Module: 4	**CONTROL LOGIC DESIGN:** Control organization, Hardwired control, microprogram control, control of processor unit, micro-program sequencer, micro-programmed CPU organization, horizontal and vertical micro instructions,

CONTROL LOGIC DESIGN

The process of logic design is a complex undertaking. The binary information found in a digital system is stored in processor or memory registers, and it can be either data or control information.

The logic design of a digital system is a process for deriving the digital circuits that perform data processing and the digital circuits that provide control signals. The timing for all registers in a synchronous digital system is controlled by a master-clock generator. The clock pulses are applied to all flip-flops and registers in the system, including the flip-flops and registers in the control unit.

Two representations which are helpful in the design of systems that need a control are timing diagrams and flowcharts. A ***timing diagram*** clarifies the timing sequence and other relationships among the various control signals in the system.A ***flowchart*** is a convenient way to specify the sequence of procedural steps and decision paths for an algorithm.

The design of control logic cannot be separated from the algorithmic development necessary for solving a design problem. Moreover, the control logic is directly related to the data-processor part of the system that it controls.

CONTROL ORGANIZATION

Once a control sequence has been established, the sequential system that implements the control operations must be designed. Since the control is a sequential circuit, it can be designed by a sequential logic procedure.

Disadvantages of sequential control logic are

- Large number of states
- Excessive number of flip-flops and gates
- Design methods uses state and excitation tables but in practice they are cumbersome

Goal of control logic design should be development of a circuit that implements the desired control sequence in a logical andstraightforward 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.

We consider four methods of control organization

- One flip-flop per state methods
- Sequence register and decoder method
- PLA control
- Micro-program control

The first two methods result in a circuit that must use SSI and MSI circuits for the implementation. A control unit implemented with SSI and MSI devices is said to be a hard-wired control. If any alterations or modifications are needed, the circuits must be rewired to fulfill the new requirements.

The PLA or micro-program control which uses an LSI device such as a **Programmable Logic Array** or a Read-Only Memory. Any alterations or modifications in a micro-program control can be easily achieved without wiring changes by removing the ROM from its socket and inserting another ROM programmed to fulfill the new specifications.

One flip-flop per state methods

This method uses one flip-flop per state in the control sequential circuit. Only 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 the other under the control of decision logic. In such an array, each flip-flop represents a state and is activated only when the control bit is transferred to it.

In this method, maximum numbers of flip-flops were used. Example: A sequential circuit with 12 states requires a minimum of four flip-flops because $2^3 < 12 < 2^4$. Control circuit uses 12 flip-flops, one for each state

The ***advantage*** of this method is the simplicity with which it can be designed. This type of controller can be designed by inspection from the state diagram that describes the control sequence. This also offers other advantages like savings in design effort, an increase in operational simplicity, and a potential decrease in the combinational circuits required to implement the complete sequential circuit. The ***disadvantage*** is that this method would increase system cost since more flip-flops are used.

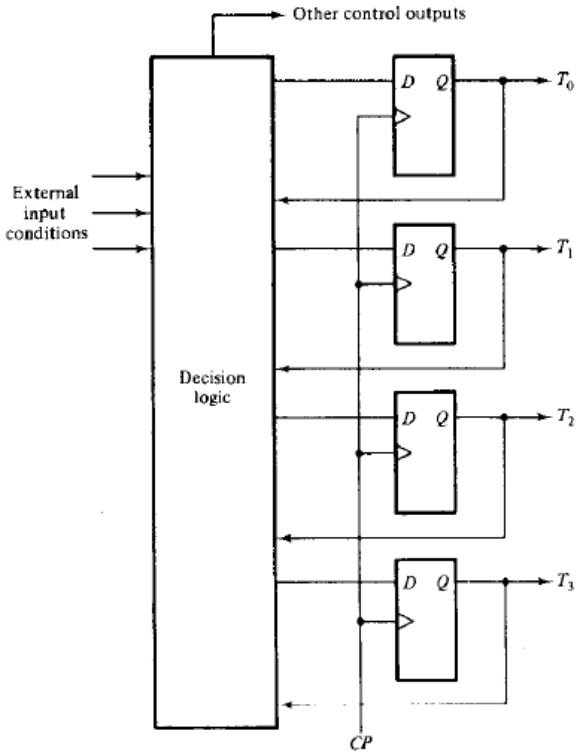


Figure shows the configuration of a four-state sequential control logic that uses four D-type flip-flops: one flip-flop per state T_i, i = 0, 1, 2, 3.

At any given time interval between 2 clock pulses, only one flip-flop is equal to 1; all others are 0. The transition 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-flops is connected to the data processing section of the digital system to initiate certain micro-operations. The control outputs are a function of the T's and external inputs. These outputs may also initiate micro-operations

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 over 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 continuously shifts from one position to the next in a circular manner.

For this reason, this method is also called **ring counter controller**

Sequence Register and Decoder Method

This method uses a 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" states and the decoder will have 2" outputs.

For example, a 4-bit register can be in any one of 16 states. A 4 x 16 decoder will have 16 outputs, one for each state of the register. Both the sequence register and decoder are MS1 devices.

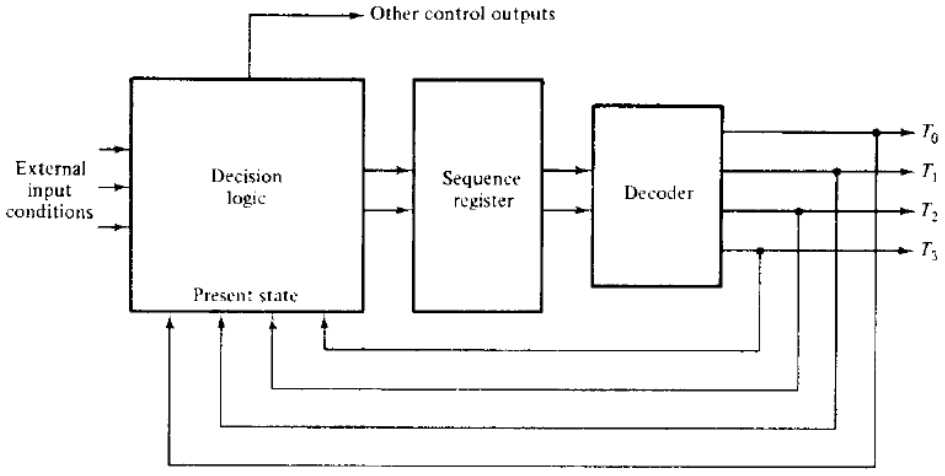
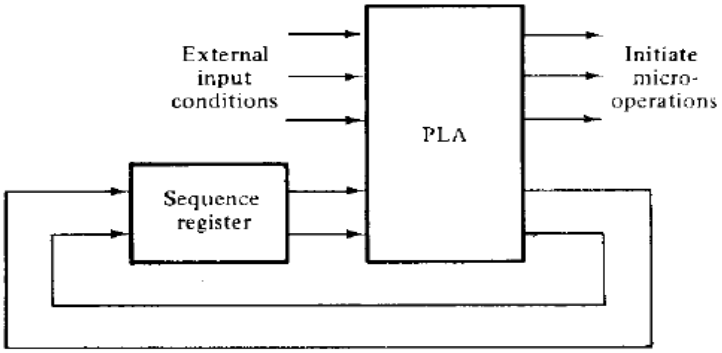


Figure shows the configuration of a four-state sequential control logic. The sequence register has two flip-flops and the decoder establishes separate outputs for each state in the register. The transition to the next state in the sequence register is a function of the present state and the external input conditions.

