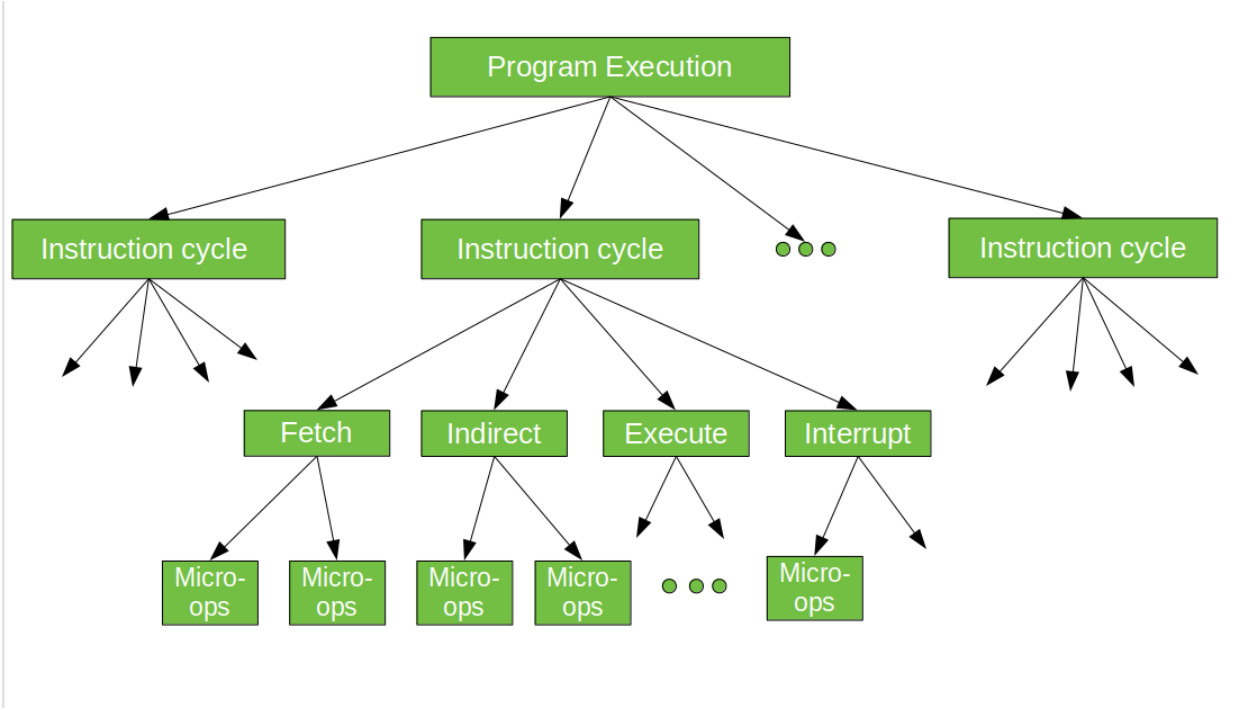
Control Unit

Micro-operation

In computer central processing units, **micro-operations** (also known as micro-ops) are the functional or atomic, operations of a processor. These are low level instructions used in some designs to implement complex machine instructions. They generally perform operations on data stored in one or more registers. They transfer data between registers or between external buses of the CPU, also performs arithmetic and logical operations on registers.
In executing a program, operation of a computer consists of a sequence of instruction cycles, with one machine instruction per cycle. Each instruction cycle is made up of a number of smaller units – *Fetch, Indirect, Execute and Interrupt cycles.* Each of these cycles involves series of steps, each of which involves the processor registers. These steps are referred as micro-operations. The prefix micro refers to the fact that each of the step is very simple and accomplishes very little. Figure below depicts the concept being discussed here.

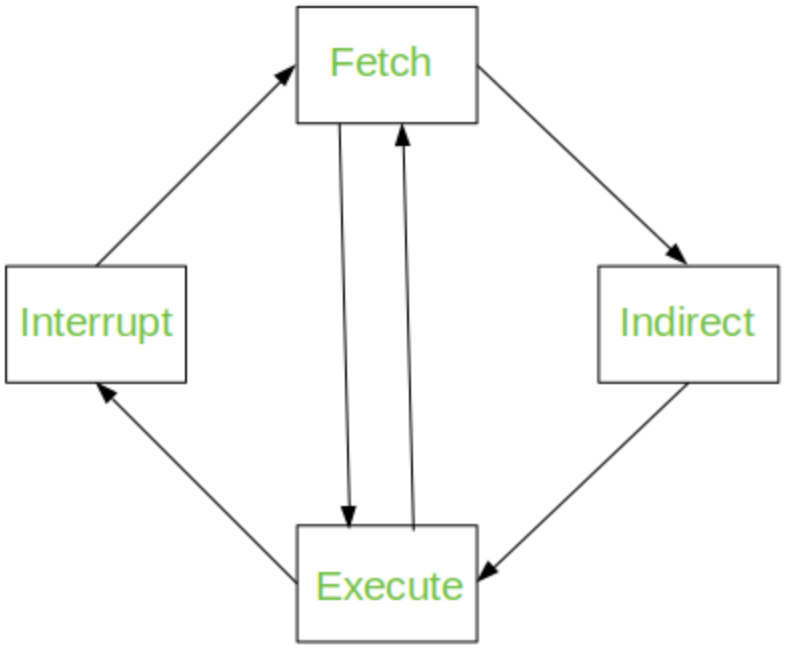


Registers Involved In Each Instruction Cycle:

- **Memory address registers(MAR)** : It is connected to the address lines of the system bus. It specifies the address in memory for a read or write operation.
- **Memory Buffer Register(MBR)** : It is connected to the data lines of the system bus. It contains the value to be stored in memory or the last value read from the memory.
- **Program Counter(PC)** : Holds the address of the next instruction to be fetched.
- **Instruction Register(IR)** : Holds the last instruction fetched.

The Instruction Cycle –

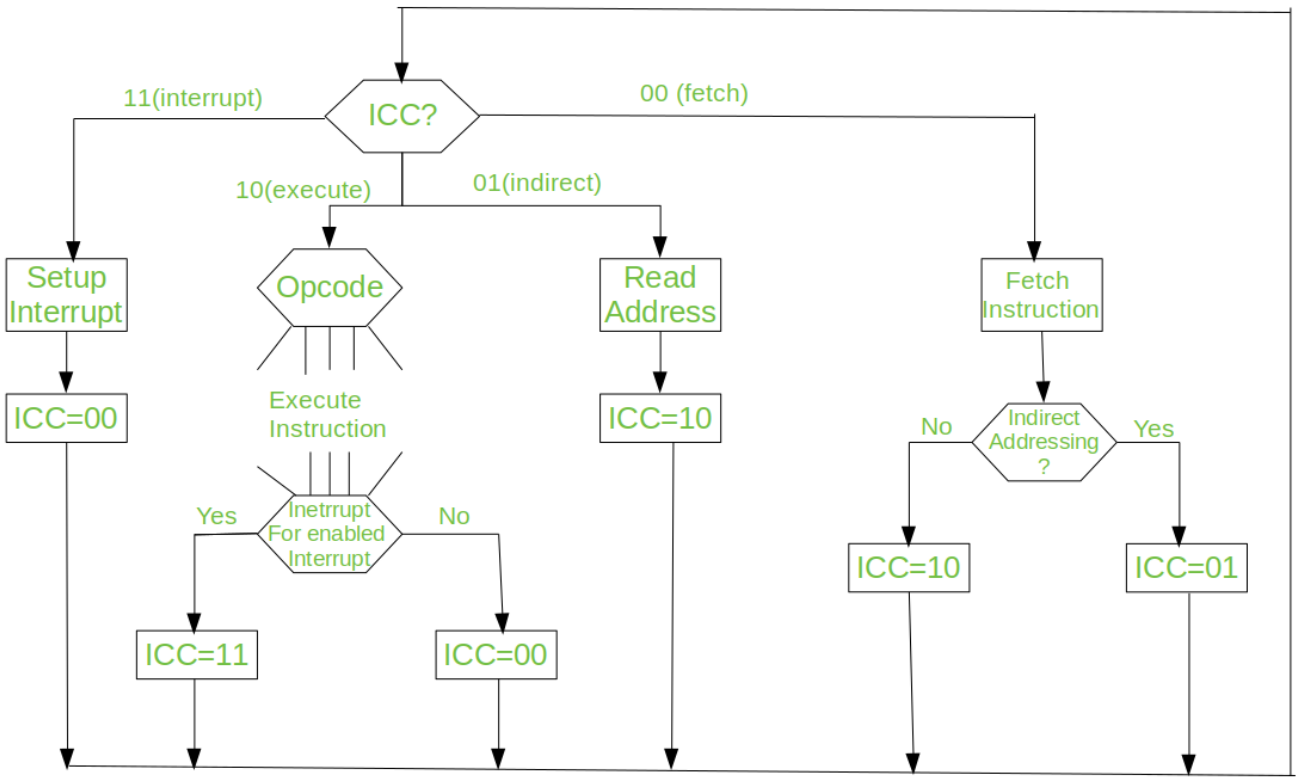
Each phase of Instruction Cycle can be decomposed into a sequence of elementary micro-operations. In



The Instruction Cycle

the above examples, there is one sequence each for the *Fetch, Indirect, Execute and Interrupt Cycles*.

The *Indirect Cycle* is always followed by the *Execute Cycle*. The *Interrupt Cycle* is always followed by the *Fetch Cycle*. For both fetch and execute cycles, the next cycle depends on the state of the system.



