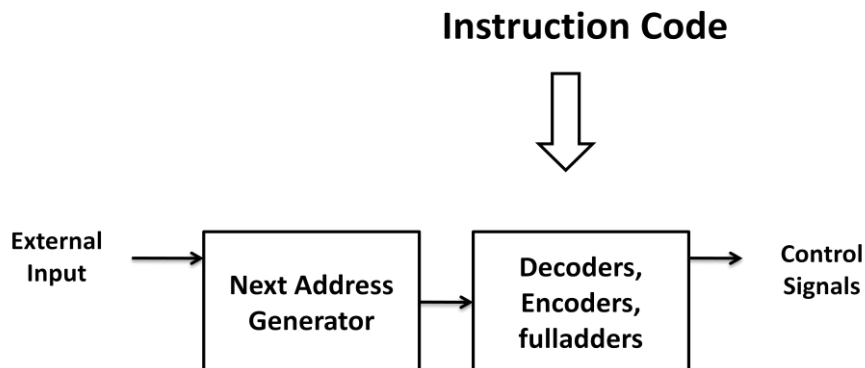## Microprogrammed Control

➢ The major functional parts in a digital computer are Central Processing Unit (CPU), Memory, and Input-output.

➢ The main digital hardware functional units of CPU are control unit, arithmetic and logic unit, and registers.

➢ The function of control unit in a digital computer is to initiate sequence of micro operations.

➢ Two methods are implementing control unit are hardwired control and microprogrammed control.

Hardwired Control Unit –

The control hardware can be viewed as a state machine that changes from one state to another in every clock cycle, depending on the contents of the instruction register, the condition codes and the external inputs. The outputs of the state machine are the control signals. The sequence of the operation carried out by this machine is determined by the wiring of the logic elements and hence named as "hardwired".

- Fixed logic circuits that correspond directly to the Boolean expressions are used to generate the control signals.
- Hardwired control is faster than micro-programmed control.
- A controller that uses this approach can operate at high speed.

Instruction Code

Micro-programmed Control Unit –

- The control signals associated with operations are stored in special memory units inaccessible by the programmer as Control Words.
- Controls 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.

Instruction Code
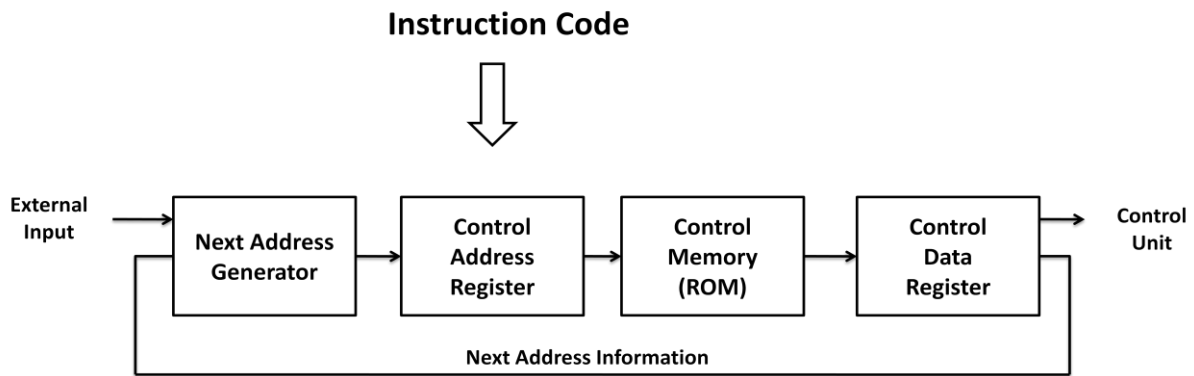
Control Memory:
Control Memory is the storage in the microprogrammed control unit to store the microprogram.

Control Word:
The control variables at any given time can be represented by a control word string of 1 's and 0's called a control word.