If the control circuit does not need external inputs the sequence register reduces to a counter that continuously sequence through the four states so called **counter decoder method**

PLA control

The external sequence register establishes the present state of the control circuit. The PLA outputs determine which micro-operations 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 sequence register 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 implements a sequential circuit by specifying the links that must be connected to the flip-flops in manner that the gate links are specified

Micro-program Control

The purpose of control unit is to initiate a series of sequential steps of micro-operations. At any given time certain operations are to be initiated while all others remain idle. The control variable at any given time can be represented by a string of 1's and 0's called ***control word***. The 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 ***micro-programmed control*** unit. Each control word of memory is called ***Microinstruction*** and Sequence of microinstructions is called ***Micro-program***.

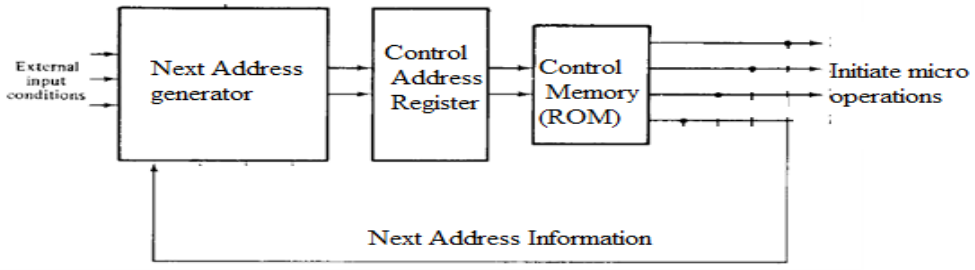
Control memory is usually ROM since an alteration of micro-program is seldom needed. The use of micro-program involves placing all control variables in words of the ROM for use by the control unit through successive read operations. The content of the word in the ROM at a given address specifies the micro-operations for the system.

Dynamic micro-programming permits a micro-program to be loaded initially from the computer console or from an auxiliary memory such as 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

Control memory address register specifies the control word read from control memory. The ROM operates as a combinational 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 address register.

If the address registers changes while the ROM word is still in use then the word out of the ROM should be transferred to a buffer register. 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 micro-operations for the components of the system.



Once these operations are executed, the control unit must determine its next address. The location of the next microinstruction may be next one in the sequence or it may locate somewhere else in the control memory. 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 address generator circuit and then transferred into the control address register to read the next microinstruction.

HARDWIRED CONTROL

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”. Control logic derived in this section is a hardwired control of the one flip-flop per state method. The 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

The problem here is to implement with hardware the addition and subtraction of two fixed-point binary numbers represented in sign-magnitude form. Complement arithmetic may be used, provided the final result is in sign-magnitude form.

The addition of two numbers stored in registers of finite length may result in a sum that exceeds the storage capacity of the register by one bit. The extra bit is said to cause an overflow. The circuit must provide a flip-flop for storing a possible overflow bit.

Equipment Configuration

The two signed binary numbers to be added or subtracted contain n bits. The magnitudes of the numbers contain k = n - 1 bits and are stored in registers A and B. The sign bits are stored in flip-flops A_s and B_s.

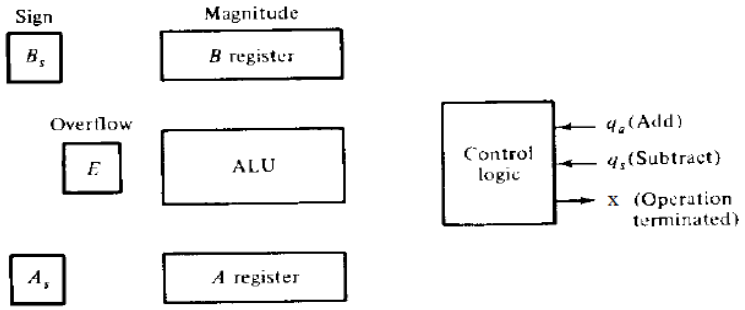


Figure 10-6 Register configuration for the adder-subtractor

Figure shows the registers and associated equipment. The ALU performs the arithmetic operations and the 1-bit register E serves as the overflow flip-flop. The output carry from the ALL) is transferred to E.

It is assumed that the two numbers and their signs have been transferred to their respective registers and that the result of the operation is to be available in registers A and A_s. Two input signals in the control specify the add (q_a) and subtract (q_s) operations. Output variable x indicates the end of the operation.

The control logic communicates with the outside environment through the input and output variables. Control recognizes input signal q_a or q_s and provides the required operation. Upon completion of the operation, control informs the external environment with output x that the sum or difference is in registers A and A, and that the overflow bit is in E.

Derivation of the Algorithm

Designate the magnitude of the two numbers by A and B. When the numbers are added or subtracted algebraically, there are eight different conditions to consider, depending on the sign of the numbers and the operation performed. The eight conditions may be expressed in a compact form as follows:

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

If the arithmetic operation specified is subtraction, we change the sign of B and add. This is evident from the relations:

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

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

This reduces the number of possible conditions to four, namely:

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

The four possible combination are

When the signs of A and B are the same: $(+A) + (+B) = +(A + B)$

$$(-A) + (-B) = -(A + B)$$

When the signs of A and B are not the same

$$(+A) + (-B) = +(A - B) \quad [\text{ if(A>B) }]$$

$$-(B - A) \quad [\text{ if(B>A) }]$$

$$(-A) + (+B) = -(A - B) \quad [\text{ if(A>B) }]$$

$$+(B - A) \quad [\text{ if(B>A) }]$$

The flowchart shows how we can implement sign-magnitude addition and subtraction with the equipment as shown in previous diagram.

For more details about flowchart, refer text book "Digital Logic and Computer Design" by M. Morris Mano; Page 418.