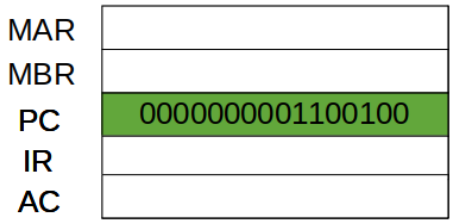
Flowchart for Instruction Cycle

We assumed a new 2-bit register called *Instruction Cycle Code* (ICC). The ICC designates the state of processor in terms of which portion of the cycle it is in:

00 : Fetch Cycle
01 : Indirect Cycle
10 : Execute Cycle
11 : Interrupt Cycle

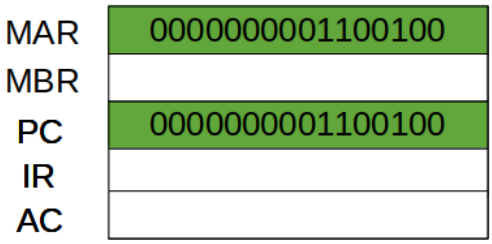
At the end of the each cycles, the ICC is set appropriately. The above flowchart of *Instruction Cycle* describes the complete sequence of micro-operations, depending only on the instruction sequence and the interrupt pattern (this is a simplified example). The operation of the processor is described as the performance of a sequence of micro-operation.
Different Instruction Cycles:

1. **The Fetch Cycle –**
   At the beginning of the fetch cycle, the address of the next instruction to be executed is in the *Program Counter*(PC).



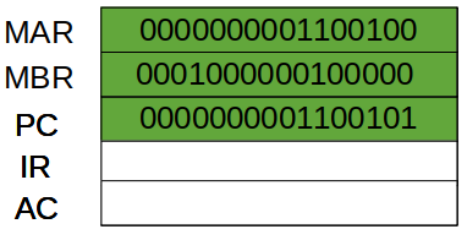
BEGINNING

Step 1: The address in the program counter is moved to the memory address register (MAR), as this is the only register which is connected to address lines of the system bus.



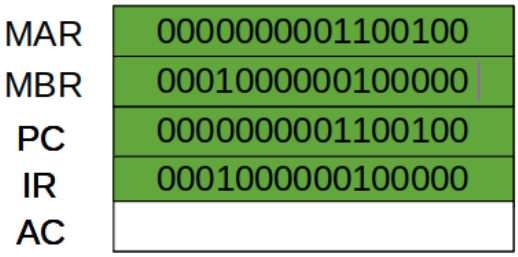
FIRST STEP

Step 2: The address in MAR is placed on the address bus, now the control unit issues a READ command on the control bus, and the result appears on the data bus and is then copied into the memory buffer register (MBR). Program counter is incremented by one, to get ready for the next instruction.(These two action can be performed simultaneously to save time)



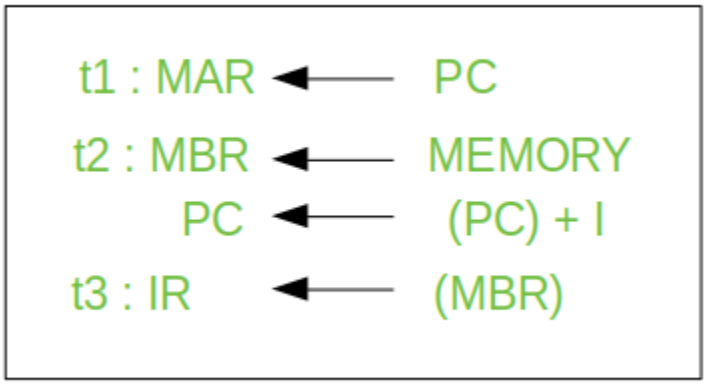
SECOND STEP

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



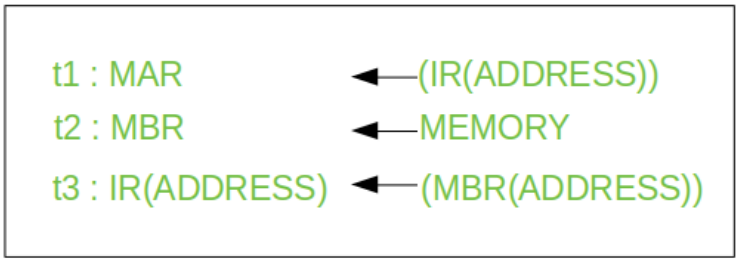
FIRST STEP

Thus, a simple *Fetch Cycle* consist of three steps and four micro-operation. Symbolically, we can write these sequence of events as follows:-



Here 'I' is the instruction length. The notations (t1, t2, t3) represents successive time units. We assume that a clock is available for timing purposes and it emits regularly spaced clock pulses. Each clock pulse defines a time unit. Thus, all time units are of equal duration. Each micro-operation can be performed within the time of a single time unit.
First time unit: Move the contents of the PC to MAR.
Second time unit: Move contents of memory location specified by MAR to MBR. Increment content of PC by I.
Third time unit: Move contents of MBR to IR.
Note: Second and third micro-operations both take place during the second time unit.