- The control memory address register specifies the address of the microinstruction, and the control data register holds the microinstruction read from memory.
- The microinstruction contains a control word that specifies one or more microoperations for the data processor. Once these operations are executed, the control must determine the next address.
- The location of the next microinstruction may be the one next in sequence, or it may be located somewhere else in the control memory.
- While the microoperations are being executed, the next address is computed in the next address generator circuit and then transferred into the control address register to read the next microinstruction.
- Thus a microinstruction contains bits for initiating microoperations in the data processor part and bits that determine the address sequence for the control memory.
- The next address generator is sometimes called a *micro-program sequencer*, as it determines the address sequence that is read from control memory.
- The control data register holds the present microinstruction while the next address is computed and read from memory.
- The data register is sometimes called a *pipeline register*.
- It allows the execution of the microoperations specified by the control word simultaneously with the generation of the next microinstruction.
- This configuration requires a two-phase clock, with one clock applied to the address register and the other to the data register.
- The main advantage of the micro programmed control is the fact that once the hardware configuration is established; there should be no need for further hardware or wiring changes.
- If we want to establish a different control sequence for the system, all we need to do is specify a different set of microinstructions for control memory.

Address Sequencing

- Microinstructions are stored in control memory in groups, with each group specifying a *routine*.
- To appreciate the address sequencing in a micro-program control unit, let us specify the steps that the control must undergo during the execution of a single computer instruction.

Step-1:

- An initial address is loaded into the control address register when power is turned on in the computer.
- This address is usually the address of the first microinstruction that activates the instruction fetch routine.
- The fetch routine may be sequenced by incrementing the control address register through the rest of its microinstructions.
- At the end of the fetch routine, the instruction is in the instruction register of the computer.

Step-2:

- The control memory next must go through the routine that determines the effective address of the operand.
- A machine instruction may have bits that specify various addressing modes, such as indirect address and index registers.
- The effective address computation routine in control memory can be reached through a branch microinstruction, which is conditioned on the status of the mode bits of the instruction.
- When the effective address computation routine is completed, the address of the operand is available in the memory address register.

Step-3:

- The next step is to generate the microoperations that execute the instruction fetched from memory.
- The microoperation steps to be generated in processor registers depend on the operation code part of the instruction.
- Each instruction has its own micro-program routine stored in a given location of control memory.
- The transformation from the instruction code bits to an address in control memory where the routine is located is referred to as a *mapping* process.
- A mapping procedure is a rule that transforms the instruction code into a control memory address.

Step-4:

- Once the required routine is reached, the microinstructions that execute the instruction may be sequenced by incrementing the control address register.
- Micro-programs that employ subroutines will require an external register for storing the return address.
- Return addresses cannot be stored in ROM because the unit has no writing capability.
- When the execution of the instruction is completed, control must return to the fetch routine.

- This is accomplished by executing an unconditional branch microinstruction to the first address of the fetch routine.

In summary, the address sequencing capabilities required in a control memory are:
1. Incrementing of the control address register.
2. Unconditional branch or conditional branch, depending on status bit conditions.
3. A mapping process from the bits of the instruction to an address for control memory.
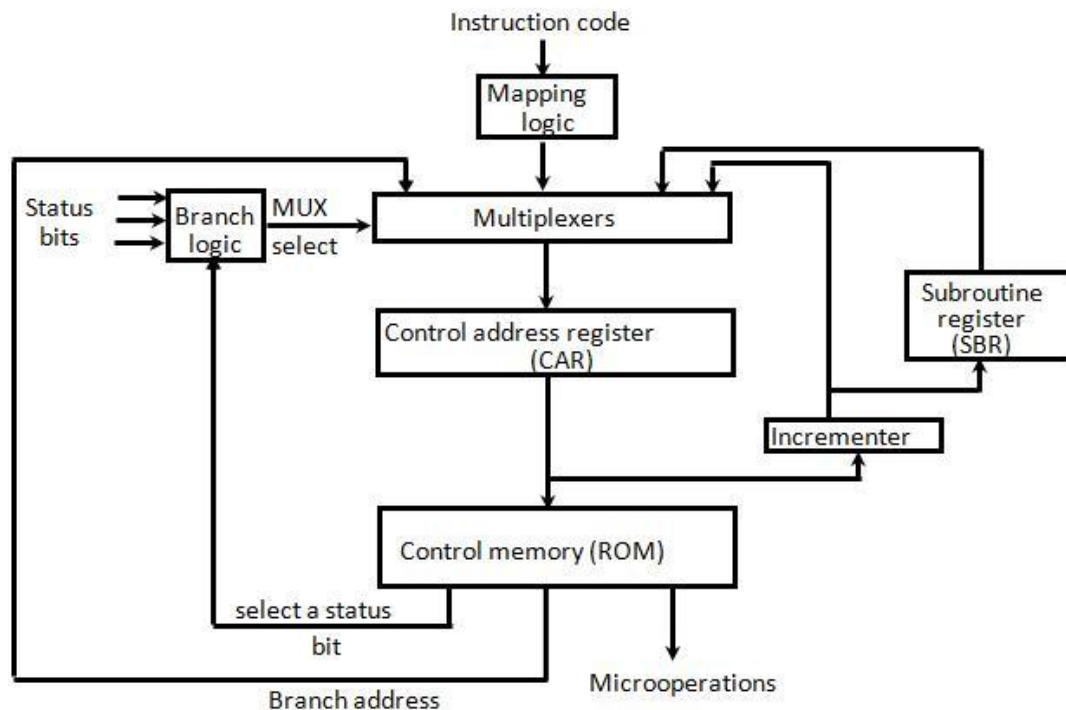4. A facility for subroutine call and return.

