Fundamental Concepts and Processor Organization

- Introduction of Fundamental Concepts
- Single Bus CPU organization
- 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 of Fundamental Concepts

- Processor fetches one instruction at a time, and performs the operation specified.
- Instructions are fetched from successive memory locations until a branch or a jump Instruction is encountered.
- Processor keeps track of the address of the memory location containing the next Instruction to be fetched using Program Counter (PC).
- When an instruction is fetched, it is placed in the instruction register(IR) and the IR holds the instruction until its execution is completed.

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

- Fetch the contents of the memory location pointed to by the PC. The contents of this location is the instruction to be executed; hence they are loaded into the IR. (Fetch phase)

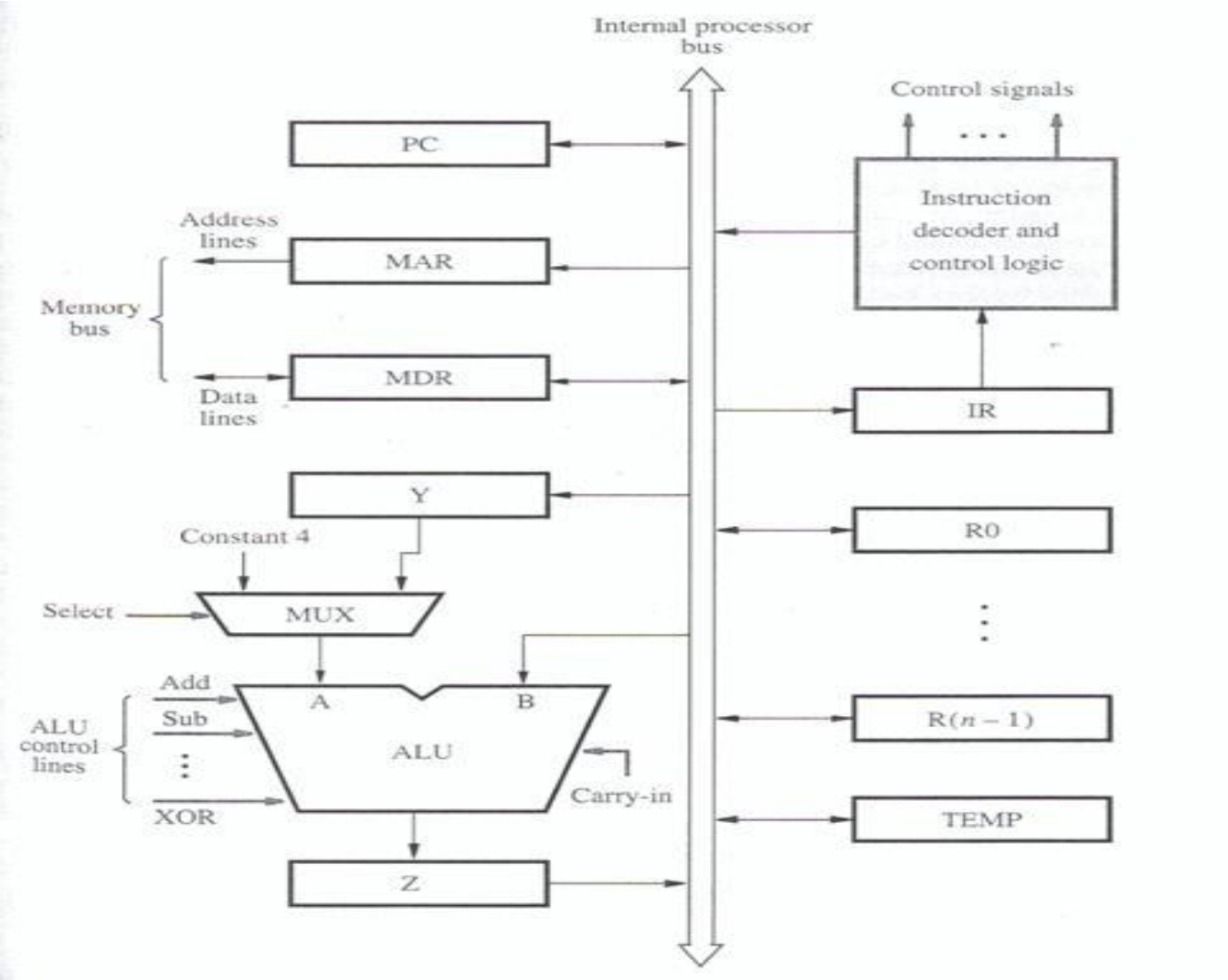
$$IR \leftarrow [[PC]]$$

- Increment the PC to point to the next instruction. Assuming that the memory is byte addressable, the PC is incremented by 4. (Fetch phase)

$$PC \leftarrow [PC] + 4$$

