Memory unit has a capacity of 4096 words, and each word contains 16 bits.
- The Data Register (DR) contains 16 bits which hold the operand read from the memory location.
- The Memory Address Register (MAR) contains 12 bits which hold the address for the memory location.
- The Program Counter (PC) also contains 12 bits which hold the address of the next instruction to be read from memory after the current instruction is executed.
- The Accumulator (AC) register is a general purpose processing register.
- The instruction read from memory is placed in the Instruction register (IR).
- The Temporary Register (TR) is used for holding the temporary data during the processing.
- The Input Registers (IR) holds the input characters given by the user.
- The Output Registers (OR) holds the output after processing the input data.

COMMON BUS SYSTEM

We shall study the common bus system of a very basic computer in this article. A basic computer has 8 registers, memory unit and a control unit. To avoid excessive wiring, memory and all the register are connected via a common bus. The specific output that is selected for the bus is determined by S2S1S0. The register whose LD (Load) is enable receives the data from the bus.

Registers can be incremented by setting the INR control input and can be cleared by setting the CLR control input. The Accumulator's input must come via the Adder & Logic Circuit. This allows the Accumulator and Data Register to swap data simultaneously. The address of any memory location being accessed must be loaded in the Address Register.

COMMON BUS SYSTEM



S_2 S_1 S_0	Register
0 0 0	x
0 0 1	AR
0 1 0	PC
0 1 1	DR
1 0 0	AC
1 0 1	IR
1 1 0	TR
1 1 1	Memory

Outputs of the six registers and memory unit are connected to the common bus of 16 lines. The specific output is selected from the binary value of the selection variables S2, S1, S0.

The lines from the common bus are connected to the inputs of each register and the data input of the memory. The particular register whose LD (Load) input is enabled receives the data from the bus.

INPR receives the a character from an input device and which is then transferred to AC. OUTR receive the character from AC and delivers it to an output device.

Connections:

The outputs of all the registers except the OUTR (output register) are connected to the common bus. The output selected depends upon the binary value of variables S2, S1 and S0. The lines from common bus are connected to the inputs of the registers and memory. A register receives the information from the bus when its LD (load) input is activated while in case of memory the Write input must be enabled to receive the information. The contents of memory are placed onto the bus when its Read input is activated.

Various Registers:

4 registers DR, AC, IR and TR have 16 bits and 2 registers AR and PC have 12 bits. The INPR and OUTR have 8 bits each. The INPR receives character from input device and delivers it to the AC while the OUTR receives character from AC and transfers it to the output device. 5 registers have 3 control inputs LD (load), INR (increment) and CLR (clear). These types of registers are similar to a binary counter.

Abbreviation	Register name
OUTR	Output register
TR	Temporary register
IR	Instruction register
INPR	Input register
AC	Accumulator
DR	Data register
PC	Program counter
AR	Address register

Adder and logic circuit:

The adder and logic circuit provides the 16 inputs of AC. This circuit has 3 sets of inputs. One set comes from the outputs of AC which implements register micro operations. The other set comes from the DR (data register) which are used to perform arithmetic and logic micro operations. The result of these operations is sent to AC while the end around carry is stored in E as shown in diagram. The third set of inputs is from INPR.

COMPUTER INSTRUCTIONS

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

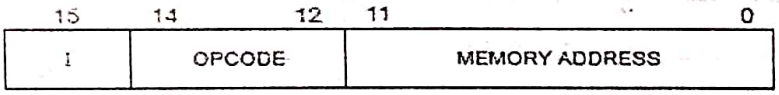
An instruction comprises of groups called fields. These fields include:

- The Operation code (Opcode) field which specifies the operation to be performed.
- The Address field which contains the location of the operand, i.e., register or memory location.
- The Mode field which specifies how the operand will be located.

A basic computer has three instruction code formats which are:

1. **Memory - reference instruction**
2. **Register - reference instruction**
3. **Input-Output instruction**

Memory - reference instruction



In Memory-reference instruction, 12 bits of memory is used to specify an address and one bit to specify the addressing mode 'I'.

Memory Reference – These instructions refer to memory address as an operand. The other operand is always accumulator. Specifies 12-bit address, 3-bit opcode (other than 111) and 1-bit addressing mode for direct and indirect addressing.

Example

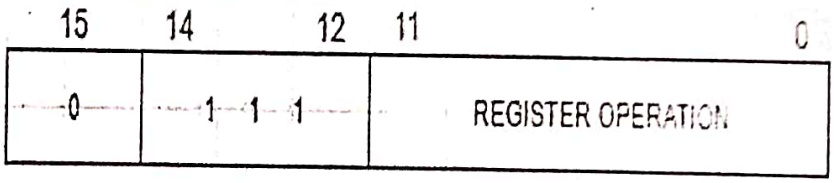
IR register contains = 0001XXXXXXXXXXX, i.e. ADD after fetching and decoding of instruction we find out that it is a memory reference instruction for ADD operation.

Hence, DR ← M[AR] AC ← AC + DR, SC ← 0

Basic Memory-Reference Instructions

Symbol	Details	Operation Decoder	Symbolic Description
AND	Performing bitwise AND operation	D_0	AC ← AC ∧ M[AR]
ADD	Adds together its two operands	D_1	AC ← AC + M[AR] →, E ← C_{out}
LDA	LoaD Accumulator with the contents from memory	D_2	AC ← M[AR]
STA	STore Accumulator contents in memory	D_3	M[AR] ← AC
BUN	The Branch Unconditionally (BUN) instruction can send the instruction that is determined by the effective address	D_4	PC ← AR
BSA	Branch and Save return Address	D_5	M[AR] ← PC , PC ← AR + 1
ISZ	The Increment if Zero (ISZ) instruction increments the word determined by effective address	D6	M[AR] ← M[AR] + 1, if M[AR] + 1 = 0 then PC ← PC + 1

Register - reference instruction



The Register-reference instructions are represented by the Opcode 111 with a 0 in the leftmost bit (bit 15) of the instruction.

A Register-reference instruction specifies an operation on or a test of the AC (Accumulator) register.

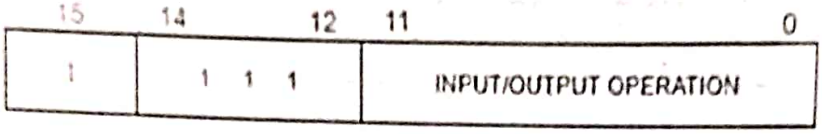
Register Reference – These instructions perform operations on registers rather than memory addresses. The IR(14 – 12) is 111 (differentiates it from memory reference) and IR(15) is 0 (differentiates it from input/output instructions). The rest 12 bits specify register operation.

Example

IR register contains = 0111001000000000, i.e. CMA after fetch and decode cycle we find out that it is a register reference instruction for complement accumulator.

Hence, AC ← ~AC

Input-Output instruction



Just like the Register-reference instruction, an Input-Output instruction does not need a reference to memory and is recognized by the operation code 111 with a 1 in the leftmost bit of the instruction. The remaining 12 bits are used to specify the type of the input-output operation or test performed.

Input/output – These instructions are for communication between computer and outside environment. The IR (14 – 12) is 111 (differentiates it from memory reference) and IR (15) is 1 (differentiates it from register reference instructions). The rest 12 bits specify I/O operation.

Example

IR register contains = 1111100000000000, i.e. INP after fetch and decode cycle we find out that it is an input/output instruction for inputting character. Hence, INPUT character from peripheral device.

