Basic Processing Unit

Basic Processing Unit.

Fundamental Concepts, Execution of a Complete instruction, Multiple-Bus Organization, Hardwired Control. Multiprogrammed control.

Fundamental Concepts:-

While executing a program the processor fetches one instruction at a time and performs the Operations specified. Instructions are fetched from successive memory locations until a branch or jump instruction is encounted. The processor keeps track of the address of the memory location containing the next instruction to be fetched using the program Counter PC. A branch instruction may load a different value into the PC.

Instruction register (IR) is used to process instruction in the processor. To execute an instruction, the processor has to perform the following three steps:

① Fetch the Contents of the memory location pointed to by the PC into the IR register.

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

② Increment the Contents of the PC by 1

$$PC \leftarrow [PC] + 1$$

Note:- Here plus 1 means the address of next instruction. For example if each instruction takes 4 bytes PC will be incremented by 4 [i.e +4]

③ Carry out the actions Specified by the instruction in the IR.

Suppose if an instruction Occupies more than one word step1 and step2 must be repeated as many times as necessary to fetch the Complete instruction. First two Steps are usually referred to as the fetch phase, step3 Considered as excecution phase. The internal Organization of the processor is shown below with all main building blocks.

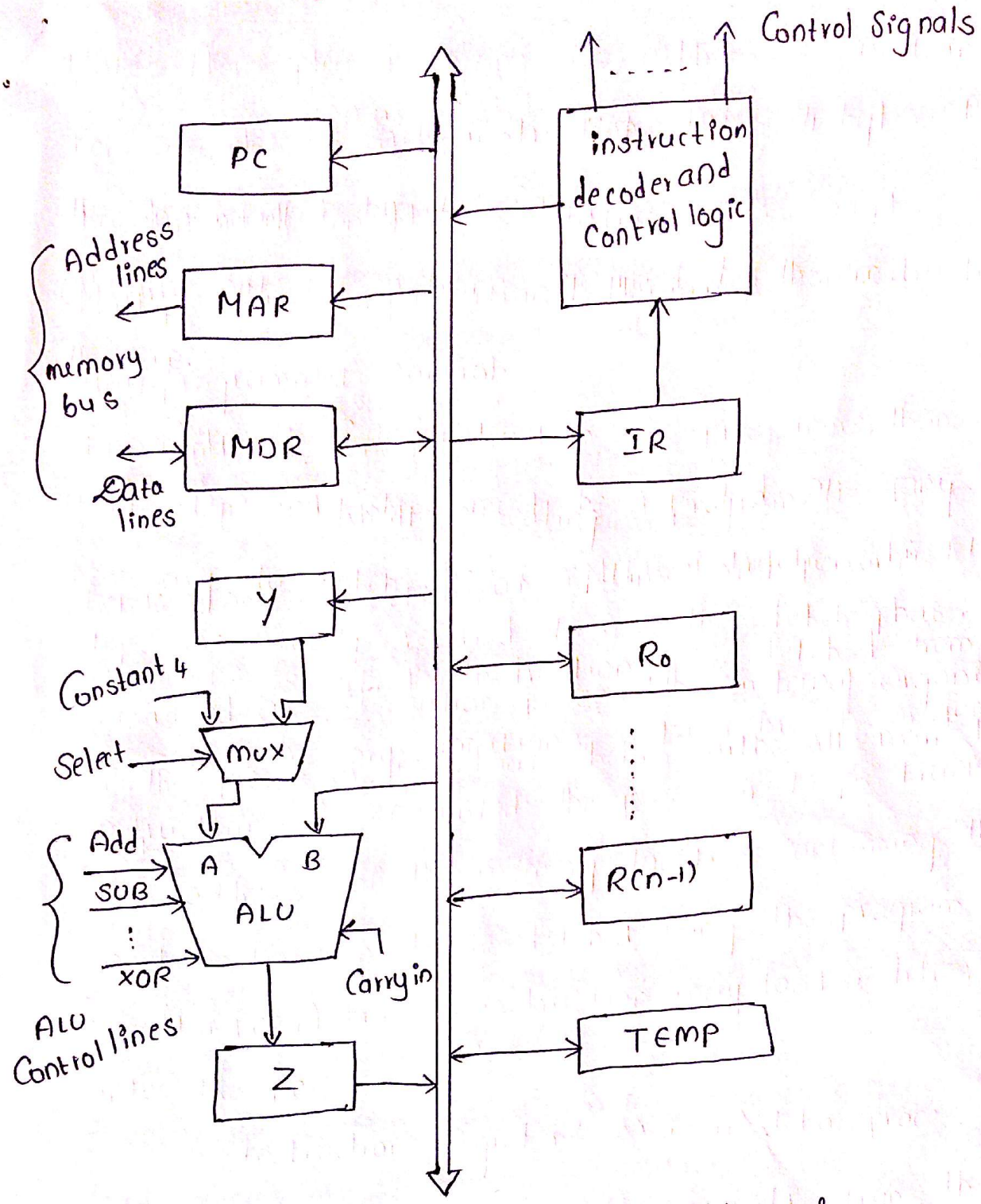
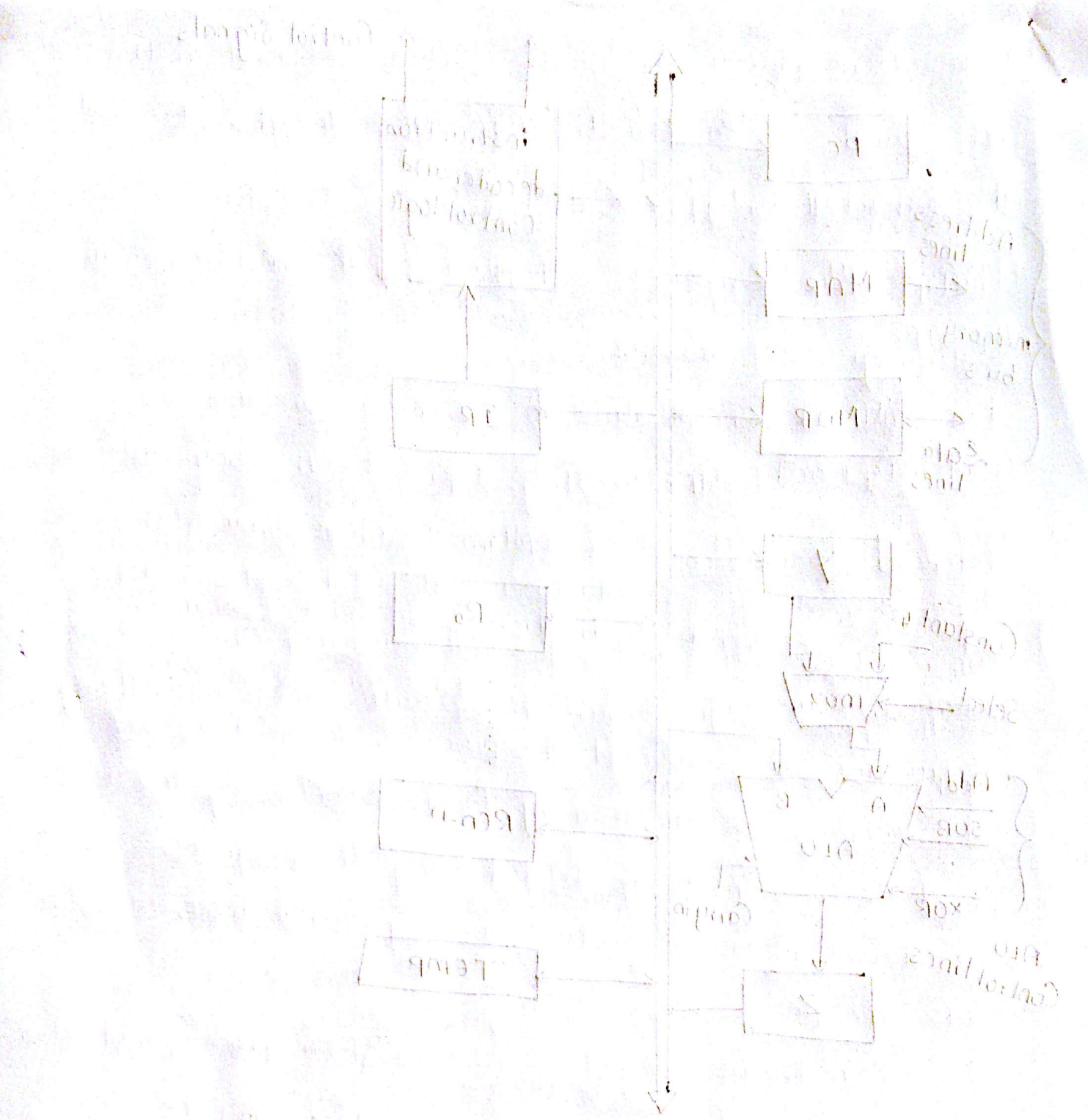


