Hardwired Control Unit VS. Microprogrammed Control Unit

The Control Unit is classified into two major categories:

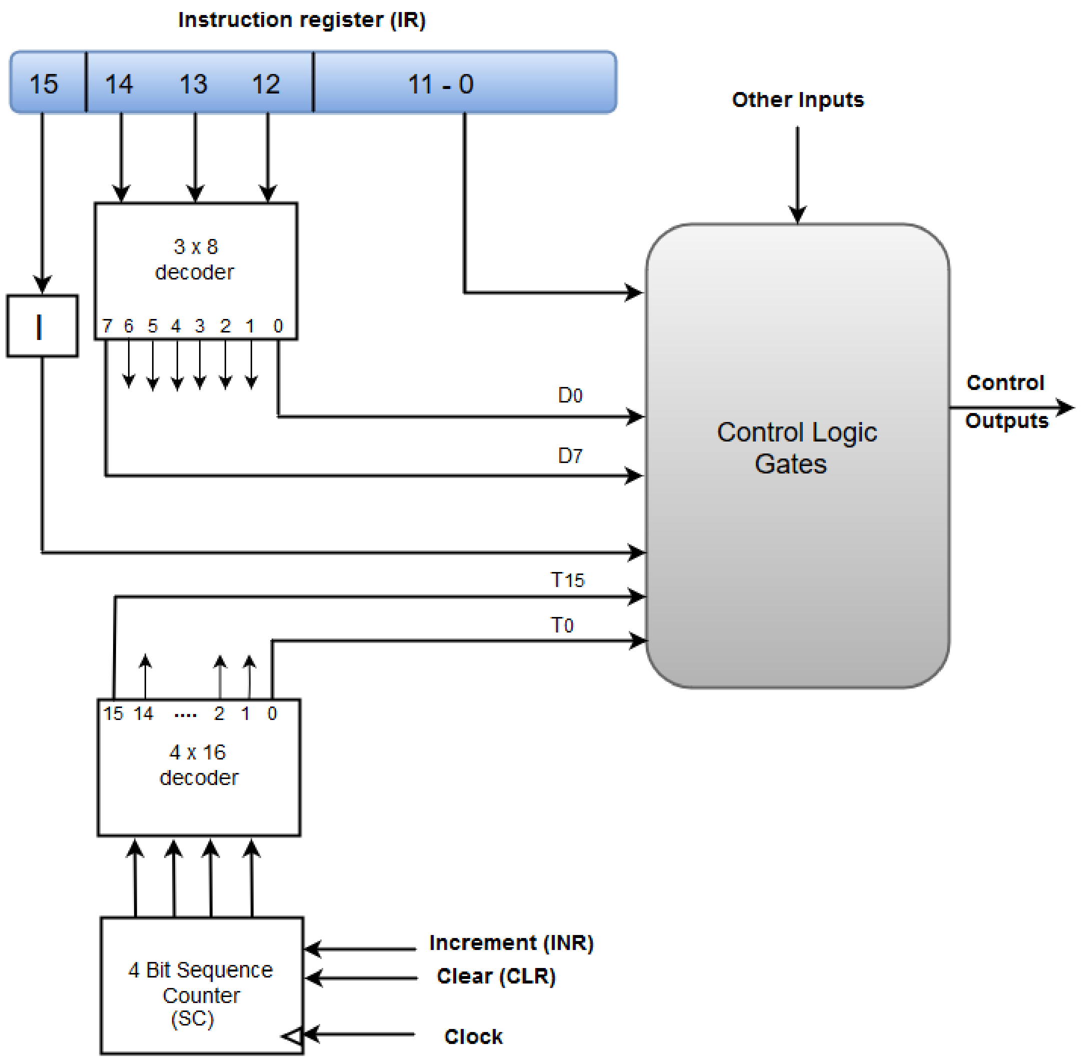
1. Hardwired Control
2. Microprogrammed Control

1. **Hardwired Control Unit:**
   The Hardwired Control organization involves the control logic to be implemented with gates, flip-flops, decoders, and other digital circuits.

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.

The following image shows the block diagram of a Hardwired Control organization.

Control Unit of a Basic Computer:



- A Hard-wired Control consists of two decoders, a sequence counter, and a number of logic gates.
- An instruction fetched from the memory unit is placed in the instruction register (IR).
- The component of an instruction register includes; I bit, the operation code, and bits 0 through 11.
- The operation code in bits 12 through 14 are coded with a 3 x 8 decoder.
- The outputs of the decoder are designated by the symbols D0 through D7.
- The operation code at bit 15 is transferred to a flip-flop designated by the symbol I.

- The operation codes from Bits 0 through 11 are applied to the control logic gates.
- The Sequence counter (SC) can count in binary from 0 through 15.

Control Memory

Control units that use dynamic microprogramming use a writable control memory. This type of memory can be used for writing (to change the microprogram) but is used mostly for reading. A memory that is part of a control unit is called a **control memory**.

The control unit in a digital computer initiates sequences of **micro-operations**. The control variables can be represented by a string of 1's and 0's called a **control word**. The control information is shared in a control memory, in a **micro-programmed** organization. A microprogrammed control unit is a control unit whose binary control variables are stored in memory.

Each word in control memory contains within it is a **microinstruction.** The control memory is programmed to begin the desired sequence of micro-operation.