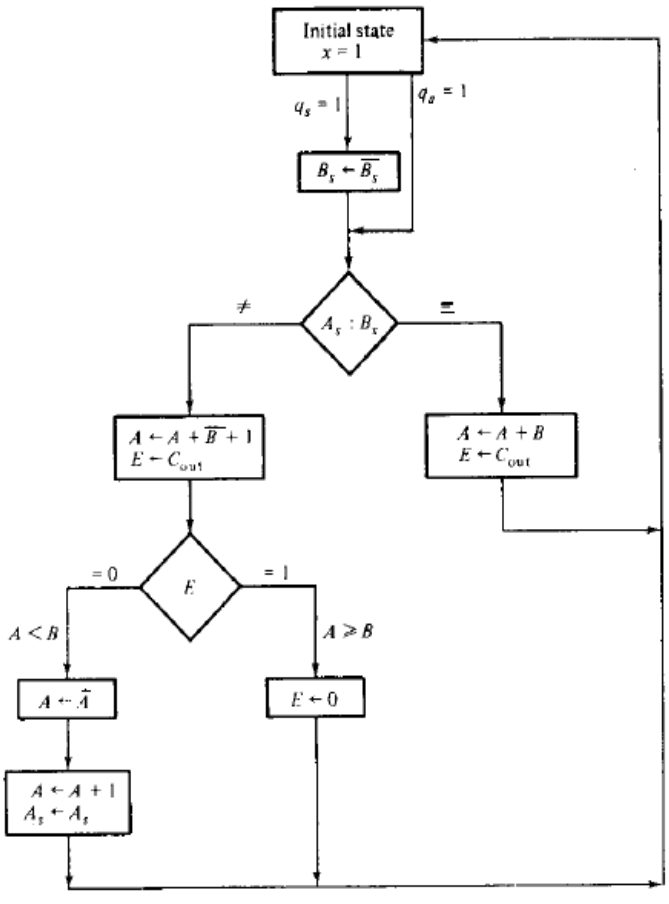


Figure 10-7 Flowchart for sign-magnitude addition and subtraction

Data Processor Specification

Figure (a) shows the data-processor with the required control variables. This ALU has four selection variables, as shown in the diagram. The variable L loads the output of the ALU into register A and also the output carry into E. Variables y, z. and w complement B, and A, and clear E, respectively.

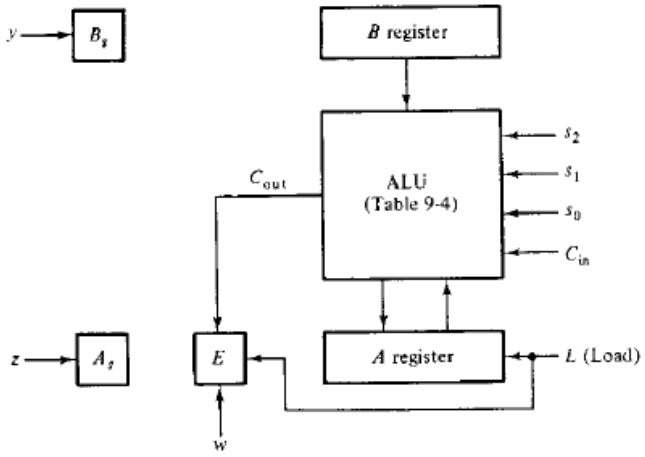


Figure a. Data Processor Register and ALU

The block diagram of the control logic is shown in figure (b). The control receives five inputs: two from the external environment and three from the data-processor. To simplify the design, we define a new variable

$$S = A_s \oplus B_s$$

This variable gives the result of the comparison between the two sign bits. The exclusive-OR operation is equal to 1 if the two signs are not the same, and it is equal to 0 if the signs are both positive or both negative. The control provides an output x for the external circuit. It also selects the operations in the ALU through the four selection variables S_2, S_1, S_0, and C_{in}. The other four outputs go to registers in the data-processor as specified in the diagram.

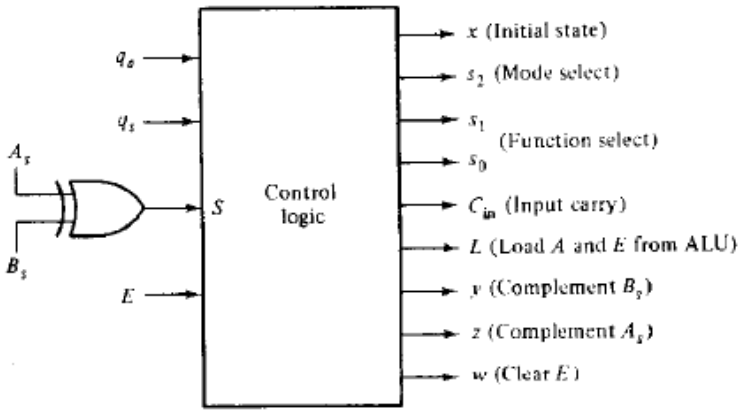


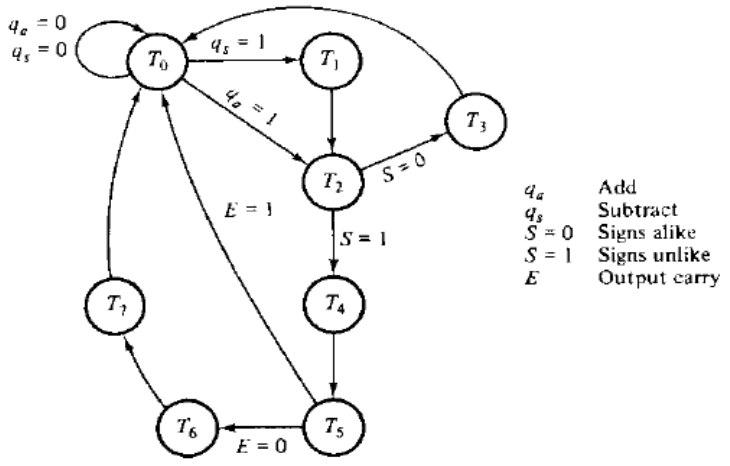
Figure b. Control Block Diagram

Control State Diagram

The design of a hard-wired control is a sequential-logic problem. As such, it may be convenient to formulate the state diagram of the sequential control. The function boxes in a flowchart may be considered as states of the sequential circuit, and the decision boxes as next-state conditions.

The micro-operations that must be executed at a given state are specified within the function box. The conditions for the next state transition are specified inside the decision box or in the directed lines between two function boxes. Consequently, different designers may produce different state diagrams for the same flowchart, and each may be a correct representation of the system. The control state diagram and the corresponding register-transfer operations are derived in below figure.