2. **The Indirect Cycles –**
   Once an instruction is fetched, the next step is to fetch source operands. *Source Operand* is being fetched by indirect addressing (it can be fetched by any addressing mode, here it's done by indirect addressing). Register-based operands need not be fetched. Once the opcode is executed, a similar process may be needed to store the result in main memory. Following *micro-operations* takes place:-



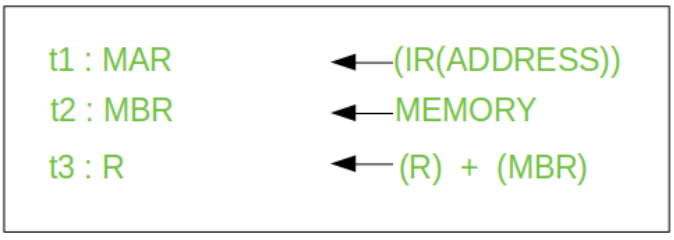
Step 1: The address field of the instruction is transferred to the MAR. This is used to fetch the address of the operand.
Step 2: The address field of the IR is updated from the MBR.(So that it now contains a direct addressing rather than indirect addressing)
Step 3: The IR is now in the state, as if indirect addressing has not been occurred.

Note: Now IR is ready for the execute cycle, but it skips that cycle for a moment to consider the *Interrupt Cycle.*

3. **The Execute Cycle**

The other three cycles (*Fetch, Indirect and Interrupt*) are simple and predictable. Each of them requires simple, small and fixed sequence of micro-operation. In each case same micro-operation are repeated each time around.
Execute Cycle is different from them. Like, for a machine with N different opcodes there are N different sequence of micro-operations that can occur.
Let's take an hypothetical example:
consider an add instruction:

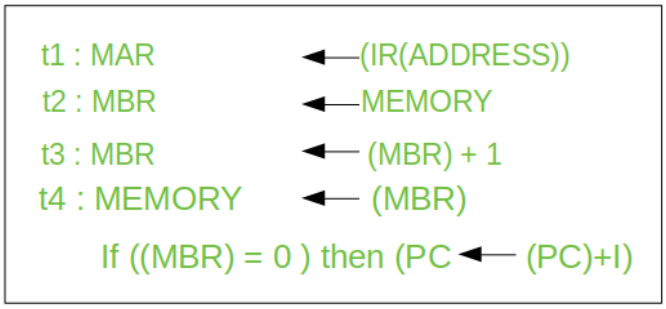
Here, this instruction adds the content of location X to register R. Corresponding micro-operation will be:-



We begin with the IR containing the ADD instruction.
Step 1: The address portion of IR is loaded into the MAR.
Step 2: The address field of the IR is updated from the MBR, so the reference memory location is read.
Step 3: Now, the contents of R and MBR are added by the ALU.

Lets take a complex example :-

Here, the content of location X is incremented by 1. If the result is 0, the next instruction will be skipped.
Corresponding sequence of micro-operation will be :-



Here, the PC is incremented if (MBR) = 0. This test (is MBR equal to zero or not) and action (PC is incremented by 1) can be implemented as one micro-operation.
Note : This test and action micro-operation can be performed during the same time unit during which the updated value MBR is stored back to memory.

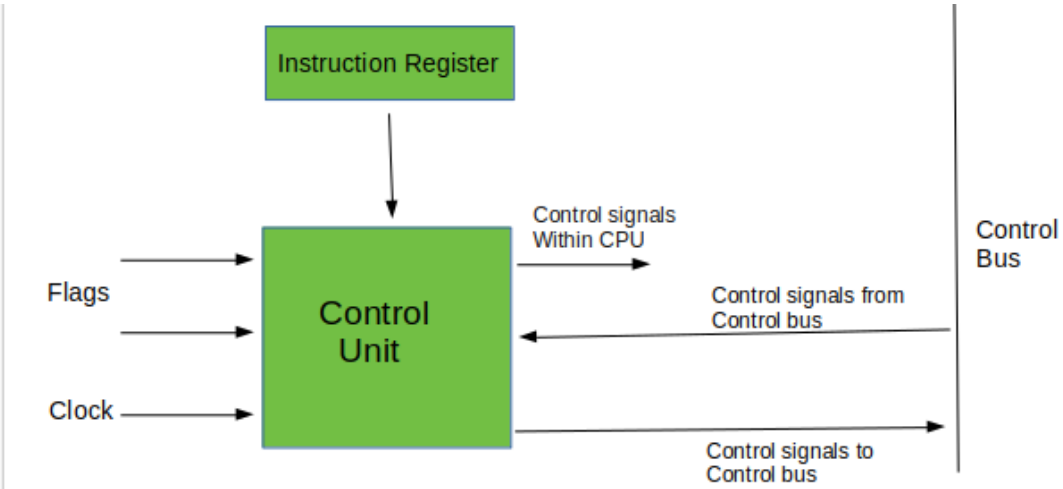
4. **The Interrupt Cycle**:
   At the completion of the Execute Cycle, a test is made to determine whether any enabled interrupt has occurred or not. If an enabled interrupt has occurred then Interrupt Cycle occurs. The nature of this cycle varies greatly from one machine to another.
   Let's take a sequence of micro-operation:-

Step 1: Contents of the PC is transferred to the MBR, so that they can be saved for return.
Step 2: MAR is loaded with the address at which the contents of the PC are to be saved.
PC is loaded with the address of the start of the interrupt-processing routine.
Step 3: MBR, containing the old value of PC, is stored in memory.