The set of instructions incorporated in16 bit IR register are:

1. Arithmetic, logical and shift instructions (and, add, complement, circulate left, right, etc)
2. To move information to and from memory (store the accumulator, load the accumulator)
3. Program control instructions with status conditions (branch, skip)
4. Input output instructions (input character, output character)

Note

- The three operation code bits in positions 12 through 14 should be equal to 111. Otherwise, the instruction is a memory-reference type, and the bit in position 15 is taken as the addressing mode I.
- When the three operation code bits are equal to 111, control unit inspects the bit in position 15. If the bit is 0, the instruction is a register-reference type. Otherwise, the instruction is an input-output type having bit 1 at position 15.

TIMING AND CONTROL

All sequential circuits in the Basic Computer CPU are driven by a master clock, with the exception of the INPR register.

At each clock pulse, the control unit sends control signals to control inputs of the bus, the registers, and the ALU.

Control unit design and implementation can be done by two general methods:

A **hardwired** ***control unit*** is designed from scratch using traditional digital logic design techniques to produce a minimal, optimized circuit. In other words, the control unit is like an ASIC (application-specific integrated circuit).

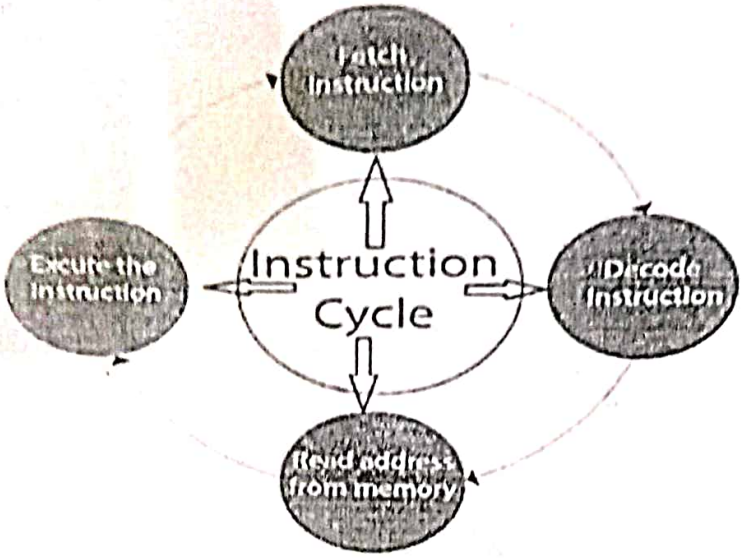
A **micro-programmed** ***control unit*** is built from some sort of ROM. The desired control signals are simply stored in the ROM, and retrieved in sequence to drive the micro-operations needed by a particular instruction.

INSTRUCTION CYCLE

The **instruction cycle** is related to the CPU (Central processing unit), a list of steps that follows when the computer system starts until the computer system is shut down. The other name of the instruction cycle is the **fetch-decode-execute cycle.** There are basically four phases in the instruction cycle: fetching an instruction from memory, decoding of the fetched instruction, reading of the address from memory, and the last phase include the instruction execution. The computer processor executes it.

In the computer system, all the instructions are executed in the RAM of the computer system. The CPU is responsible for executing the instruction. The CPU system first fetches the data and instruction from the main memory and store in the temporary memory, which is known as registers. This phase is known as the fetch cycle. After fetching an instruction from memory, the

next step taken by the CPU is decoding of fetched instruction. This phase is known as the decod phase.



There are basically five stages of the instruction cycle which are described below:

1. Initiating Cycle

In this phase, the computer system boots up, and the operating system loads in the main memory (read-only memory) of the central processing unit. It immediately begins when the computer system starts.

2. Fetching of Instruction

The fetching of instruction is the first phase. The fetch instruction is common for each instruction executes in a central processing unit. In this phase, the central processing unit sends the PC to MAR and then sends the READ command into a control bus. After sending a read command on the data bus, the memory returns the instruction, which is stored at that particular address in the memory. Then, the CPU copies data from the data bus into MBR and then copies the data from MBR to registers. After all this, the pointer is incremented to the next memory location so that the next instruction can be fetched from memory.

3. Decoding of Instruction

The decoding of instruction is the second phase. In this phase, the CPU determines which instruction is fetched from the instruction and what action needs to be performed on the instruction. The opcode for the instruction is also fetched from memory and decodes the related operation which needs to be performed for the related instruction.

4. Read of an Effective Address

The reading of an effective address is the third phase. This phase deals with the decision of the operation. The operation can be of any type of memory type non-memory type operation. Memory instruction can be categorized into two categories direct memory instruction and indirect memory instruction.

5. Executing of Instruction

The executing of instruction is the last phase. In this stage, the instruction is finally executed. The instruction is executed, and the result of the instruction is stored in the register. After the execution of an instruction, the CPU prepares itself for the execution of the next instruction. For every instruction, the execution time is calculated, which is used to tell the processing speed of the processor.

Importance of Instruction Cycle

- It is important for the processor system for the central processing unit as the instructions are the basic operations that are performed in the main memory of the central processing unit.
- It is a set of steps that help to understand the flow of instruction. By the instruction cycle, the end to end the flow of instructions can be visualized in the computer processor.
- It is common for all instruction set it needs to properly understand so that all the operations can be performed easily.
- By the instruction cycle, the processing time of the program can be easily calculated, which helps to determine the speed of the processor.
- As the speed of the processor tells how many instructions can be simultaneously executed in the central processing unit.

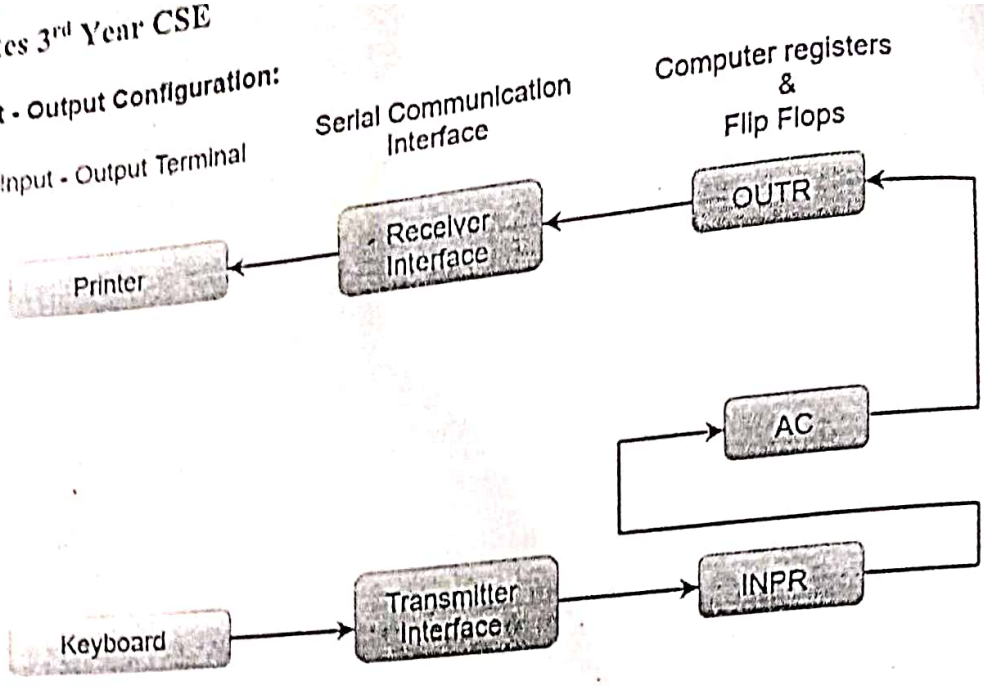
INPUT-OUTPUT CONFIGURATION

In computer architecture, input-output devices act as an interface between the machine and the user.