fig: Single bus Organization of the datapath inside a processor.

The data and address lines are of the external memory bus are Connected to the internal processor bus via the memory data register MDR and memory address register MAR respectively. Here MDR is a bidirectional register and MAR is a unidirectional



The processor registers R0 through R(n-1) vary b/w the processor and they are used for general purpose use by the programmer. Some Special registers Y, Z and temp which can be used as temporary purpose by the processor while processing the instruction. The multiplexer 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 programmer Counter.

With few exceptions an instruction can be excecuted 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 an arthmetic 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.

As instruction excecution progress, data are transferred from one register to another, often passing through the ALU to perform Some arithmetic or logic operation.

Register Transfer :-

Processing the instruction includes a Sequence of steps in which data are transferred from one register to another. For each register, two Control Signals are used one is to place the Contents of register on the bus (Rout) and second is to load the data on the bus into the register [Rin]

When Rin is set to 1, the data on the bus one are loaded into R. Similarly when Rout is set to 1, the Contents of Register R are placed on the bus. While Rout is equal to 0, the bus can be used for transferring data from other registers.

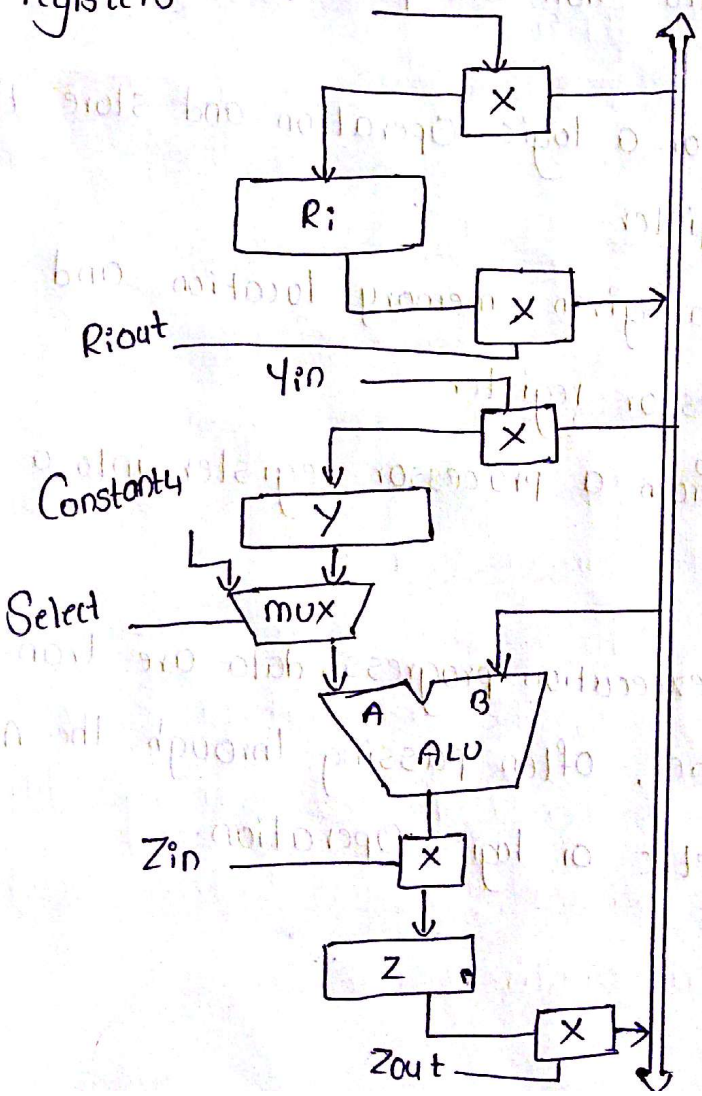


fig: input and output gating for the registers.