Note: In step 2, two actions are implemented as one micro-operation. However, most processor provide multiple types of interrupts, it may take one or more micro-operation to obtain the save address and the routine address before they are transferred to the MAR and PC respectively.

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.



Block Diagram of the Control Unit

Inputs of Control Unit

a. Clock: One or set of parallel micro-instructions per clock cycle.
b. Instruction Register
   - Op-code for current instruction

- Determines which micro-instructions are performed.

c. Flag
   - State of CPU
   - Results of previous operations.

d. Control signals from control bus
   - Interrupts
   - Acknowledgements

Outputs of Control Unit

a. Within CPU
   - Cause data movements
   - Activate specific functions

b. Via control bus
   - To memory
   - To I/O modules

Functions of the Control Unit

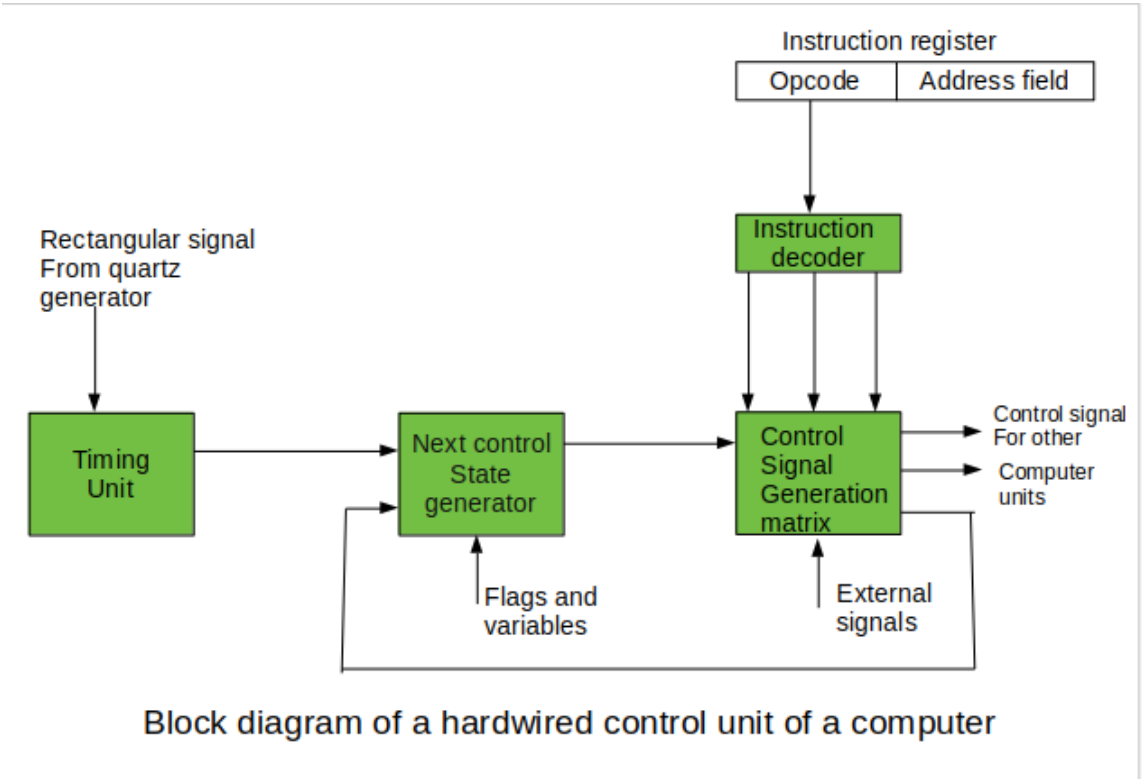
1. It coordinates the sequence of data movements into, out of, and between a processor's many sub-units.
2. It interprets instructions.
3. It controls data flow inside the processor.
4. It receives external instructions or commands to which it converts to sequence of control signals.
5. It controls many 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 Microprogrammable 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.