Figure: Selection of address for control memory

- Above figure shows a block diagram of a control memory and the associated hardware needed for selecting the next microinstruction address.
- The microinstruction in control memory contains a set of bits to initiate microoperations in computer registers and other bits to specify the method by which the next address is obtained.
- The diagram shows four different paths from which the control address register (CAR) receives the address.
- The incrementer increments the content of the control address register by one, to select the next microinstruction in sequence.
- Branching is achieved by specifying the branch address in one of the fields of the microinstruction.
- Conditional branching is obtained by using part of the microinstruction to select a specific status bit in order to determine its condition.
- An external address is transferred into control memory via a mapping logic circuit.
- The return address for a subroutine is stored in a special register whose value is then used when the micro-program wishes to return from the subroutine.
- The branch logic of figure provides decision-making capabilities in the control unit.

- The status conditions are special bits in the system that provide parameter information such as the carry-out of an adder, the sign bit of a number, the mode bits of an instruction, and input or output status conditions.
- The status bits, together with the field in the microinstruction that specifies a branch address, control the conditional branch decisions generated in the branch logic.
- A 1 output in the multiplexer generates a control signal to transfer the branch address from the microinstruction into the control address register.
- A 0 output in the multiplexer causes the address register to be incremented.

Mapping of an Instruction

- A special type of branch exists when a microinstruction specifies a branch to the first word in control memory where a microprogram routine for an instruction is located.
- The status bits for this type of branch are the bits in the operation code part of the instruction.
  For example, a computer with a simple instruction format as shown in figure 4.3 has an operation code of four bits which can specify up to 16 distinct instructions.
- Assume further that the control memory has 128 words, requiring an address of seven bits.
- One simple mapping process that converts the 4-bit operation code to a 7-bit address for control memory is shown in figure 4.3.
- This mapping consists of placing a 0 in the most significant bit of the address, transferring the four operation code bits, and clearing the two least significant bits of the control address register.
- This provides for each computer instruction a microprogram routine with a capacity of four microinstructions.
- If the routine needs more than four microinstructions, it can use addresses 1000000 through 1111111. If it uses fewer than four microinstructions, the unused memory locations would be available for other routines.
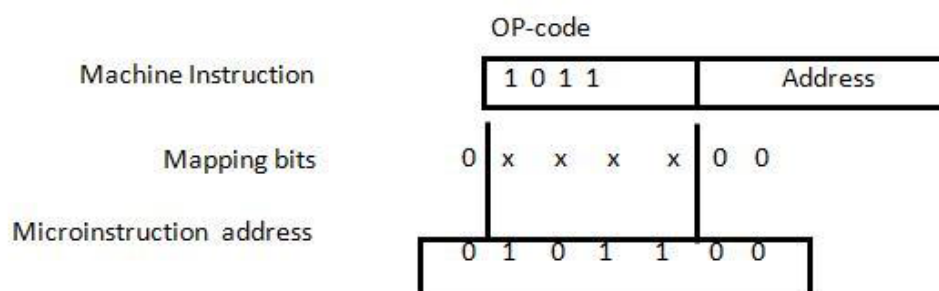
OP-code

| Machine Instruction | 1 0 1 1 | Address |

| Mapping bits | 0 | x | x | x | x | 0 0 |

Microinstruction address   0 1 0 1 1 0 0

Figure : Mapping from instruction code to microinstruction address

- One can extend this concept to a more general mapping rule by using a ROM to specify the mapping function.
- The contents of the mapping ROM give the bits for the control address register.
- In this way the microprogram routine that executes the instruction can be placed in any desired location in control memory.
- The mapping concept provides flexibility for adding instructions for control memory as the need arises.

