# CS 202 Computer Organization and Architecture
# Module 6
# Control Logic Design

Sheena N

Assistant Professor
Dept. of CSE
College of Engineering & Management Punnapra

*sheena.cemp@gmail.com*

May 7, 2017

## Overview

1. Control Organization
2. One flip flop per state methods
3. Sequence Register and Decoder Method
4. Hardwired Control
5. PLA Control
6. Microprogram Control
7. Hardwired Control
8. Control of Processor Unit
9. PLA Control
10. Micro-programmed Control
11. Micro-program Sequencer
12. Microprogrammed CPU Organization

## Control Organization

- Sequential logic procedure was used to implement control operations
- Disadvantages of sequential control logic
    - Large number of states
    - Excessive number of flip flops and gates
    - Design methods uses state and excitation tables but in practice they are cubersome
- Goal of control logic design should be development of a circuit that implements the desired control sequence in a logical and straightforward manner
- Designers used specialized methods for control logic design which is considered as the extension of the classical sequential logic method combined with register transfer method

## Control Organization (cont.)

4 methods of control organisation

1. One flip flop per state methods
2. Sequence register and decoder method
3. PLA control
4. Microprogram control

## One flip flop per state methods

- One flip flop per state in the control sequential circuit
- One flip flop is set at any particular time ; all others are cleared
- A single bit is made to propagate from one flip flop to other under the control of a decision logic
- Each flip flop represents a state and is activated only when the control bit is transferred to it
- Uses maximum number of flip flops
  Eg:- A sequential circuit with 12 states requires a minimum of 4 flip flops because $2^3 < 12 < 2^4$. Control circuit uses 12 flip flops one for each state

## One flip flop per state methods (cont.)

- Advantages
    - Simple to design by inspecting the state diagram that describe the control sequence
    - Offers a savings in design effort
    - An incease in operational simplicity
    - Decrease in the combinational ciruits required to implement the complete sequential circuit
- Disadvantage
    - Increase system cost since more flip flops are used

## One flip flop per state methods (cont.)

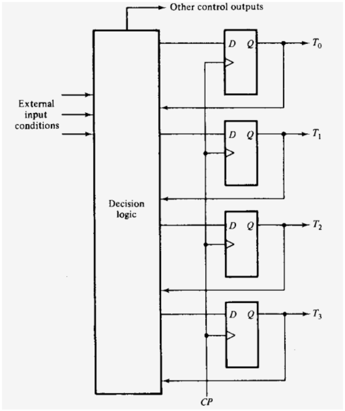


Figure: Control logic with one flip flop per state

## One flip flop per state methods (cont.)

- 4 stage sequential control logic that uses 4 D-type flip flops: one flip flop per state $T_i$, i=0, 1, 2, 3
- At any given time interval between 2 clock pulses ony one flip flop is equal to 1 all others are 0
- The trasition from the present state to next state is a function of the present $T_i$ that is a 1 and certain input conditions
- The next state is manifested when the previous flip flop is cleared and a new one is set
- Each of the flip flop is connected to the data processing section of the digital system to initiate certain microoperations
- The control outputs are a function of the T's and external inputs
- These outputs may also initiate microopeartions

## One flip flop per state methods (cont.)

- If control circuit does not need external inputs for its sequencing, the circuit reduces to straight shift register with a single bit shifted from one position to the next
- If control sequence must repeated ove and over again the control reduces to ring counter
- Ring counter is a shift register with the output of last flip flop connected to the input of the first flip flop.
- In a ring counter single bit continously shifts from one position to the next in a circular manner.
- Also called ring counter controller

## Sequence Register and Decoder Method

- Uses register to sequence the control states
- The register is decoded to provide one output for each state
- For n flip flops in the sequence register the circuit will have $2^n$ states and decoder will have $2^n$ outputs
- Eg :- A 4 bit register can be in any one of 16 states
  A 4X16 decoder will have 16 outputs one for each state of the register

## Sequence Register and Decoder Method (cont.)

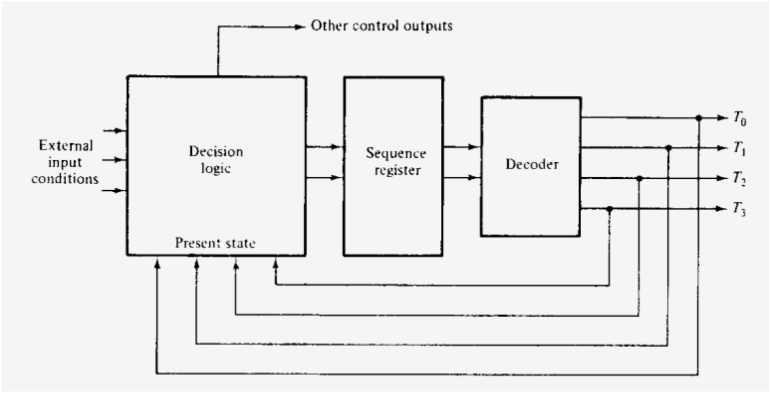


Figure: Control logic with sequence register and decoder

## Sequence Register and Decoder Method (cont.)

- 4 state sequential control logic
- The sequence register has 2 flip flops and the decoder establishes separate outputs for each state in the register
- The transition to the next state in the sequence register is a function of the present state and the external input conditions
- If the control circuit does not need external inputs the sequence register reduces to a counter that continously sequence through the four states so called counter decoder method

## Hardwired Control

-

## PLA Control

- PLA control is similar to the sequence register and decoder method except that all combinational circuit are implemented with a PLA including the decoder and the decision logic
- By using PLA for combinational circuit reduce the number of ICs and the number of interconnection wires

## PLA Control (cont.)

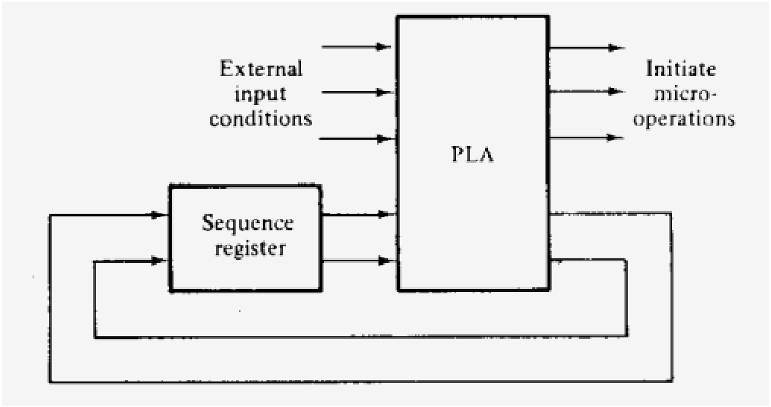


Figure: PLA control logic

## PLA Control (cont.)

- The external sequence register establishes the present state of the control circuit
- The PLA outputs determine which microoperations should be initiated depending on the external input conditions and the present state of the sequence register
- At the same time other PLA outputs determine the next state of the sequence register
- The sequenc eregister is external to the PLA if the unit implements only combinational circuits
- Some PLAs include not only gates but also flip flops within the unit.
    - This implement a sequential circuit by specifying the links that must be connected to the flip flops in manner that the gate links are specified

## Microprogram Control

- Purpose of control unit is to initiate a series of sequential steps of microoperations
- Any given time certain operations are to be initiated while all others remain idle
- Control variable at any given time can be represented by a string of 1's and 0's called control word
- Control words can be programmed to initiate the various components in the system in an organized manner
- A control unit whose control variables are stored in a memory called a microprogrammed control unit
- Microinstruction :- Each control word of memory
- Microprogram :- Sequence of microinstructions

## Microprogram Control (cont.)

- Control memory is usually ROM since alterations of microprogram is seldom needed
- Use of microprogram involves placing all control control variables in words of the ROM for use by the control unit through successive read opeartions.
- The content of the word in the ROM at a igven address specifies the microooprations for the system
- Dynamic microprogramming
  - Permits a microprogram to be loaded initially from the computer console or from an auxiliary memory such aas magnetic disk
  - writable control memory(WCM) can be used for writing but used mostly for reading
- A ROM, PLA or WCM when used in a control unit is referred as a control memory

## Microprogram Control (cont.)

- Control memory address register specifies the control word read from control memeory
- ROM operates as a combiantional circuit with address value as the input and the corresponding word as the output
- The content of the specified word remains on the output wires as long as the address value remains in the adddress register
- The word out of the ROM should be transferred to a buffer register if the address register changes while the ROM word is still in use
- If the change in address and ROM word can occur simultaneously no buffer register is needed
- The word read from memory represents a microinstruction.
- The microinstruction specifies one or more microoperations for the components of the system

## Microprogram Control (cont.)

- Once these opearations are executed the control unit must determine its next adddress
- The location of the next microinstruction may be next one in the sequence or it may locate somewhere else in the control memeory
- Some bits of the microinstruction to control the generation of the address for the next microinstruction
- The next address may be function of external input conditions
- The next address is computed in the next addess generator circuit and then transferred into the control address register to read the next miccroinstruction

## Hardwired Control

- Control logic derived in this section is a hardwired control of the one flip flop per state method
- Design of hardwired control is carried out in 5 consecutive steps
    1. The problem is stated
    2. An initial equipment configuration is assumed
    3. An algorithm is formulated
    4. The data processor part is specified
    5. The control logic is designed
- **Statement of the problem**
    - implement with hardware the addition or subtraction of 2 fixed point binary numbers represented in sign magnitude form
    - Complement arithmetic used and final result in sign magnitude form