Instructions and data stored in the memory must come from some input device. The results are displayed to the user through some output device.

The following block diagram shows the input-output configuration for a basic computer.

Input - Output Configuration:



- The input-output terminals send and receive information.
- The amount of information transferred will always have eight bits of an alphanumeric code.
- The information generated through the keyboard is shifted into an input register 'INPR'.
- The information for the printer is stored in the output register 'OUTR'.
- Registers INPR and OUTR communicate with a communication interface serially and with the AC in parallel.
- The transmitter interface receives information from the keyboard and transmits it to INPR.
- The receiver interface receives information from OUTR and sends it to the printer serially.

Design of a Basic Computer

A basic computer consists of the following hardware components.

1. A memory unit with 4096 words of 16 bits each
2. Registers: AC (Accumulator), DR (Data register), AR (Address register), IR (Instruction register), PC (Program counter), TR (Temporary register), SC (Sequence Counter), INPR (Input register), and OUTR (Output register).
3. Flip-Flops: I, S, E, R, IEN, FGI and FGO
4. Two decoders: a 3 x 8 operation decoder and 4 x 16 timing decoder
5. A 16-bit common bus
6. Control Logic Gates
7. The Logic and Adder circuits connected to the input of AC.

PROGRAM INTERRUPT:

Program interrupt defines the transfer of program control from a currently running program to another service program as a result of an external or internal created request. Control returns to the initial program after the service program is implemented.

There are three major types of program interrupts that are as follows –

External Interrupts

External interrupts come from input-output (I/0) devices, from a timing device, from a circuit monitoring the power supply, or from any other external source. The timeout interrupt can result from a program that is in an endless loop and thus exceeded its time allocation. Power failure interrupt can have as its service routine a program that transfers the complete state of the CPU into a non-destructive memory in a few milliseconds before power ceases.

Internal Interrupts

Internal interrupts arise from illegal or erroneous use of an instruction or data. Internal interrupts are also called traps. These error conditions generally appear as a result of premature termination of the instruction execution. The service program that processes the internal interrupt determines the corrective measure to be taken.

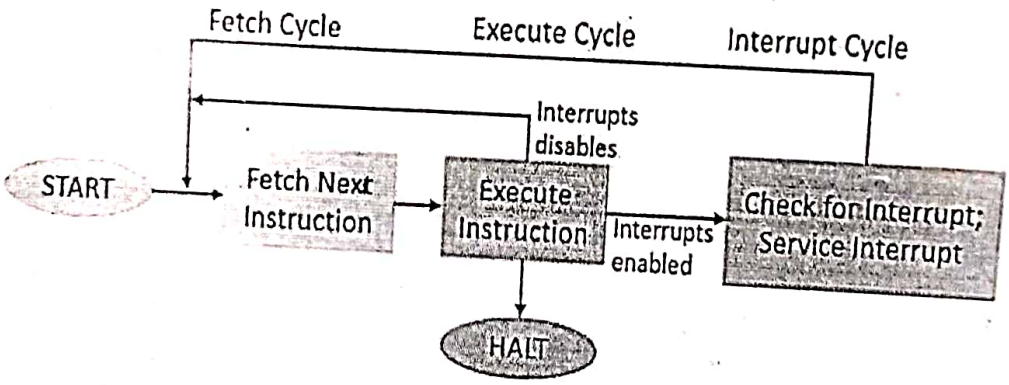
The main difference between internal and external interrupts is that the internal interrupt is initiated by some exceptional condition caused by the program itself rather than by an external event. Internal interrupts are synchronous with the program while external interrupts are asynchronous. If the program is rerun, the internal interrupts will appear in the same place each time. External interrupts depend on external conditions that are independent of the program being executed at the time.

Software Interrupts

A software interrupt is initiated by executing an instruction. A software interrupt is a special call instruction that behaves like an interrupt rather than a subroutine call. It can be used by the programmer to initiate an interrupt procedure at any desired point in the program.

INTERRUPT CYCLE

- An instruction cycle (sometimes called fetch-and-execute cycle, fetch-decode-execute cycle, or FDX) is the basic operation cycle of a computer. It is the process by which a computer retrieves a program instruction from its memory, determines what actions the instruction requires, and carries out those actions. This cycle is repeated continuously by the central processing unit (CPU), from boot up to when the computer is shut down.
- Consider the condition that the interrupts are enabled. In this case, if an interrupt occurs then the processor halt the execution of the current program. Thereby it saves the address of the instruction that has to be executed next and service the occurred interrupt.
- To process the interrupts the processor set the program counter with starting address of the interrupt service routine. This would let the processor fetch the first instruction of interrupt service routine and service the occurred interrupt. Once the interrupt is serviced the processor resumes back to the execution of the program it has halted to service the interrupt. It set the program counter with the address of the next instruction to be executed.
- If the interrupts are disabled then the processor will simply ignore the occurrence of interrupts. The processor will smoothly execute the currently running program and will check the pending interrupts once the interrupts are enabled.



Instruction Cycle with Interrupts

Block diagram of Interrupt Cycle

- After the execute cycle is completed, a test is made to determine if an interrupt was enabled (e.g. so that another process can access the CPU)
- If not, instruction cycle returns to the fetch cycle
- If so, the interrupt cycle might perform the following tasks: (simplified...)

- move the current value of PC into MBR
- move the PC-save-address into MAR
- move the interrupt-routine-address into PC
- move the contents of the address in MBR into indicated memory cell
- continue the instruction cycle within the interrupt routine
- after the interrupt routine finishes, the PC-save-address is used to reset the value of PC and program execution can continue