- Carry out the operation specified by the instruction in the IR.(Execution phase)

Single Bus CPU organization



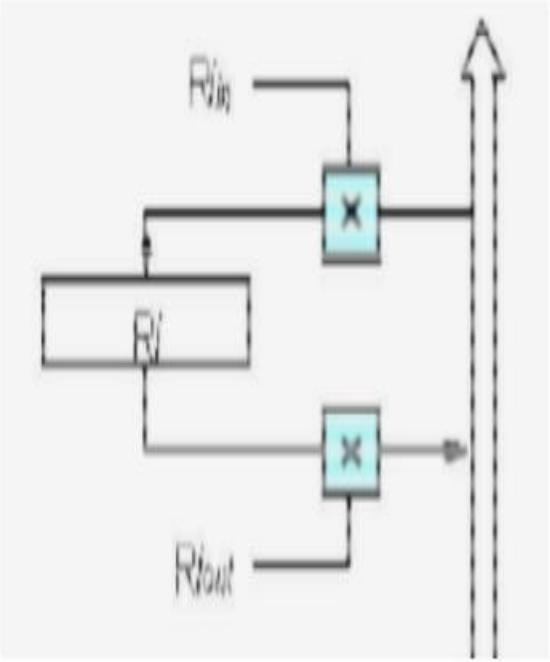
- ALU and all the registers are interconnected via a single common bus.
- The data and address lines of the external memory bus connected to the internal processor bus via the memory data register, MDR, and the memory address register, MAR respectively.
- Register MDR has two inputs and two outputs.
- 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.
- This unit is responsible for issuing the signals that control the operation of all the units inside the processor and for increasing with the memory bus.
- 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.

With few exceptions, the operation specified by an instruction can be carried out by performing one or more of the following actions:

- Read the contents of a given memory location and load them into a processor register.
- Read data from one or more processor registers.
- Perform an arithmetic or logic operation and place the result into a processor register.
- Store data from a processor register into a given memory location.

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 register.