## Hardwired Control (cont.)

- **Equipment configuration**
  - Two signed binary numbers is added or subtracted contains n bits
  - The magnitude of the numbers k=n-1 bits and are stored in registers A and B
  - The sign bit are stored in flip flops $A_s$ and $B_s$
  - ALU performs arithmetic operations
  - 1 bit register E serves as the overflow filp flop
  - Output carry from ALU transferrd to E
  - Result of operation is available in registers A and $A_s$
  - 2 inputs signals of control specify the add($q_a$) and subtract ($q_s$) operations
  - Output variable x indicates end of the operation
  - Control recognizes input signal $q_a$ and $q_s$ and provide the required opeartion
  - Upon completion control inorms external environment with the output x that sum or difference is in registers A and $A_s$ and that the overflow bt is in E

## Hardwired Control (cont.)

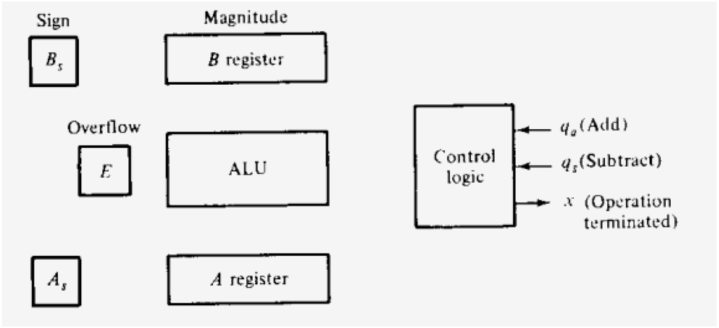


Figure: Register configuration for adder-subtractor

## Hardwired Control (cont.)

- **Derivation of the algorithm**
  - Designate magnitude of numbers by A and B
  - When the numbers are addded or subtracted there are 8 different conditionsdepending on the sign of numbers and the operation performed

    $$(\pm A) \pm (\pm B)$$

    if the arithmetic operation specified is subtraction change the sign of B and add

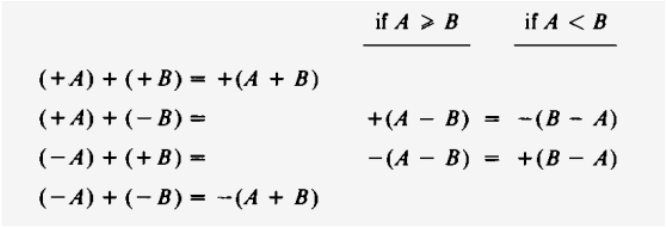
    $$\begin{aligned}(\pm A) - (\pm B) = (\pm A) + (-B)\\(\pm A) - (-B) = (\pm A) + (+B)\end{aligned}$$

    reduces to the number of possible combinations to four

    $$(\pm A) + (\pm B)$$

## Hardwired Control (cont.)

and the 4 combinations are



| | if $A \geqslant B$ | if $A < B$ |
|---|---|---|
| $(+A) + (+B) = +(A + B)$ | | |
| $(+A) + (-B) =$ | $+(A - B)$ | $= -(B - A)$ |
| $(-A) + (+B) =$ | $-(A - B)$ | $= +(B - A)$ |
| $(-A) + (-B) = -(A + B)$ | | |

## Hardwired Control (cont.)

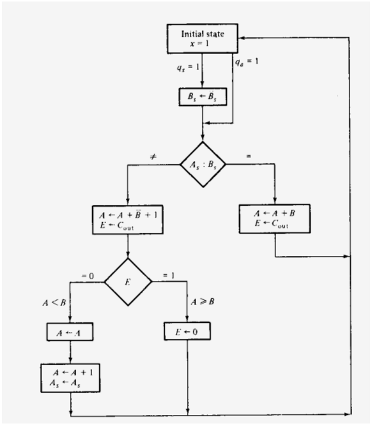


Figure: Flowchart of sign magnitude addition and subtraction

## Hardwired Control (cont.)

- **Data processor Specifiction**

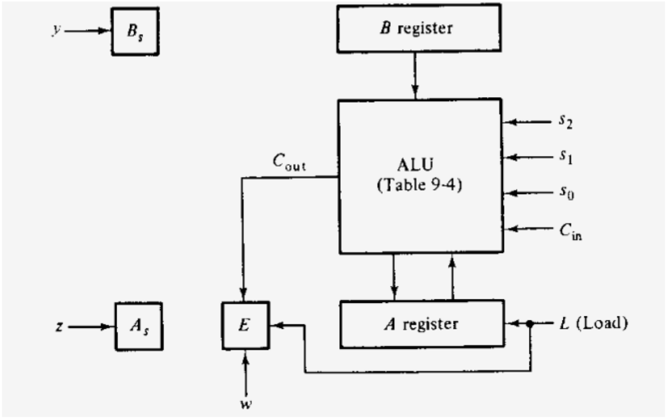


Figure: Data processor registers and ALU

## Hardwired Control (cont.)

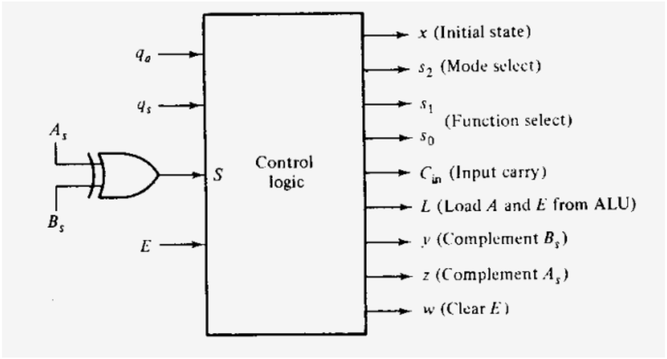


Figure: System block diagram

## Hardwired Control (cont.)

- **Control state diagram**

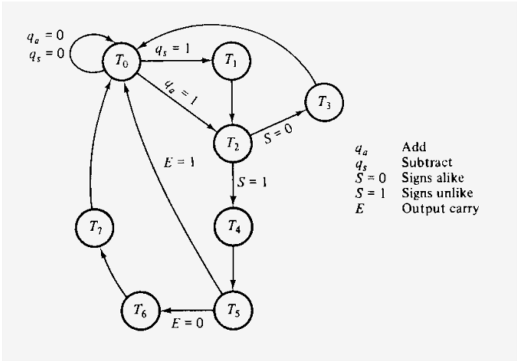
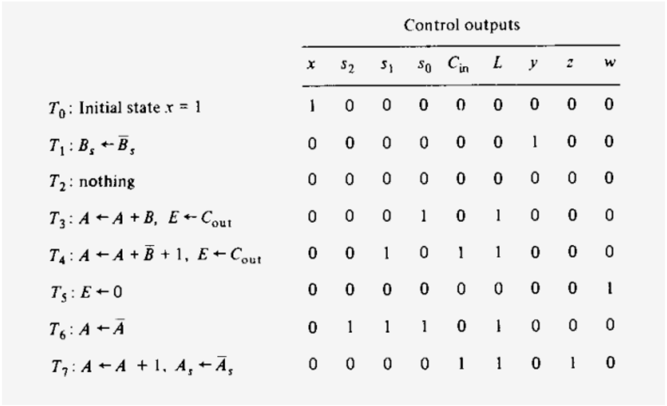


Figure: State diagram

## Hardwired Control (cont.)

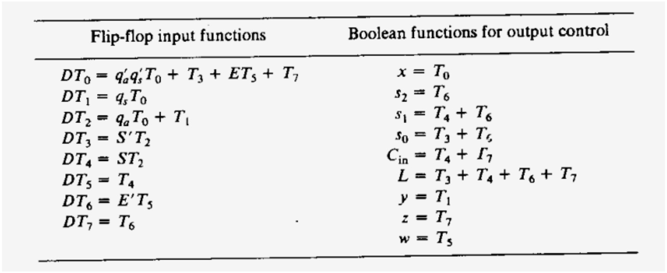


|  | Control outputs |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|
|  | $x$ | $s_2$ | $s_1$ | $s_0$ | $C_{in}$ | $L$ | $y$ | $z$ | $w$ |
| $T_0$: Initial state $x = 1$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $T_1: B_s \leftarrow \overline{B}_s$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| $T_2$: nothing | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $T_3: A \leftarrow A + B,\ E \leftarrow C_{out}$ | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| $T_4: A \leftarrow A + \overline{B} + 1,\ E \leftarrow C_{out}$ | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| $T_5: E \leftarrow 0$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| $T_6: A \leftarrow \overline{A}$ | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| $T_7: A \leftarrow A + 1,\ A_s \leftarrow \overline{A}_s$ | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |

Figure: Sequence of microoperations

## Hardwired Control (cont.)

- **Design of hardwired control**



| Flip-flop input functions | Boolean functions for output control |
|---|---|
| $DT_0 = q_a' q_s' T_0 + T_3 + ET_5 + T_7$ | $x = T_0$ |
| $DT_1 = q_s T_0$ | $s_2 = T_6$ |
| $DT_2 = q_a T_0 + T_1$ | $s_1 = T_4 + T_6$ |
| $DT_3 = S' T_2$ | $s_0 = T_3 + T_5$ |
| $DT_4 = S T_2$ | $C_{in} = T_4 + T_7$ |
| $DT_5 = T_4$ | $L = T_3 + T_4 + T_6 + T_7$ |
| $DT_6 = E' T_5$ | $y = T_1$ |
| $DT_7 = T_6$ | $z = T_7$ |
| | $w = T_5$ |