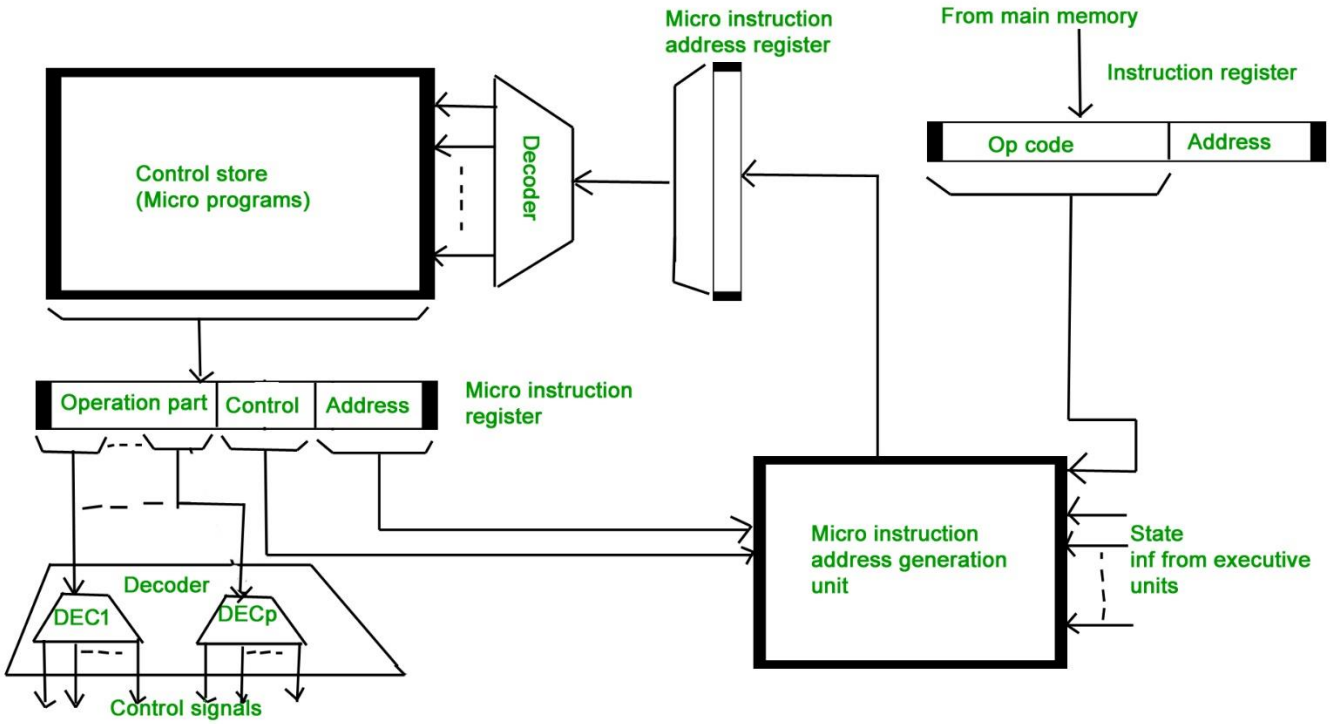
The last states in the cycle are control states that commence fetching the next instruction of the program: sending the program counter content to the main memory address buffer register and next, reading the instruction word to the instruction register of computer. When the ongoing

instruction is the stop instruction that ends program execution, the control unit enters an operating system state, in which it waits for a next user directive.

2. **Micro-programmed Control Unit**
    - The control signals associated with operations are stored in special memory units inaccessible by the programmer as Control Words.
    - Control signals are generated by a program are similar to machine language programs.
    - Micro-programmed control unit is slower in speed because of the time it takes to fetch microinstructions from the control memory.

Some Important Terms –

1. **Control Word :** A control word is a word whose individual bits represent various control signals.
2. **Micro-routine :** A sequence of control words corresponding to the control sequence of a machine instruction constitutes the micro-routine for that instruction.
3. **Micro-instruction :** Individual control words in this micro-routine are referred to as microinstructions.
4. **Micro-program :** A sequence of micro-instructions is called a micro-program, which is stored in a ROM or RAM called a Control Memory (CM).
5. **Control Store :** the micro-routines for all instructions in the instruction set of a computer are stored in a special memory called the Control Store.



Types of Micro-programmed Control Unit – Based on the type of Control Word stored in the Control Memory (CM), it is classified into two types :

1. Horizontal Micro-programmed control Unit :

The control signals are represented in the decoded binary format that is 1 bit/CS. Example: If 53 Control

signals are present in the processor than 53 bits are required. More than 1 control signal can be enabled at a time.

- It supports longer control word.
- It is used in parallel processing applications.
- It allows higher degree of parallelism. If degree is n, n CS are enabled at a time.
- It requires no additional hardware(decoders). It means it is faster than Vertical Microprogrammed.
- It is more flexible than vertical microprogrammed

2. Vertical Micro-programmed control Unit :

The control signals re represented in the encoded binary format. For N control signals- Log2(N) bits are required.

- It supports shorter control words.
- It supports easy implementation of new conrol signals therefore it is more flexible.
- It allows low degree of parallelism i.e., degree of parallelism is either 0 or 1.
- Requires an additional hardware (decoders) to generate control signals, it implies it is slower than horizontal microprogrammed.
- It is less flexible than horizontal but more flexible than that of hardwired control unit.

Microinstruction Sequencing

The two basic tasks performed by a microprogrammed control unit are as follows:

- Microinstruction Sequencing : Get the next microinstruction from the control memory.
- Microinstruction Execution: Generate the control signals needed to execute the microinstruction.