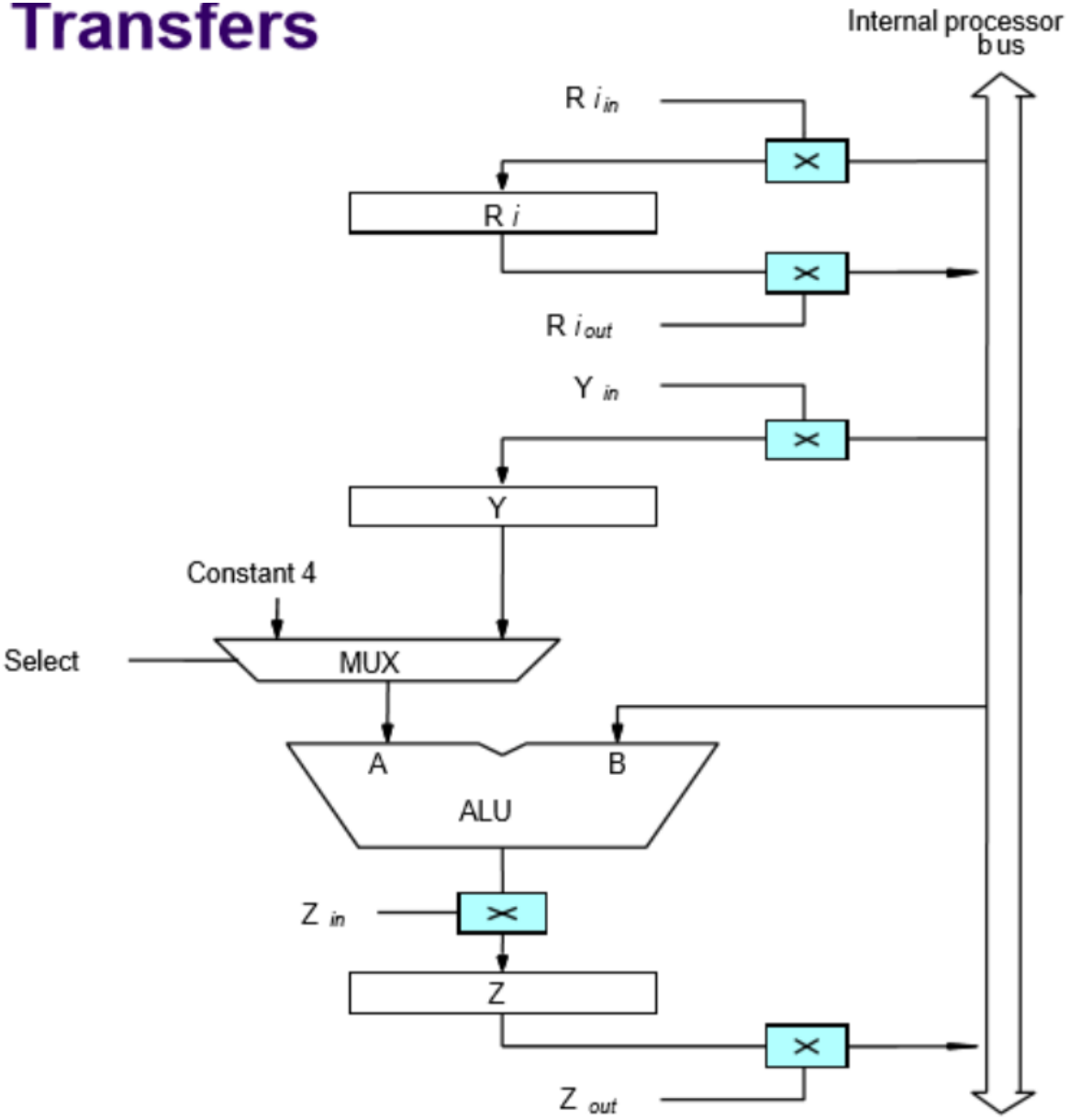
• The input and output of register Ri in and Ri out is set to 1,

• When Ri in is set to 1, the data on the bus are loaded into R i.

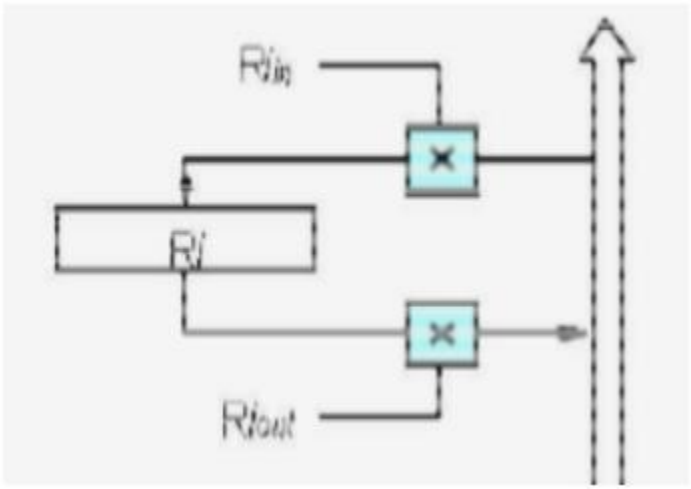
• Similarly, when Ri out is set to 1, the contents of register Ri are placed on the bus.