Figure: Boolean function for control

## Control of Processor Unit

- General purpose microprogram control unit must have control memory large enough to store many microinstructions
- Include all possible control variables in the system - controlling ALU, Multiplexer, status bits
- Provision must be available to accept an external addresss to initiate many operations
- Advantage of Microprogram control
  - Once the hardware configuration is established there should be no need for further hardware or wiring changed
  - Only change microprogram reside in the control memory for different operations

## Control of Processor Unit (cont.)

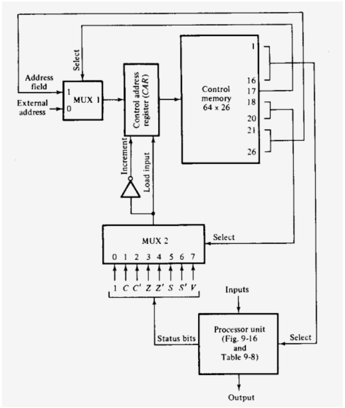


Figure: Microprogram control for processing unit

## Control of Processor Unit (cont.)

- Control memory of 64 words with 26 bits
- To select 64 words require an address of 6 bits
- To select 8 status bits 3 selection lines for multiplexer
- 1 bit of the microinstruction selects between an external address and address firld of the microinstruction
- First 16 bits of the microinstruction select the microoperation for the processor
- The other 10 bits select the next address for the control address register
- The status bits fro the processor are applied to the inputs of the multiplexer
- Both normal and complement values are used for status bits except for overflow bit V

## Control of Processor Unit (cont.)

- Input 0 of the of MUX2 is connected to a binary constant which is always 1
- The load of CAR is enabled when the input is selected by bits 18,19 and 20 in the microinstruction causes a transfer of information from output of MUX1 into CAR
- The input into CAR is a function of bit 17 in the microinstruction
  Bit $17 = 1$ : CAR receives the address field of microinstruction
  Bit $17 = 0$ : External address is loaded into CAR
- External addresss is for the purpose of initiating a new sequence of microinstructionswhich can be specified by the external environment

## Control of Processor Unit (cont.)

- How status bits are affected by each microopeartions
  - The sign(S) and Zero(Z) bits are affected by all opeartions
  - The Carry(C) and Overflow(V) bits do not change after the following ALU operations
    - The 4 lgic operationsOR, AND, XOR and complement
    - The increment and decrement opeartions
  - The output carry from the ALU goes into the C bit of the status register also affected after a circular shift wih carry opeartion

**Microprogram example**

- Count the number of 1's presently stored in processor register R1 and sets processor register R2 to that number

## Control of Processor Unit (cont.)

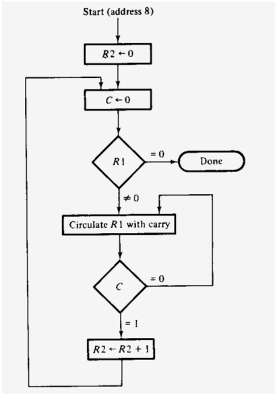
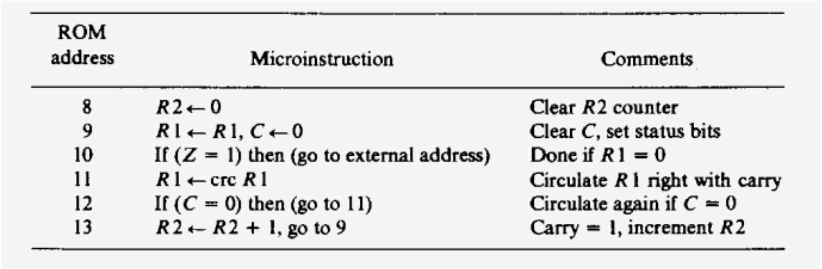


Figure: Flowchart for counting the number of 1's in register R1

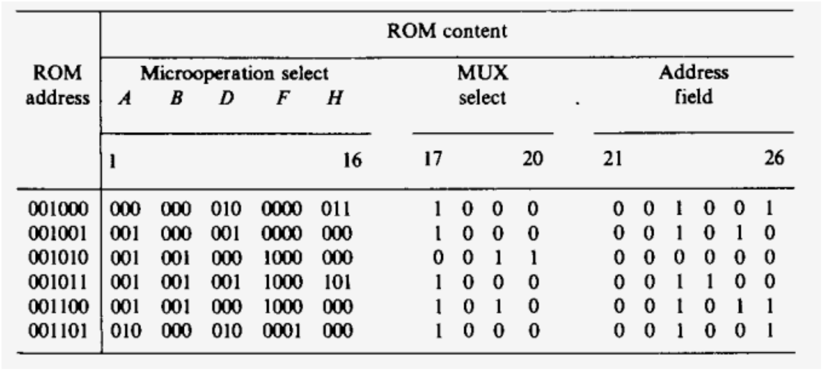
## Control of Processor Unit (cont.)



| ROM address | Microinstruction | Comments |
|---|---|---|
| 8 | $R2 \leftarrow 0$ | Clear $R2$ counter |
| 9 | $R1 \leftarrow R1, C \leftarrow 0$ | Clear $C$, set status bits |
| 10 | If $(Z = 1)$ then (go to external address) | Done if $R1 = 0$ |
| 11 | $R1 \leftarrow$ crc $R1$ | Circulate $R1$ right with carry |
| 12 | If $(C = 0)$ then (go to 11) | Circulate again if $C = 0$ |
| 13 | $R2 \leftarrow R2 + 1$, go to 9 | Carry $= 1$, increment $R2$ |

Figure: Symbolic mcroprogram to count the number of 1's in R1

## Control of Processor Unit (cont.)



| ROM address | ROM content | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Microoperation select | | | | | MUX select | | | | Address field | | | | | |
| | *A* | *B* | *D* | *F* | *H* | | | | | | | | | | |
| | 1 | | | | 16 | 17 | | | 20 | 21 | | | | | 26 |
| 001000 | 000 | 000 | 010 | 0000 | 011 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 001001 | 001 | 000 | 001 | 0000 | 000 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 001010 | 001 | 001 | 000 | 1000 | 000 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 001011 | 001 | 001 | 001 | 1000 | 101 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 001100 | 001 | 001 | 000 | 1000 | 000 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 001101 | 010 | 000 | 010 | 0001 | 000 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

Figure: Binary microprogram to count the number of 1's in R1

## PLA Control

-

## Micro-programmed Control

- Consisting od 2 parts
  - Control memory :- Stores the microinstructions
  - Microprogram sequencer :- Control the generation of the next address
- Sequences microinstructions in control memory
- Inspects certain bits of the microinstruction from which it determines the next address for control memory
- A typical sequencer provides the following address sequencing capabilities
  1. Increments the present address for cntrol memory
  2. Branches to an address as specified by the address field of the microinstruction
  3. Branches to a given ddress if a specified status bit is equal to 1
     Transfers control to a new address as specified by an external source
  4. Has facility for subroutine calls and returns

## Micro-program Sequencer

-

## Microprogrammed CPU Organization

- Digital computer consists of
  - Central Processing Unit(CPU)
  - Memory unit
  - Input-output devices
- CPU can be divides into 2 distinct and interactive sections
  - Processing section
    - Useful device for constructing the processor section of a CPU
  - Control section
    - Controlling the entire units of computer
    - Microprogram sequencer - constructing a microprogram control of CPU
- Microprogrammed computer - A computer CPU use the microprogram sequencer
- Microprogrammed computer consists of
  - Memory unit

## Microprogrammed CPU Organization (cont.)

    - Stores instructions and data supplied by the user through an input device
  - 2 processor unit
    1. Data processor :- Manipulates data
    2. Address processor :- Manipulates the address information received from memory

    - Combined into one unit or separate to provide distinct bus for memory address
  - A microprogram sequencer
  - A control memory
  - Other digtal functions
- An instruction extracted from memory unit during fetch cycle goes into instruction register
- The instruction code bits in the instruction register specify afor control memory macrooperation

## Microprogrammed CPU Organization (cont.)

- Code trasformation constitutes a mapping function that is needed to convert the operation code bits of an instruction into a starting address for the control memory and is implemented with ROM or PLA
- Mapping concept provide flexibility for adding instructions or macrooperations for control memory as need arises
- The address generated in code trnsformation mapping function i applied to the external address(EXA) input of the sequencing
- Microprogram control unit consists of
  - The sequencer
    - Generates next address
  - A control memory
    - Reads the next macroinstruction while present macroinstruction are being executed in the other units of the CPU
    - For storing microinstructions
  - A multiplexer

## Microprogrammed CPU Organization (cont.)

    - selects one of the many status bits and applies to the T(test) input of the sequencer
    - One of the input of the multiplexer is always I to provide an unconditional branch opeartion
  - A pipiline register
    - Speed up the control opeartion
    - Allows next address to be generated and the output of control memory to change while the control word in pipeline register initiates the microopertaions given by present microinstruction
    - Not always necessary
    - The output of control memory can go directly to the control inputs of the various units in the CPU
- Microinstruction format :-
  - contains 6 fields
  - First 3 fields(I,SL, BRA) provide information to the sequencer to determine the next address for control memory
    - I field ( 3 bits) :- Supplies input information for the sequencer

## Microprogrammed CPU Organization (cont.)

- SL field :- Selects a status bit for the multilpxer
- BRA field :- Address field of microinstruction and supplies a branch address(BRA) to the sequencer

- The next 3 fields(MC, PS, DF) are for controlling microoperations in the processor and the memory units
    - Memory control (MC) field :- Controls the address processor and the read and write opeartions in the memory unit
    - The processor select(PS) field :- Controls the operations in the data processor unit
    - The data field(DF) :- Used to introduce constants into the processor
- Output from data field may be used to set up control registers and introduce data in processor registers
    - Eg:- Constant in the data field may be added to a processor register to increment its contents by a specified value
      Setting a sequence counter to a constant value. Sequence counter s then used to count the number of times a microprogram loop is traversed and is usually required in a multiply or divide routine

## Microprogrammed CPU Organization (cont.)

-

The End

# CHAPTER -6

# Control Unit Design

**Objectives are:**

- **Control Logic Design**
  - Control Organization
  - Design of Hardwired Control
  - Control of Processor Unit
  - PLA Control
- **Micro-Programmed Control**
  - Microinstructions
  - Horizontal and Vertical Microinstructions
  - Micro-Program Sequencer
  - Micro Programmed CPU Organization

## 6.1 Control Logic Design

Control unit is the nerve centre of a computer system. The main responsibility of control unit is the generation of timing and ~~control signals~~. In this section, we are discussing the different methods of control unit design.

