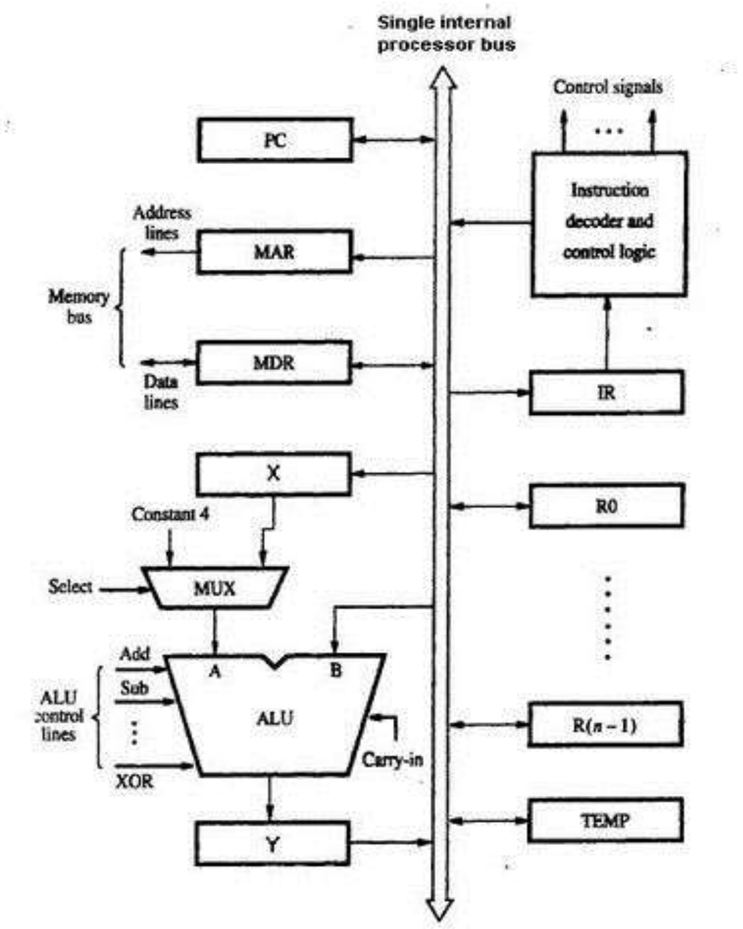
COMPUTER ORGANIZATION & ARCHITECTURE
Module 5

- Prof. Bimbitha Swarna Gowri M

Q1. Explain with a neat diagram, Single bus organisation of data path inside a processor. (10m)

Single Bus Organization



Single Bus Organization

Fig 1: Single bus organization of the datapath inside processor.

Let us assume an instruction of 4 bytes stored in one memory word. To execute an instruction the processor has to perform following actions.

1. Fetch contents of memory location pointed to by PC. The contents of this location are loaded to special purpose register called instruction register (IR).
   i.e. IR ← [PC]
2. Increment contents of PC to get next word in memory.
   PC ← PC+4
3. Carry out actions specified by instruction in IR.

Step 1 & step 2 are referred to as **fetch phase** and step 3 constitutes **execution phase.**

Fig 1 shows simple organization in which ALU and all registers are connected using a single common bus.

- This bus is internal to the processor.
- Data and address lines of external memory are connected to internal processor bus via memory data register (MDR) and memory address register (MAR) respectively.
- Data may be loaded into MDR either from memory bus or from internal processor bus.
- The input of MAR is connected to internal bus and its output is connected to external bus.
- The control lines of memory bus are connected to instruction decoder and control logic block.
- This unit is responsible for issuing the signals that control the operation of all units inside processor and for interacting with the memory bus.
- Functional elements like PC, MAR, MDR, R0, R1 etc., are connected to internal bus. MAR and MDR are also connected to external bus.
- Three registers Y, Z, TEMP are used by processor for temporary storage during execution of some instructions.
- The multiplexer MUX selects either the o/p of register Y or a constant value 4.The o/p of MUX is provided as i/p A to the ALU.
- The B i/p of ALU is obtained directly from the processor bus.
- The registers, the ALU and the interconnecting bus are collectively referred to as **datapath** of the processor.
- The instructions are usually categorized into three classes: memory-reference, arithmetic/logic and branch instructions. The action required to complete the instructions depend on instruction class.
- Generally instruction can be executed by performing one or more of following operations in some specified sequence.
  i. Transfer a word of data from one processor register to other or to ALU.
  ii. Perform an arithmetic or logic operation and store result in processor register.
  iii. Fetch the contents of a given memory location and load them into a processor register.
  iv. Store a word of data from processor register into a given memory location.