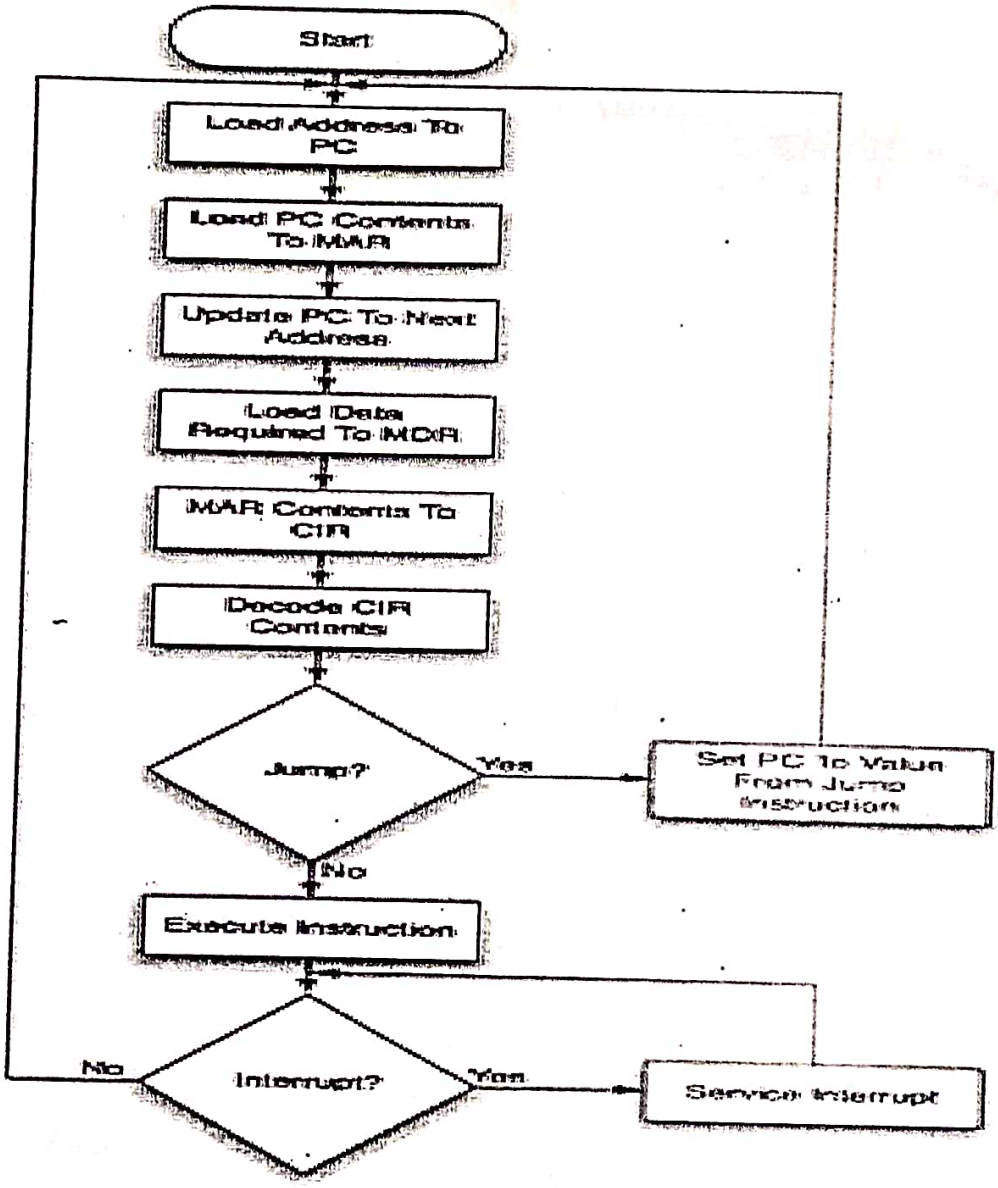


Figure: Flowchart of interrupt cycle

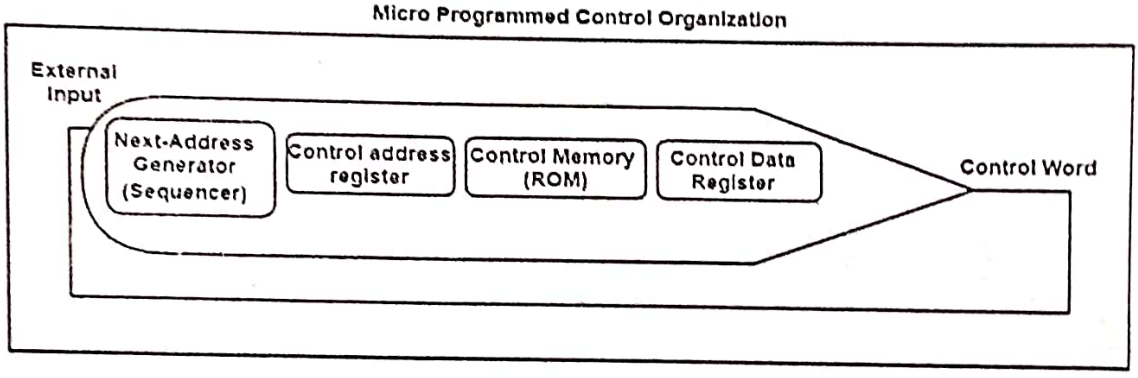
MICRO-PROGRAMMED CONTROL ORGANIZATION:

A control unit whose binary control values are saved as words in memory is called a micro-programmed control unit.

A controller results in the instructions to be implemented by constructing a definite collection of signals at each system clock beat. Each of these output signals generates one micro-operation including register transfer. Thus, the sets of control signals are generated definite micro-operations that can be saved in the memory.

Each bit that forms the microinstruction is linked to one control signal. When the bit is set, the control signal is active. When it is cleared the control signal turns inactive. These microinstructions in a sequence can be saved in the internal '**control**' memory. The control unit of a micro-program controlled computer is a computer inside a computer.

The following image shows the block diagram of a Micro-programmed Control organization.



There are the following steps followed by the micro-programmed control are –

- It can execute any instruction. The CPU should divide it down into a set of sequential operations. This set of operations are called microinstruction. The sequential micro-operations need the control signals to execute.
- Control signals saved in the ROM are created to execute the instructions on the data direction. These control signals can control the micro-operations concerned with a microinstruction that is to be performed at any time step.
- The address of the microinstruction is executed next is generated.
- The previous 2 steps are copied until all the microinstructions associated with the instruction in the set are executed.

The address that is supported to the control ROM originates from the micro counter register. The micro counter received its inputs from a multiplexer that chooses the output of an address ROM, a current address Incrementer, and an address that is saved in the next address field of the current microinstruction.

Advantages of Micro-programmed Control Unit

There are the following advantages of micro-programmed control are as follows –

- It can more systematic design of the control unit.
- It is simpler to debug and change.
- It can retain the underlying structure of the control function.
- It can make the design of the control unit much simpler. Hence, it is inexpensive and less error-prone.
- It can orderly and systematic design process.
- It is used to control functions implemented in software and not hardware.
- It is more flexible and used to complex function is carried out easily.

Disadvantages of Micro-programmed Control Unit

There are the following disadvantages of micro-programmed control are as follows –

- Adaptability is obtained at more cost.
- It is slower than a hardwired control unit.

CONTROL MEMORY:

A control memory is a part of the control unit. Any computer that involves micro-programmed control consists of two memories. They are the main memory and the control memory. Programs are usually stored in the main memory by the users. Whenever the programs change, the data is also modified in the main memory. They consist of machine instructions and data.

The control memory consists of microprograms that are fixed and cannot be modified frequently. They contain microinstructions that specify the internal control signals required to execute register micro-operations.

The machine instructions generate a chain of microinstructions in the control memory. Their function is to generate micro-operations that can fetch instructions from the main memory,

compute the effective address, execute the operation, and return control to fetch phase an continue the cycle.

Here, the control is presumed to be a Read-Only Memory (ROM), where all the control information is stored permanently. ROM provides the address of the microinstruction. The other register, that is, the control data register stores the microinstruction that is read from the memory. It consists of a control word that holds one or more micro-operations for the data processor.

The next address must be computed once this operation is completed. It is computed in the next address generator. Then, it is sent to the control address register to be read. The next address generator is also known as the micro-program sequencer. Based on the inputs to a sequencer, it determines the address of the next microinstruction. The microinstructions can be specified in several ways.