### 6.1.1 Control Organization

The control unit is the circuitry that controls the flow of information through the processor, and coordinates the activities of the other units within it. In a way, it is the "brain within the brain", as it controls what happens inside the processor, which in turn controls the rest of the PC. Control units control the flow of information with the help of control signals.

ALU (log/Arith)
t ALU | M. Memory
directional
structure

**Functions of Control Unit**

The control unit directs the entire computer system to carry out stored program instructions. The control unit must communicate with both the arithmetic logic unit (ALU) and main memory. The control unit instructs the arithmetic logic unit that which logical or arithmetic operation is to be performed .The control unit co-ordinates the activities of the all other units as well as all peripherals and auxiliary storage devices linked to the computer.

**Design of Control Unit**

To execute an instruction, the control unit of the CPU must generate the required control signal in the proper sequence. As for example, during the fetch phase, CPU has to generate $PC_{out}$ signal along with other required signal in the first clock pulse. In the second clock pulse CPU has to generate $PC_{in}$ signal along with other required signals. So, during fetch phase, the proper sequence for generating the signal to retrieve from and store to PC is $PC_{out}$ and $PC_{in}$.

To generate the control signal in proper sequence, a wide variety of techniques exists. Most of these techniques, however, fall into one of the two categories.

**1. Hardwired Control**
**2. Micro programmed Control**

Hardwired control units are constructed using digital circuits and once formed it cannot be changed. A micro programmed control unit itself decodes and execute instructions by means of executing micro programs.

### 6.1.2 Design of Hardwired Control

In this hardwired control techniques, the control signals are generated by means of hardwired circuit. The main objective of control unit is to generate the control signal in proper sequence.

Consider the sequence of control signal required to execute the add instruction. It is obvious that seven non-overlapping time slots are required for proper execution of the instruction represented by this sequence. Each time slot must be at least long enough for the function specified in the corresponding step to be completed. Since , the control unit is implemented by hardwire device and every device is having a propagation delay , due to which it requires some time to get the stable output signal at the output port after giving the input signal. So, to find out the time slot is a complicated task. For the moment, for simplicity, let us assume that all slots are equal in diameter. Therefore the required controller may be implemented based upon the use of a counter driven by clock. Each state, or count, of this counter corresponds to one of the steps of the control sequence of the instructions of the CPU.

In the previous discussion (Refer chapter 2, section 2.1.3), we have mentioned control sequence for execution of two instructions only (one is for add and other one is for branching). Like that we need to design the control sequence of all the instructions. By looking into the design of the CPU, we may say that there are various instruction for add operation. As for example,

| | |
|---|---|
| **Add (R3), R1** | Add the contents of memory location specified by R3 to the contents of register R1.<br>R1 ← R1 + [R3] |
| **Add R2, R1** | Add the contents of register R2 to the contents of register R1<br>R1 ← R1 + R2 |

The control sequence for execution of these two adds instructions are different of course, the fetch phase of all the instruction remain same. It is clear that control signals depend on the instruction, i.e. the contents of the instruction register. It is also observed that execution of some of the instructions depend on the contents of condition code or status flag register where the control sequence depends in conditional branch instruction. Hence, the required control signals are uniquely determined by the following information:

- Contents of the control counter.
- Contents of the instruction register.
- Contents of the condition code and other status flags.

The status flags represent the various state of the CPU and various control lines connected to it, such as MFC status signal. The structure of control unit can be represented in a simplified view by putting it in block diagram. The detailed hardware involved may be explored step by step. The simplified view of the control unit is given in the figure 6.1.

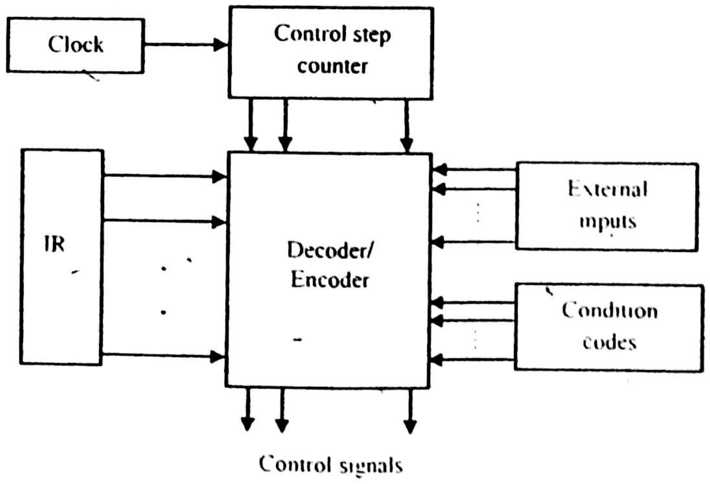


**Fig 6.1: Control Unit Organization**

The decoder/encoder block is simply a combinational circuit that generates the required control outputs depending on the state of all its input. The decoder part of decoder/encoder part provides a separate signal line for each control step, or time slot in the control sequence. Similarly, the output of the instruction decoder consists of a separate line for each machine instruction loaded in the IR, one of the output lines $INS_1$ to $INS_n$ be set to 1 and all other lines are set to 0.

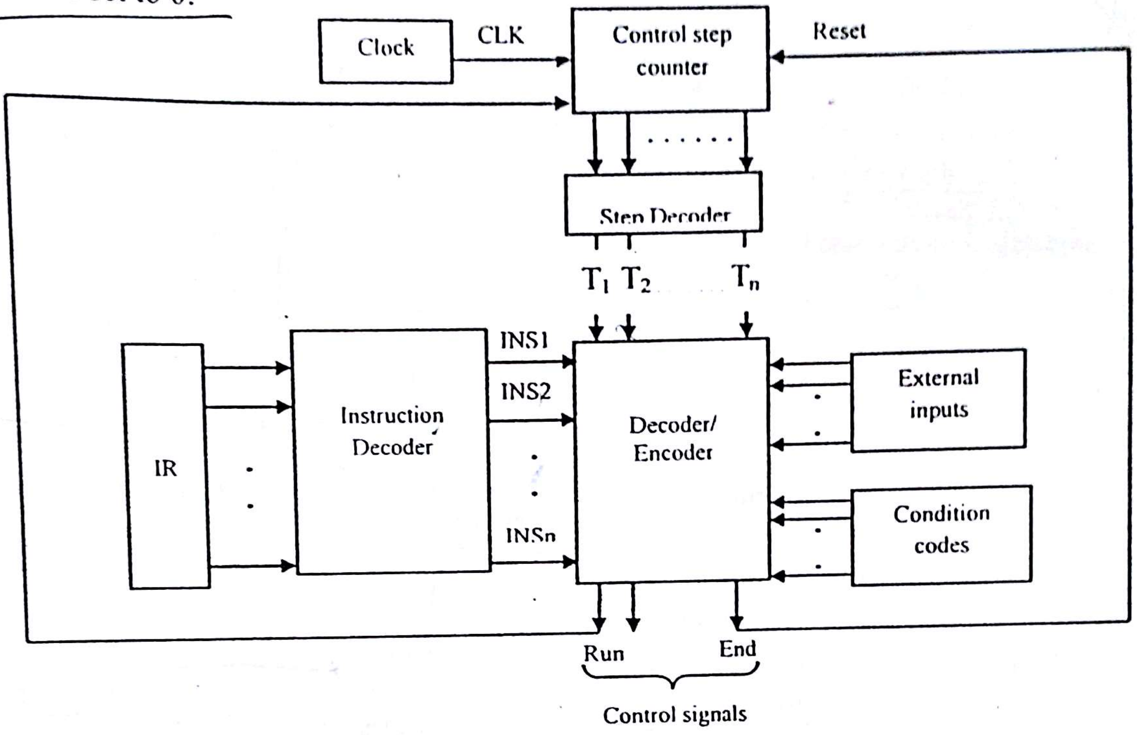


**Fig 6.2: Control Unit Organization**

All input signals to the encoder block should be combined to generate the individual control signals. Control step counter keeps tracks of the count of control steps. Each state or count of this control step corresponds to one control step. The step decoder provides a separate signal line for each step or time slot in the control sequence. The output of the instruction decoder consists of a separate signal line for each instruction. For any instruction loaded in the IR, one of the output lines $INS_1$ through $INS_n$ is set to 1 and all other lines are set to zero. The encoder circuit combines all of these inputs and generates the control signals $Y_{in}$, $Z_{out}$ etc.