Q2. Explain multiple bus/three bus organization, with a neat diagram. (10m)

<u>Multiple Bus Organization</u>

- Fig 2 shows three-bus organization of CPU. All general purpose registers are combined into a single block called a **register file**.
- The register file has three parts:
  i. Two outputs allowing the contents of two different registers to be simultaneously placed on the buses A and B.
  ii. The third part allows the data on bus C to be loaded into a third register during the same clock cycle.
- Buses A and B can transfer two source operands to A and B inputs of the ALU. The result is transferred to the destination over C bus during the same cycle.
- Therefore, a three address instruction of the form OP Rsrc1, Rsrc2, Rdst can be executed in one clock cycle after the fetch phase. Hence no need of temporary registers Y, Z and TEMP.

- The increment unit is used to update the value of PC, which eliminates the need to add a value 4 to program counter using the ALU.

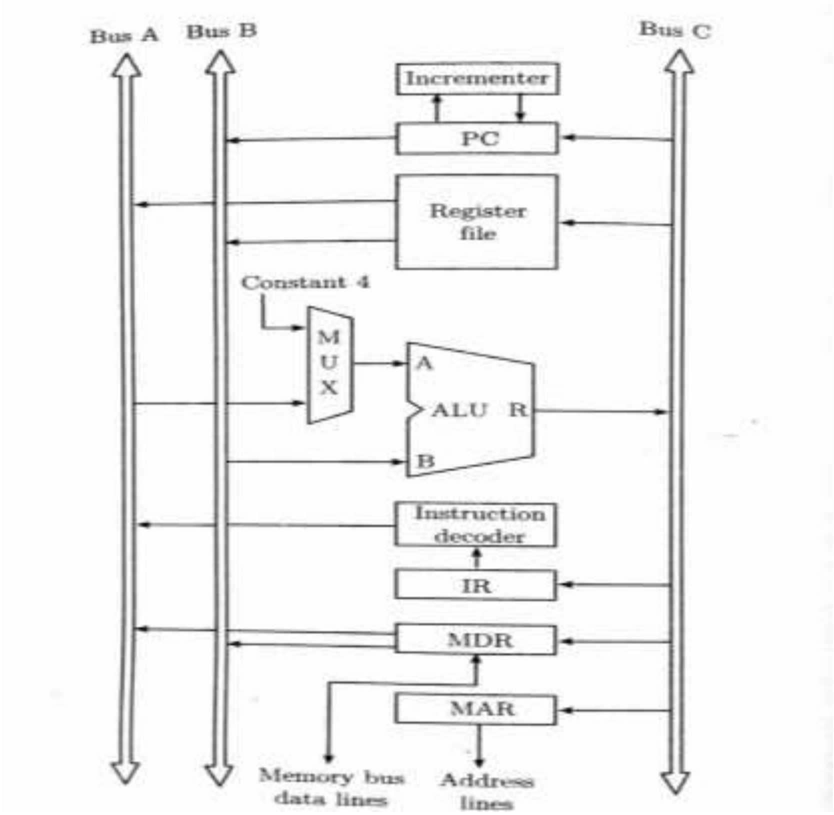
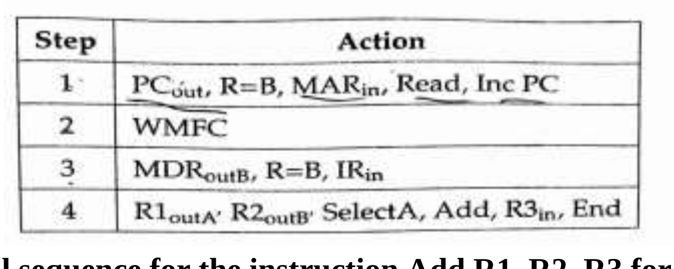


Fig 2: Three-bus Organization of the datapath inside a processor.