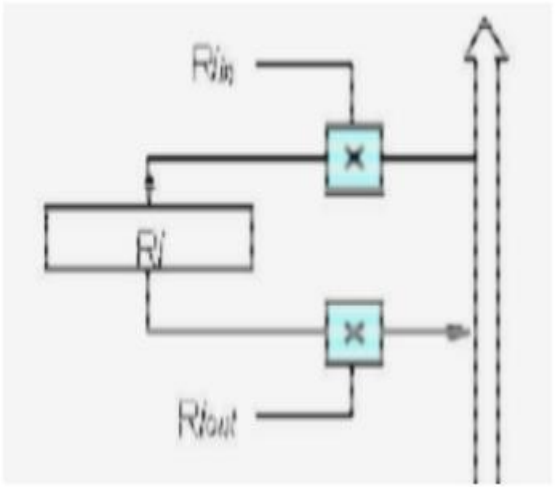
• While Ri out is equal to 0, the bus can be used for transferring data from other registers.

Register Transfers



Example: Move R1, R4





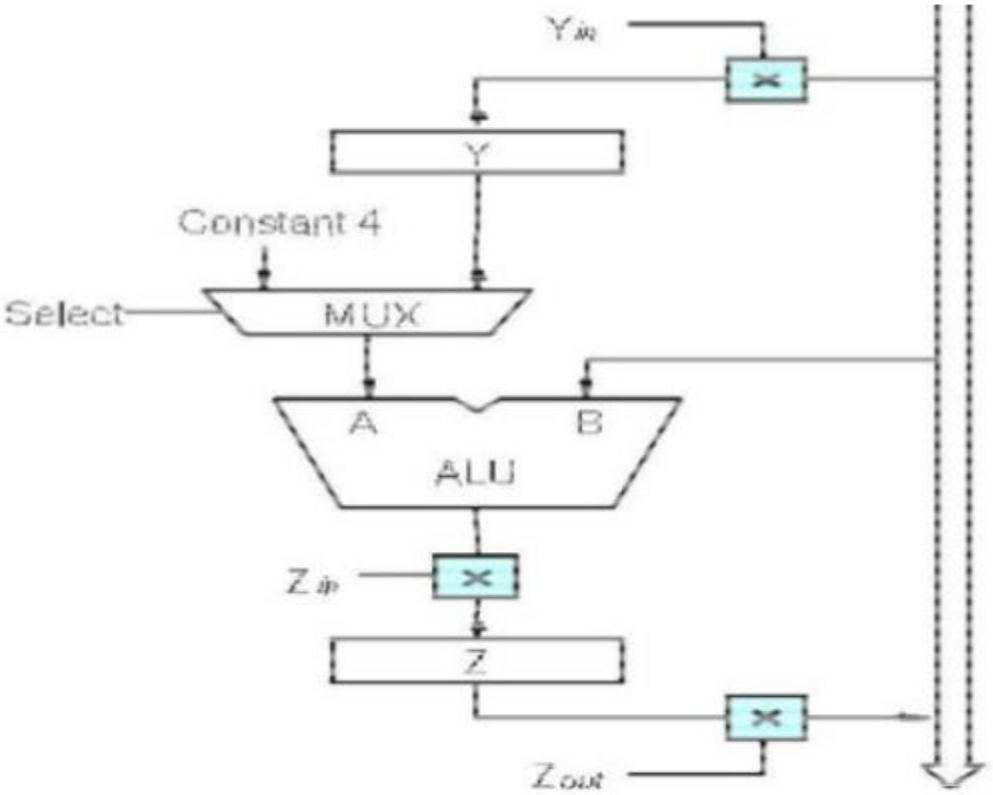
R1 R4

Transfer the contents of register R1 to register R4. This can be accomplished as follows

- Enable the output of registers R1 by setting R1out to 1. This places the contents of R1 on the processor bus.
- Enable the input of register R4 by setting R4in to 1. This loads data from the processor bus into register R4.
- All operations and data transfers within the processor take place wit in time periods defined by the processor clock.
- The control signals that govern a particular transfer are asserted at the start of the clock cycle.

Performing an Arithmetic or Logic Operation

- The ALU is a combinational circuit that has no internal storage.
- ALU gets the two operands from MUX and bus. The result is temporarily stored in register Z.