It is required to generate many control signals by the control unit. These are basically coming out from the encoder circuit of the control signal generator. The control signals are lie $PC_{in}$ , $PC_{out}$ , $Z_{in}$ , $Z_{out}$ , $MAR_{in}$ , add , End etc. In the previous discussions (chapter 2), we have mentioned the control sequence of the instruction, "Add (R3), R1", and "Control sequence for an unconditional branch instruction (BR)". Consider those CPU instructions.

By looking into the above instructions, we can write the logic function for $Z_{in}$ as:

$$Z_{in} = T_1 + T_6.ADD + T_4.BR + \ldots\ldots\ldots\ldots$$

This means, control signal $Z_{in}$ have to be generated for time cycles T1for all instructions, in time cycle T6 for add instruction and in time cycle $T_4$ of BR instruction and so on.

Similarly, the Boolean logic function for add signal is

$$add = T_1 + T_6.ADD + T_4.BR + \dots\dots\dots\dots$$

These logic functions can be implemented by a two level combinational circuit of AND and OR gates. Similarly, the End control signal is generated by the logic function:

$$End = T_7.\ ADD + T_5.BR + (T_5.N + T_4.\overline{N}).BRN + \dots\dots\dots\dots$$

This End signal indicates the End of the execution of an instruction, so this End signal can be used to start a new instruction fetch cycle by resetting the control step counter to its starting value. The circuit diagram (Partial) for generating $Z_{in}$ and End signal is shown in the diagram.

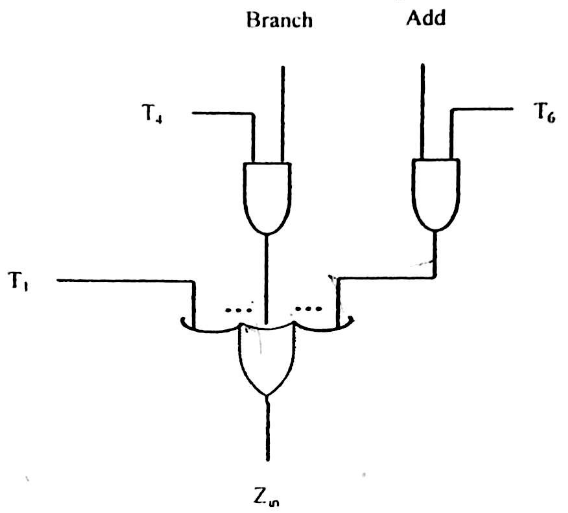


**Fig 6.3: Generation of $Z_{in}$ signal for the processor**

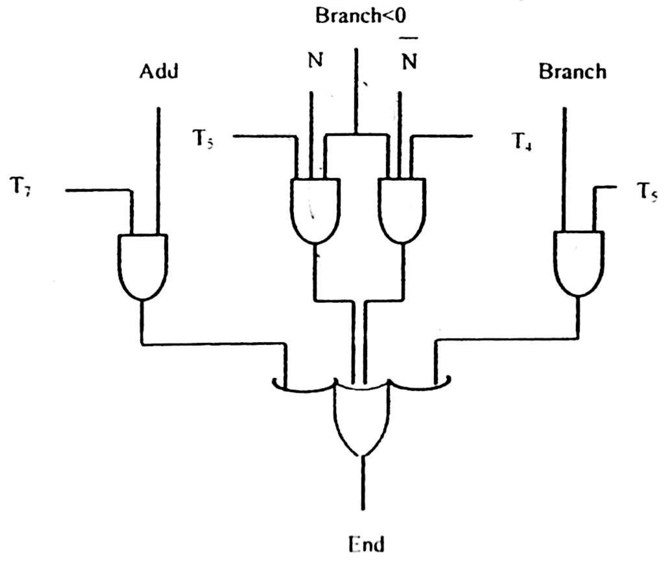


**Fig 6.4: Generation of the End control signal**

The signal ADD, BR, BRN etc are coming from instruction decoder circuits which depends on the contents of IR. The signal $T_1$, $T_2$, $T_3$ etc are coming out from step decoder depends on control step counter. The signal N (Negative) is coming from condition code register. When wait for MFC (WMFC) signal is generated, then CPU does not do any works and it waits for an MFC signal from memory unit. In this case, the desired effect is to delay the initiation of the next control step until the MFC signal is received from the main memory. This can be incorporated by inhibiting the advancement of the control step counter for the required period. Let us assume that the control step counter is controlled by a signal called RUN. When set to 1, RUN causes the counter to be incremented by one at the end of every clock cycle. When RUN equal to zero, the counter stops counting. This is needed whenever the WMFC signal is issued, ~~to cause the processor to wait for the~~ reply from the memory. End signal starts a new instruction fetch cycle by resetting the control step counter to its starting value. The control hardwire we have discussed now 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, condition codes and external inputs. The output of the state machines are the control signals. The sequence of operations carried ~~out by this~~ machine is determined by the wiring of the logic elements; hence it is named as "Hardwired unit".

**Advantages**

1. They Operates at high speed.

**Disadvantages**

1. Have little flexibility.
2. Complexity of the instruction set it can implement is limited.

**A complete Processor**

A complete processor can be designed using the following structure as shown in figure 6.5.This structure has an instruction unit that fetches instruction from an instruction cache or from the main memory. To deal with integer and floating point data it has separate processing units. A data cache is inserted between the main memory and integer and floating point units. The processor is connected to the system bus and hence to the rest of the computer by means of a bus interface. A processor may include several units of integer and floating point units so that the rate of instruction execution will be increased.

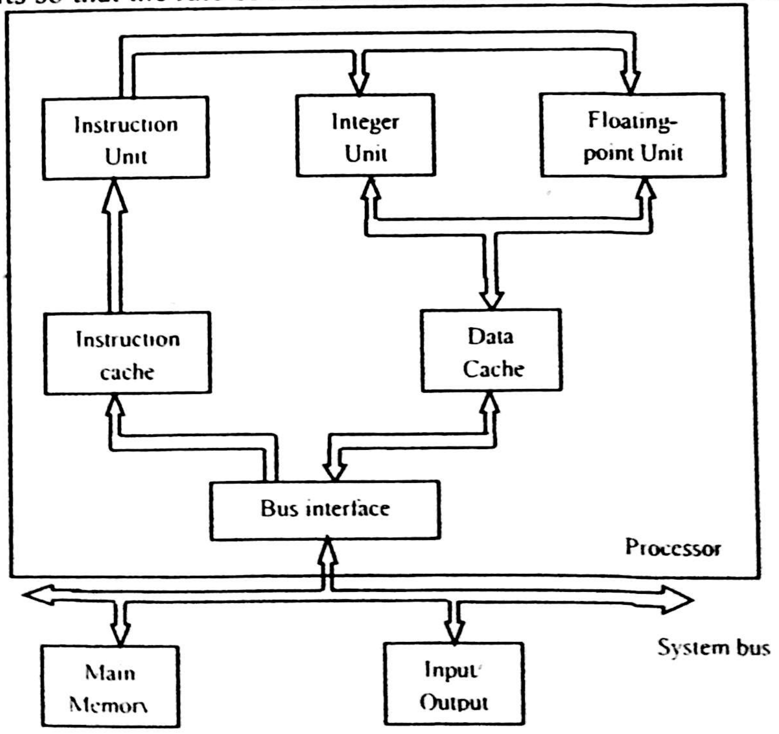


Fig 6.5: Block diagram of a complete Processor

### 6.1.3 Control of Processor Unit

In this section we are discussing the hardwire configuration of micro program organization to include the control of entire processing unit. Processor unit consists of ALU, Shifter, status register, general purpose registers etc. A micro operation is selected with a control word of 16 bits.

A micro program organization for controlling the processor unit is shown in the following figure 6.6. It has a control memory of 64 words, with 26 bits per word. For selecting 64 words, we need 6 bits address. To select 8 status bits, we need 3 selection lines for the multiplexer.

One bit of the microinstruction selects between an external address and the address field of the microinstruction. 16 bits is in need for selecting the microoperation in the processor.

The diagram shows the connection of processor unit to micro program control unit. The first 16 bits of the microinstruction selects the microoperation for the processor. The other 10 bits selects the next address for the Control Address Register (CAR). So, total 26 bits for each microinstruction. The status bits from the processor are applied to the inputs of a multiplexer. Both the normal and complement values are used (except status bit V).

Input 0 of MUX2 is always a binary 1. The load input to CAR is enabled when this input is selected by bits 18, 19 and 20 in the microinstruction. This causes a transfer of information from the output of MUX 1 into CAR. The input into CAR is a function of bit 17 in the microinstruction. If bit 17 is 1, CAR receives the address field of the microinstruction. If bit 17 is 0, an external address is loaded into CAR (this is needed for initiating a new sequence of microinstructions).

The status bits selected by bits 18, 19 and 20 of the microinstruction may be equal to 1 or 0. If the selected bit is 1, the input address is loaded into CAR. If the selected bit is 0, CAR is incremented.

To construct correct micro programs, it is necessary to specify exactly how the status bits are affected by each microoperation in the processor. The sign (S) and Zero (Z) flags are affected by all operations. The C (carry) and V (overflow) bits do not change after the following ALU operations.

1. The four logic operations OR, AND, XOR and complement.
2. The increment and decrement operations.

For all other operations, the output carry from the ALU goes into the C bit of the status register. The C bit is also affected after a circular shift with carry operation.