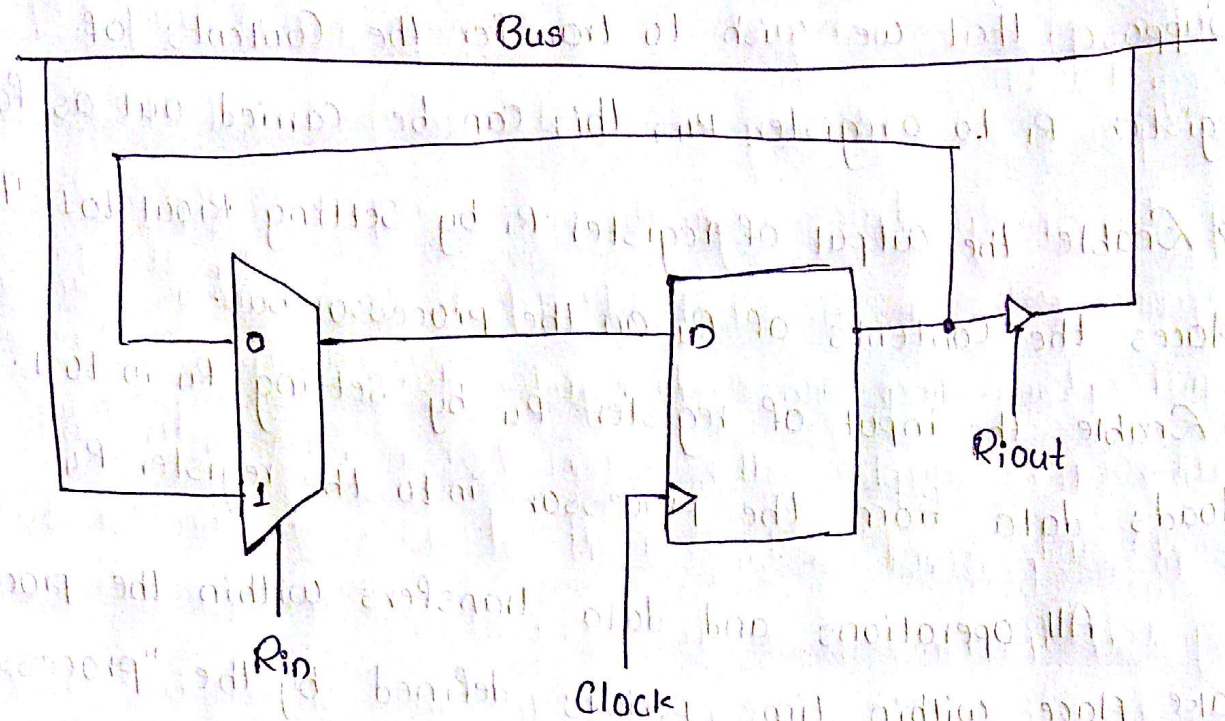
Suppose that we wish to transfer the Contents of Register R_1 to a register R_4, this can be carried out as follows:

⇒ Enable the output of register R_1 by setting R_1out to 1. This places the Contents of R_1 on the processor bus.

⇒ Enable the input of register R_4 by setting R_4in to 1. This loads data from the processor into the register R_4.

All operations and data transfers within the processor take place within time periods defined by the "processor clock". The registers Consist of edge-triggered flip-flops when edge triggered flip-flops are not used, two or more Clock signals may be needed to guarantee proper transfer of data. This is known as "Multiphase Clocking".

An implementation for one bit of register R_i shown below. A two-input multiplexer is used to select data applied to the input of an edge-triggred D flip-flop when the Control input R_in is equal to 1, the multiplexer selects the data on the bus. This data will be loaded into the flip-flop at the rising edge of the clock when R_in is equal to 0, the multiplexer feeds back the value currently stored in the flip-flop.