- Consider a three operand instructions.

  Add R1, R2, R3

  The control sequence is shown in fig 3.



Step	Action
1	PC_{out}, R=B, MAR_{in}, Read, Inc PC
2	WMFC
3	MDR_{outB}, R=B, IR_{in}
4	$R1_{outA}$, $R2_{outB}$, SelectA, Add, $R3_{in}$, End

Fig 3: Control sequence for the instruction Add R1, R2, R3 for the three-bus organization.

Step 1: Contents of PC are passed through ALU using R= B control signal and loaded into MAR to start a memory read operation. Pc is incremented by 4 to point to next instruction in the sequence.

Step 2: Processor waits for MFC signal from memory.

Step 3: The instruction code is received in MDR and transferred to IR. This completes the fetch phase.

Step 4: The instruction is decoded and the add operation takes place in a single step.

Q3. Illustrate the connection and control signals for register MDR with a neat diagram. (6m)

OR

Q3. Explain the process of fetching a data word from memory using respective registers of a processor with neat diagram. (10m)

Fetching a Word from Memory

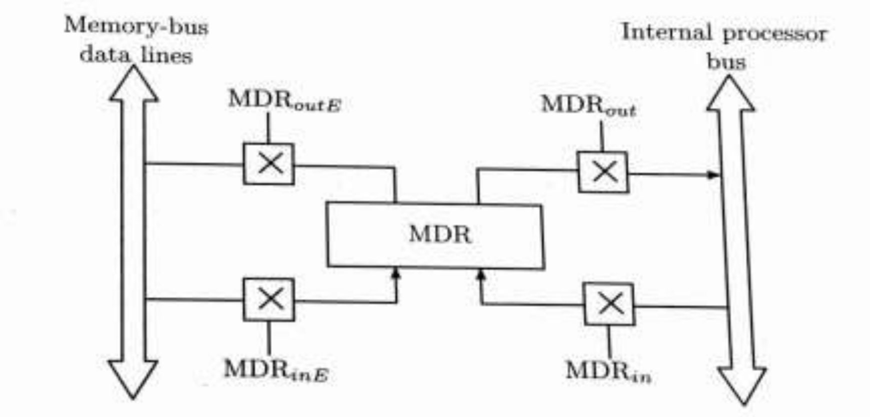


Fig 4: Connection and control signals for register MDR

- To fetch any information from memory, processor has to specify the address of memory location where this word is stored and issue a Read control signal.
- The processor transfers the required address to MAR from PC.
- The o/p of MAR is connected to address lines of memory bus.
- The processor uses the control lines of memory bus to indicate that a read operation is needed.
- The processor waits until it receives an indication from memory that the requested read operation has been completed. This action is done by the signal called **Memory-Function-Completed (MFC**)
- The memory sets this signal to 1 to indicate that the content of addressed location are available on data lines of memory bus.
- Now, CPU reads the information into MDR.

Fig 4 shows connections for register MDR. MDRin and MDRout control the direction of data to internal bus, whereas MDRinE and MDRoutE control the direction of data to external bus.

Assume instruction MOVE (R1), R2. Sequence of steps are:

1. MAR ← [R1]
2. Start a Memory Read Operation
3. Wait for MFC signal from memory
4. Load MDR from external bus
5. R2 ← [MDR]

- Consider that the o/p of MAR is enabled all the time.