In designing a control unit, these tasks must be considered together, because both affect the format of the microinstruction and the timing of the control unit. Two concerns are involved in the design of a microinstruction sequencing technique: the size of the microinstruction and the address-generation time. The first concern is obvious; the minimizing the size of the control memory reduces the cost of the component. The second concern is simply a desire to execute microinstructions as fast as possible. In executing a microprogram, the address of the next microinstruction to be executed is in one of the these categories:

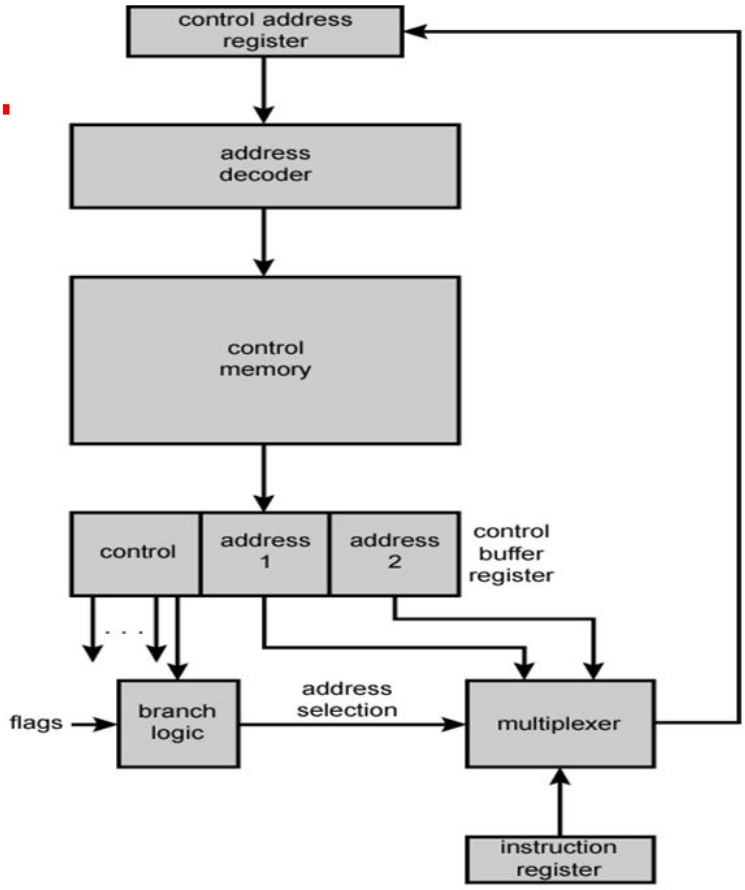
- Determined by instruction register
- Next sequential address
- Branch

The first category occurs only once per instruction cycle, just after an instruction is fetched. The second category is the most common in most designs. However, the design cannot be optimized just for sequential access. Branches, both conditional and unconditional, are a necessary part of a microprogram. Furthermore, microinstruction sequences tend to be short; one out of every three or four microinstructions could be a branch. Thus, it is important to design compact, time efficient techniques for microinstruction branching.