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. As shown in above diagrams the result produced by the ALU is stored temporarily in register Z.

The Sequence of operations to add the Contents of register R_1 to those of register R_2 and store the result in register R_3 is

1. R_1 Out, Yin
2. R_2 Out, Select Y, Add Zin
3. Zout, R_3 in

The Signals whose names are given in any step are activated for the duration of the Clock Cycle Corresponding to that step. So in Step 1 Output of R_1 and input of Y

registers are enabled. In step2 the multiplexers select signals is set to select Y, Causing the multiplexer select signal is set to selecty, Causing the multiplexer to gate the Contents of register Y to input A of the ALU. At the same time the Contents of register R_2 are gated onto the bus and hence to input B. The function performed by ALU is based on the Control Signals. In Step3 the Contents of register Z are transferred to the destination register.

Fetching a Word from Memory:-

Processor need to specify the address when trying to read information from the memory. The processor transfers the required address to the MAR, whose output is Connected to the address lines of the memory bus. At the same time it 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.

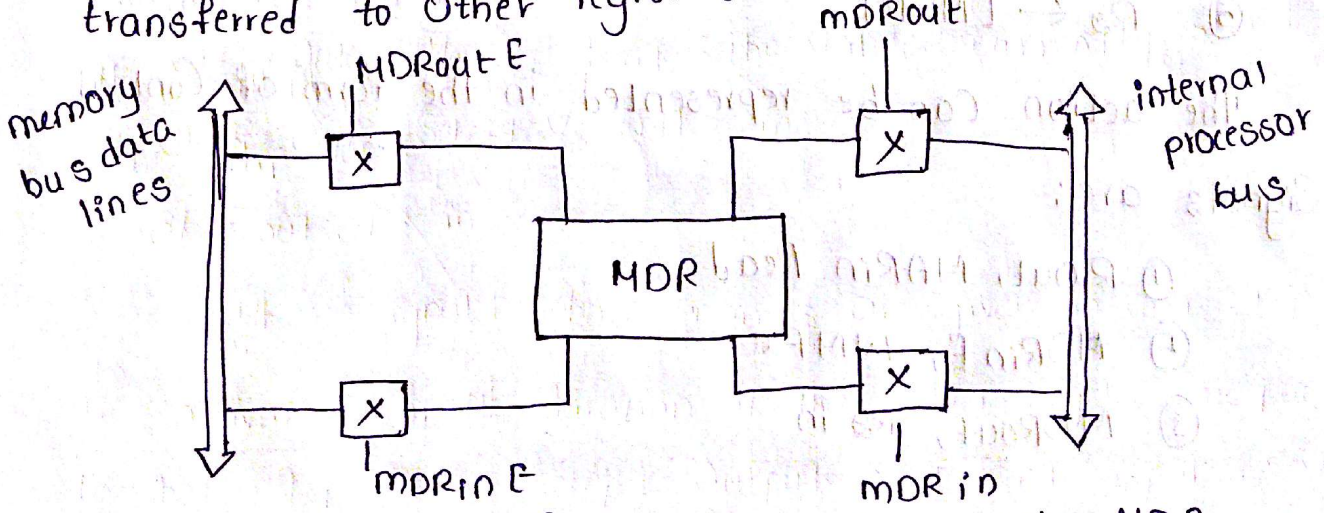


fig: Connection and Control Signals for register MDR

MDR has four signals: MDRin and MDRout Control the Connection to the internal bus, and MDRinE and MDRoutE Control the Connection to the external bus. During memory read and write operations, the timing of internal processor operations must be Coordinated with the response of the addressed device on the memory bus. The processor Completes one internal data transfer in one Clock Cycle.

To accomodate the variability in response time, of different memories, the processor waits until it receives an indication that the requested read operation has been Completed we will assume that a Control Signal called memory-function Completed [MFC] is used for this purpose.