- When a new address is loaded into MAR, it will appear on the memory bus at the beginning of the next clock cycle as shown in fig 5.
- A read control signal is activated at the same time MAR is loaded.
- While waiting a response from memory, MDRinE signal can be activated.
- Thus, information received from the memory are loaded into MDR at the end of the clock cycle in which MFC signal is received
- In next cycle, MDRout is activated to transfer the data to register R2.
- Therefore, a memory read operation requires these three steps
    1. R1out, MARin, Read
    2. MDRinE, WMFC
    3. MDRout, R2in

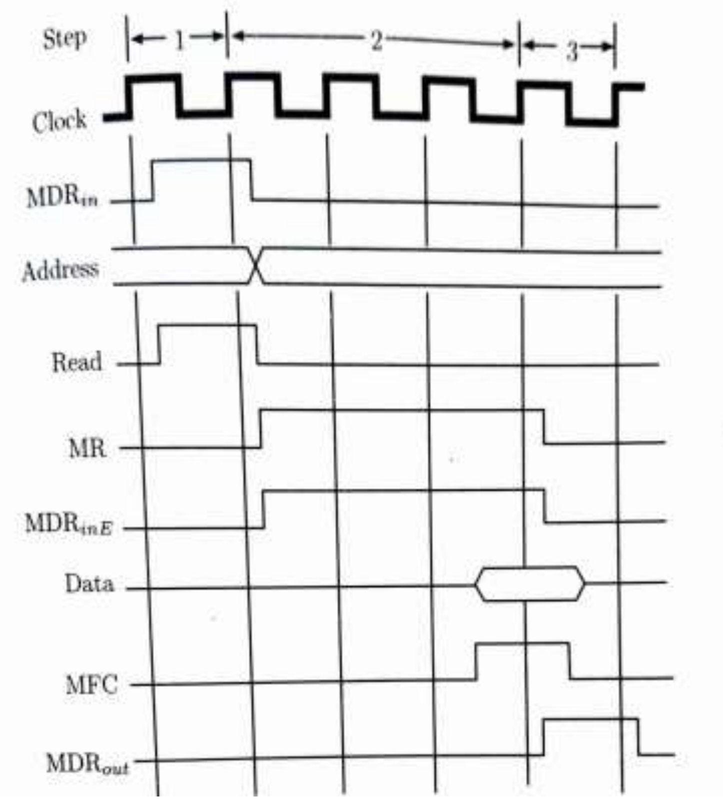


Fig 5: Timing of a memory read operation

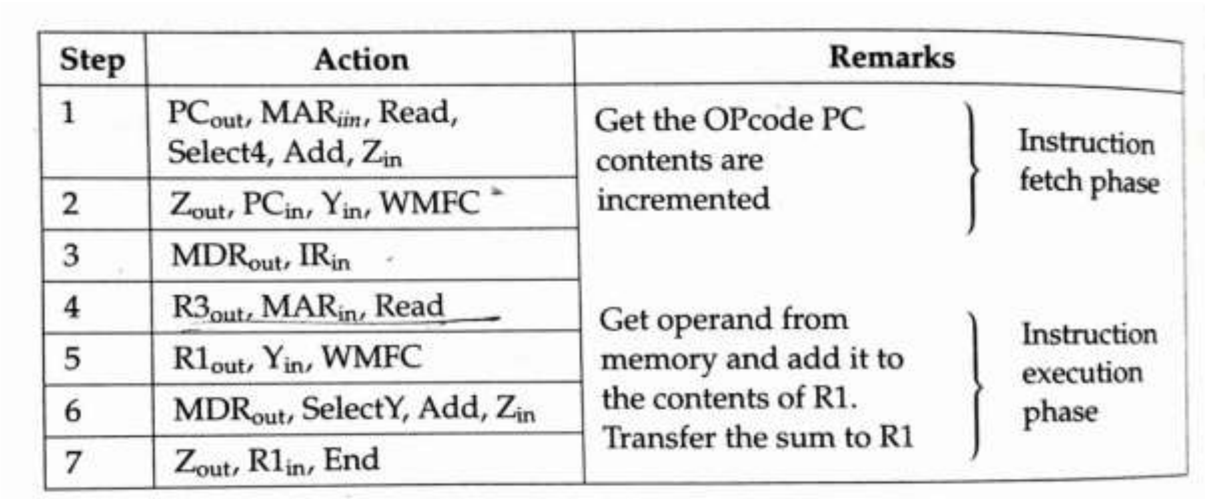
Storing a Word in Memory

- The desired address is loaded into MAR.
- Then, data to be written is loaded into MDR and a write command is issued.
- Eg: data is in register R2 and the address of the memory location is in register R1. Then, to store the contents of register R2 in memory,
  the instruction is Move R2, (R1). This requires following three steps:
    1. R1out, MARin
    2. R2out, MDRin, Write
    3. MDRoutE, WMFC

Q4. List the actions required to execute a complete instruction ADD (R3), R1. (10m)

This instruction adds the content of a memory location pointed to by R3 to the contents of register R1 and place result in R1. Following actions are needed.