A sequence of microinstructions constitutes a **microprogram**. When the control signals are generated by hardware, it is **hardwired**. In a bus-oriented system, the control signals that specify micro-operations are groups of bits that select the paths in multiplexers, decoders, and ALUs.

The control memory is usually a ROM, which stores all control information permanently. The **control address register** (CAR) specifies the address of the microinstruction, and the control data register holds the microinstruction read from memory.

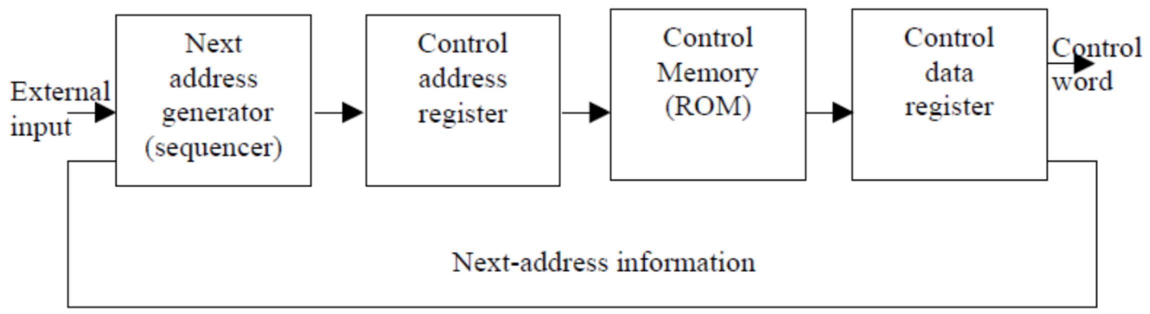


Figure: Micro-programmed Control Organization

The microinstruction contains a control word that specifies one or more micro-operations for the data processor. Once these operations are executed, the control must determine the next address. The location of the next microinstruction is generally the one next in sequence, otherwise, it may be located somewhere else in the control memory. For this reason it is necessary to use some bits of the present microinstruction to control the generation of the address of the next microinstruction. The next address may also be a function of external input conditions.

While the micro-operations 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 micro-instruction. Hence a microinstruction contains bits for initiating micro-operations in the data processor part and bits that determine the address sequence for the control memory.

A microprogram **sequencer** is the next address generator, as it determines the address sequence that is read from control memory. The address of the next microinstruction can be specified in several ways depending on the sequencer inputs.

Typical functions of a microprogram sequencer are:

- incrementing the CAR by one
- loading into the CAR and address from control memory
- transferring an external address
- loading an initial address to start the control operations

The **control data register** (CDR) stores the present microinstruction while the next address is computed and read from memory. The data register is also called a **pipeline register**. It allows the execution of the micro-operations 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 that once the hardware configuration is built, there should be no need for further hardware or wiring changes. If we want to make a different control sequence for the system, all we need to do is to specify a different set of microinstructions for control memory. The hardware configuration should not be changed for different operations. We have to change only the microprogram residing in control memory.

Timing and Control

Timing pulses are used in sequencing the micro-operations in an instruction. A master clock generator is used for controlling the timing for all register in a computer system. A state of a register cannot be changed by a clock pulse until it is enabled by the control signal, which are generated in the control unit and provide control inputs for multiplexers, processor register, and micro-operations. The control organization is of two types; hardwired control and microprogrammed control.

Hardwired Control

In a hardwired control, the control signals are generated by using the collection of combinational circuits. The main advantage of the hardwired control is that, it can be optimized to produce a fast mode of operation. Whenever a change or modification is to be done in the design, then the wiring among the various components needs to be done.

Micro-Programmed Control

In a micro-programmed control, a control memory is used for storing control information which is also programmed for initiating the sequence of micro-operations. Whenever any change or modification is required in the design, it can be done by updating the micro-program in the control memory.

Control unit consists of two decoders, a sequence counter, and number of logic gates. When an instruction is read from the memory, it is placed in the instruction register (IR). The instruction register is divided into three parts; addressing mode, opcode, and address. Control unit is responsible for interpreting the instruction code and providing the necessary control needed for processing these instructions. Control unit uses the instruction format for interpreting the instruction.

Timing is generated by 4-bit sequence counter and 4x16 decoder. The SC can be incremented or cleared. Example: T_0, T_1, T_2, T_3, T_4, T_0, T_1, . . .

Assume: At time T_4, SC is cleared to 0 if decoder output D_3 is active. This is expressed as:

D_3T_4: SC $\leftarrow$ 0

Initially, the CLR input of SC is active. The last three waveform on figure below shows how SC is cleared when $D_3T_4 = 1$. Output D_3 from the operation decoder becomes active at the end of the timing signal T_2. When timing signal T_4 becomes active, the output of the AND gate that implements the control function D_3T_4 becomes active. This signal is applied to the CLR input of SC. On the next positive clock transition the counter is cleared to 0. This causes the timing signal T_0 to become active instead of T_5 that would have been active if SC were incremented instead of cleared.