We start by assigning an initial state, To, to the sequential controller. We then determine the transition to other states T_1, T_2, T_3, and so on. For each state, we determine the micro-operations that must be initiated by the control circuit. The figure below shows the sequence of register transfers. [Refer flowchart]



The end carry is transferred to E during the subtraction, and control then goes to state T_5.

It must be realized that the end carry from the ALU is transferred to E with a clock pulse. This happens with the same clock pulse that causes the control to go from state T_4 to T_5. Although we show the microoperation:

$$E \leftarrow C_{out}$$

with timing variable T_4 this operation is not executed until a clock pulse occurs. Once this clock pulse executes the operation, control finds itself in state T_5. Therefore, the value of E for an end carry should not be checked until control reaches state T_5. The value of E is checked to determine the relative magnitudes of A and B. If $E = 1$, it indicates that $A \geq B$. For this case, E must be cleared and the operation is completed. If $E = 0$, it indicates that $A < B$. Control then goes to states T_6 and T_7 to complement A and A_s. Note that E is cleared while the control is in state T_5. This is done whether E is 1 or 0, since trying to clear a flip-flop that is already 0 leaves the flip-flop in the 0 state anyway. Note also that E is cleared with the clock pulse that causes control to go out of state T_5. It must be realized that clearing E and transferring control to state T_0 or T_6 is done with one common clock pulse without a conflict. The original value of E at time T_5 determines the next state even though this flip-flop is cleared while the clock pulse goes through an edge transition.

[Refer ALU Table in page 23 of Module 2]

	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

Design of Hardwired Control

The control can be designed using the classical sequential-logic procedure. This procedure requires a state table with eight states, four inputs, and nine outputs. The sequential circuit to be derived from such a state table will not be easy to obtain because of the large number of variables.

The circuit obtained by using this method may have a minimum number of gates, but it will have an irregular pattern and will be difficult to analyze if a malfunction occurs. These difficulties are removed if the control is designed by the one flip-flop per state method.

A control organization that uses one flip-flop per state has the convenient characteristic that the circuit can be derived directly from the state diagram by inspection. No state or excitation tables are needed if D flip-flops are employed.

Remember that the next state of a D flip-flop is a function of the D input and is independent of the present state. Since the method requires one flip-flop for each state, we choose eight D flip-flops and label their outputs T_0, T_1, T_2, . . , T_7. The condition for setting a given flip-flop is specified in the state diagram.

For example, flip-flop T_2 is set with the next clock pulse if T_1= I or if $T_0 = 1$ and $q_a = 1$. This condition can be defined with the Boolean function:

$$DT_2 = q_aT_0 + T_1$$

where DT_2 designates the D input of flip-flop T_2.If there is more than one directed line going into a state, all conditions must be ORed. Using this procedure for the other flip-flops, we obtain the input functions given in table below. [Refer flowchart and the above table]

Flip-flop input functions	Boolean functions for output control
$DT_0 = q_a'q_s'T_0 + T_3 + ET_5 + T_7$	$x = T_0$
$DT_1 = q_sT_0$	$s_2 = T_6$
$DT_2 = q_aT_0 + T_1$	$s_1 = T_4 + T_6$
$DT_3 = S'T_2$	$s_0 = T_3 + T_5$
$DT_4 = ST_2$	$C_{in} = T_4 + T_7$
$DT_5 = T_4$	$L = T_3 + T_4 + T_6 + T_7$
$DT_6 = E'T_5$	$y = T_1$
$DT_7 = T_6$	$z = T_7$
	$w = T_5$

Initially, flip-flop T_0 is set and all others are cleared. At any given time, only one D input is in the 1 state while all others are maintained at 0. The next clock pulse sets the flip-flop whose D input is 1 and clears all others.

The circuit for the control logic is not drawn but can be easily obtained from the Boolean functions in table. The circuit can be constructed with eight D flip-flops, seven AND gates, six OR gates, and four inverters. Note that five control outputs are taken directly from the flip-flop outputs.

MICROPROGRAM CONTROL

In a microprogrammed control the **control variables** that initiate **micro operations** are stored in **memory**. The control memory is usually a ROM. since the control sequence is permanent and needs no alteration. The control variable stored in memory are read one at a time to initiate the sequence of micro operations for the system.

The **words** stored in a control memory are **micro instructions** and each microinstruction specifies one or more **microoperations** for the components in the system. Once these microoperations are executed the control unit must determine its next address. Therefore, a few bits of micro instruction are used to control the generation of the address for the next microinstruction. Thus, a micro instruction contains bits for **initiating microoperations** and bits that **determine the next address** for the control memory itself. In addition to the control memory, a microprogram control unit must include special circuits for selecting the next address as specified by the micro instruction.

The digital system considered here is too small for a microprogram controller and in practice hardwired control would be more efficient. The micro program control organisation is more efficient in large and complicated systems.

A state in control memory is represented by the address of a micro instruction. An **address** for control memory specifies a **control word** within a microinstruction, just as a **state** in a sequential circuit specifies a **microoperation.** Since there are 8 states in the control, we choose a control memory with 8 words having addresses 0 through 7. The address of the control memory corresponds to subscript number under the T's in the state diagram

Inspection of the state diagram reveals that the address sequencing in the microprogram control must have the following capabilities

1. Provision for loading an external address as a result of the occurrence of external signals qa and qs.
2. Provision for sequencing consecutive addresses
3. Provision for choosing between two addresses as a function of present values of the status variables S and E.