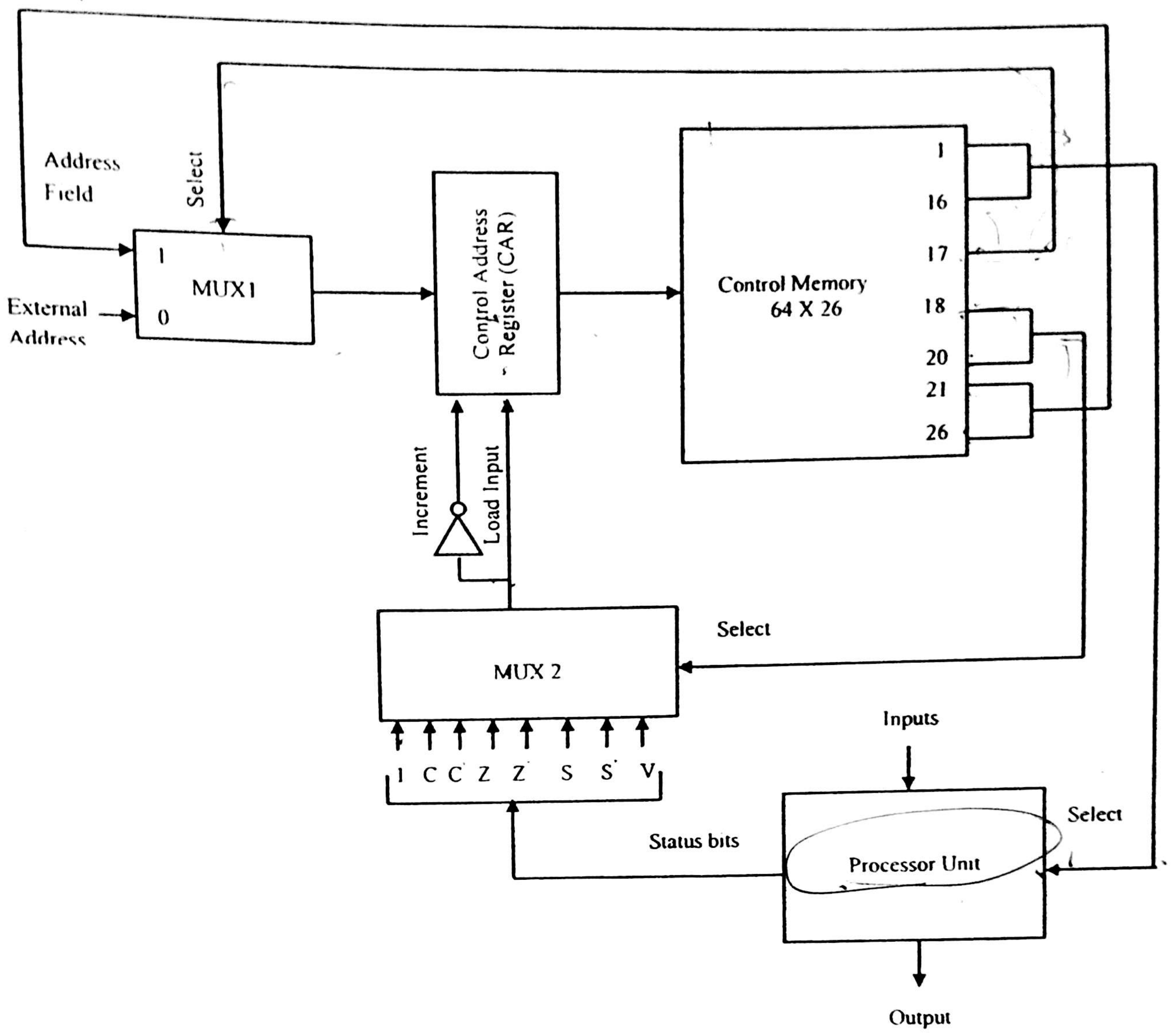


Fig 6.6: A Micro program organization for controlling the processor unit

## 6.1.4 PLA Control (Programmable Logic Array Control)

A programmable logic array is a large scale integration device that can implement any ~~complex combinational circuit~~. In PLA control, all combinational circuits are implemented with a PLA (Programmable Logic Array) including the decoder and decision logic, so that the number of ICs and number of interconnection wires can be reduced.

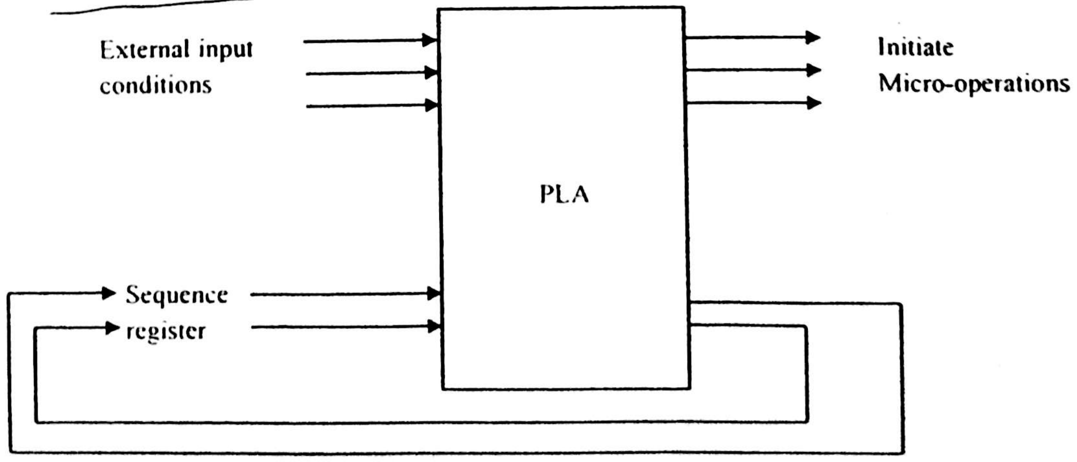


Fig 6.7: Block diagram of PLA control

Here, sequence register establishes the present state of the control circuit. Depending on the present state of the control circuit and external input conditions, PLA will determine the micro operations which have to be initiated. At the same time, other PLA outputs determine the next state of sequence register. The sequence register is external to the PLA if the unit implements only combinational circuits.

## 6.2 Micro-Programmed Control

Here the control signals are generated by a program similar to machine language programs instead of using a decoder/encoder circuit.

### 6.2.1 Microinstructions

Micro instruction is normally structured as assign one bit position to each control signal

Consider the following control sequence:

1. $PC_{out}$, $MAR_{in}$, Read, Select4, Add
2. $Z_{out}$, $PC_{in}$, $Y_{in}$ ,WMFC
3. $MDR_{out}$, $IR_{in}$
4. $R3_{out}$, $MAR_{in}$, Read
5. $R1_{out}$,$Y_{in}$,WMFC
6. $MDR_{out}$, Select4,Add, $Z_{in}$
7. $Z_{out}$,$R1_{in}$,End

Here so many control signals are there like PCin, PCout, $MAR_{in}$, $Z_{in}$, $Z_{out}$ etc. The following table shows the micro instructions corresponding to the given control sequence.

| Micro instructions | .... | $PC_{in}$ | $PC_{out}$ | $MAR_{in}$ | Read | Add | $Z_{in}$ | $Z_{out}$ | $MDR_{out}$ | $IR_{in}$ | $R3_{out}$ | $R1_{out}$ | $Y_{in}$ | $R1_{in}$ | WMFC | End |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 3 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 5 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 6 | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

control signals

controlo

Table 6.1: An example of micro instruction for the above control sequence

Each ~~row in~~ this table is referred as a control word. Individual bits in the control word represent various control signals. Each control word is a combination of sequence of 0's and 1's. A sequence of control words corresponding to the control sequence of a machine instruction is referred as micro routine. Individual control words in the micro routine are called micro instructions.

Microroutine

## 6.2.2 Horizontal and Vertical Microinstructions

Microinstructions can be organized in two different ways named as Horizontal and Vertical microinstructions

### Horizontal Microinstructions

The above represented scheme of micro instruction by assigning one bit position to each control signal is known as horizontal microinstruction.

**Example**

011100010110010

**Drawback of this scheme**

Assigning individual bits to each control signal results in long micro instructions because the number of required signals is usually large. For any given micro instruction, only few bits are set to 1, which means available bit space is purely used.

In the earlier discussion of the processor organization we had discussed, it needs total of 42 control signals. (Assuming that here four general purpose registers R0 to R3.)If we are using the simple encoding scheme now we have discussed, total of 42 bits would be needed for each micro instruction.

### Vertical Microinstructions

We can reduce the length of the horizontal micro instructions so easily by implementing another method known as vertical micro instructions. In all cases, most signals are not needed simultaneously, and many signals are mutually exclusive.

For example, only one function of the ALU can be activated at a time. The source for a data transfer must be unique because it is not possible to gate the contents of two different registers on to the bus at the same time. Read and write signals to the memory cannot be active simultaneously. All of these cases suggest that signals can be grouped so that all mutually exclusive signals are placed in the same group. A binary coding scheme can be used to represent the signals within a group. The following example will illustrate this.