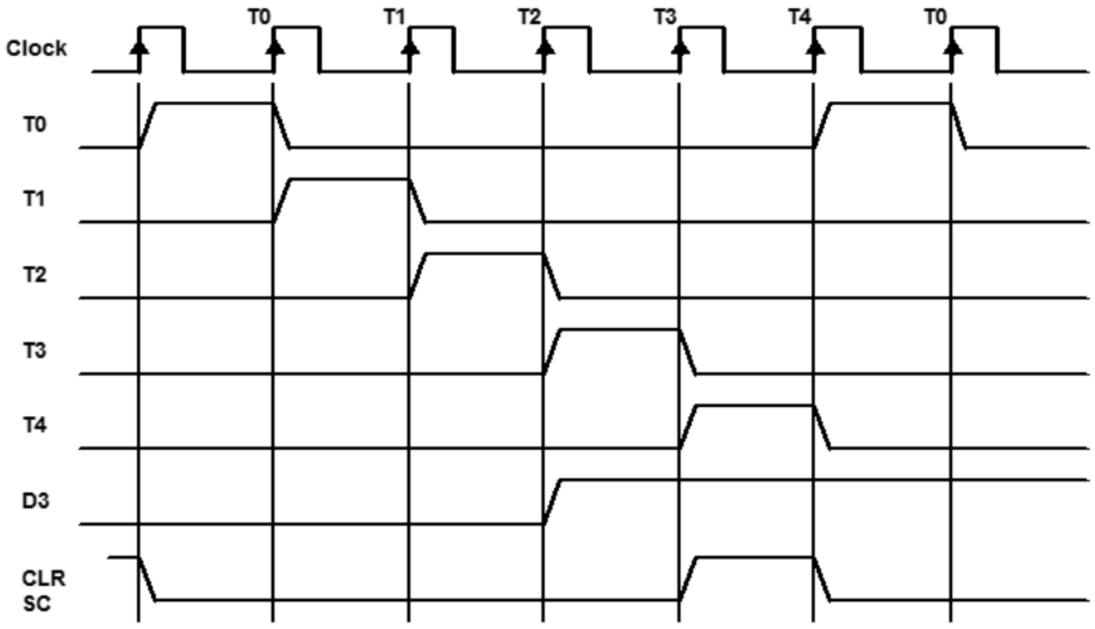


Figure: Example of Control Timing Signal

Address Sequencing

The microprogram consists of microinstructions that specify various internal control signals for execution of register micro-operations. Process of finding address of next microinstruction to be executed is called **address sequencing**. Address sequencer must have capabilities of finding address of next micro-instruction in following situations:

- In-line Sequencing

- Conditional Branch
- Subroutine Call and Return
- Looping
- Mapping from instruction opcode to address in control memory Unconditional Branch

Microinstructions are stored in control memory in groups, with each group specifying a routine. Each computer instruction has its own microprogram routine to generate the micro-operations.

The hardware that controls the address sequencing of the control memory must be capable of sequencing the microinstructions within a routine and be able to branch from one routine to another.