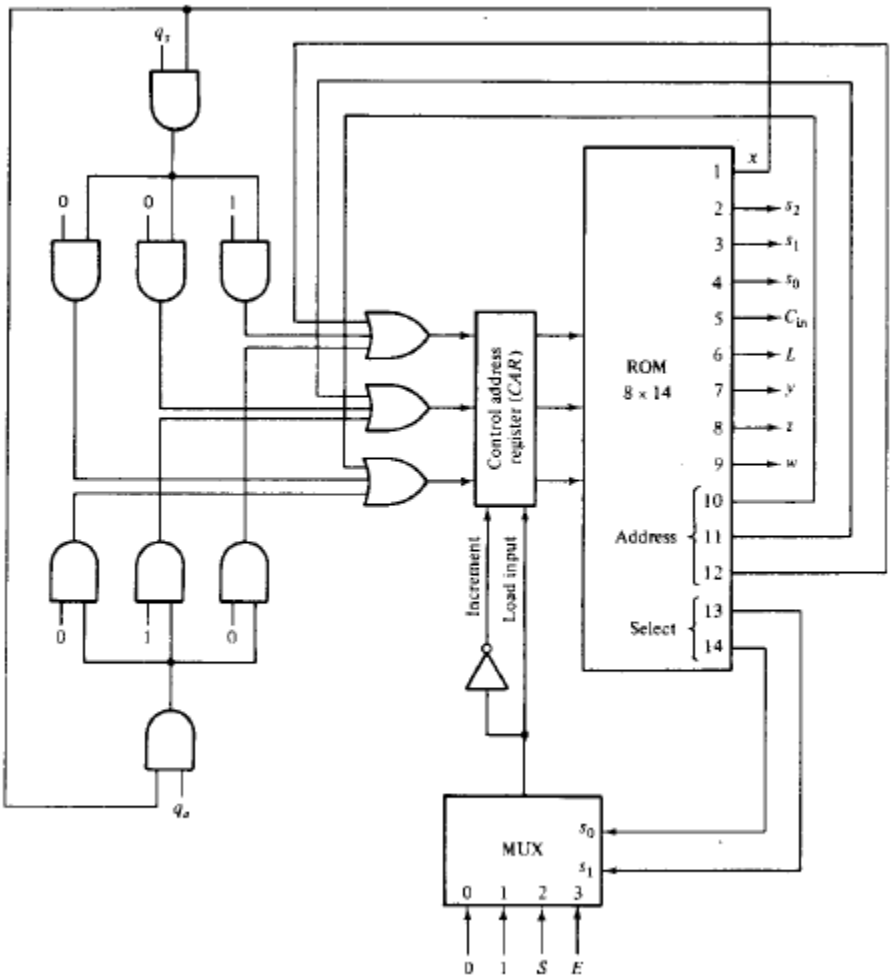
Hardware Configuration

The organisation of the microprogram control is shown in figure.

- The control memory is an 8 word by 14 bit ROM.
- The first 9 bits of a microinstruction word contain the control variables that initiate the micro-operations.
- The last five bits provide information for selecting the next address.
- The control address register CAR holds the adddress for the control memory.
- The register receives an input value when its load control is enabled otherwise it is incremented by 1.
- Bits 10 11 and 12 of a micro instruction contain an address for CAR.
- Bits 13 and 14 select an input for a multiplexer.

Bit 1 provides the initial state condition denoted by variable x and also enables an external address when qs and qa are equal to 1.

When x = 1, the address field of the micro instruction must be 000.
Then if qs =1, address 001 is available at the input of CAR,
but if qa=1, address 010 is applied to CAR.
If both qs and qa are 0, the zero address from bits 10 11 and 12 are applied to the input of CAR. In this way control memory stays at address 0 until an external variable is enabled.



ROM bits 13	14	MUX select function
0	0	Increment *CAR*
0	1	Load input to *CAR*
1	0	Load input to *CAR* if $S = 1$, increment *CAR* if $S = 0$
1	1	Load inputs to *CAR* if $E = 1$, increment *CAR* if $E = 0$

Figure 10-10 Microprogram control block diagram

The multiplexer has 4 inputs that are selected with 13 and 14 of the micro instruction. The functions of the multiplexer select bits are tabulated.

- If bits 13 and 14 are 00, a multiplexer input that is equal to 0 is selected. The output of the multiplexer is 0 and increment input to CAR is enabled. This configuration increments CAR to choose the next address in the sequence.
- An input of 1 is selected by the multiplexer when bits 13 and 14 are equal to 01. The output of the multiplexer is 1 and external input is loaded onto the CAR.
- Status variable S is selected when bits 13 and 14 or equal to 10. If S=1, the output of the multiplexer is 1 and address bits of the micro instructions are loaded into CAR (provided x=0). If S=0, the output of a multiplexer is 0 and CAR is incremented.
- With bits 13 and 14 equal to 11, status variable E is selected and the address field is loaded into CAR. If E=1, but CAR is incremented if equal to 0.
- Thus, the multiplexer allows the control to choose between two addresses depending on the value of the status with selected.

Microprogram

Once the configuration of a microprogram control unit is established, the designer's task is to generate the microcode for the control memory. This code generation is called **microprogramming** and is a process that determines the bit configuration for each and all words in control memory. Let's derive the micro program for the adder subtractor example.

The control memory has 8 word and each word contain 14 bits. To microprogram the control memory, we must determine the bit values of each of the eight words. The register transfer method can be adopted for developing a micro program. The micro-operation sequence can be specified with the register transfer statements.

Instead of a control function, we specify an address with each register transfer statement. The address associated with each symbolic statement corresponds to the address where the micro instruction is to be stored in memory. The sequencing from one address to the next can be indicated by means of conditional control statements. This type of statement can specify the address to which control goes depending on status conditions. Once the symbolic micro program is established, it is possible to translate the register transfer statements to their equivalent binary form.

Microprogram in **symbolic** form is given in the table. [Refer previous flowchart]

TABLE 10-2 Symbolic microprogram for control memory

ROM address	Microinstruction	Comments
0	$x = 1$, if ($q_s = 1$) then (go to 1), if ($q_a = 1$) then (go to 2), if ($q_s \wedge q_a = 0$) then (go to 0)	Load 0 or external address
1	$B_s \leftarrow \bar{B}_s$	$q_s = 1$, start subtraction
2	If ($S = 1$) then (go to 4)	$q_a = 1$, start addition
3	$A \leftarrow A + B$, $E \leftarrow C_{out}$, go to 0	Add magnitudes and return
4	$A \leftarrow A + \bar{B} + 1$, $E \leftarrow C_{out}$	Subtract magnitudes
5	If ($E = 1$) then (go to 0), $E \leftarrow 0$	Operation terminated if $E = 1$
6	$A \leftarrow \bar{A}$	$E = 0$, complement A
7	$A \leftarrow A + 1$, $A_s \leftarrow \bar{A}_s$, go to 0	Done, return to address 0

The symbolic designation is a convenient method for developing the microprogram in a way that people can read and understand. But this is not the way that the microprogram is stored in control memory. The symbolic microprogram must be translated to binary which was this is the form that goes into memory. The translation is done by dividing the bits of each microinstruction into their functional parts called **fields**. Here we have three functional parts.

- Bits 1 through 9 specify the control word for initiating the micro operations.
- Bits 10 through 12 specify address field
- Bits 13 and 14 select multiplexer input.
-

For each microinstruction listed in symbolic form we must choose the appropriate bits in the corresponding microinstruction fields. The **binary** form of the micro program is given in table. [Refer MUX table, symbolic form table]

TABLE 10-3 Binary microprogram for control memory

ROM address			ROM outputs												
			x	s_2	s_1	s_0	C_{in}	L	y	z	w	*Address*			*Select*	
			1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
0	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	1	1	0	0	0	1	0	1	0	1
1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1
1	1	0	0	1	1	1	0	1	0	0	0	1	1	1	0	1
1	1	1	0	0	0	0	1	1	0	1	0	0	0	0	0	1

The address for the ROM control memory are listed in binary. The content of each word of ROM is also given in binary. The first nine bits in each ROM word give the control word that initiates the specified microoperations. These bit values are taken directly from figure 10.9 (b). The last five bits in each row in word are derived from the conditional control statements in the same symbolic program.