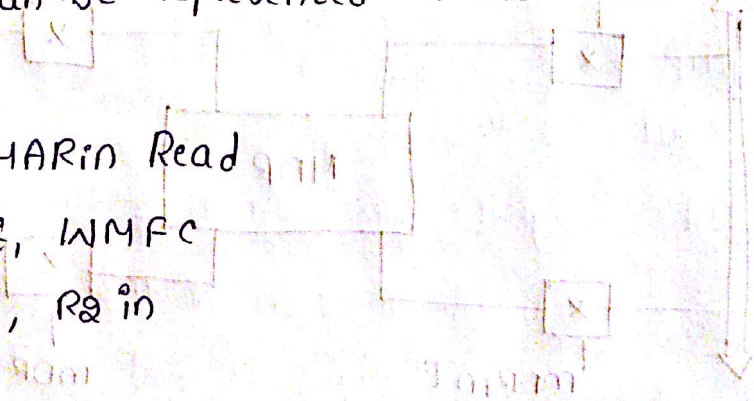
Consider an example instruction MOV [R1], R2.

The actions need to excecute 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]

The action Can be represented in the form of Control Signals are:

① R1out, MARin Read
② MDRinE, WMFC
③ MDRout, R2in



Storing a Words in MEMORY:-

Writing a word into a memory location follows a similar procedure like reading a word. The desired address is loaded into MAR. Then the data to be written are loaded into MDR, and a write Command is issued.

For excecuting the instruction MOV $R_2,[R_1]$ requires the following Sequence

1. R_1 Out, MARin
2. R_2 Out, MDRin Write
3. MDRout E, WMFC

Excecution of a Complete Instruction:-

The processor uses the sequence of elementary Operations for excecuting the instruction. Consider the instruction

Add $(R_3),(R_1$

Which adds the Contents of a memory location pointed to by R_3 to register R_1. Excecuting this instruction requires the following actions

1. Fetch the instruction
2. Fetch the first Operand
3. Perform the addition
4. Load the result into R_1

The Control sequence required for excecution of above

instructions are as follows:

Step	Action
1	PCout, MARin, Read, Select4, Add, Zin
2	Zout, PCin, Yin, WMFC
3	MDRout, IRin
4	R_3Out, MARin, Read
5	R1out, Yin, WMFC
6	MDRout, SelectY, Add Zin
7	Zout, Rin, End

In step1 instruction fetch Operation is initiated by loading the Contents of PC into the MAR and sending a read request to the memory. The select signal selects Constant 4 signal. This value is added to the Operand of B input which is not PC value. Step1 to Step3 Continues the instruction fetch phase. In step3 the word fetched from the memory is located into the IR register. Step4 to step7 performs the instruction execution.

Branch Instructions:-

A branch instruction replaces the Contents of PC with the branch target address. This address is usually Obtained by adding an offset x, which is given in the branch instruction, to the updated value of the PC. The following Control sequence implements an unconditional

branch instruction

Step	Action
1	PCout, MARin, Read, Select4, Add, Zin
2	Zout, PCin, Yin, WMFC
3	MDRout, IRin
4	Offset-field of-IRout, Add, Zin
5	Zout, PCin, End.

The offset x used in a branch instruction is usually the difference b/w the branch target address and the address immediately following the branch instruction.

Multiple Bus Organization:-

Only one data item can be transferred over the bus in a clock cycle while using single bus organization. To reduce the no.of steps needed multiple internal paths are that enable several transfers to take place in parallel. The following diagram gives the three-bus structure to Connect the registers and the ALU of a processor.