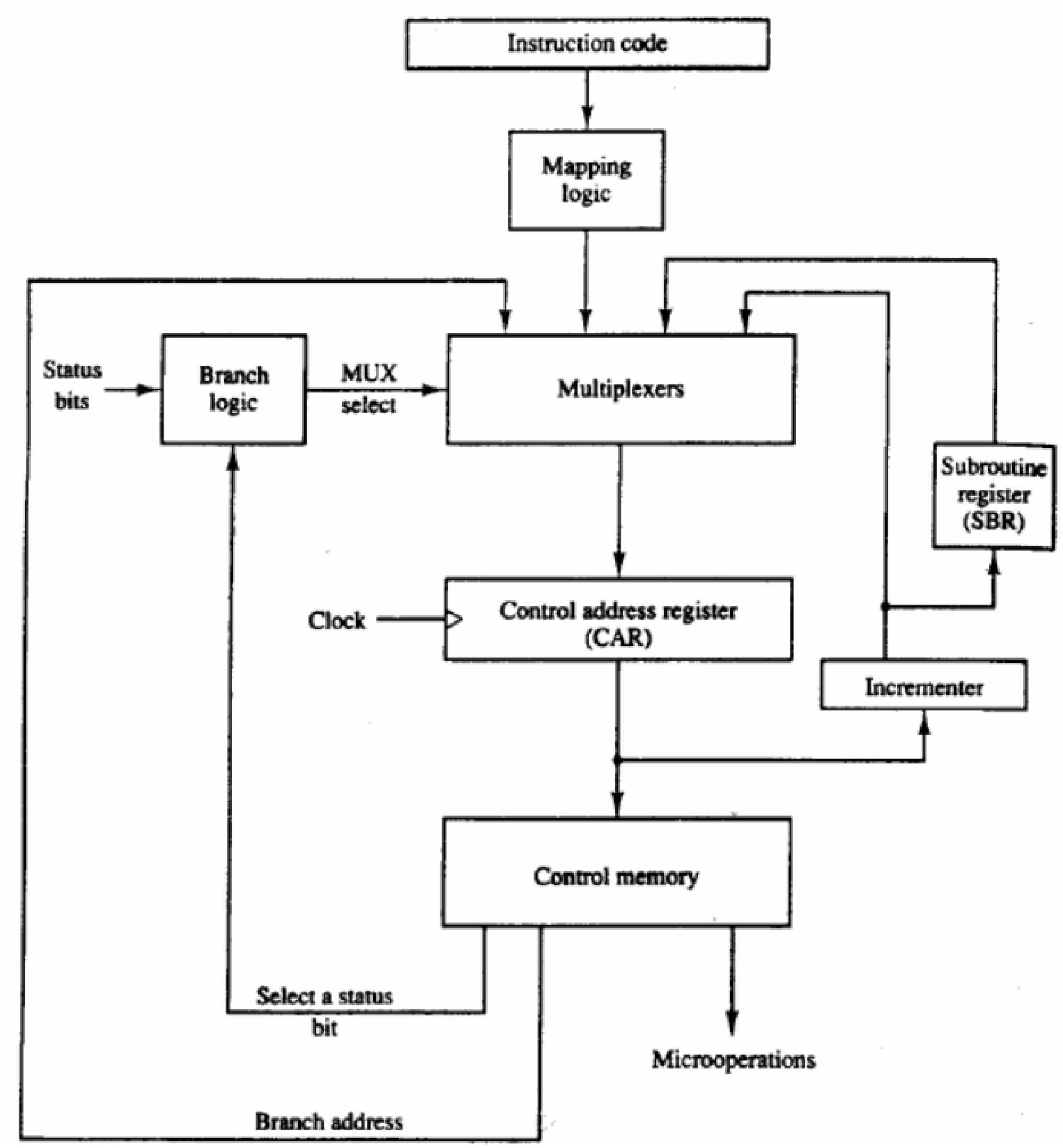


Figure: Selection of Address for Control Memory

The steps that the control must undergo during the execution of a single computer instruction are:

• Load an initial address into the CAR when power is turned on in the computer. This address is usually the address of the first microinstruction that activates the instruction fetch routine. IR holds instruction.

• The control memory then goes through the routine to determine the effective address of the operand. AR holds operand address.
• The next step is to generate the micro-operations that execute the instruction by considering the opcode and applying a mapping.
• After execution, control must return to the fetch routine by executing an unconditional branch.

The microinstruction 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 next address is obtained.

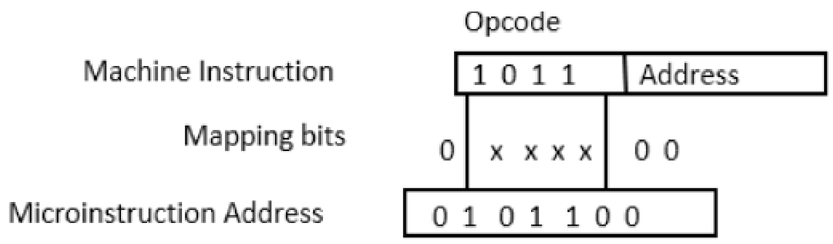
Conditional Branching
Conditional branching is obtained by using part of the microinstruction to select a specific status bit in order to determine its condition. The status conditions are special bits in the system that provide parameter information such as the carry-out of an adder, the sign bit of a number, the mode bits of an instruction, and i/o status conditions. The status bits, together with the field in the microinstruction that specifies a branch address, control the branch logic. The branch logic tests the condition, if met then branches, otherwise, increments the CAR. If there are 8 status bit conditions, then 3 bits in the microinstruction are used to specify the condition and provide the selection variables for the multiplexer.
If Condition is true, set the appropriate field of status register to 1. Conditions are tested for O (overflow), N (negative), Z (zero), C (carry), etc. Then test the value of that field if the value is 1 take branch address from the next address field of the current microinstruction). Otherwise simple increment the address.

Unconditional Branching
For unconditional branching, fix the value of one status bit to be one load the branch address from control memory into the CAR.
Mapping of Instruction
A special type of branch exists when a microinstruction specifies a branch to the first word in control memory where a microprogram routine is located. The status bits for this type of branch are the bits in the opcode. Assume an opcode of four bits and a control memory of 128

locations. The mapping process converts the 4-bit opcode to a 7-bit address for control memory. This provides for each computer instruction a microprogram routine with a capacity of four microinstructions.

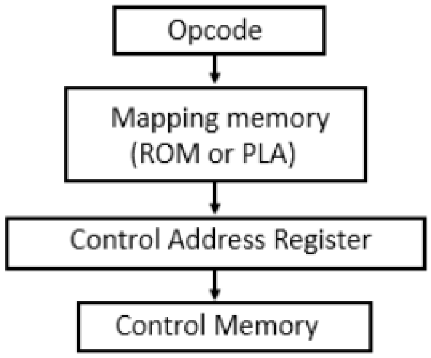
Another Approach of Mapping

Modify opcode to use it as an address of control memory.



Mapping Function Implemented by ROM or PLA

Use opcode as address of ROM where address of control memory is stored and then use that address as an address of control memory.



Subroutines

Subroutines are programs that are used by other routines to accomplish a particular task and can be called from any point within the main body of the microprogram. Frequently many micro-programs contain identical section of code. Microinstructions can be saved by employing subroutines that use common sections of microcode.