Example: Add R3,R2,R1

- Add the contents of register R1 to those of R2 and store the result in R3
    - R1out, Yin
    - R2out, SelectY, Add, Zin
    - Zout, R3in

• All other signals are inactive.

• In step 1, the output of register R1 and the input of register Y are enabled, causing the contents of R1 to be transferred over the bus to Y.

• Step 2, the multiplexer's select signal is set to Select Y, causing the multiplexer to gate the contents of register Y to input A of the ALU.

• At the same time, the contents of register R2 are gated onto the bus and, hence, to input B.

• The function performed by the ALU depends on the signals applied to its control lines.

• In this case, the ADD line is set to 1, causing the output of the ALU to be the sum of the two numbers at inputs A and B.

• This sum is loaded into register Z because its input control signal is activated.

- In step 3, the contents of register Z are transferred to the destination register R3. This last transfer cannot be carried out during step 2, because only one register output can be connected to the bus during any clock cycle.

Fetching a Word from Memory

- The processor has to specify the address of the memory location where this information is stored and request a Read operation.
- This applies whether the information to be fetched represents an instruction in a program or an operand specified by an instruction.
- The processor transfers the required address to the MAR, whose output is connected to the address lines of the memory bus.

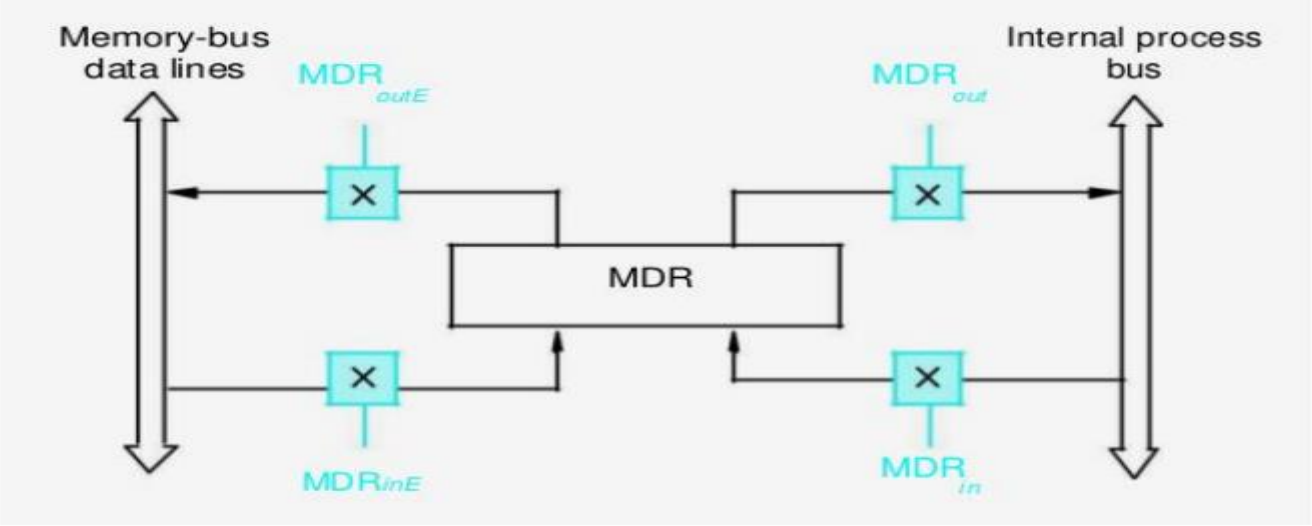


Figure: Connection and control signals MDR.

- 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 register MDR, from where they can be transferred to other registers in the processor.
- The response time of each memory access varies (cache miss, memory-mapped I/O etc.)
- To accommodate this, the processor waits until it receives an indication that there quested operation has been completed (Memory-Function-Completed, MFC).

