UNIT-6: BASIC PROCESSING UNIT

Processing Unit: Some fundamental Concepts: Register transfers, Performing an arithmetic or logic Operation, Fetching a word from memory, Storing a word in memory, Execution of a complete instruction. Hardwired Control, Microprogrammed Control.

Introduction:

The processing unit executes machine instructions and coordinates the activities of other units. This unit is often called the ***Instruction Set Processor*** (ISP) or simply the ***processor.*** It is also called as the "central processing unit".

Some fundamental concepts:

To execute an instruction, the processor has to perform the following three steps:

Step 1: Fetch the contents of the memory location pointed by the *PC*. The contents of this location are interpreted as an instruction to be executed. Hence, they are loaded into the *IR*. Symbolically, this can be written as ***IR ← [[PC]]***

Step 2: Assuming that the memory is byte addressable, increment the contents of the *PC* by *4*, that is, ***PC ←[PC]+4*** (Here 4 is added because each instruction comprises of 4 bytes)

Step 3: Carry out the actions specified by the instruction in the *IR.*

Step 1 and Step 2→*fetch phase*

Step 3 → *decode phase & execution phase*

Processor Organization

To study these operations in detail there is a need to examine the internal organization of the processor. The following figure shows an organization in which the ALU and all the registers are interconnected via a single common bus. This bus is internal to the processor and should not be confused with the external bus that connects the processor to the memory and I/O devices.

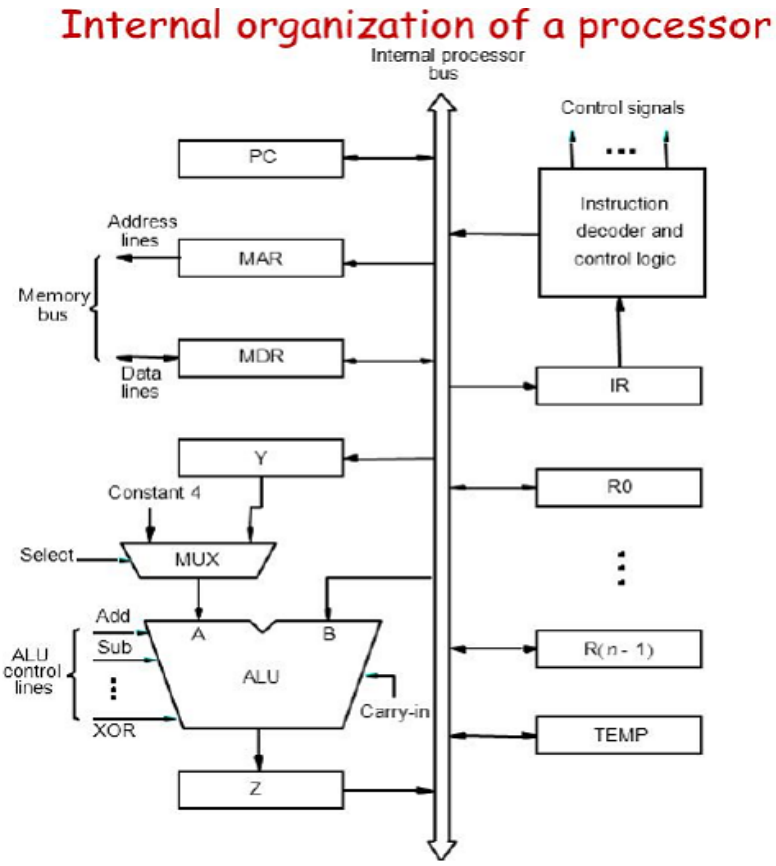


Fig-1: Single-bus organization of the datapath inside a processor

- The data and address lines of the external memory bus are connected to the internal processor bus via the memory data register, MDR, and the memory address register, MAR respectively.
- Register MDR has two inputs and two outputs.
- Data may be loaded into MDR either from the memory bus or from the internal processor bus.
- The data stored in MDR may be placed on either bus.
- The input of MAR is connected to the internal bus, and its output is connected to the external bus.
- The control lines of the memory bus are connected to the instruction decoder and control logic block.
- This unit is responsible for issuing the signals that control the operation of all the units inside the processor and for interacting with the memory bus.
- The number and use of the processor registers R0 through R(n-1) vary from one processor to another.
- Three registers X, Y &TEMP are used by the processor for temporary storage during the execution of some instructions.
- The MUX selects either the output of register Y or a constant value 4 to be provided as input A of the ALU.
- The constant 4 is used to increment the contents of the program counter.
- The two possible values of the MUX control input Select are referred to as Select4 and SelectY for selecting the constant ***4*** or register ***Y*** respectively.
- The registers, the ALU and the interconnecting bus are collectively referred to as the ***datapath.***