Microprogram Example

Once we have a configuration of a computer and its micro-programmed control unit, the designer generates the microcode for the control memory. Code generation of this type is called microprogramming and is similar to conventional machine language programming. The block diagram of computer consists of:

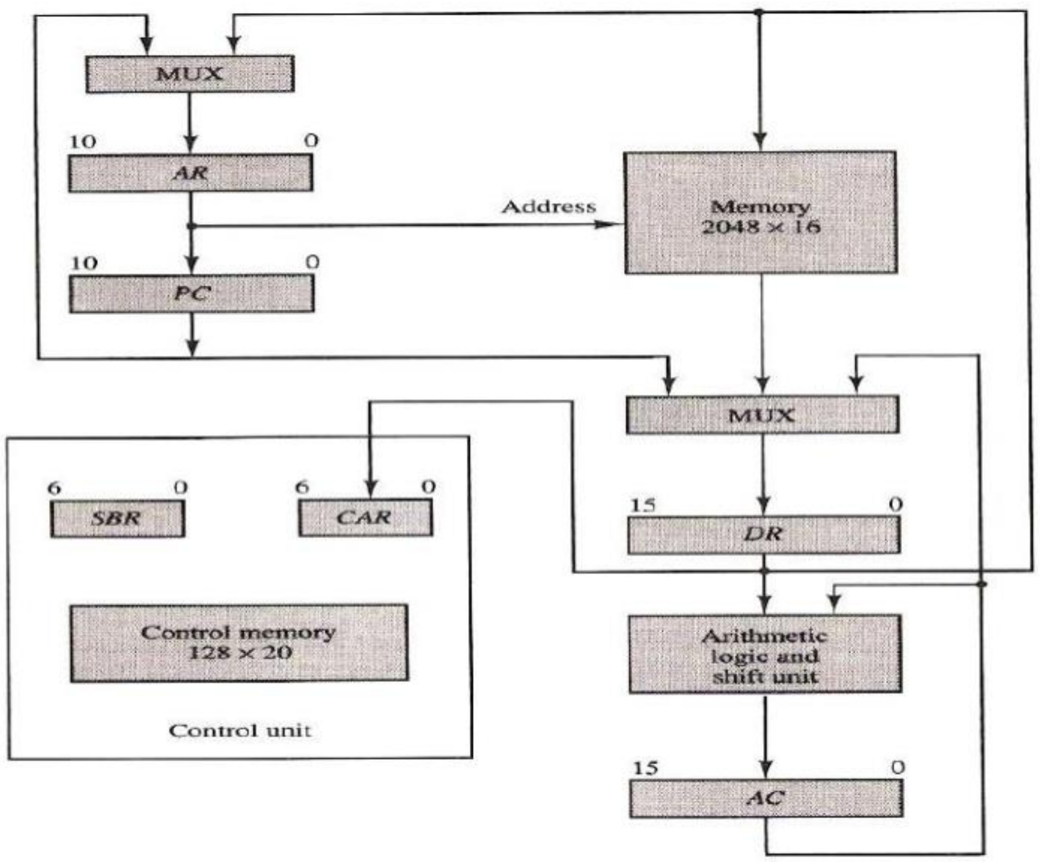


Figure: Computer hardware configuration

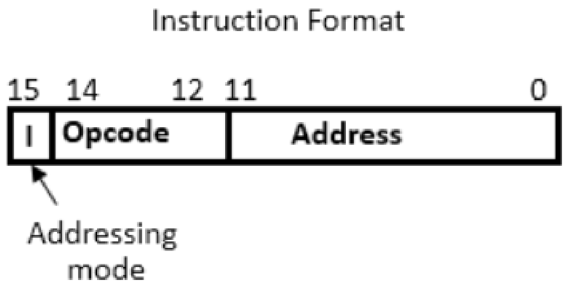
Transfer of information among registers in the processor is through Multiplexers rather than a bus.

Two memory units:
Main memory – stores instructions and data
Control memory – stores microprogram

Four processor registers:
Program counter – PC
Address register – AR
Data register – DR
Accumulator register – AC

Two control unit registers:

Control address register – CAR
Subroutine register – SBR

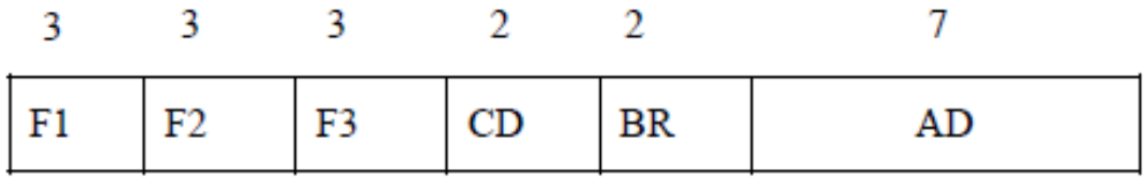


Three fields for an instruction:
I = 1-bit for indirect addressing
Opcode = 4-bit
Address Field = 11-bit

The example will only consider the following 4 of the possible 16 memory instructions

Symbol	**Opcode**	**Description**
ADD	0000	AC ← AC + M[EA]
BRANCH	0001	If (AC < 0) then (PC ← EA)
STORE	0010	M[EA] ← AC
EXCHANGE	0011	AC ← M[EA], M[EA] ← AC

Note: (EA is the effective address)
The microinstruction format is composed of 20 bits with four parts:



Three fields F1, F2, and F3 specify micro-operations for the computer [3 bits each].
The **CD** field selects status bit conditions [2 bits]
The **BR** field specifies the type of branch to be used [2 bits]
The **AD** field contains a branch address [7 bits]