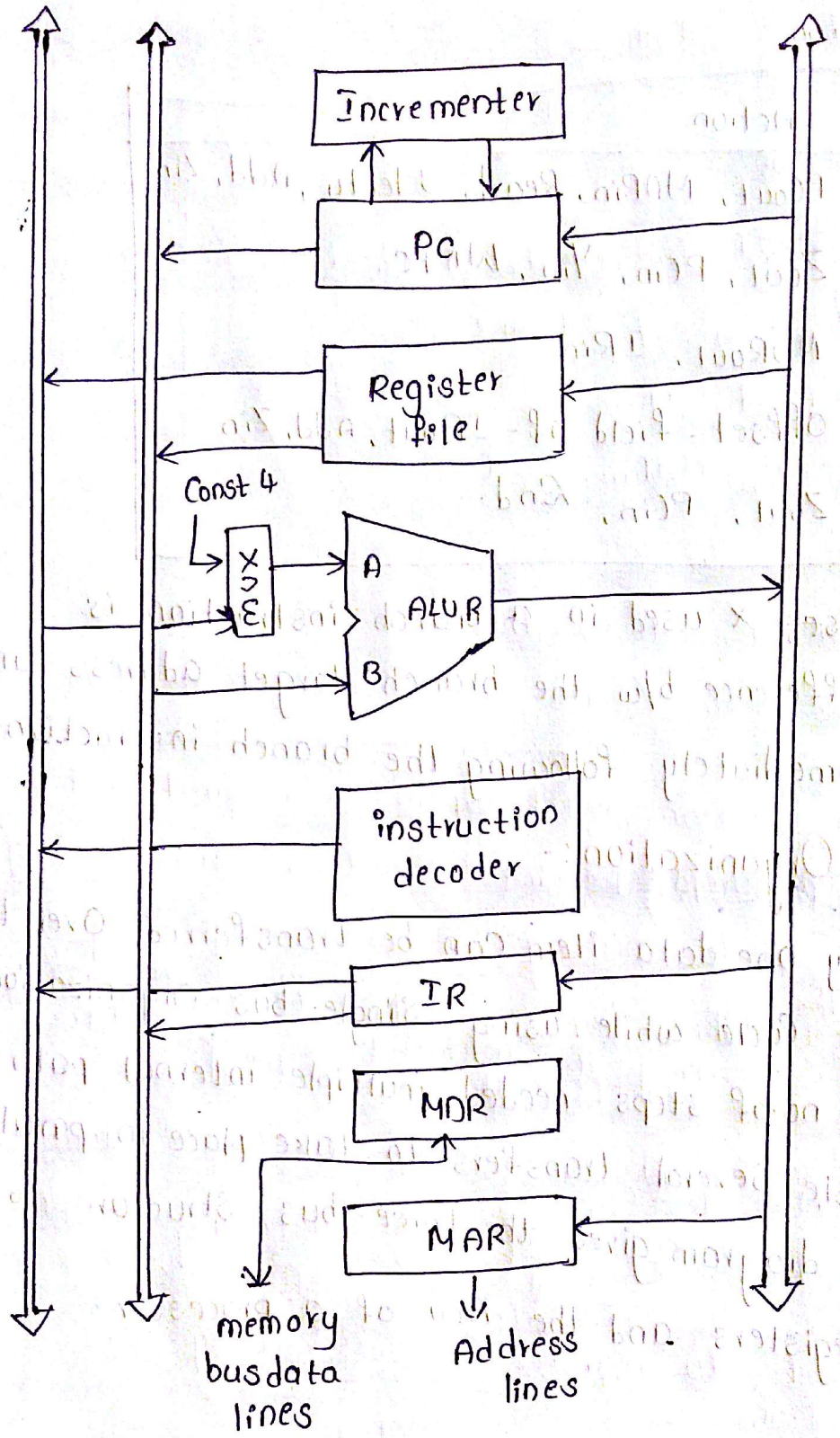


fig :- Three-bus Organization of the datapath.

All general purpose registers are Combined into a single block called the "register file". The register file have three parts. Two parts are used as two outputs, allowing the Contents of twoo different registers to be accessed

Simultaneously and have their contents placed on base A and B. The third port allows the data on bus C to be loaded into a third register during same clock cycle.

Buses A and B are used to transer the data as inputs A and B of ALU and Bus C is used as transferring the result to destination after arithmetic and logical operation performed. When R=A or R=B for such operations to inputs are modified to bus C. The three bus arrangement obviates the need for registers y and z.

The incrementer unit is used to increment the PC by 4 using the incrementer eliminates the need to add 4 to the PC using the main ALU. The source for the constant 4 at the ALU input multiplexer is still usefull when we want to increment other addresses, such as in load multiple and store Multiple instructions.

Consider the three Operand instruction

Add R4, R5, R6

Step	Action
1	PCout, R=B MARin Read inc PC
2	WMFC
3	MDRoutB R=B, IRin
4	R4outA, R5outB, SelectA, Add
	Rin, END

fig: Control Sequence diagram for above instruction.