With a few exceptions, an instruction can be executed by performing one or more of the following operations in some specified sequence:

- Transfer a word of data from one processor register to another or to the ALU.
- Perform arithmetic or a logic operation and store the result in a processor register.
- Fetch the contents of a given memory location and load them into a processor register.
- Store a word of data from a processor register into a given memory location.

1. Register Transfers

Instruction execution involves a sequence of steps in which data are transferred from one register to another.

•For each register two control signals are used to place the contents of that register on the bus or to load the data on the bus into the register.

This is represented symbolically in the following figure.

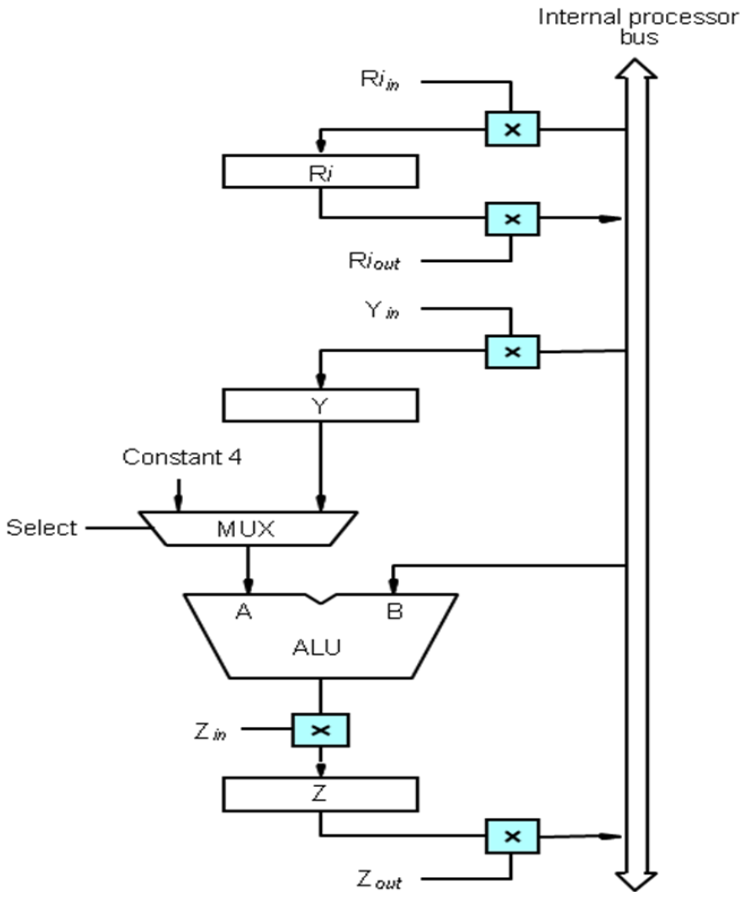


Fig-2: Input and output gating for the registers in Fig-1

•The input and output of register ***Ri*** are connected to the bus via switches controlled by the signals Ri_{in} and Ri_{out} respectively.

•When Ri_{in} is set to 1, the data on the bus are loaded into ***Ri.***

•Similarly, when Ri_{out} is set to 1, the contents of register ***Ri*** are placed on the bus.

Example

Suppose we wish to transfer the contents of register ***R1*** to register ***R4***.

This can be accomplished as follows.

•Enable the output of registers ***R1*** by setting $R1_{out}$ to ***1***. This places the contents of ***R1*** on the processor bus.

• Enable the input of register ***R4*** by setting $R4_{in}$ to ***1***. This loads data from the processor bus into register ***R4***.

2. Performing an arithmetic or logic operation

• The ALU is a combinational circuit that has no internal storage.

•It performs arithmetic and logic operations on the two operands applied to its A and B inputs.

•ALU gets the two operands from MUX and bus. The result is temporarily stored in register Z.

Example

What is the sequence of operations to add the contents of register R1 to those of R2 and store the result in R3?