1. Fetch the instruction.
2. Fetch the first operand from memory
3. Perform addition
4. Store result in R1



Step	Action	Remarks	
1	PC_{out}, MAR_{iin}, Read, Select4, Add, Z_{in}	Get the OPcode PC contents are incremented	Instruction fetch phase
2	Z_{out}, PC_{in}, Y_{in}, WMFC		
3	MDR_{out}, IR_{in}		
4	$R3_{out}$, MAR_{in}, Read	Get operand from memory and add it to the contents of R1. Transfer the sum to R1	Instruction execution phase
5	$R1_{out}$, Y_{in}, WMFC		
6	MDR_{out}, SelectY, Add, Z_{in}		
7	Z_{out}, $R1_{in}$, End		

Fig 6: Control sequence for execution of instruction Add (R3), R1

Fig 6 gives the sequence of control steps required to perform these operations for the single-bus structure.

Step 1: Fetch operations is initiated by loading address in PC into MAR and sending a read request to the memory. Select signal is to select4, which causes the Mux to select constant value 4. This value is added to the operand at i/p B (contents of PC) and result is stored in register Z.

Step 2: The updated value in Z is moved to PC while waiting for memory to respond.

Step 3: Once MFC signal is received from the memory, the fetched instruction will be moved into MDR and then to IR.

These three steps constitute **instruction fetch phase**.

Step 4: The instruction decoding circuit interprets the contents of IR and the processor starts the execution phase. Contents of R3 are loaded into MAR and a read signal is issued.

Step 5: While waiting for memory to respond, contents of R1 are transferred into Y register.

Step 6: Memory provides data on the bus, which is moved into MDR and onto B i/p of ALU. The contents of Y (R1 contents) are gated into A i/p of ALU using Y signal of MUX. Add control signal is activated. After the addition, the result is transferred to Z.

Step 7: Finally, the sum is moved out of register Z into register R1. The End signal causes a new instruction fetch cycle to begin by returning to step 1.

- Prof. Bimbitha Swarna Gowri M

Q5. Explain hardwired Control Unit Organisation. (10m)

Or

Explain the control signal generation required for proper sequence of instructions in the processor. (10m)

To ensure proper operation of the processor, efficient design of control unit is very much necessary. The control design approaches fall in two categories: Hardwired control and microprogrammed control.