- At address 000, we have 0 one for the select field. This allows an external address to be loaded into CAR if qs or qa is equal to 1. Otherwise address 000 is transferred to CAR.
- In address 001, the microinstruction select field is 01 and address field is 010. From table in figure 10.10, we find that the clock pulse that indicates the microoperation Bs <- Bs (dash) also transfer the address field into CAR.
- The next microinstruction out of ROM will be the one stored in address 010. The select field at address 001 could have been chosen to be 00. This would have caused CAR to increment and go to address 010.

- Inspection of the select field in which 13 and 14 shows that when these two bits are equal to 01, the address field is next address.
- When these two bits are 10, status variable S is selected and when they are 11, status variable E is selected.

- In the last two cases the next address is the one specified in the address field if the selected status bit=1. If the selected status bit=0, the next address is one next in sequence because CAR is incremented.

Advantages of Micro programmed control

- It simplifies the design of control unit. Thus it is both, cheaper and less error prone implement.
- Control functions are implemented in software rather than hardware.
- The design process is orderly and systematic
- More flexible, can be changed to accommodate new system specifications or to correct the design errors quickly and cheaply.
- Complex function such as floating point arithmetic can be realized efficiently.

Disadvantages

- A micro programmed control unit is somewhat slower than the hardwired control unit, because time is required to access the microinstructions from CM
- The flexibility is achieved at some extra hardware cost due to the control memory and its access circuitry.

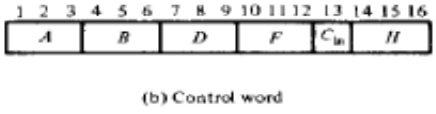
CONTROL OF PROCESSOR UNIT

General purpose processor unit has 7 registers, an ALU, a shifter and a status register [Module 2 – page 25- Processor Unit block diagram]. A micro-operation is selected with a control word of 16 bits. [Fig 9.16]

A microprogram organization for controlling the processor unit is shown in Fig 10-11. ie; control of a processor unit by a control unit that is designed using microprogrammed control.

Microprogram is stored in control memory. A micro program is a sequence of micro instructions which contain control signals for ALU, PC, MUX etc. A typical example of microprogram controlled processor unit is given here.

- A control memory of 64 words.
- Each word with 26 bits.
- 64 words = 2^6 = 6 bits **[21-26]** needed to address.
- 8 status bits = 2^3 = 3 selection lines **[18-20]** for multiplexer.
- 1 bit **[17]** of the micro instruction selects between an external address and address field of the microinstruction.
- First 16 bits **[1-16]** selects the micro operation for the processor.



(b) Control word

Figure 9-16 Processor unit with control variables

- Remaining 10 bits **[17-26**] select the next address for the control address.

- The status bits of the processor are applied to the inputs of the multiplexer.
- Normal and complement values are used for status with except for overflow bit V.
- Input 0 of the MUX2 is connected to a 1 - for unconditional branch.

- 18-20 bits will enable the load input and address through MUX1 will be transferred to CAR.

  If bit 17 = 0, CAR is loaded with external address (to initiate new sequence of microinstructions)
  If bit 17 = 1, CAR receives the address from the address field of micro instruction.

- Status bit selected by 18-20 bits of microinstruction can be 1 or 0.

  If selected bit = 1, input address is loaded to CAR
  If selected bit = 0, iincrement CAR

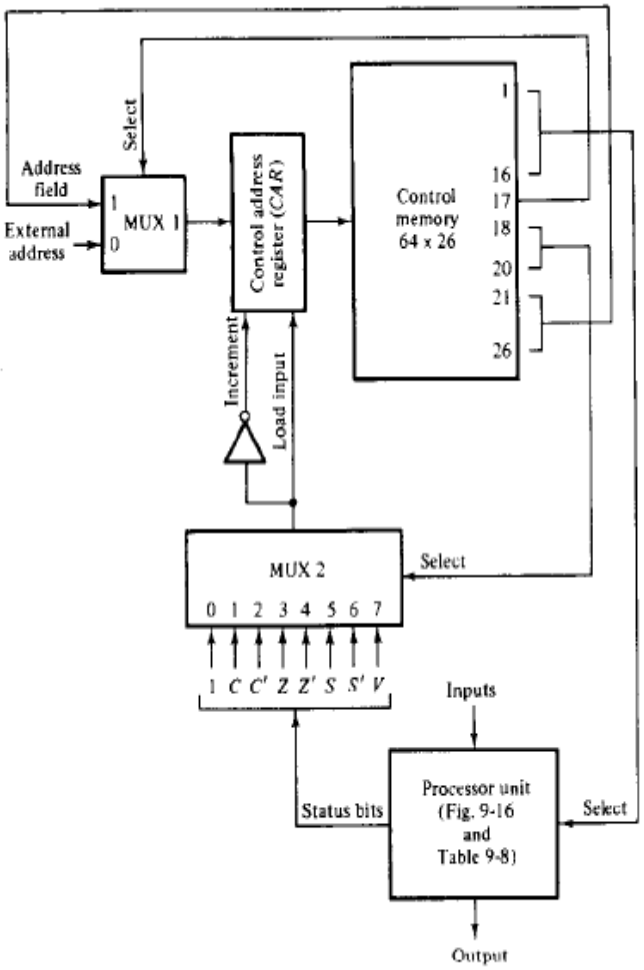


Figure 10-11 Microprogram control for processor unit

Micro-program Sequencer

Micro program sequencer is a control unit which does the tasks of Micro-program sequencing. There are two important factors must be considered while designing the micro program sequencer.