1. $R1_{out}$, Yin

2. $R2_{out}$, SelectY, Add, Z_{in}

3. Z_{out}, $R3_{in}$

3. Fetching a word from memory

• To fetch a word from memory the processor has to specify the address of the memory location and request a Read operation.

• The processor then transfers the required address to the MAR, whose output is connected to the address lines of the memory bus.

• At the same time the processor uses the control lines of the memory bus to indicate that a Read operation is needed.

• When the requested data are received from the memory they are stored in MDR from where they can be transferred to other registers in the processor.

The connections for the register MDR are given in the figure below.

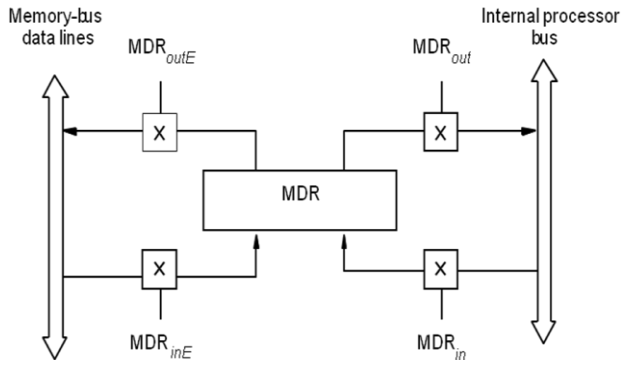


Fig-3: Connection and control signals for register MDR

It has 4 control signals:

1. MDR_{in} 2.MDR_{out} 3.MDR_{inE} 4.MDR_{outE}

   MDR_{in} and MDR_{out} control the connection to the internal bus and MDR_{inE} and MDR_{outE} control the connection to the external bus.

Example

As an example of read operation, consider the instruction ***Move (R1), R2***.

The actions needed to execute this instruction are:

1. MAR ← [R1]
2. Start a Read operation on the memory bus
3. Wait for the MFC response from the memory
4. Load MDR from the memory bus
5. R2 ← [MDR]

Memory read operations requires three steps, which can be described by the signals being activated as follows:

- $R1_{out}$, MAR_{in}, Read
- MDR_{inE}, WMFC
- MDR_{out}, $R2_{in}$

Note: WMFC (Wait for Memory Function Complete) is a control signal used to check whether the memory functions like Read and Write operations are completed or not.

4. Storing a word in memory

Writing a word into a memory location follows a similar procedure:

- The desired address is loaded into MAR.
- Then, the data to be written are loaded into MDR, and a write command is issued.

Example

Move R2, (R1) requires the following steps:

1. $R1_{out}$, MAR_{in}

2. $R2_{out}$, MDR_{in}, Write

3. MDR_{outE}, WMFC

Execution of a complete instruction

Let us now put together the sequence of elementary operations required to execute one instruction.

Example

Consider the instruction ***Add (R3), R1*** which adds the contents of a memory location pointed to by ***R3*** to register ***R1***.Executing this instruction requires the following actions:

1. Fetch the instruction
2. Fetch the first operand (the contents of the memory location pointed to by R3)
3. Perform the addition
4. Load the result into R1

The figure below gives the sequence of control steps required to perform these operations.

Step	Action
1	PC_{out} , MAR_{in} , Read,Select4,Add, Z_{in}
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} , SelectY,Add, Z_{in}
7	Z_{out} , $R1_{in}$, End

Fig-4: Control sequence for the execution of the instruction Add (R3), R1

Instruction execution proceeds as follows:

- In step 1, the instruction fetch operation is initiated by loading the contents of the PC into the MAR and sending a Read request to the memory. The Select signal is set to Select4, which causes the multiplexer MUX to select the constant 4. This value is added to the operand at input B, which is the contents of the PC, and the result is stored in register Z.
- The updated value is moved from register Z back into the PC during step 2, while waiting for the memory to respond.
- In step 3, the word fetched from the memory is loaded into the IR.

  Steps 1 through 3 constitute the instruction *fetch phase*, which is the same for all instructions. The instruction decoding circuit interprets the contents of the IR at the beginning of step 4. This enables the control circuitry to activate the control signals for steps 4 through 7, which constitute the *execution phase*.
- The contents of register R3 are transferred to the MAR in step 4, and a memory read operation is initiated.
- Then the contents of R1 are transferred to register Y in step 5, to prepare for the addition operation.
- When the Read operation is completed, the memory operand is available in register MDR, and the addition operation is performed in step 6.
- The contents of MDR are gated to the bus, and thus also to the B input of the ALU, and register Y is selected as the second input to the ALU by choosing Select Y. The sum is stored in register Z, and then transferred to R1 in step 7. The End signal causes a new instruction fetch cycle to begin by returning to step 1.