Each of the three micro-operation fields can specify one of seven possibilities. Therefore only 21 micro-operations are used. No more than three micro-operations can be chosen for a microinstruction. If fewer than three are needed, the code 000 = NOP is used.
Five letters are used to specify a transfer-type micro-operation. First two designate the **source register**, Third is a '**T**', and Last two designate the **destination register.**

F1	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC + DR$	ADD
010	$AC \leftarrow 0$	CLRAC
011	$AC \leftarrow AC + 1$	INCAC
100	$AC \leftarrow DR$	DRTAC
101	$AR \leftarrow DR(0\text{–}10)$	DRTAR
110	$AR \leftarrow PC$	PCTAR
111	$M[AR] \leftarrow DR$	WRITE

F2	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC - DR$	SUB
010	$AC \leftarrow AC \vee DR$	OR
011	$AC \leftarrow AC \wedge DR$	AND
100	$DR \leftarrow M[AR]$	READ
101	$DR \leftarrow AC$	ACTDR
110	$DR \leftarrow DR + 1$	INCDR
111	$DR(0\text{-}10) \leftarrow PC$	PCTDR

F3	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC \oplus DR$	XOR
010	$AC \leftarrow \overline{AC}$	COM
011	$AC \leftarrow$ shl AC	SHL
100	$AC \leftarrow$ shr AC	SHR
101	$PC \leftarrow PC + 1$	INCPC
110	$PC \leftarrow AR$	ARTPC
111	Reserved	

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

BR	Symbol	Function
00	JMP	$CAR \leftarrow AD$ if condition = 1 $CAR \leftarrow CAR + 1$ if condition = 0
01	CALL	$CAR \leftarrow AD$, $SBR \leftarrow CAR + 1$ if condition = 1 $CAR \leftarrow CAR + 1$ if condition = 0
10	RET	$CAR \leftarrow SBR$ (Return from subroutine)
11	MAP	$CAR(2\text{–}5) \leftarrow DR(11\text{–}14)$, $CAR(0,1,6) \leftarrow 0$

Table: Symbols and Binary Code for Microinstruction Fields

Symbolic Microinstructions

A symbolic microprogram can be translated into its binary equivalent by means of an assembler. Each line of an assembly language microprogram defines a symbolic microinstruction and is divided into five fields: Label, micro-operations, CD, BR, and AD. The fields specify the following information:

1. The label field may be empty or it may specify a symbolic address. Terminate with a colon (:)
2. The micro-operations field consists of 1-3 symbols, separated by commas. Only one symbol from each F field. If NOP, then translated to 9 zeros
3. The condition field specifies one of the four conditions: U, I, S or Z
4. The branch field has one of the four branch symbols
5. The address field has three formats

a. A symbolic address – must also be a label
b. The symbol NEXT to designate the next address in sequence
c. Empty if the branch field is RET or MAP and AD is converted to 7 zeros

Design of Control Unit

After getting the micro-operations we have to execute these micro-operations but before that we need to decode them. The 9-bits of the micro-operation field are divided into 3 subfields of 3 bits each. The control memory output of each subfield must be decoded to provide distinct micro-operations. The outputs of the decoders are connected to the appropriate inputs in the processor unit. The Figure below shows 3 decoders and connections that must be made from their outputs.