ADDRESS SEQUENCING:

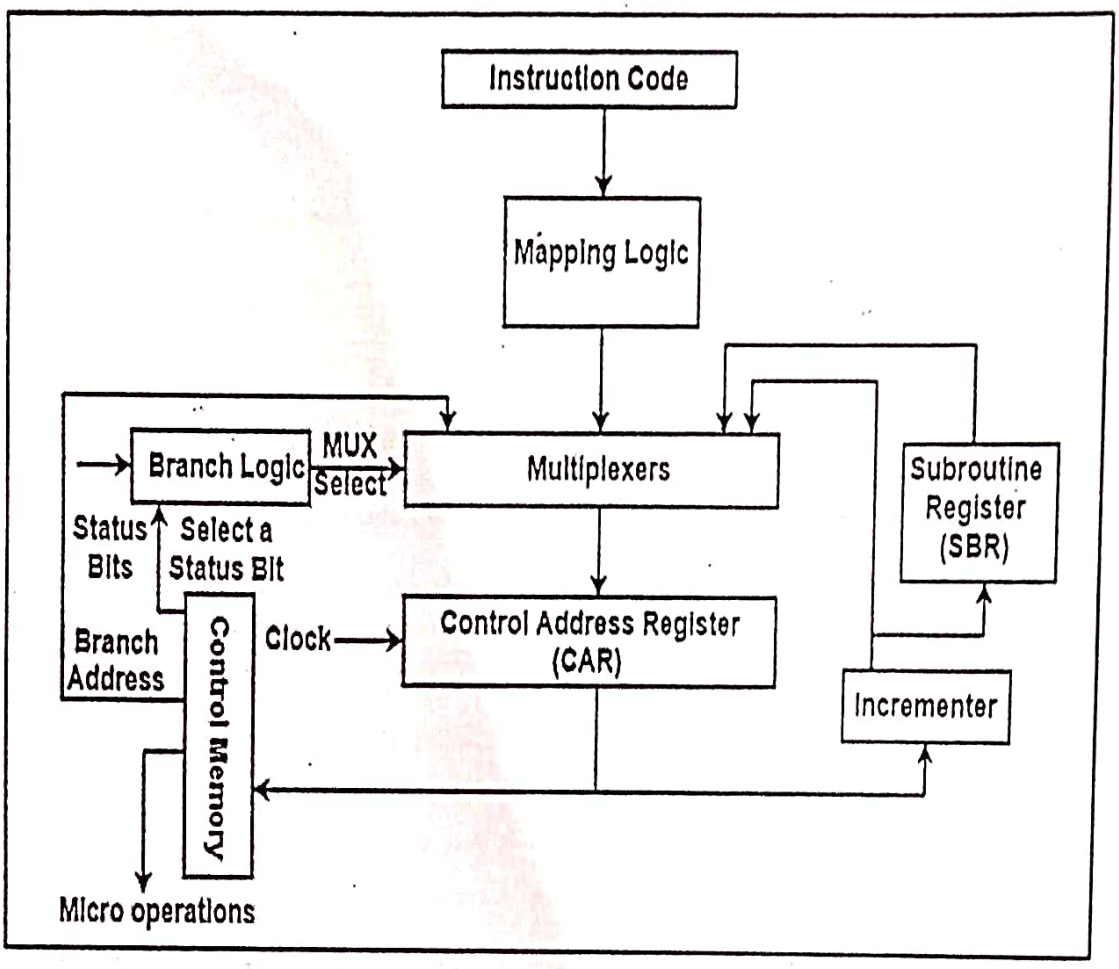
Microinstructions are saved in control memory in groups. These groups describe routines. Each computer instruction has its micro-program routine that can create micro-operations. These micro-operations can execute instructions. The hardware consists of controls for the address sequencing of the microinstructions of a similar routine. They also branch the microinstructions.

There are the following phases that the control has while implementing a computer instruction –

- When power is turned on, and address is initially loaded into the control address register. (This is the address of the first microinstruction).
- The control address register is incremented resulting in sequencing the fetch routine.
- After the fetch routine, the instruction is present in the IR of the computer.
- Next, the control memory retrieves the effective address of the operand from the routine.
- Thus, the mapping process appears from the instruction bits to a control memory address.
- It depends on the Opcodes of instruction the microinstructions of the processor registers are generated. Each of these microinstructions has a separate micro-program routine stored.
- The instruction code bits are changed into the address where the routine is placed and is known as the mapping process. A mapping process transforms the microinstruction into a control memory address.
- Next, subroutines are called and processes are returned.

- After the completion of the routine, the control address register is incremented to sequence the instruction is implemented.

Selection of Address for Control Memory



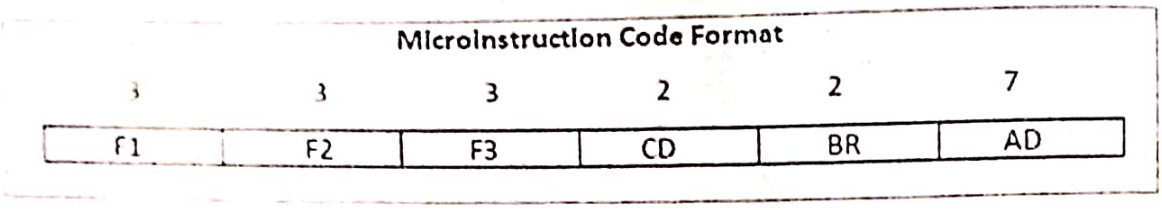
The diagram shows the block diagram of a control memory and its associated hardware to support in choosing the next microinstruction. The microinstruction present in the control memory has a set of bits that facilitate to start off the micro-operations in registers.

There are four different directions are showed in the figure from where the control address register recovers its address. The CAR is incremented by the incrementer and selects the next instruction. In multiple fields of microinstruction, the branching address can be determined to result in branching.

It can specify the condition of the status bits of microinstruction, conditional branching can be applied. A mapping logic circuit can share an external address. A special register can save the return address so that when the micro-program needs to return from the subroutine, it can need the value from the unique register.

MICRO INSTRUCTION FORMAT

A microinstruction format includes 20 bits in total. They are divided into four elements as displayed in the figure.



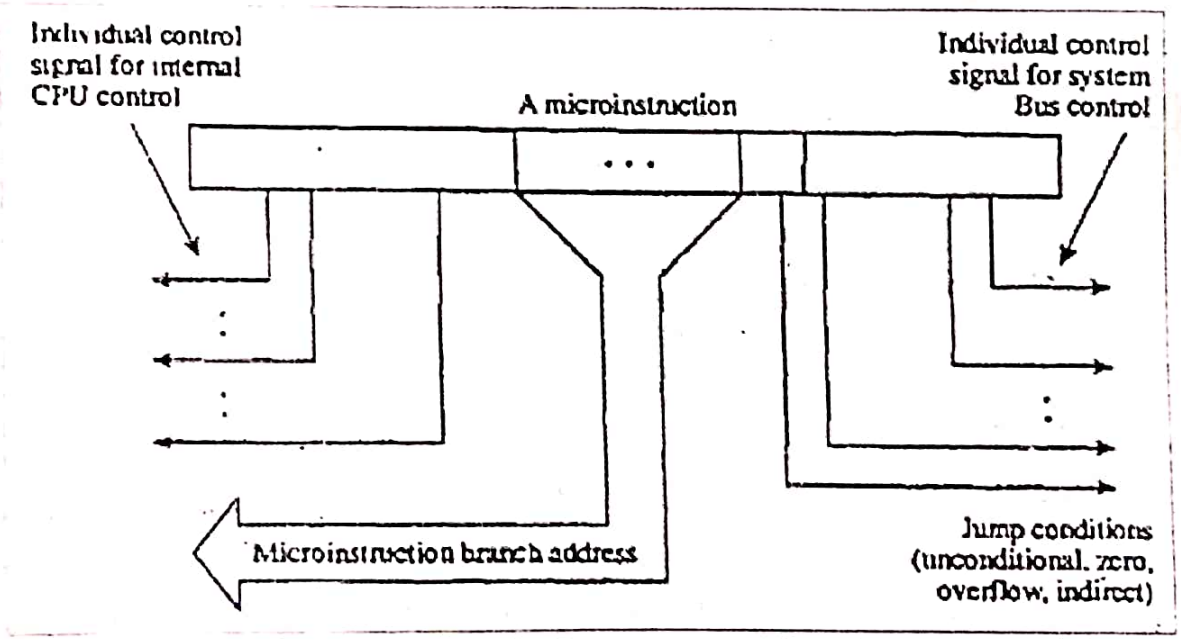
F1, F2, F3 are the micro-operation fields. They determine micro-operations for the computer.