Computer Hardware Configuration
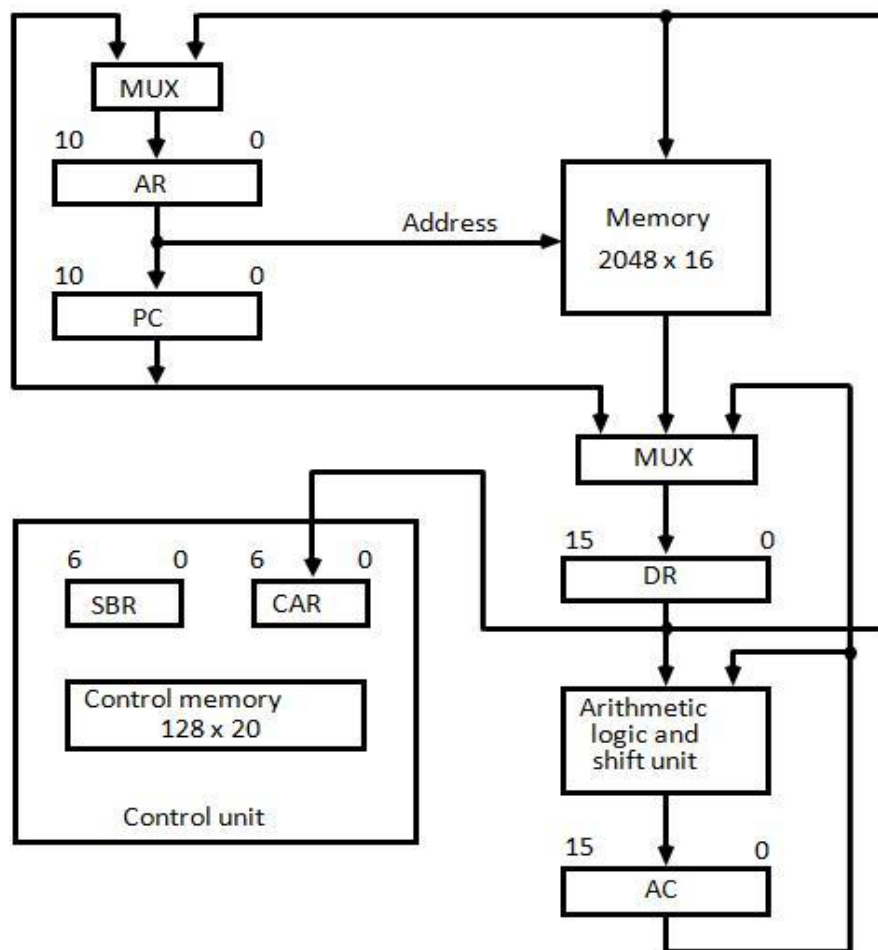
Figure: Computer Hardware Configuration

The block diagram of the computer is shown in Figure it consists of

1. Two memory units:

   Main memory → for storing instructions and data, and

   Control memory → for storing the microprogram.

2. Six Registers:

   Processor unit register: AC(accumulator), PC(Program Counter), AR(Address Register), DR(Data Register)

   Control unit register: CAR (Control Address Register), SBR(Subroutine Register)

3. Multiplexers:

   The transfer of information among the registers in the processor is done through multiplexers rather than a common bus.

4. ALU:

   The arithmetic, logic, and shift unit performs microoperations with data from AC and DR and places the result in AC.

- DR can receive information from AC, PC, or memory.
- AR can receive information from PC or DR.
- PC can receive information only from AR.
- Input data written to memory come from DR, and data read from memory can go only to DR.