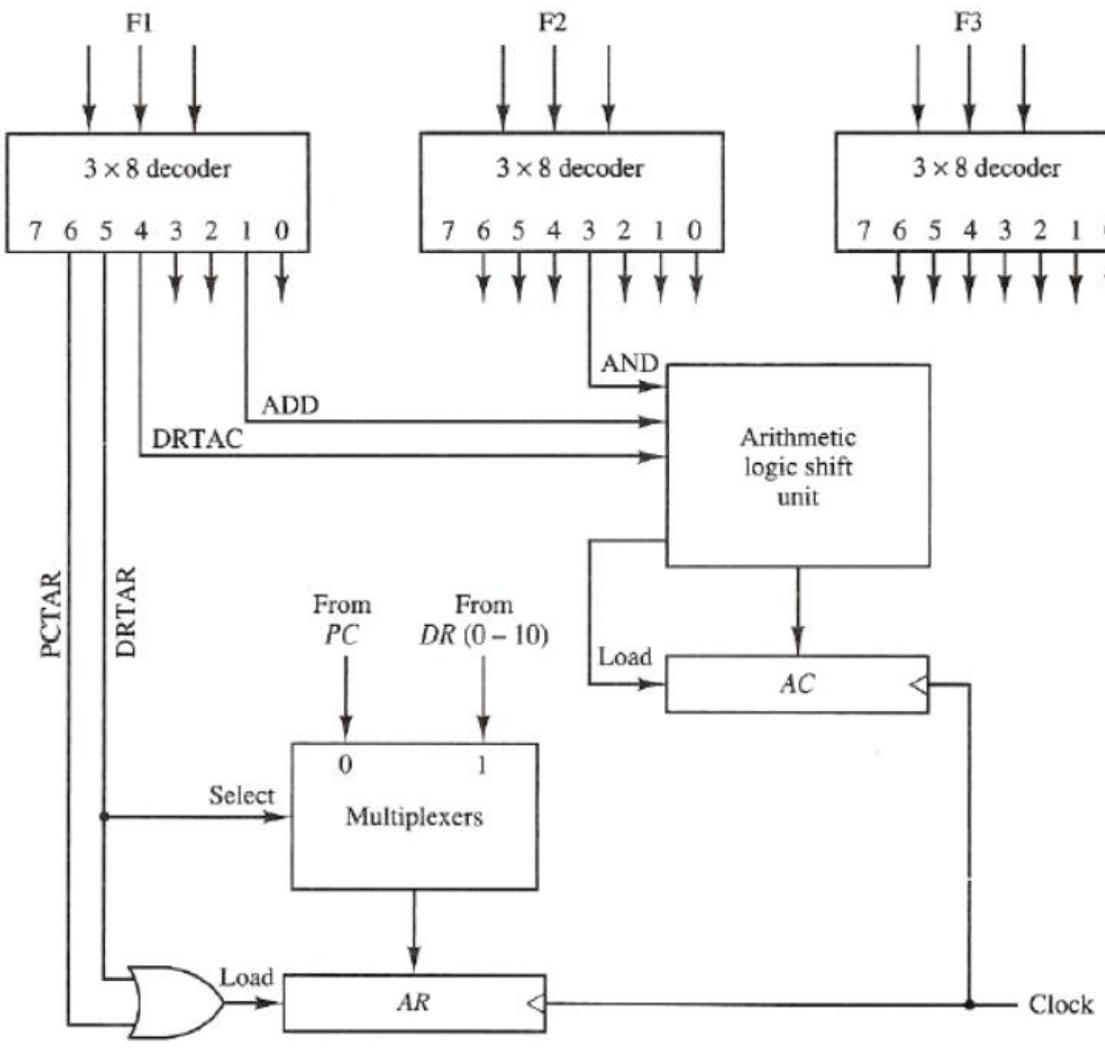


Fig. Decoding of Micro-operation Fields.

Three decoders and some of the connections that must be made from their outputs. Each of
the three fields of the microinstruction presently available in the output of control memory are
decoded with a 3x8 decoder to provide eight outputs.

- when F1 = 5, transfers the content of DR(0-10) to AR (DRTAR)
- when F1 = 6 there is a transfer from PC to AR (PCTAR)
- Outputs 5 and 6 of decoder F1 are connected to the load input of AR so that information
is transferred to AR.

The multiplexers select the information from DR when output 5 is active and from PC when output 5 is inactive. Because we have 8 micro-operations represented with the help of 3 bits in every table and also we have 3 such tables possible we have decoded these micro-operations field bits with three 3 x 8 decoders.

After getting the micro-operations, we have to give it to particular circuits, the data manipulation type of micro-operations like AND, ADD, Sub and so on we give to ALU and the corresponding results moved to AC. The ALU has been provided data from AC and DR. And for data transfer type of instructions like in the case of PCTAR or DRTAR we need to simply transfer the values. Because we have two options for data transfer in AR we are taking the help of MUX to choose one. We will take 2 x 1 MUX and one select line which is attached with DRTAR micro-operation signal. That means if DRTAR is high then MUX will choose DR to transfer the data to AR else PC's data will be moved to AR. And the corresponding data movement will be done with the help of load high or not. If any of the values is high the value will be loaded to AR. The clock signal is provided for the synchronization of micro-operations. Instead of using gates to generate the control signals marked by the symbols AND, ADD, and DR. These inputs will now come from the outputs of the decoders associated with the symbols AND, ADD, and DRTAC respectively. The other outputs of the decoders that are associated with
an AC operation must also be connected to the arithmetic logic shift unit in a similar fashion.

Microprogram Sequencer:
The basic components of a micro-programmed control unit are the control memory and the circuits that select the next address. The address selection part is called a micro-program sequencer. It can be constructed with digital functions to suit a particular application. Main purpose is to present an address to the control memory so that a microinstruction may be read and executed.

Design of input logic:

The input logic circuit in the figure below has three inputs, I_0, I_1, and T, and three outputs S_0, S_1,
and L. Variables S_0 and S_1 select one of the source addresses for CAR. Variable L enables the
load input in SBR. The binary values of the two selection variables determine the path in the
multiplexer.
For example, with S_1S_0 = 10, multiplexer input number 2 is selected and establishes a transfer
path from SBR to CAR. Note that each of the four inputs as well as the output of MUX 1 contains a 7-bit address.
The truth table can be used to obtain the simplified Boolean functions for the input logic circuit:
$S_1=I_1$
$S_0 = I_1I_0+I'_1T$
$L = I'_1I_0T$
The circuit can be constructed with three AND gates, an OR gate and an inverter. The

truth
table for the input logic circuit is shown in table below:

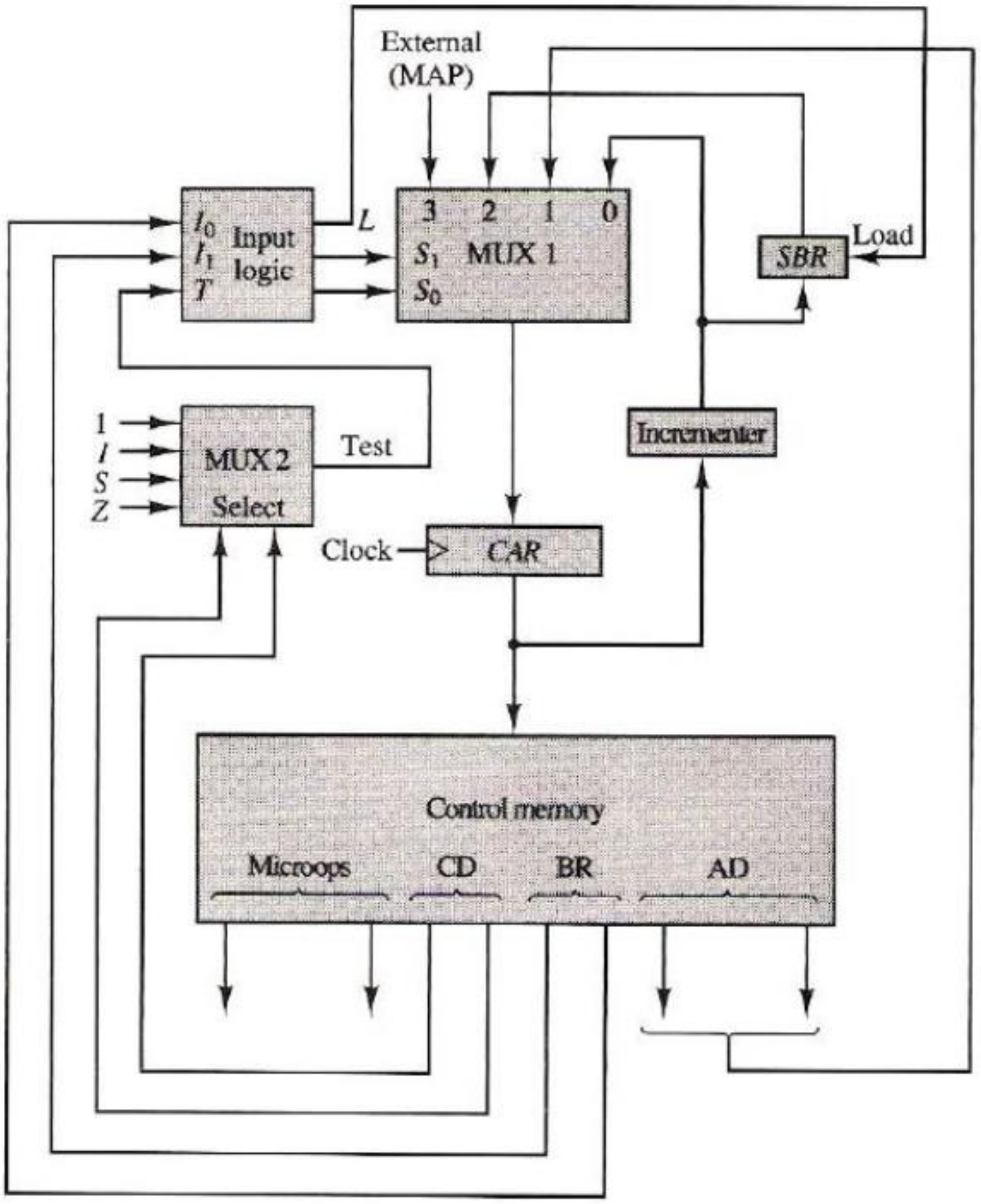
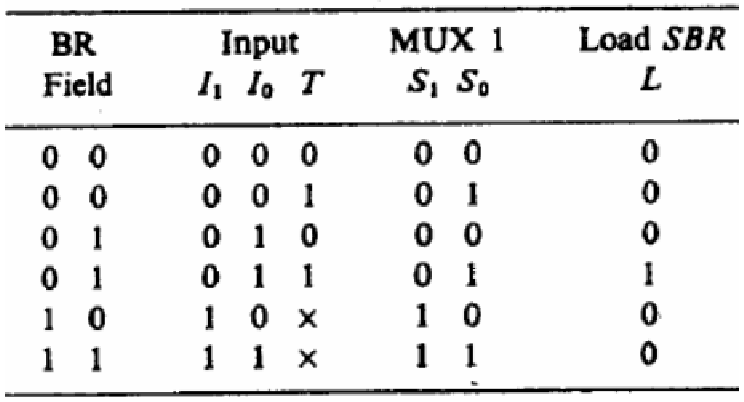


Figure: Microprogram Sequencer for a Control Memory



BR	Input			MUX 1		Load SBR
Field	I_1	I_0	T	S_1	S_0	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	×	1	0	0
1 1	1	1	×	1	1	0

A hardwired control differs from microprogrammed control in the following ways:

HARDWIRED CONTROL UNIT

1. The control unit whose control signals are generated by the hardware through a sequence of instructions is called a hardwired control unit.
2.The control logic of a hardwired control is implemented with gates, flip flops, decoders etc.
3.Wiring changes are made in the hardwired control unit if there are any changes required in the design.
4.Hardwired control unit are faster and known to have complex structure.

MICROPROGRAMMED CONTROL UNIT

1. The control unit whose control signals are generated by the data stored in control memory and constitute a program on the small scale is called a microprogrammed control unit.
2. The control logic of a micro-programmed control is the instructions that are stored in control memory to initiate the required sequence of microoperations.
3. Changes in a microprogrammed control unit are done by updating the microprogram in control memory.
4. Microprogrammed control unit is comparatively slow compared but are simple in structure.

The control unit whose control signals are generated by the hardware through a sequence of instructions is called a hardwired control unit. The control logic of a hardwired control is implemented with gates, flip flops, decoders etc.