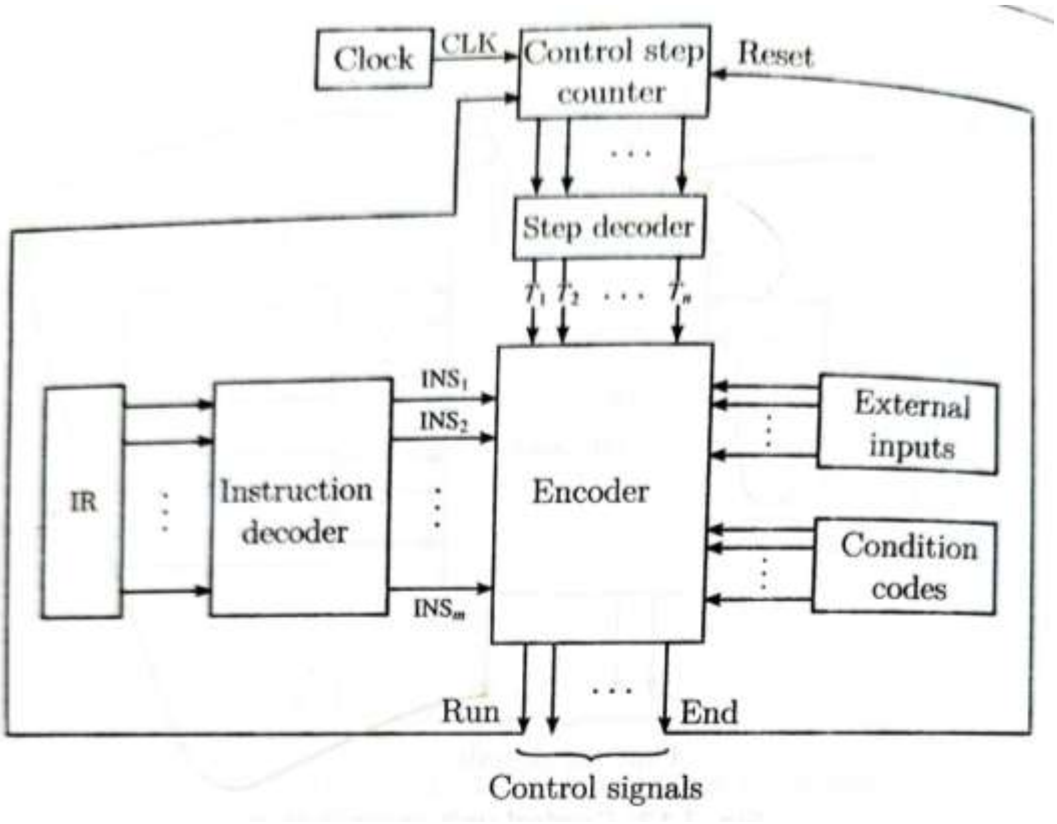


Fig 7: Control Unit Organization

Fig 7 shows organization of hardwired control unit. Since each step is performed in one clock cycle, a counter driven by a clock signal CLK, can be used to keep track of the control steps.

The required control signals are determined by following information.

- Contents of control step counter
- Contents of instruction register
- Contents of condition code flags
- External i/p signal, such as MFC, and interrupt requests

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

Fig 7 shows detailed block diagram where the step decoder provides a separate signal line for each step in the control sequence.

The output of instruction decoder consists of separate line for each machine instruction.

For any instruction loaded in IR, one of the output lines INS1 through INSm is set to 1 and all others are set to 0.

The i/p signals to encoder block are combined to generate individual control signals Zin, Add, PCout and so on.

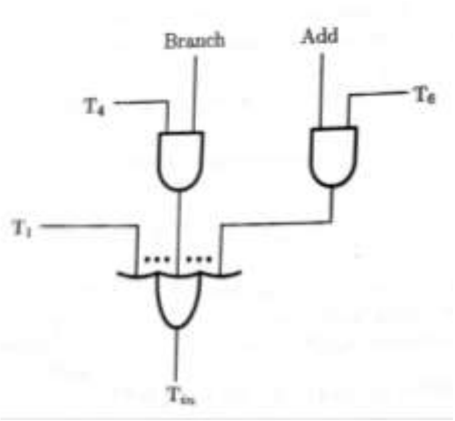


Fig 8: Generation of the Zin control signal.

Fig 8 shows the generation of control signal Zin

$$Zin = T1 + T6.ADD + T4.BR + \dots$$

The signal Zin is asserted during time slot T1 for all instructions, during T4 for an unconditional branch instruction, during T6 for an Add instruction and so on.

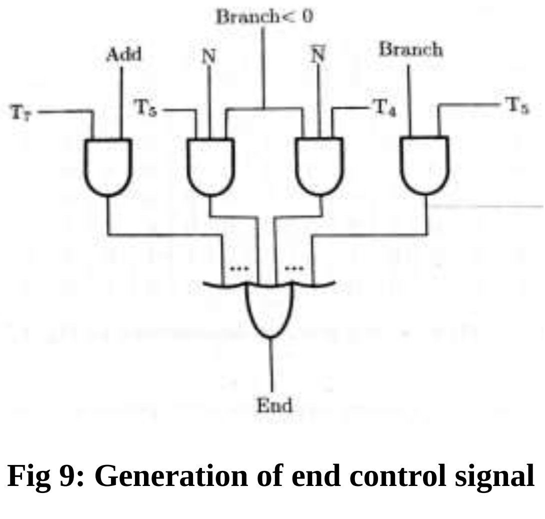


Fig 9: Generation of end control signal

Fig 9 shows the generation of End control signal.