Note: In the control sequence explained above all control signals except Y_{in} in step-2 are explained. There is no need to copy the updated contents of the PC into register Y when executing the Add instruction. But in Branch instructions the updated value of the PC is needed to compute the branch target address

Branch instructions

- A branch instruction replaces the contents of PC with the branch target address, which is usually obtained by adding an offset X given in the branch instruction to the updated value of PC.
- The following figure gives a control sequence that implements an unconditional branch instruction.

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4,Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	Offset-field-of-IR_{out}, Add, Z_{in}
5	Z_{out}, PC_{in}, End

Fig-5: Control sequence for an unconditional Branch instruction.

- Processing starts as usual with the fetch phase. This phase ends when the instruction is loaded into the IR in step 3.
- The offset value is extracted from the IR by the instruction decoding circuit.
- Since the value of the updated PC is already available in register Y, the offset is gated onto the bus in step 4 and an addition operation is performed.
- The result which is the branch target address is loaded into the PC in step 5.
- The offset X used on a branch instruction is usually the difference between the branch target address and the address immediately following the branch instruction.

Hardwired Control

To execute instructions the processor must have some means of generating the control signals in proper sequence. Control signals are generated by the control unit.

Control unit can be designed by using the following techniques:

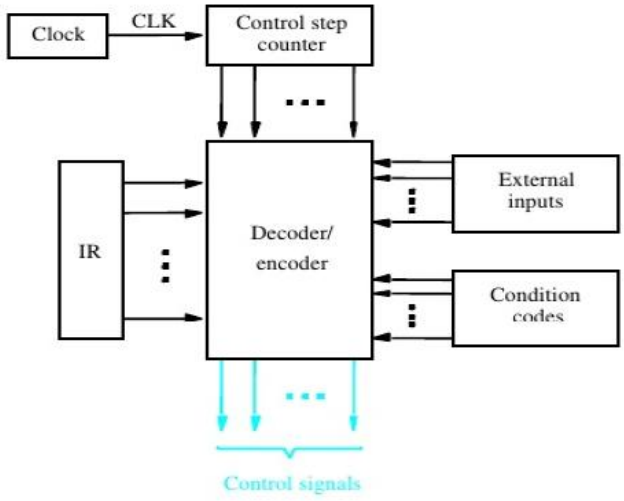
1. Hardwired control

2. Micro programmed control.

Hardwired Vs Microprogrammed

Hardwired Control	Microprogrammed control
1) It uses flipflops, decoders, logic gates and other digital circuits.	1) It uses sequence of micro-instructions in micro programming language.
2) Speed is high.	2) Speed is low.
3) More costlier.	3) Cheaper.
4) Occurrence of error is more	4) Occurrence of error is less
5) Control functions are implemented in hardware.	5) Control functions are implemented in software.
6) Not flexible to accommodate new system specification.	6) More flexible to accommodate new system specification.
7) Complicated design process.	7) Orderly, systematic and simple design process.

Simple Hardwired Control unit organization



The decoder/encoder block is a combinational circuit that generates the required control outputs, depending on the state of all its inputs.

The required control signals are determined by the following information:

•Contents of the control step counter

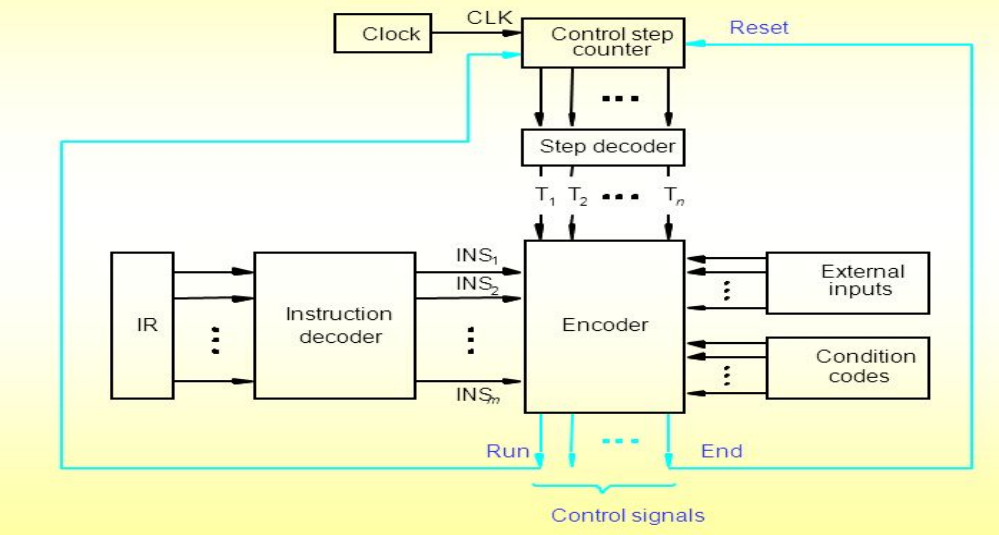
•Contents of the instruction register

•Contents of the condition code flags

•External input signals such as MFC and interrupt requests

Detailed Hardwired Control unit organization

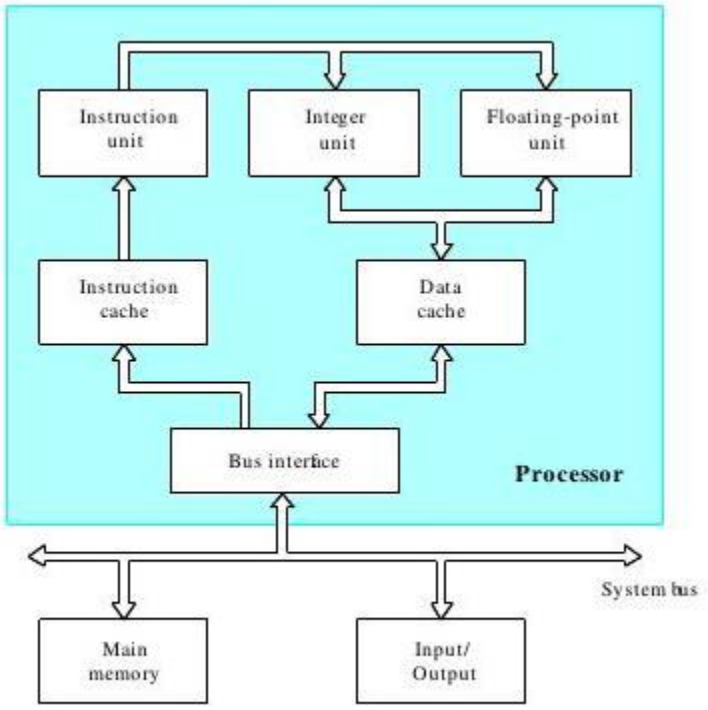
By separating the encoding and decoding functions we obtain the more detailed block diagram which is given below.



- The step decoder provides a separate signal line for each step in the control sequence.
- The output of the instruction decoder consists of a separate line for each machine instruction.
- For any instruction loaded into the IR, one of the output lines INS_1 to INS_m is set to *1* and all the others are set to *0*.
- The input signals to the encoder block are combined to generate individual control signals Y_{in}, PC_{out}, Add, End and so on.

A complete processor

- The instruction unit fetches instructions from the instruction cache or from the main memory when the desired instructions are not there in the cache.
- It has separate processing units to deal with integer data and floating-point data.
- A data cache is inserted between these units and the main memory.
- The processor is connected to the system bus and to the rest of the computer by means of a bus interface.



Microprogrammed Control

- In microprogrammed control unit, the logic of the control unit is specified by a microprogram.
- A microprogram consists of a sequence of instructions (micro-instructions) in a microprogramming language.
- Microinstructions specify micro-operations.
- Microprograms are stored in microprogram memory and the execution is controlled by microprogram counter (µPC).

Important terminologies:

Control Word (CW) :

- Control word is defined as a word whose individual bits represent the various control signals. Therefore each of the control steps in the control sequence of an instruction defines a unique combination of 0s and 1s in the CW.

Micro Program/Microroutine:

- A sequence of control words (CWs) corresponding to the control sequence of a machine instruction constitutes the microprogram for that instruction.

Micro Instruction:

- The individual control words in this microroutine are referred to as microinstructions.

Control store:

- The microroutines for all instructions in the instruction set of a computer are stored in a special memory called the control store.