Example: Move (R1), R2

- The output of MAR is enabled all the time.
- Thus the contents of MAR are always available on the address lines of the memory bus.
- When a new address is loaded into MAR, it will appear on the memory bus at the beginning of the next clock cycle.
- A read control signal is activated at the same time MAR is loaded.
- This means memory read operations requires three steps, which can be described by the signals being activated as follows

o R1 out ,MAR in ,Read
o MDR inE ,WMFC
o MDR out ,R2 in

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.

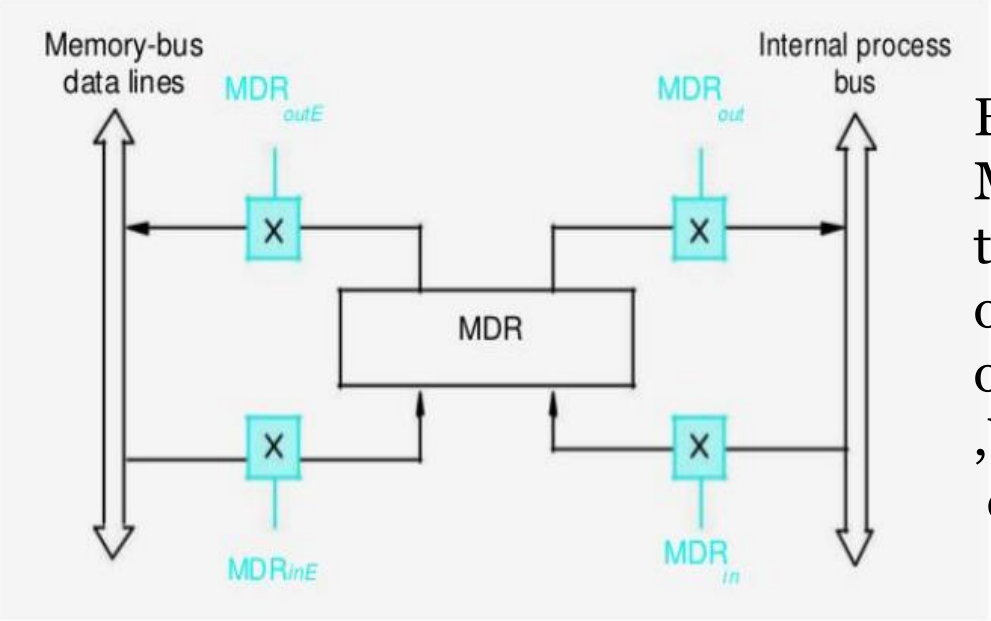


Figure: Connection and control signals MDR.

Example:
Move R2,(R1) requires the following steps
o R1 out ,MAR in
o R2 out ,MDR in ,Write
o MDR outE ,WMFC

Execution of a complete Instruction

Control sequence for execution of the instruction
Add (R3),R1.

- Fetch the first instruction
- Fetch the first operand
- Perform the addition
- Load the result in R1

Step Action

1. PC out , MAR in , Read, Select4,Add, Z in
2. Z out , PC in ,Yin,WMF C
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

Execution of Branch Instruction

1. PC out , MAR in , Read, Select4,Add, Z in
2. Z out , PC in ,Yin,WMF C
3. MDR out , IR in
4. Offset_Field_of_IRout, Add,Zin
5. Zout,Pcin,End

Hardwired Control

- To execute instructions, the processor must have some means of generating the control signals needed in the proper sequence.
- The hardwired control is used to solve this problem.
- Hardwired system can operate at high speed; but with little flexibility