- The size of the microinstruction
- The address generation time

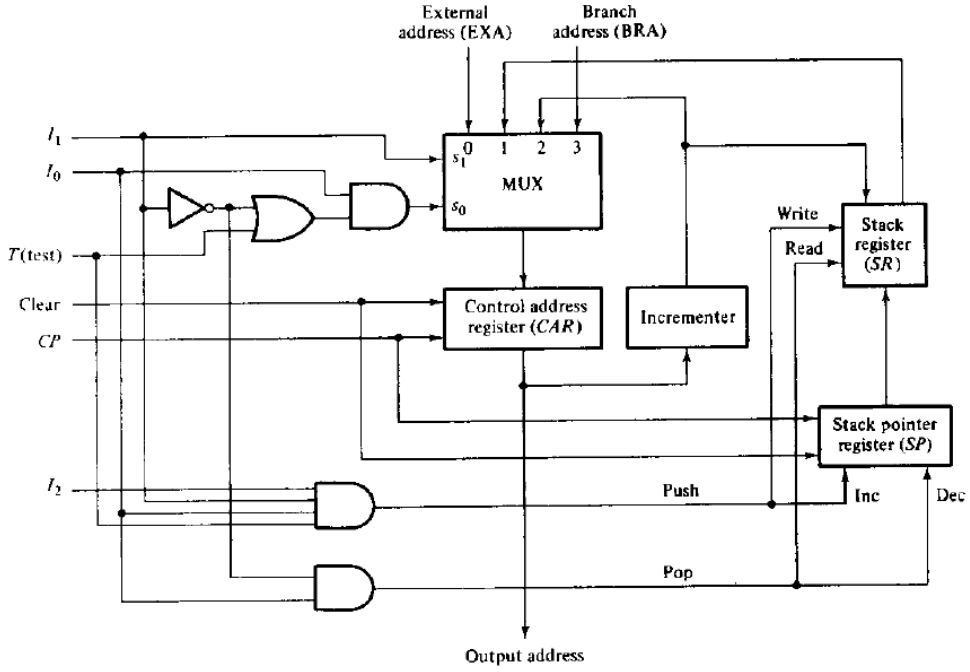
Micro-program sequencer is attached to the control memory. It inspects certain bits in the microinstruction to determine the next address for control memory. A typical sequencer has the following address sequencing capabilities.

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.

The address generation part is sometimes called a **microprogram sequencer**, since it sequences the micro instructions in control memory.

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



The block diagram of **Micro-program Sequencer** is shown in the figure. It consists of a multiplexer that selects an address from **four sources** and routes it into a Control Address Register (CAR).

1. Transfers control to a new address specified by an external source (EXA).
2. Transfers control to a new address for subroutines call (SR) and returns.
3. Increments the present address of control memory (Incrementer)
4. Branches to an address which will be specified in the bits of microinstruction (BRA).

- **Multiplexer**: Selects an address from four source s and routes it into CAR.
- The output from CAR provides the address for control memory to read a microinstruction.
- The contents of CAR are incremented and applied to the multiplexer and to the stack register file (SR).

- The register selected in the stack is determined by stack pointer (SP).
- Inputs (I_0-I_2) specify the operation for the sequencer and input.
- T is the test point for a status bit. I

Initially the address register is cleared to zero and clock pulse synchronizes the loading into registers. [In table X indicates don't care]

Function table

I_2	I_1	I_0	T	s_1	s_0	Operation	Comments
X	0	0	X	0	0	$CAR \leftarrow EXA$	Transfer external address
X	0	1	X	0	1	$CAR \leftarrow SR$	Transfer from register stack
X	1	0	X	1	0	$CAR \leftarrow CAR + 1$	Increment address
0	1	1	0	1	0	$CAR \leftarrow CAR + 1$	Increment address
0	1	1	1	1	1	$CAR \leftarrow BRA$	Transfer branch address
1	1	1	0	1	0	$CAR \leftarrow CAR + 1$	Increment address
1	1	1	1	1	1	$CAR \leftarrow BRA, SR \leftarrow CAR + 1$	Branch to subroutine

For more details about this, refer text book "Digital Logic and Computer Design" by M. Morris Mano; Page 446 - 449

Micro-programmed CPU Organization

Digital computer consists of: Central Processing Unit(CPU), Memory unit and Input-output devices. CPU can be divides into 2 distinct and interactive sections namely

- *Processing section*: useful device for constructing the processor section of a CPU and
- *Control section*: controlling the entire units of computer

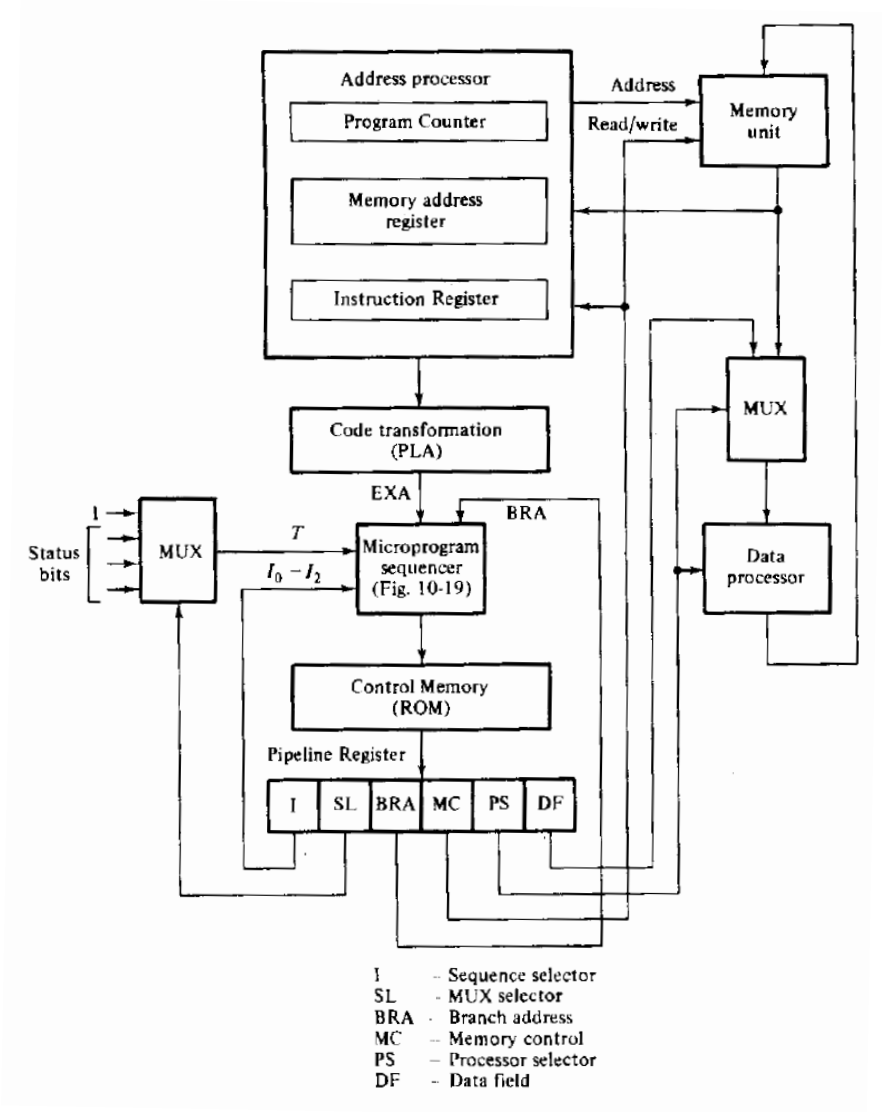
Mmicro-program sequencer: constructing a micro-program control of CPU

Micro-programmed computer

A computer CPU uses the micro-program sequencer. Micro-programmed computer consists of

1. Memory unit: Stores instructions and data supplied by the user through an input device
2. Two processor unit:
   Data processor: Manipulates data
   Address processor: Manipulates the address information received from memory
3. A Micro-program sequencer
4. A control memory
5. Other digital functions
   - An instruction extracted from memory unit during fetch cycle goes into instruction register and is decoded.
   - Corresponding to each opcode (Add, sub), there will be a micro routine in the control memory.