CD is the condition for branching. They choose the status bit conditions.

BR is the branch field. It determines the type of branch.

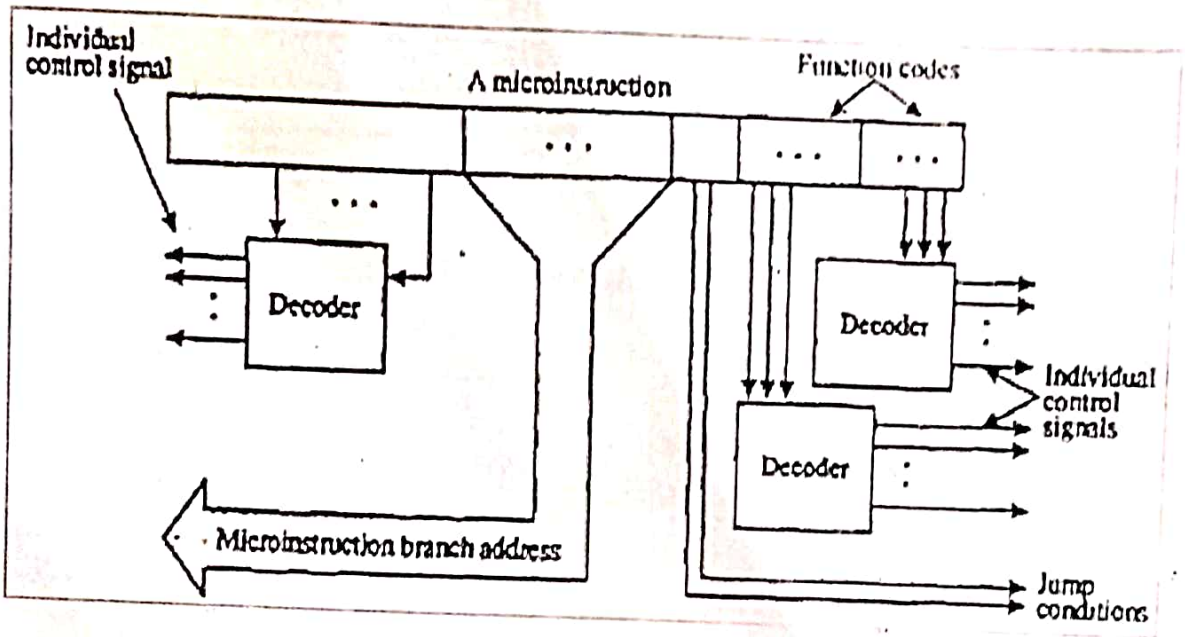
AD is the address field. It includes the address field whose length is 7 bits.

The two widely used formats employed for micro-instructions are vertical and horizontal. In horizontal micro-instruction every bit of micro-instruction signifies a control signal that directly controls a single bus line or occasionally a gate in the machine. Though the length of such a micro-instruction can be hundreds of bits. A typical horizontal micro-instruction with its related fields is demonstrated in Figure below.



(a) Horizontal Micro-instruction

In a vertical micro-instruction several similar control signals can be encoded in a few micro-instruction bits. For example, for 16 ALU operations that may need 16 individual control bits in horizontal micro-instruction only 4 encoded bits are required in vertical micro-instruction. In the same way in a vertical micro-instruction only 3 bits are required to select one of eight registers. Though these encoded bits need to be passed from the respective decoders to get the individual control signals.