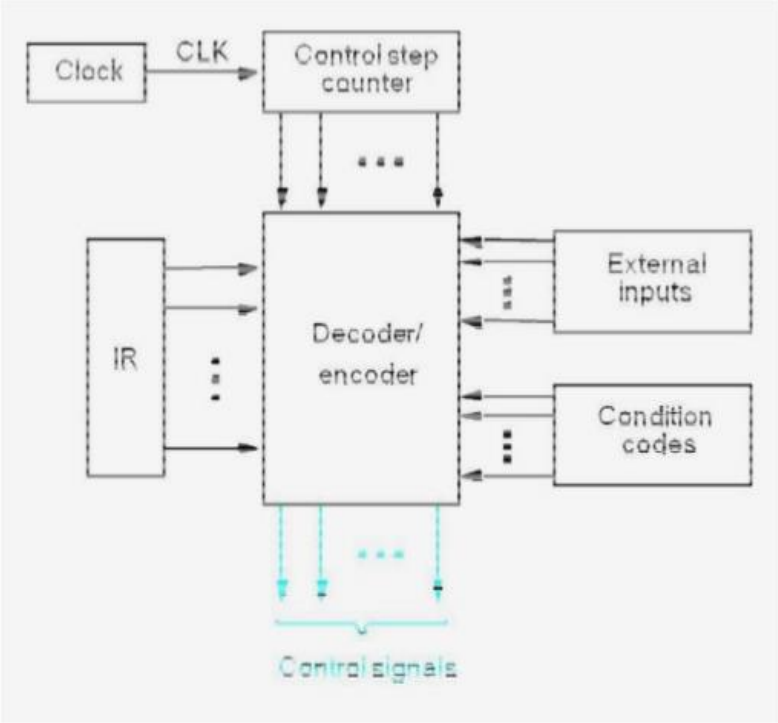
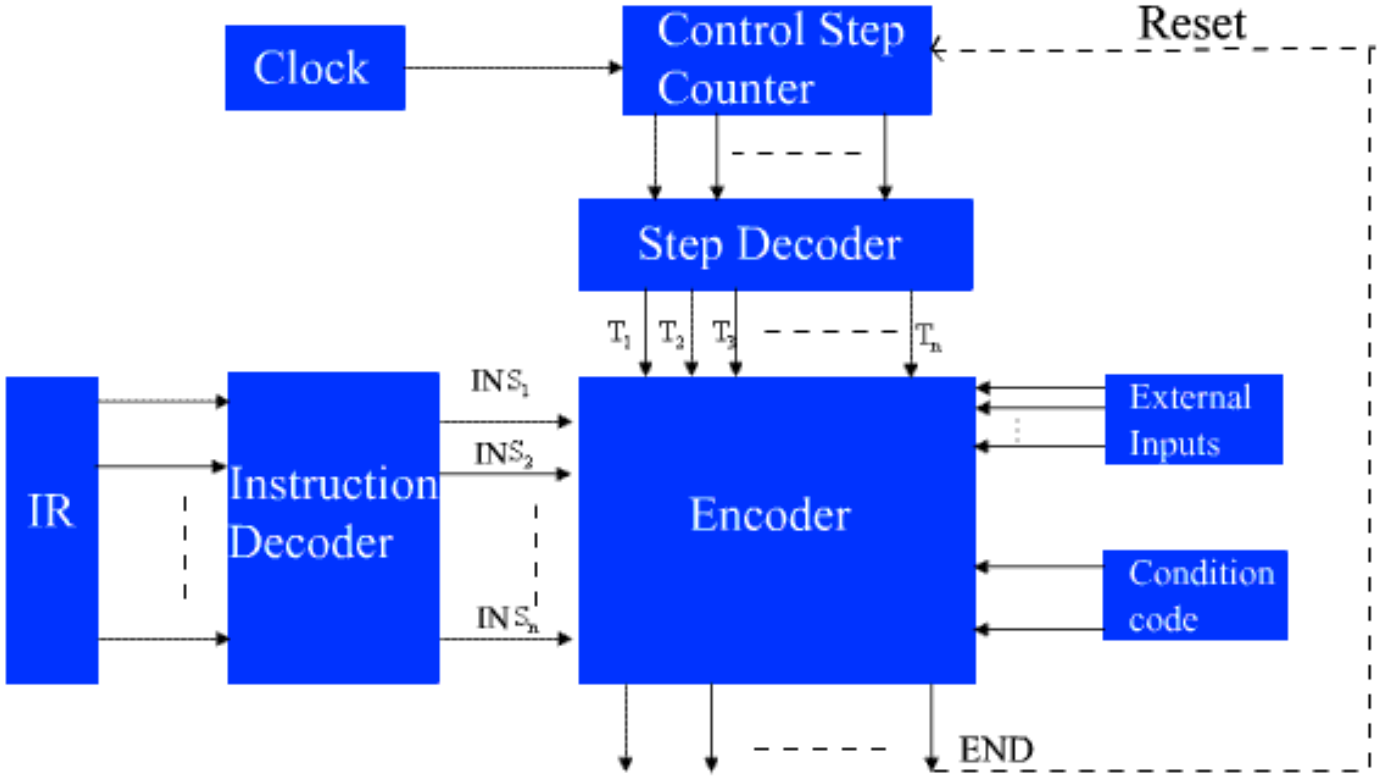