- Code transformation constitutes a mapping function that is needed to convert the op code bits of an instruction into a starting address for the control memory.
- It is implemented with ROM or PLA.
- Mapping concept provides flexibility for adding instructions or micro-operations for control memory as need arises.
- The address generated in code transformation mapping function is applied to the external address (EXA) input of the sequencer.



Micro-programmed Computer Organization

Micro-program control unit consists of

1. *The sequencer*: Generates next address
2. *A control memory:* Reads the next microinstruction while present microinstruction are being executed in the other units of the CPU and for storing microinstructions

3. *A multiplexer:* 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 operation

A **pipeline register**: Speed up the control operation, allows next address to be generated and the output of control memory to change while the current control word in pipeline register initiates the micro-operations given by present microinstruction. It's not always necessary because the output of control memory can go directly to the control inputs of the various units in the CPU

Microinstruction format contains six 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

SL field: Selects a status bit for the multiplexer

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 micro-operations in the processor and the memory units

Memory control (MC) field: Controls the address processor and the read and write operations 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.

MICRO-INSTRUCTIONS

- A microprogram consists of a set of instructions.
- Each microinstruction contains a control word which is a sequence of 0's and 1's
- Collection of micro instruction is **microprogram** or **microroutine**.

Structuring of microinstruction can be done in two ways:

- Horizontal organization
- Vertical organization

1. Horizontal Micro-instructions

Simplest way to structure microinstructions is to assign **one bit position** to each control signal required in the CPU.

Example: Control Sequence for instruction **Add (R3), R1:**

1. PCout, MARin, Read, Select4, Add, Zin
2. Zout, PCin, Yin, Wait for the MFC
3. MDRout, IRin
4. R3out, MARin, Read
5. R1out, Yin, Wait for MFC
6. MDRout, Select Y, Add, Zin
7. Zout, R1in, End

The microinstruction for the above control sequence can be expressed as follows.

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

In a horizontal microinstruction every bit in the control field attaches to a controller. Horizontal microinstructions represent several micro-operations that are executed at the same time. However, in extreme cases, each horizontal microinstruction controls all the hardware resources of the system.

Advantages: Fast execution

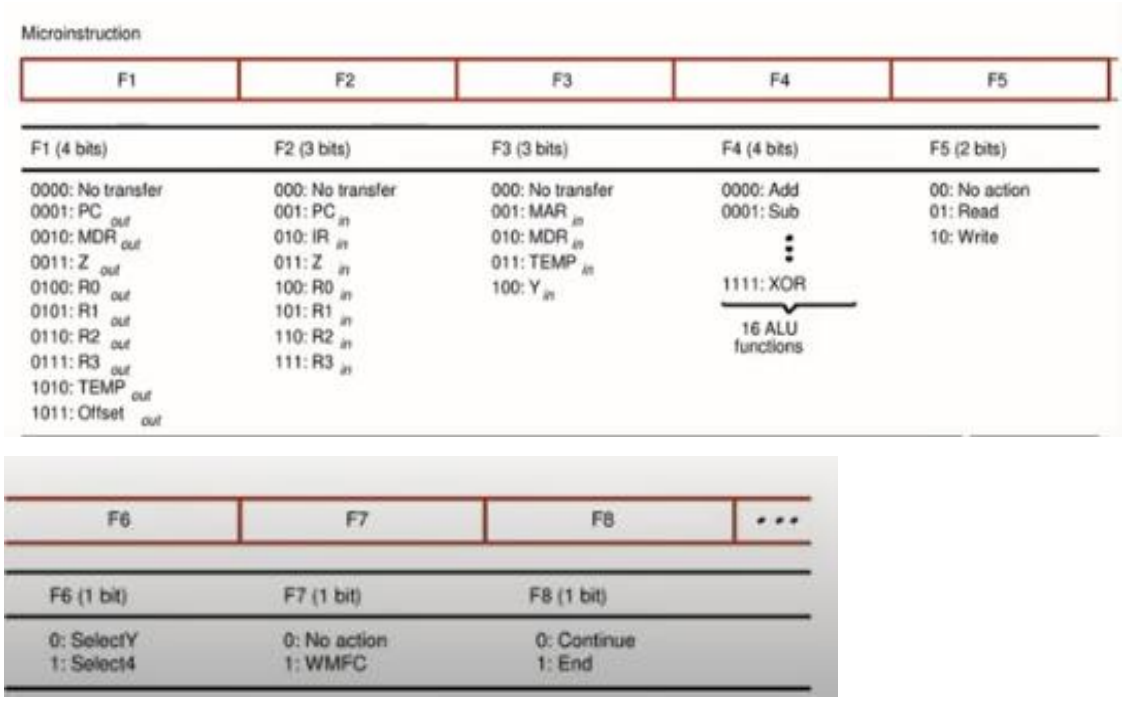
Disadvantages:

i. Size of micro instruction is too large
ii. Most signals are not needed simultaneously
iii. Many signals are mutually exclusive. ie; only one function of ALU can be activated at a time.

To overcome this disadvantage, **encoding** of signal is used. ie; mutually exclusive signals are grouped together.

2. Vertical Micro-Instructions

We can reduce the length of the horizontal micro-instruction so easily by implementing another method known as vertical micro-instructions. In this case, most signals are not needed simultaneously and many others are mutually exclusive. **Mutually exclusive signals are grouped together.**



In a vertical microinstruction, a code is used for each action to be performed and the decoder translates this code into individual control signals. The vertical microinstruction resembles the conventional machine language format comprising one operation and a few operands. As opposed to horizontal microinstructions, the vertical microinstruction represents single micro-operations.

Advantage: Fewer bits are required in the microinstruction.

Disadvantage: Vertical approach results in slower operations speed.

Comparison between Horizontal and Vertical Organization

Horizontal	Vertical
Long formats.	Short formats
Ability to express a high degree of parallelism.	Limited ability to express parallel micro operations.
Little encoding of the control information.	Considerable encoding of the control information
Useful when higher operating speed is desired.	Slower operating speeds.
The minimally encoded scheme in which resources can be controlled with a single instruction is called horizontal organization	Highly encoded scheme that use compact codes to specify only a small number of control functions in each micro instruction are referred to as a vertical organization