Microinstruction Format

The microinstruction format for the control memory is shown in below figure. The 20 bits of the microinstruction are divided into four functional parts as follows:

1. The three fields F1, F2, and F3 specify microoperations for the computer. The microoperations are subdivided into three fields of three bits each. The three bits in each field are encoded to specify seven distinct microoperations. This gives a total of 21 microoperations.
2. The CD field selects status bit conditions.
3. The BR field specifies the type of branch to be used.
4. The AD field contains a branch address. The address field is seven bits wide, since the control memory has $128 = 2_7$ words.

3	3	3	2	2	7
F1	F2	F3	CD	BR	AD

F1, F2, F3: Microoperation fields
CD: Condition for branching
BR: Branch field
AD: Address field

Figure: Microinstruction Format

*   As an example, a microinstruction can specify two simultaneous microoperations from F2 and F3 and none from F1.

    DR←M[AR] with F2 = 100
    PC←PC + 1 with F3 = 101

*   The nine bits of the microoperation fields will then be 000 100 101.
*   The CD (condition) field consists of two bits which are encoded to specify four status bit conditions as listed in Table 1.

CD	Condition	Symbol	Comments
00	Always = 1	U	Unconditional branch
01	DR(15)	I	Indirect address bit
10	AC(15)	S	Sign bit of AC
11	AC = 0	Z	Zero value in AC

Table 1: Condition Field

*   The BR (branch) field consists of two bits. It is used, in conjunction with the address field AD, to choose the address of the next microinstruction shown in Table 2.

BR	Symbol	Function
00	JMP	CAR ← AD if condition = 1
		CAR ← CAR + 1 if condition = 0
01	CALL	CAR ← AD, SBR ← CAR + 1 if condition = 1
		CAR ← CAR + 1 if condition = 0
10	RET	CAR ← SBR (Return from subroutine)
11	MAP	CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0

Table 2: Branch Field

Symbolic Microinstruction:

- Each line of the assembly language microprogram defines a symbolic microinstruction.
- Each symbolic microinstruction is divided into five fields: label, microoperations, CD, BR, and AD. The fields specify the following Table 3.

1.	Label	The label field may be empty or it may specify a symbolic address. A label is terminated with a colon (:).
2.	Micro operations	It consists of one, two, or three symbols, separated by commas, from those defined in Table 5.3. There may be no more than one symbol from each F field. The NOP symbol is used when the microinstruction has no microoperations. This will be translated by the assembler to nine zeros.
3.	CD	The CD field has one of the letters U, I, S, or Z.
4.	BR	The BR field contains one of the four symbols defined in Table 5.2.
5.	AD	The AD field specifies a value for the address field of the microinstruction in one of three possible ways: i. With a symbolic address, this must also appear as a label. ii. With the symbol NEXT to designate the next address in sequence. iii. When the BR field contains a RET or MAP symbol, the AD field is left empty and is converted to seven zeros by the assembler.

Table 3: Symbolic Microinstruction

Micro programmed sequencer for a control memory:

Microprogram sequencer:

- The basic components of a microprogrammed control unit are the control memory and the circuits that select the next address.
- The address selection part is called a microprogram sequencer.
- A microprogram sequencer can be constructed with digital functions to suit a particular application.
- To guarantee a wide range of acceptability, an integrated circuit sequencer must provide an internal organization that can be adapted to a wide range of applications.
- The purpose of a microprogram sequencer is to present an address to the control memory so that a microinstruction may be read and executed.

- Commercial sequencers include within the unit an internal register stack used for temporary storage of addresses during microprogram looping and subroutine calls.
- Some sequencers provide an output register which can function as the address register for the control memory.
- The block diagram of the microprogram sequencer is shown in figure 4.6.
- There are two multiplexers in the circuit.
- The first multiplexer selects an address from one of four sources and routes it into a control address register CAR.
- The second multiplexer tests the value of a selected status bit and the result of the test is applied to an input logic circuit.
- The output from CAR provides the address for the control memory.
- The content of CAR is incremented and applied to one of the multiplexer inputs and to the subroutine registers SBR.
- The other three inputs to multiplexer 1 come from the address field of the present microinstruction, from the output of SBR, and from an external source that maps the instruction.
- Although the below figure shows a single subroutine register, a typical sequencer will have a register stack about four to eight levels deep. In this way, a number of subroutines can be active at the same time.
- The CD (condition) field of the microinstruction selects one of the status bits in the second multiplexer.
- If the bit selected is equal to 1, the T (test) variable is equal to 1; otherwise, it is equal to 0.
- The T value together with the two bits from the BR (branch) field goes to an input logic circuit.
- The input logic in a particular sequencer will determine the type of operations that are available in the unit.