Figure: Control unit organization

The required control signals are determined by the following information:

1. Contents of the control step counter
2. Contents of the instruction register
3. Content of the condition code flags
4. External input signals, such as MFC and interrupt requests

Separation of Encoding and Decoding function



- The decoder/encoder block is a combinational circuit that generates the required control outputs, depending on the state of all its inputs.
- 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 line for each machine instruction.
- For any instruction loaded in the IR, one of the output lines INS1 through INSm is set to 1, and all other lines are set to 0.
- The input signals to the encoder block is combined to generate the individual control signals Yin, PCout, Add, End< and so on.
- An example of how the encoder generates the Z in control signal for the processor organization

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

Microprogrammed Control

- Control signals are generated by a program similar to machine language programs.
- Control Word (CW); microroutine; microinstruction
- Control signals are generated for each execution step based on the instruction in the IR.
- In hardwired control, these signals are generated by circuits that interpret the contents of the IR as well as the timing signals derived from a step counter.
- Instead of employing such circuits, it is possible to use a “software" approach, in which the desired setting of the control signals in each step is determined by a program stored in a special memory.
- The control program is called a microprogram to distinguish it from the program being executed by the processor.
- The microprogram is stored on the processor chip in a small and fast memory called the microprogram memory or the control store.

Micro programmed Control

- Control signals are generated by a program similar to machine language programs.
- Control Word (CW); microroutine; microinstruction

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	

Figure 7.15 An example of microinstructions for Figure 7.6

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

Figure 7.6. Control sequence for execution of the instruction Add (R1),R1

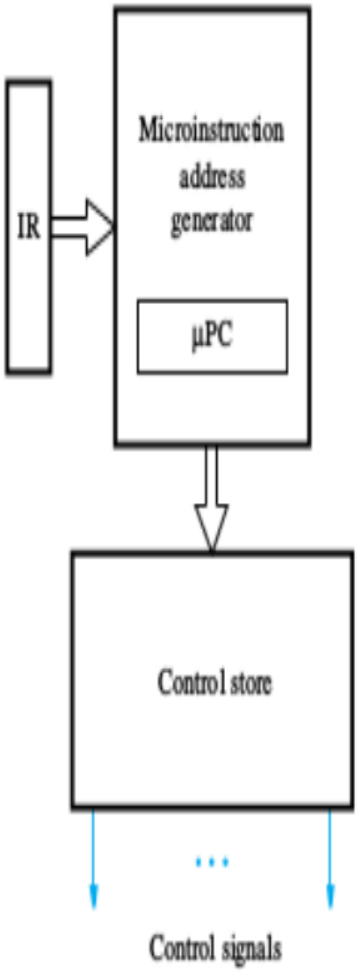


Figure depicts a typical organization of the hardware needed for microprogrammed control.

- It consists of a microinstruction address generator, which generates the address to be used for reading microinstructions from the control store.
- The address generator uses a microprogram counter, µPC, to keep track of control store addresses when reading microinstructions from successive locations.
- The microinstruction address generator decodes the instruction in the IR to obtain the starting address of the corresponding micro routine and loads that address into the µPC.
- This is the address that will be used in the following clock cycle to read the control word. As execution proceeds, the microinstruction address generator increments the µPC to read microinstructions from successive locations in the control store.
- One bit in the microinstruction, which we will call End, is used to mark the last microinstruction in a given microroutine. When End is equal to 1, the address generator returns to the microinstruction, which causes a new machine instruction to be fetched.