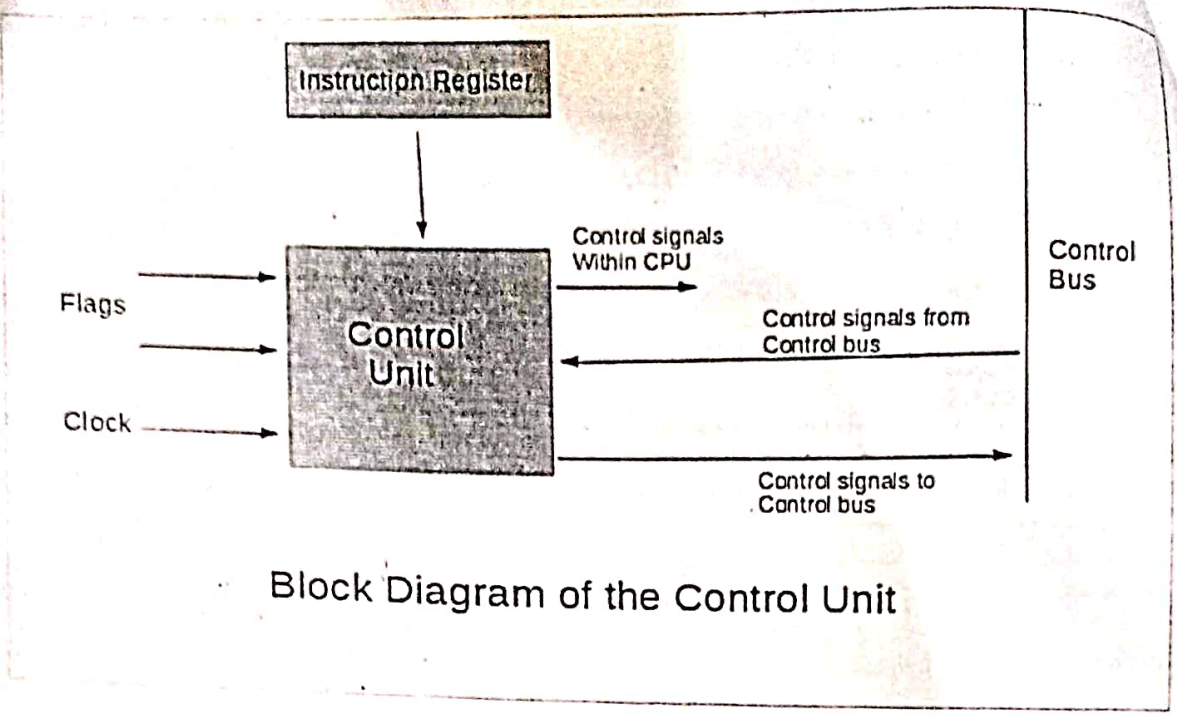


(b) Vertical Micro-instructions

DESIGN OF CONTROL UNIT

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

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



Block Diagram of the Control Unit

Functions of the Control Unit –

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

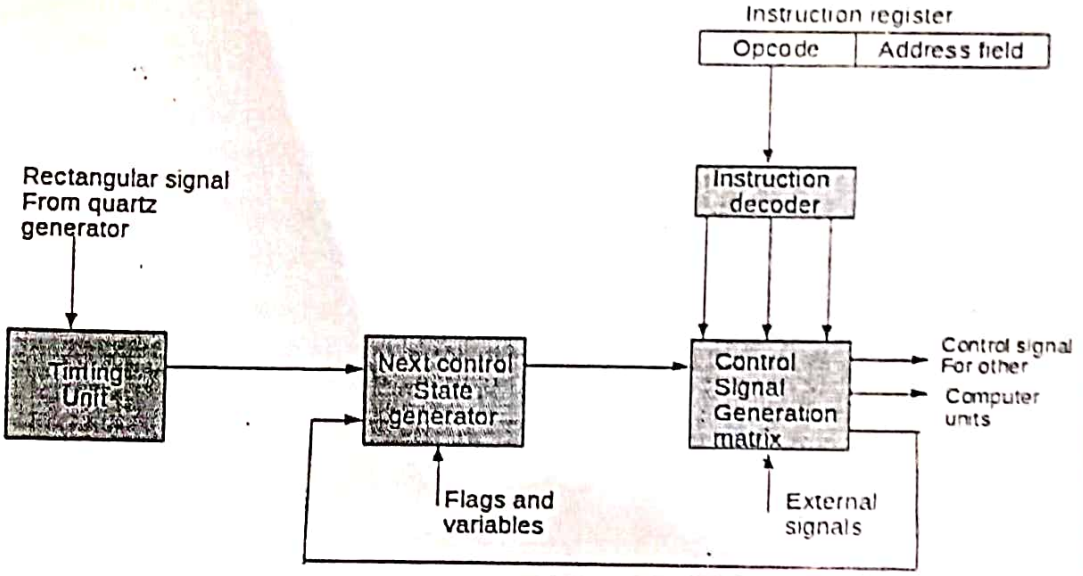
Types of Control Unit –

There are two types of control units: Hardwired control unit and Micro-programmable control unit.

1. **Hardwired Control Unit –**

   In the Hardwired control unit, the control signals that are important for instruction execution control are generated by specially designed hardware logical circuits, in which we can not modify the signal generation method without physical change of the circuit structure. The operation code of an instruction contains the basic data for control signal generation. In the instruction decoder, the operation code is decoded. The instruction decoder constitutes a set of many decoders that decode different fields of the instruction opcode.

As a result, few output lines going out from the instruction decoder obtains active signal values. These output lines are connected to the inputs of the matrix that generates control signals for executive units of the computer. This matrix implements logical combinations of the decoded signals from the instruction opcode with the outputs from the matrix that generates signals representing consecutive control unit states and with signals coming from the outside of the processor, e.g. interrupt signals. The matrices are built in a similar way as a programmable logic arrays.



Block diagram of a hardwired control unit of a computer

Control signals for an instruction execution have to be generated not in a single time point but during the entire time interval that corresponds to the instruction execution cycle. Following the structure of this cycle, the suitable sequence of internal states is organized in the control unit.

A number of signals generated by the control signal generator matrix are sent back to inputs of the next control state generator matrix. This matrix combines these signals with the timing signals, which are generated by the timing unit based on the rectangular patterns usually supplied by the quartz generator. When a new instruction arrives at the control unit, the control units is in the initial state of new instruction fetching. Instruction decoding allows the control unit enters the first state relating execution of the new instruction, which lasts as long as the timing signals and other input signals as flags and state information of the computer remain unaltered. A change of any of the earlier mentioned signals stimulates the change of the control unit state.

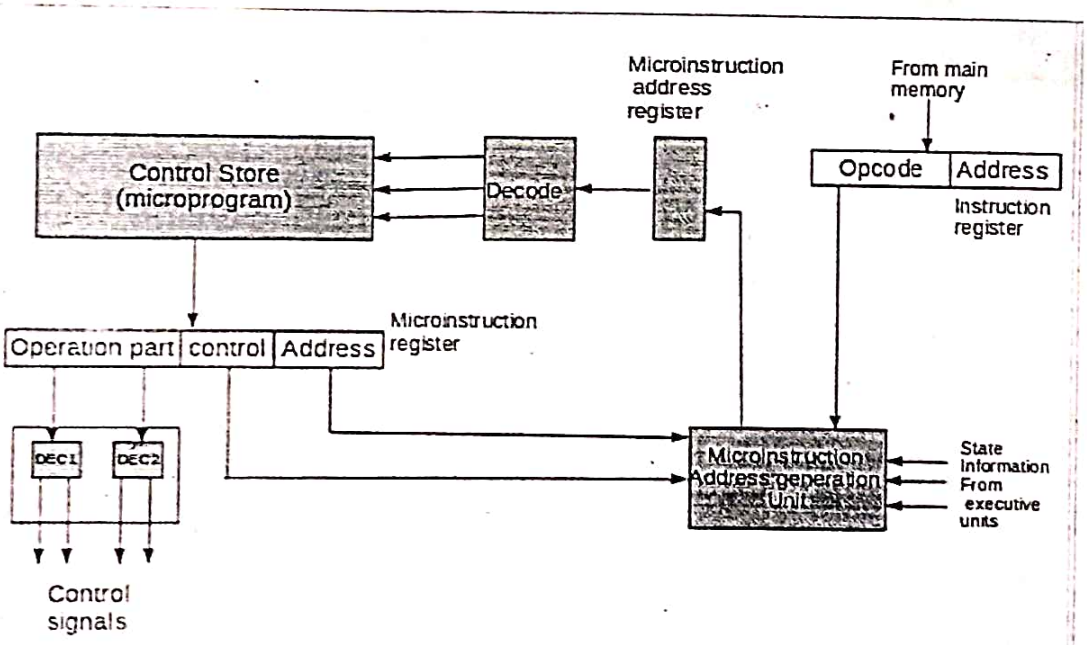
This causes that a new respective input is generated for the control signal generator matrix. When an external signal appears, (e.g. an interrupt) the control unit takes entry into a next control state that is the state concerned with the reaction to this external signal (e.g. interrupt processing). The values of flags and state variables of the computer are used to select suitable states for the instruction execution cycle.

Micro programmable control unit –

The fundamental difference between these unit structures and the structure of the hardwired control unit is the existence of the control store that is used for storing words containing encoded control signals mandatory for instruction execution. In micro-programmed control units, subsequent instruction words are fetched into the instruction register in a normal way. However, the operation code of each instruction is not directly decoded to enable immediate control signal generation but it comprises the initial address of a micro-program contained in the control store.

- **With a single-level control store:**

  In this, the instruction opcode from the instruction register is sent to the control store address register. Based on this address, the first microinstruction of a micro-program that interprets execution of this instruction is read to the microinstruction register. This microinstruction contains in its operation part encoded control signals, normally as few bit fields. In a set microinstruction field decoder, the fields are decoded. The microinstruction also contains the address of the next microinstruction of the given instruction micro-program and a control field used to control activities of the microinstruction address generator.