End control signal is generated from the logic function,

$$END = T7.ADD + T5.BR + (T5.N + T4.\overline{N}).\ BRN + \dots$$

The control hardware shown in Fig: can be viewed as a **state machine** that changes state depending on contents of instruction register, condition codes and external inputs. The o/p of the state machine are the control signals.

The sequence of operations carried out are determined by wiring (fusing) of logic circuits, hence the name "**hardwired**".

Generally, hardwired control design is found in RISC.

Q6. With a neat diagram, explain microprogrammed control unit design. (6m)

Or

What is microprogrammed control? Explain its basic organization with suitable diagram and example. (10m)

Or

Describe basic organization of a microprogrammed control unit. Give an example of microinstructions. (10m)

It is another scheme for control unit design in which control signals are generated by a program similar to machine language programs.

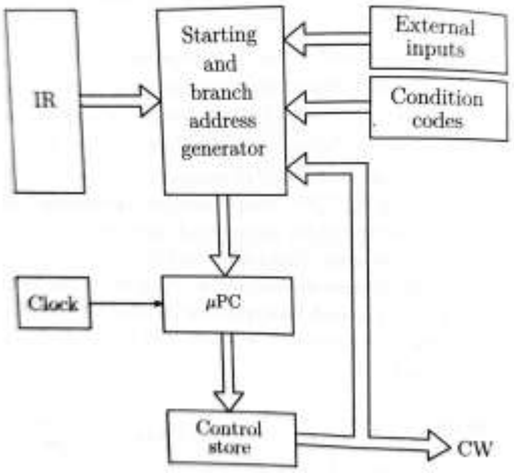


Fig 10: Basic organization of a microprogrammed control unit.

Control Word (CW): A control word is a bit pattern of 0's and 1's. Individual bits in a control word represent the various control signals like Add, End, Zin and so on. Each of control steps in a control sequence of an instruction defines unique combination of 0's and 1's.

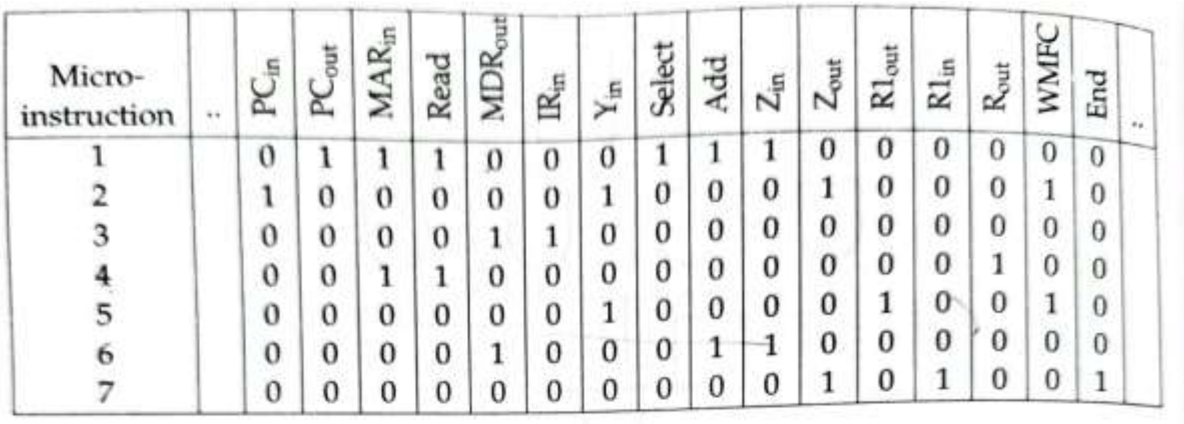
Microinstructions: Individual control words are also referred to as microinstructions.