**Micro Instruction**

| F1 | F2 | F3 | F4 | F5 |
|---|---|---|---|---|

| F1 (4 bits) | F2 (3 bits) | F3 (3 bits) | F4 (4 bits) | F5 (2bits) |
|---|---|---|---|---|
| 0000: No Transfer | 000: No Transfer | 000: No Transfer | 0000: Add | 00: No Action |
| 0001: $PC_{out}$ | 001: $PC_{in}$ | 001: $MAR_{in}$ | 0001: Sub | 01: Read |
| 0010: $MDR_{out}$ | 010: $IR_{in}$ | 010: $MDR_{in}$ | . | 10: Write |
| 0011: $Z_{out}$ | 011: $Z_{in}$ | 011: $TEMP_{in}$ | . | |
| 0100:$R0_{out}$ | 100:$R0_{in}$ | 100: $Y_{in}$ | $\underbrace{\text{1111: XOR}}_{\text{16 ALU functions}}$ | |
| 0101: $R1_{out}$ | 101:$R1_{in}$ | | | |
| 0110:$R2_{out}$ | 110:$R2_{in}$ | | | |
| 0111:$R3_{out}$ | 111:$R3_{in}$ | | | |
| 1010: $TEMP_{out}$ | | | | |
| 1011: $Offset_{out}$ | | | | |

| F6 | F7 | F8 | ........ |
|---|---|---|---|

| F6 (1 bit) | F7 (1 Bit) | F8 (1 bit) |
|---|---|---|
| 0: Select Y | 0: No Action | 0: Continue |
| 1: Select 4 | 1: WMFC | 1: End |

Here most of the fields must include one inactive code for the case in which no action is required. For example, all zero patterns in F1 indicate that none of the registers that may be specified in this field should have its contents placed on the bus. An inactive code is not needed in all fields. For example, F4 contains 4 bits that specify one of the 16 operations performed in the ALU. Since no spare code is included, the ALU is active during the execution of every micro instruction. This type of highly encoded scheme which uses compact codes to specify control functions in each micro instruction is referred as vertical micro instruction.

**Comparison of Horizontal and vertical micro instructions**

The horizontal approach is useful when a higher operating speed is desired and when the machine structure allows parallel use of resources. But in vertical micro instruction, grouping control signals into fields requires a little more hardware because decoding circuits must be used to decode the bit patterns of each field into individual control signals.

The cost of this additional hardware is more. The vertical approach results in considerably slower operating speeds because more micro instructions are needed to perform the desired control functions. To handle the execution of micro instructions only less hardware is needed.

## Design of Micro Programmed Control

The micro routines for all instructions in the instruction set of a computer are stored in a s special memory called control store. Control unit will generate the control signals by sequentially reading the control words of the corresponding micro routine from the control store. For this sequential reading micro programmed control uses a Micro program Counter. The following figure illustrates the working of micro programmed organization.

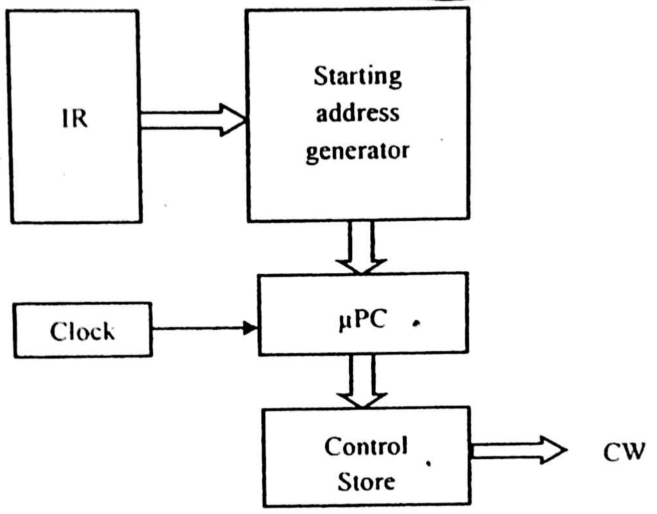


**Fig 6.8: Basic organization of a micro programmed control unit**

Every time a new instruction is loaded into IR, the output of the starting address generator is loaded into the micro program counter. This counter is automatically incremented by the clock. So that successive micro instructions can be read from the control store. These control signals are delivered to various parts of the processor in correct sequence. By using this design, it is not possible to implement the complete function of the control unit. That is, when the control unit have to check the status of condition codes or external inputs, this organization will not work. In order to implement it, the above design can be little bit modified as follows:

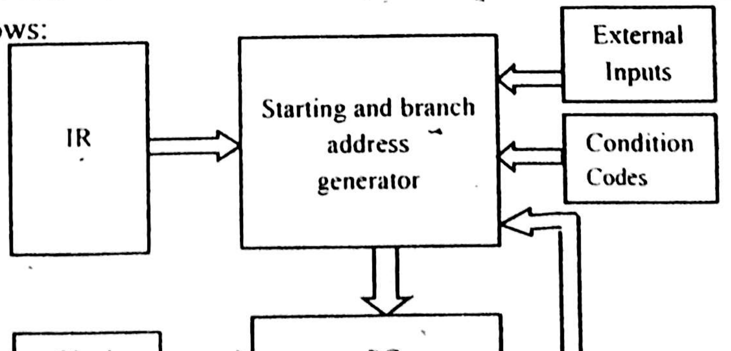


**Fig 6.9: Organization of the control unit to allow the conditional branching in the micro program.**

Here, starting address generator becomes the starting and branch address generator. This block loads a new address into the micro program counter when the micro instruction instructs it to do so. Condition codes and external inputs are also taken into consideration. Micro program counter is incremented every time a new micro instruction is fetched from the micro program memory, except in the following situations:

- When a new instruction is loaded into IR, micro program counter is loaded with the starting address of the micro routine for that instruction.
- When branch micro instruction is encountered and the branch condition is satisfied, counter is loaded with the branch address.
- When an End micro instruction is encountered, counter is loaded with the address of the first control word in the micro routine for the instruction fetch cycle.

## 6.2.3 Micro Program Sequencer

A micro program control unit consisting of two parts: control memory that stores the microinstructions and the associated circuits that control the generation of the next address. The address generation part is called micro program sequencer, since it sequences the microinstructions in control memory. Microprogram sequencer is attached to the control memory. It inspects certain bits in the microinstruction to determine the next address for control memory. A typical sequencer has the following address sequencing capabilities.

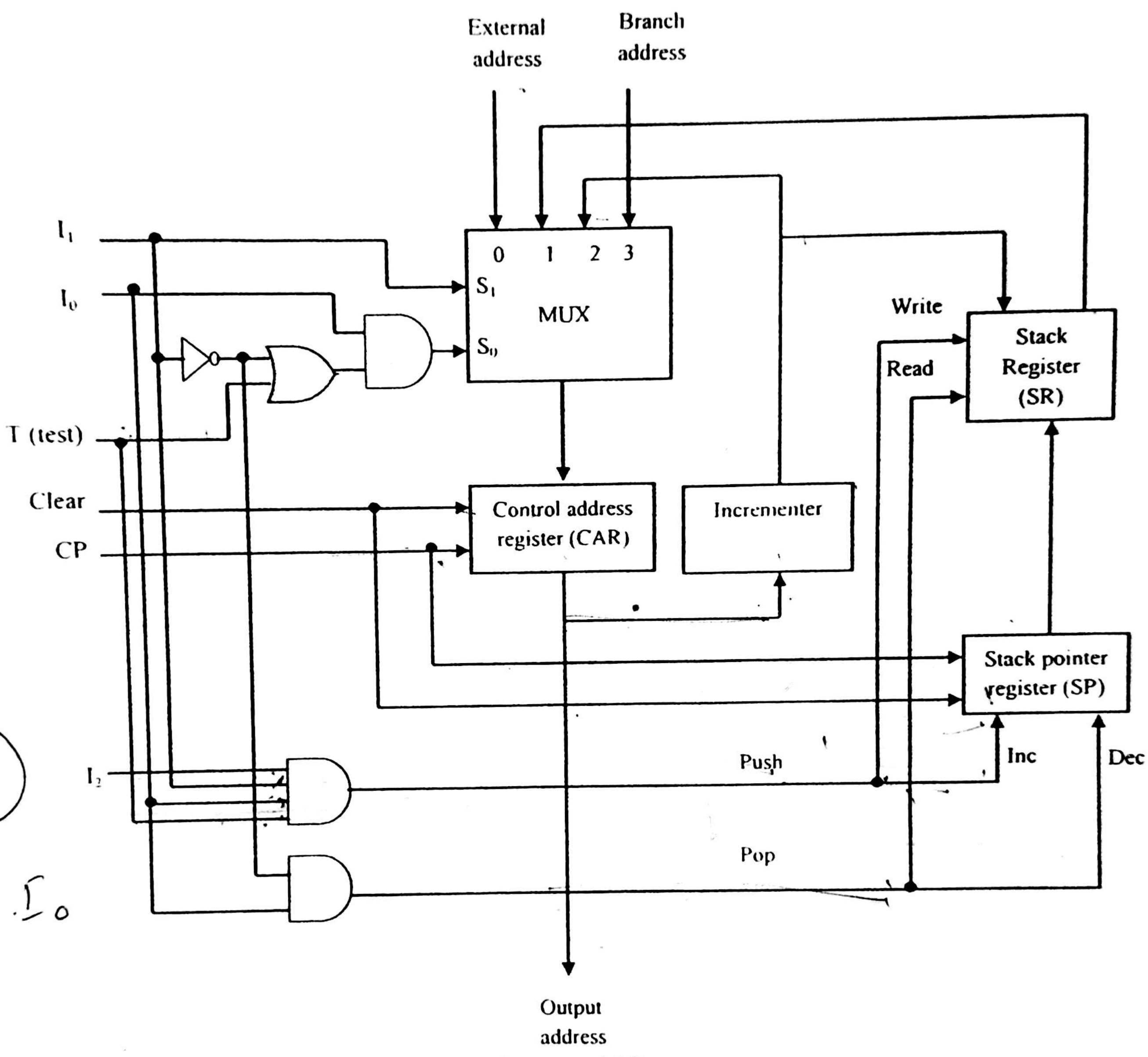
1. Increments the present address of control memory
2. Branches to an address which will be specified in the bits of microinstruction
3. Branches to a given address if a specified status bit is equal to 1.
4. Transfers control to a new address as specified by an external source
5. Has a facility for subroutines calls and returns.