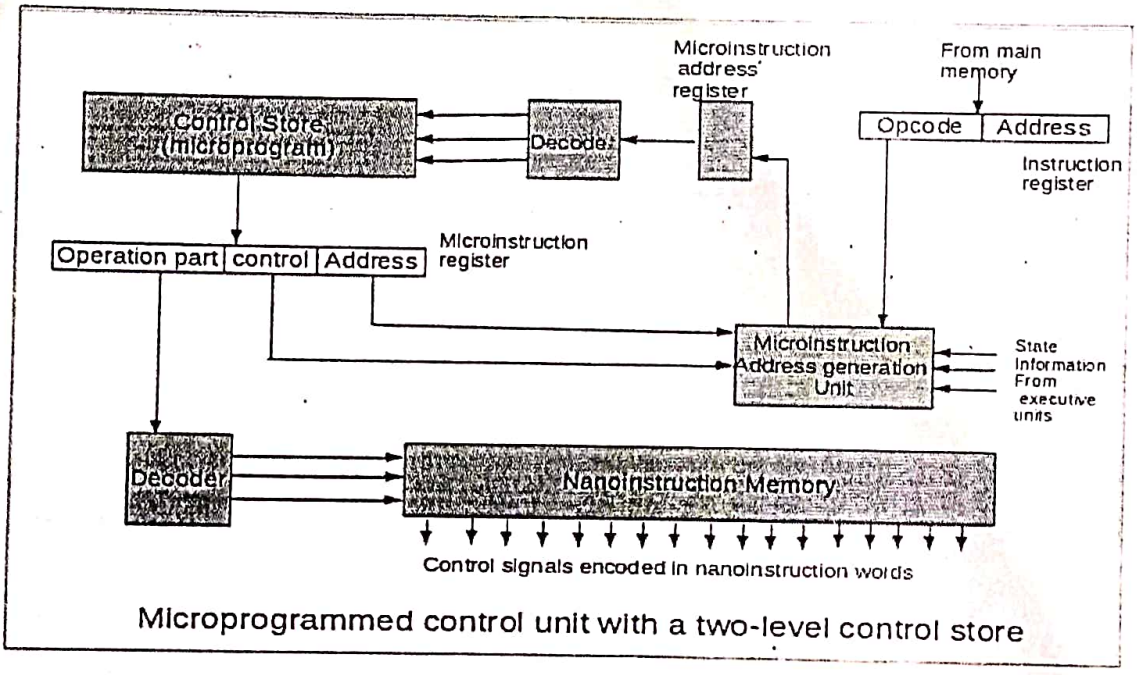


Microprogrammed control unit with a single level control store

The last mentioned field decides the addressing mode (addressing operation) to be applied to the address embedded in the ongoing microinstruction. In microinstructions along with conditional addressing mode, this address is refined by using the processor condition flags that represent the status of computations in the current program. The last microinstruction in the instruction of the given micro-program is the microinstruction that fetches the next instruction from the main memory to the instruction register.

- **With a two-level control store:**

In this, in a control unit with a two-level control store, besides the control memory for microinstructions, a Nano-instruction memory is included. In such a control unit, microinstructions do not contain encoded control signals. The operation part of microinstructions contains the address of the word in the Nano-instruction memory, which contains encoded control signals. The Nano-instruction memory contains all combinations of control signals that appear in micro-programs that interpret the complete instruction set of a given computer, written once in the form of Nano-instructions.

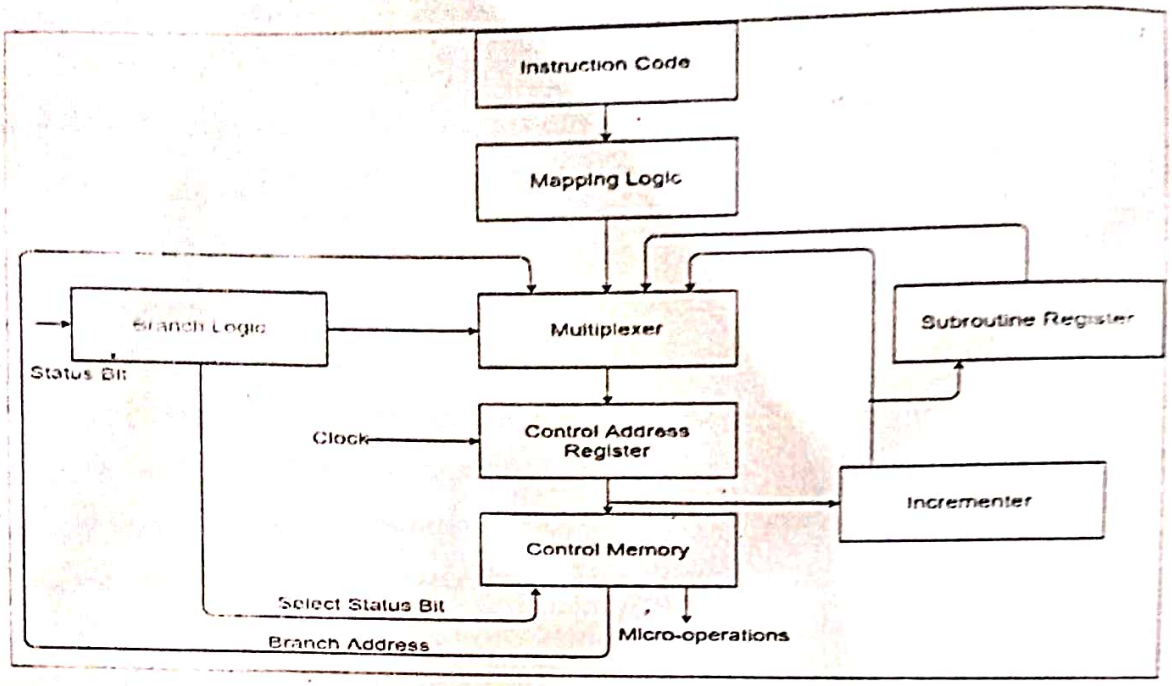


Microprogrammed control unit with a two-level control store

MICRO PROGRAM SEQUENCER

Micro Program Sequencer is a combination of all hardware for selecting the next micro-instruction address. The micro-instruction in control memory contains a set of bits to initiate micro operations in computer registers and other bits to specify the method by which the address is obtained.

Implementation of Micro Program Sequencer –



- **Control Address Register(CAR) :**

  Control address register receives the address from four different paths. For receiving the addresses from four different paths, Multiplexer is used.

- **Multiplexer :**

  Multiplexer is a combinational circuit which contains many data inputs and single data output depending on control or select inputs.

- **Branching :**

  Branching is achieved by specifying the branch address in one of the fields of the micro instruction. Conditional branching is obtained by using part of the micro-instruction to select a specific status bit in order to determine its condition.

- **Mapping Logic :**

  An external address is transferred into control memory via a mapping logic circuit.

- **Incrementer :**

  Incrementer increments the content of the control address register by one, to select the next micro-instruction in sequence.

- **Subroutine Register (SBR) :**

  The return address for a subroutine is stored in a special register called Subroutine Register whose value is then used when the micro-program wishes to return from the subroutine.

- **Control Memory :**

  Control memory is a type of memory which contains addressable storage registers. Data is

temporarily stored in control memory. Control memory can be accessed quicker than main memory.

HARDWIRED VS MICROPROGRAMMED CONTROL UNIT

To execute an instruction, there are two types of control unit.

1. **Hardwired control units** are generally faster than micro-programmed designs. In hardwired control, we saw how all the control signals required inside the CPU can be generated using a state counter and a PLA (Programmable Logic Array) circuit.
2. **A micro-programmed control unit** is a relatively simple logic circuit that is capable of (1) sequencing through microinstructions and (2) generating control signals to execute each microinstruction.

Hardwired Control Unit	Micro-programmed Control Unit
Hardwired control unit generates the control signals needed for the processor using logic circuits	Micro-programmed control unit generates the control signals with the help of micro instructions stored in control memory
Hardwired control unit is faster when compared to micro-programmed control unit as the required control signals are generated with the help of hardware's	This is slower than the other as micro instructions are used for generating signals here
Difficult to modify as the control signals that need to be generated are hard wired	Easy to modify as the modification need to be done only at the instruction level
More costlier as everything has to be realized in terms of logic gates	Less costlier than hardwired control as only micro instructions are used for generating control signals
It cannot handle complex instructions as the circuit design for it becomes complex	It can handle complex instructions
Only limited number of instructions are used due to the hardware implementation	Control signals for many instructions can be generated
Used in computer that makes use of Reduced Instruction Set Computers(RISC)	Used in computer that makes use of Complex Instruction Set Computers(CISC)