An example of microinstructions for (Add (R3),R1)

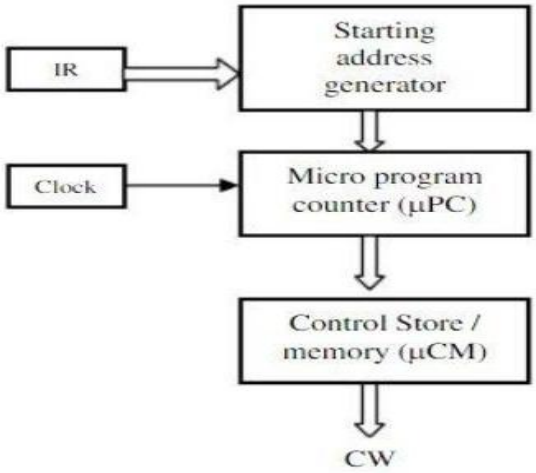
Step	Action
1.	PC_{out}, MAR_{in}, Read, Select4, Add , Z_{in}
2.	Z_{out}, PC_{in}, Y_{in}, MDR_{inE} WMFC
3.	MDR_{out}, IR_{in},
4.	$R3_{out}$, MAR_{in} Read
5.	$R1_{out}$, Y_{in}, MDR_{inE} WMFC
6.	MDR_{out}, Select Y, Add, Z_{in}
7.	Z_{out}, $R1_{in}$, End

Micro Instruction	...	PCin	PCout	MARin	Read	MDRout	IRin	Yin	Select	Add	Zin	Zout	R1out	R1in	R3out	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	1	0	1	0	0	1	

Basic organization of a microprogrammed control unit

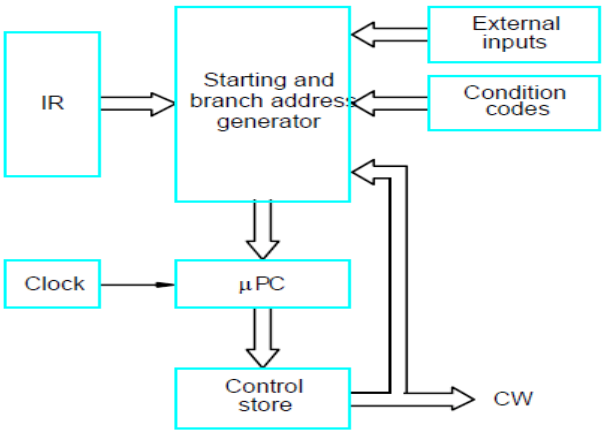
- The control unit can generate the control signals for any instruction by sequentially reading the CWs of the corresponding microroutine from the control store (microprogram memory).
- To read the control words sequentially from the control store, a microprogram counter (µPC) is used.

- Every time a new instruction is loaded into the IR, the output of the block labeled ***"starting address generator"*** is loaded into the µPC.
- The µPC is then automatically incremented by the clock, causing successive microinstructions to be read from the control store.



Organization of the control unit to allow conditional branching in the microprogram

- One important function of the control unit cannot be implemented by the basic organization of microprogrammed control unit.
- This is the situation that arises when the control unit is required to check the status of the condition codes or external inputs to choose between alternative courses of action.
- An alternative approach for microprogrammed control unit is to use conditional branch microinstructions.
- To support microprogram branching, the organization of control unit should be modified as shown in the figure below.



- The "Starting and branch address generator" loads a new address into the µPC when a microinstruction instructs it to do so.
- To allow implementation of a conditional branch, inputs to this block consists of the external inputs and condition codes as well as the contents of the IR.

- In this control unit, the µPC is always incremented everytime a new microinstruction is fetched from the microprogram memory, except in the following situations :

1. When a new instruction is loaded into the IR, the µPC is loaded with the starting address of the microroutine for that instruction.
2. When a branch microinstruction is encountered, the µPC is loaded with the branch address.
3. When an End microinstruction is encountered, the µPC is loaded with the address of the first CW in the microroutine for the instruction fetch cycle.

Microinstruction format

The figure given below specifies an example for a partial format for field-encoded microinstructions.

Microinstruction

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

F6	F7	F8	...
F6 (1 bit)	F7 (1 bit)	F8 (1 bit)	
0: SelectY 1: Select4	0: No action 1: WMFC	0: Continue 1: End	