Microroutine: A sequence of control words corresponding to the control sequence of a machine instruction constitutes the microroutine.

Control Store: The microrotine for all instructions in the instruction set of a computer are stored in a special memory called control store.

Fig 10 shows the basic structure of the microprogrammed control unit.

- The control unit can generate control signals for any instruction by sequentially reading the control words of corresponding microroutine from control store.
- To read CW sequentially, a microprogrammed counter (µPC) is used.
- Everytime a new instruction is loaded into IR, the o/p of starting address generator is loaded into microprogrammed counter.
- Then, microprogram counter is automatically incremented for each successive instruction fetch.
- Hence, control signals are delivered to various parts of the processor incorrect sequence.



Micro-instruction	..	PC_{in}	PC_{out}	MAR_{in}	Read	MDR_{out}	IR_{in}	Y_{in}	Select	Add	Z_{in}	Z_{out}	$R1_{out}$	$R1_{in}$	R_{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	

Fig 11: An example of microinstructions for fig:

The Control words correspond to the 7 steps of fig 6 are shown in fig 11

Q. Draw and explain organization of the control unit to show conditional branching in the microprogram. (6m)

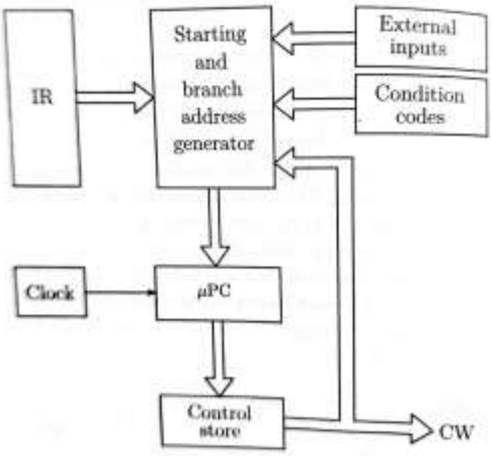
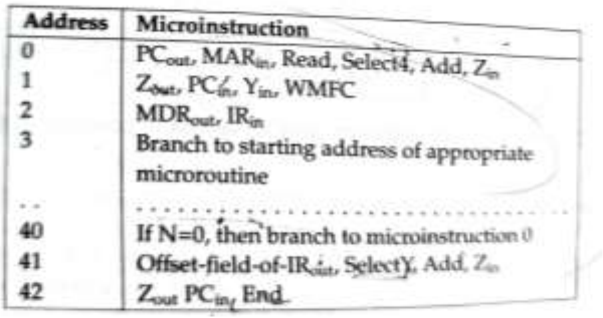


Fig 12: Organization of the control unit for handling the conditional branching.

Fig 12 shows modified control unit for handling conditional branching. Starting and branch address generator block accepts i/p from external inputs, condition codes as well as from IR. In this control unit, the µPC is incremented everytime a new instruction is fetched from the control store, except in following cases.

a) When a new instruction is loaded into IR, the µPC is loaded with the starting address of the microroutine for that instruction.
b) When a branch microinstruction is encountered and the branch condition is satisfied, the µPC is loaded with the ranch address.
c) When an END microinstruction is encountere, the µPC is loaded with the address of the first control word in the microroutine for fetch cycle.



Address	Microinstruction
0	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
1	Z_{out}, PC_{in}, Y_{in}, WMFC
2	MDR_{out}, IR_{in}
3	Branch to starting address of appropriate microroutine
..	....
40	If N=0, then branch to microinstruction 0
41	Offset-field-of-IR_{out}, SelectY, Add, Z_{in}
42	Z_{out} PC_{in}, End

Fig 13: Microroutine for the instruction Branch < 0

For example, the instruction, Branch<0 (Branch-on-negative) can be implemented by a microroutine as shown in the fig 13. After loading this instruction into IR, a branch microinstruction transfers control to the corresponding microroutine starting at some location, in the control store. This address is the o/p of the starting address generator block.

A microinstruction at this location checks the N bit of the condition code. If this bit=0, a branch takes place to location 0 to fetch a new machine instruction. Otherwise next instruction is executed to put the branch target address into register Z.