Input Logic : Truth Table

BR	Input			MUX 1		Load SBR
	I1	I0	T	S1	S0	L
0 0	0	0	0	0	0	0
0 0	0	0	1	0	1	0
0 1	0	1	0	0	0	0
0 1	0	1	1	0	1	1
1 0	1	0	X	1	0	0
1 1	1	1	X	1	1	0

Boolean Function:

$$S_0 = I_0$$
$$S_1 = I_0 I_1 + I_0'T$$
$$L = I_0' I_1 T$$

- Typical sequencer operations are: increment, branch or jump, call and return from subroutine, load an external address, push or pop the stack, and other address sequencing operations.

- With three inputs, the sequencer can provide up to eight address sequencing operations.
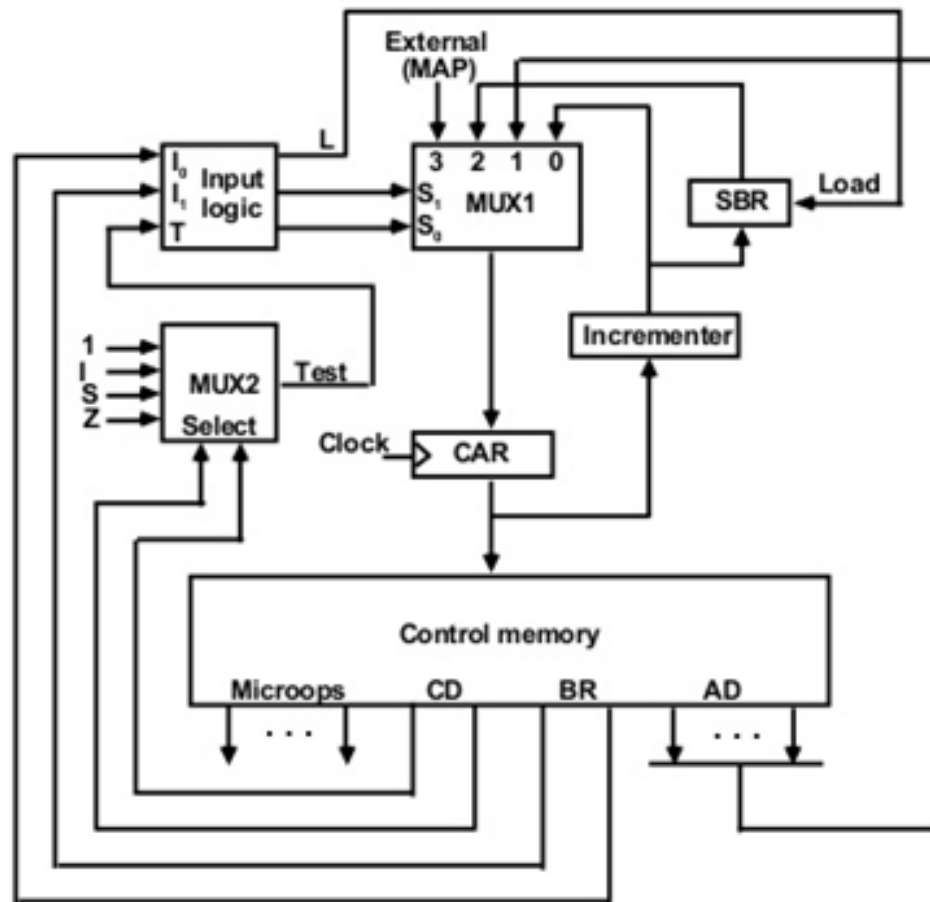- Some commercial sequencers have three or four inputs in addition to the T input and thus provide a wider range of operations.

Figure : Microprogram Sequencer for a control memory