In majority cases successive microinstructions are read from the control memory. This type of sequencing is easy by incrementing the address register of the control memory. But in some cases, address may also be a part of microinstructions. In such cases, even the incrementer also may not be needed. In some cases, control must be transferred to a non sequential microinstruction. To handle such situations sequencer must have branching capability. This is usually performed by the sequencer by looking up a special status bit. If it is 1, sequencer will transfer to the branch address which will be specified in the microinstruction. If it is 0, go to the next address in sequence.

To start a new microoperation, a new address has to be loaded. In such cases, sequencer will load new address in address register of control memory. This will be done by receiving the new address from an external input source.

Many microprograms may contain identical sections of code. Space may be wasted if the common sections are repeated in each microprogram. Instead of that such portions may shared by employing subroutines, so that we can save the control memory space. Microprograms that use subroutines must have a provision for storing the return address during a subroutine call and restoring the address during a subroutine return. This can be accomplished by placing the return address in a special register and then branching to the beginning of the subroutine. The best way to organize the register file is in LIFO (Last In First Out fashion.).

The block diagram of microprogram sequencer is shown in the below figure:



$(\bar{I}_1 + T)$

$I_2 . I_1 . T . I_0$

$I_0 . \bar{I}_1$

**Fig 6.10: Block diagram of Microprogram sequencer**

It consists of a multiplexer that selects an address from four sources and routes it into a control address register. The O/P from CAR provides the address for control memory. The contents of CAR are incremented and applied to the multiplexer and to the stack register file.

The register selected in the stack is determined by stack pointer. Inputs ($I_0$-$I_2$) specify the operation for the sequencer and input T is the test point for a status bit. Initially the address register is cleared to zero and clock pulse synchronizes the loading into registers.

The following function diagram illustrates the operation of a sequencer.

| $I_2$ | $I_1$ | $I_0$ | T | $S_1$ | $S_0$ | Operations | Comments |
|---|---|---|---|---|---|---|---|
| X | 0 | 0 | X | 0 | 0 | CAR ← EXA | Transfer external address |
| X | 0 | 1 | X | 0 | 1 | CAR ← SR | Transfer from register stack |
| X | 1 | 0 | X | 1 | 0 | CAR ← CAR+1 | Increment address |
| 0 | 1 | 1 | 0 | 1 | 0 | CAR ← CAR+1 | Increment address |
| 0 | 1 | 1 | 1 | 1 | 1 | CAR ← BAR | Transfer branch address |
| 1 | 1 | 1 | 0 | 1 | 0 | CAR ← CAR+1 | Increment address |
| 1 | 1 | 1 | 1 | 1 | 1 | CAR ← BRA, SR← CAR+1 | Branch to subroutine |

**Table 6.2: Function table for microprogram sequencer**

### 6.2.4 Microprogrammed CPU Organization

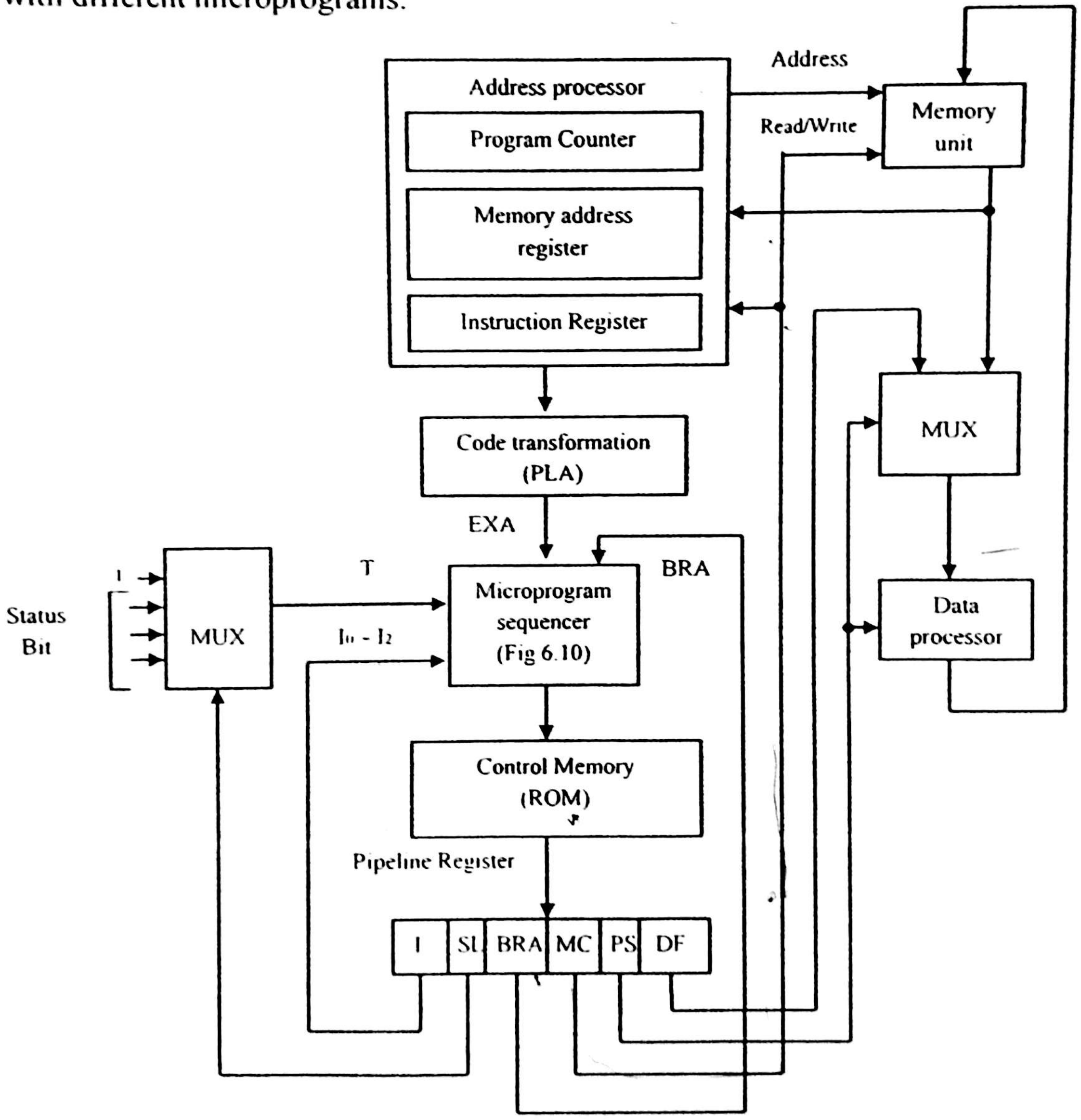
A digital computer consists of a Central Processor Unit (CPU), a memory unit, and input-output devices. CPU can be classified into processing section and control section. Here we are designing a computer CPU with micro programmed control. Microprogram sequencer is a basic element of micro programmed control for a CPU. The block diagram of the microprogrammed computer is shown below figure 6.11.

It consists of a memory unit, two processor units (one for address and one for data), a microprogram sequencer, a control memory and few other digital functions. The memory unit stores the instructions and data supplied by the user through an input device. The data processor manipulates the data and address processor manipulates the address information received from memory. Provision is also there to combine these two units into one.

The instruction extracted from the memory during fetch cycle goes into the instruction register. The instruction-code bits in the instruction register specify a macrooperation for control memory. A coded transformation is needed sometimes to convert the operation code bits of an instruction into a starting address for the control memory. This code transformation consists of a mapping function and can be implemented with a ROM or PLA. Mapping function is giving the flexibility for adding instructions or macrooperations for control memory as in need. The address generated from the PLA is applied to the External Address (EXA) input of the sequencer.

The micro program control unit consists of a control memory (for storing the microinstructions), a multiplexer and a pipeline register. The multiplexer selects the status bits and applies it to the test input of the sequencer. One input of the multiplexer is always 1 to trigger an unconditional branch instruction. The outputs from the control memory can go directly to the control inputs of the various part of the CPU. Pipeline register is an optional one. But it may speed up the control operation.

A possible micro instruction format is shown within the pipeline register. I field consists of 3 bits and supplies the input information to the sequencer. SL bits are status bits for the multiplexer. BRA field is supplies a branch address to the sequencer. Using these 3 fields microprogram sequencer is determining the next address for control memory. The other fields are for controlling the microoperations within the CPU and memory. The MC (Memory Control) bits control the address processor and the read and write operations in the memory. PS (Processor Select) bits control the operation in data processor. The DF (Data Field) bits are used to introduce constants into the processor. Data field can be used as a sequence counter to count the number of times a microprogram loop is traversed. Data field outputs can be used to setup control registers and to ~~introduce data~~ in processor registers. For example, a constant in the data field may be added to a processor register to increment its contents by a specified value. Once the hardwire configuration of microprogrammed CPU is completed, the designer can select one of the computer configuration. Initially the instructions set have to be formulated and corresponding microinstructions will be set up and stored in control memory. No hardware changes are required for a different computer configuration. Only the ROM has to be changed. This can be possible by replacing the ROM with different microprograms.



| | |
|---|---|
| I | Sequence selector |
| SL | MUX selector |
| BRA | Branch address |
| MC | Memory control |
| PS | Processor selector |
| DF | Data field |

**Fig 6.11: Microprogrammed computer organization**