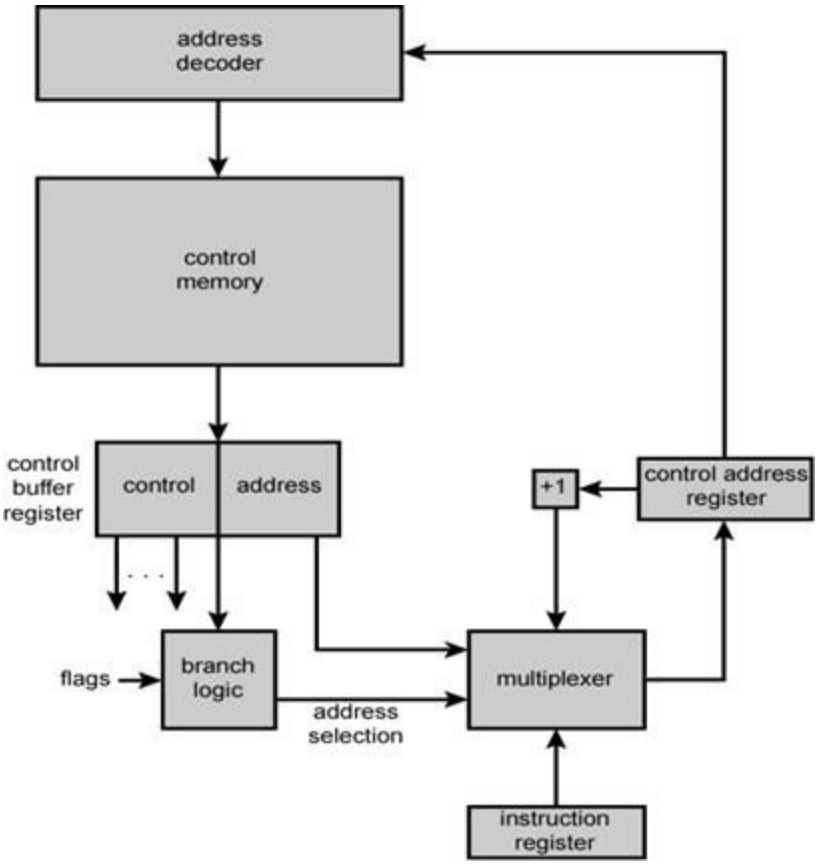
Sequencing Techniques

Based on the current microinstruction, condition flag, and the contents of the instruction register, a control memory address must be generated for the next microinstruction. A wide variety of techniques have been used. We can group them into three general categories. These categories are based on the format of the address information in the microinstruction.

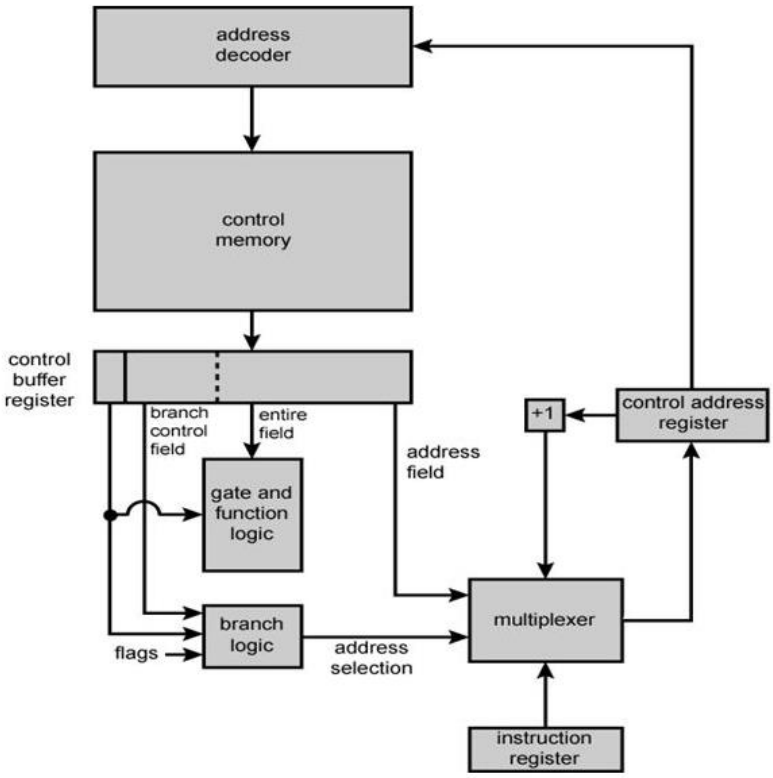
1. Two address field
   The simplest approach is to provide two address field in each microinstruction. A multiplexer is provided that search as the destination for both the address plus the instruction register. Based on an address selection i/p, the multiplexer transmit either the opcode or one of the two address CAR(Control address register). The CAR is subsequently decoded to provide the next micro-instruction address. The address selection signals are provided by a branch logic module whose o/p consist of flags plus bits from the control portion of micro-instruction. (figure below shows the branch control logic- two address format).



2. Single address field
   With single address field the option of next address are as follows: (i) address field, (ii)instruction filed, (iii)next sequential address. The address selection signal determines which option is selected. This approach reduces the no. of address filed to one. However that the address filed often will not be used. Thus, there is some inefficiency in micro instruction coding scheme.



3. Variable address field
   It provides two entirely different micro instruction format one bit designed which format being used. In one format the remaining bits are used to activate control signal. In other format some bits drives the branches logic and remaining bits provides the address. With the 1st format, the next address is either the sequential address or an address derive from the instruction register. With the 2nd format, either conditional or unconditional branch being specified.



Microinstruction Execution

The micro instruction is the basic event on the micro programmed processor. Each cycle is made up of two parts: fetch & execute. The fetch portion is determined by the generation of micro instruction address. The effect of execution of a micro-instruction is to generate control signals. Some of these signals control point interval to the processor. The remaining signals go to the external control bus or other external interface. As an incidental function, the address of next microinstruction is determined. The sequencing logic module contains the logic to perform functions.

It generates address of next micro instruction, using as inputs: the instruction register, ALU flags, the control address register(for incrementing), and the control buffer register. Control buffer register provide an actual address, control bits or both. The module is driven by clock that determines the timing of micro instruction cycle. The control logic module generates control signals as a function of some of the bits in the micro instruction. The format and content of micro instruction determines complexity of control logic module.

HARDWIRED CONTROL UNIT	MICROPROGRAMMED CONTROL UNIT
Hardwired control unit generates the control signals needed for the processor using logic circuits	Micrprogrammed control unit generates the control signals with the help of micro instructions stored in control memory
Hardwired control unit is faster when compared to microprogrammed control unit as the required control signals are generated with the help of hardwares	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)