* In Step 1 the Contents of PC are passed through the ALU using the R=B Control signal, and loaded into the MAR to Start a memory read operation. At the Sametime PC is incremented by 4. The incremented Value is loaded into PC at the end of the Clock Cycle.
* In step 2 the processor waits for MFC and loads the data received into MDR.
* In step 3 the Contents of MDR transferred into IR.
* In step 4 the excecution phase of instruction will be Completed.

∴ By providing more paths for data transfer a Significant reduction in the no.of clock cycles needed to excecute an instruction is achieved.

* Hardwired Control:-

The processor must have Capabilities of generating the Control signal needed in the proper sequence of excecuting the instructions. Computer designers uses a variety of techniques to solve this problem. These techniques are Categorized into two types. They are (i) Hardwired Control
(ii) Microprogrammed Control.

Each setup in Control sequence of excecuting the instruction Completed in one clock cycle. A Counter may be used to keep track of Control steps. The Control unit organization will be like below:

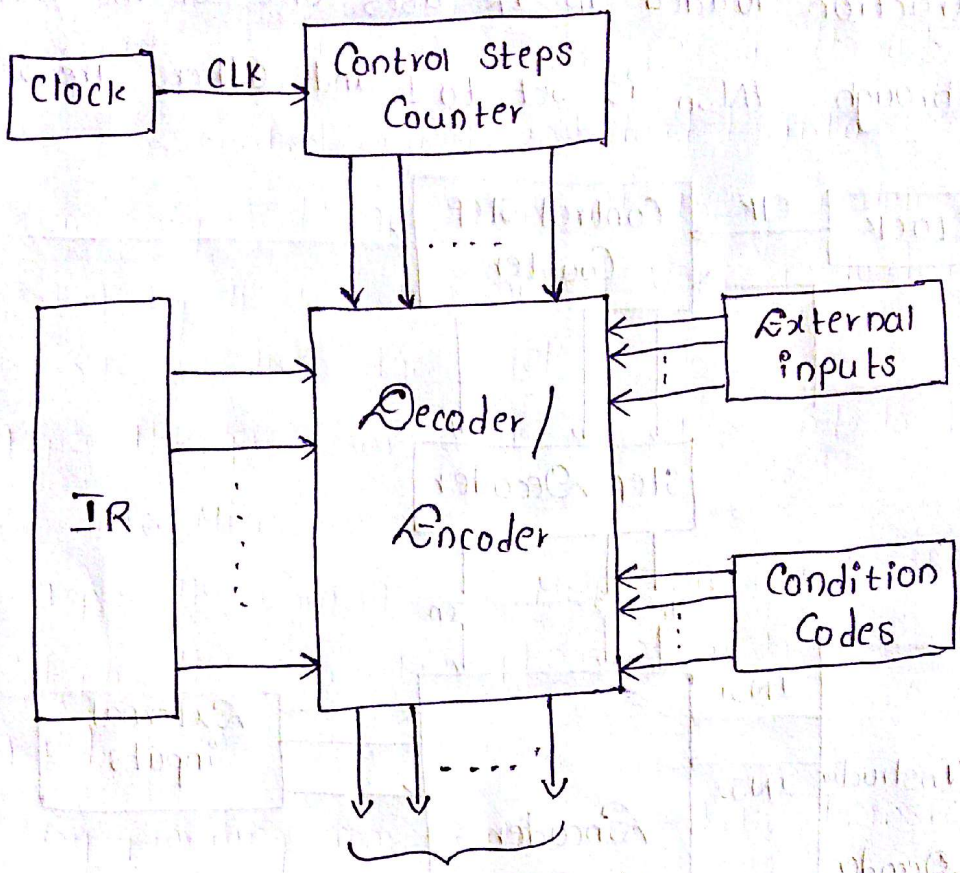


fig: Control unit organization.

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.

The decoder and encoder is a Combinational Circuit which produces required Control Signals based on all input states of it.

By Separating the decoding and encoding functions, we obtain the more detailed block diagram. The decoder provides a seperate signal line for each step in the Control Sequence. The instruction decoder Consists of a separate line for each machine instruction.

Any instruction loaded in IR uses one of the Output lines INS1 through INSn is set to '1' and others are set to '0'.

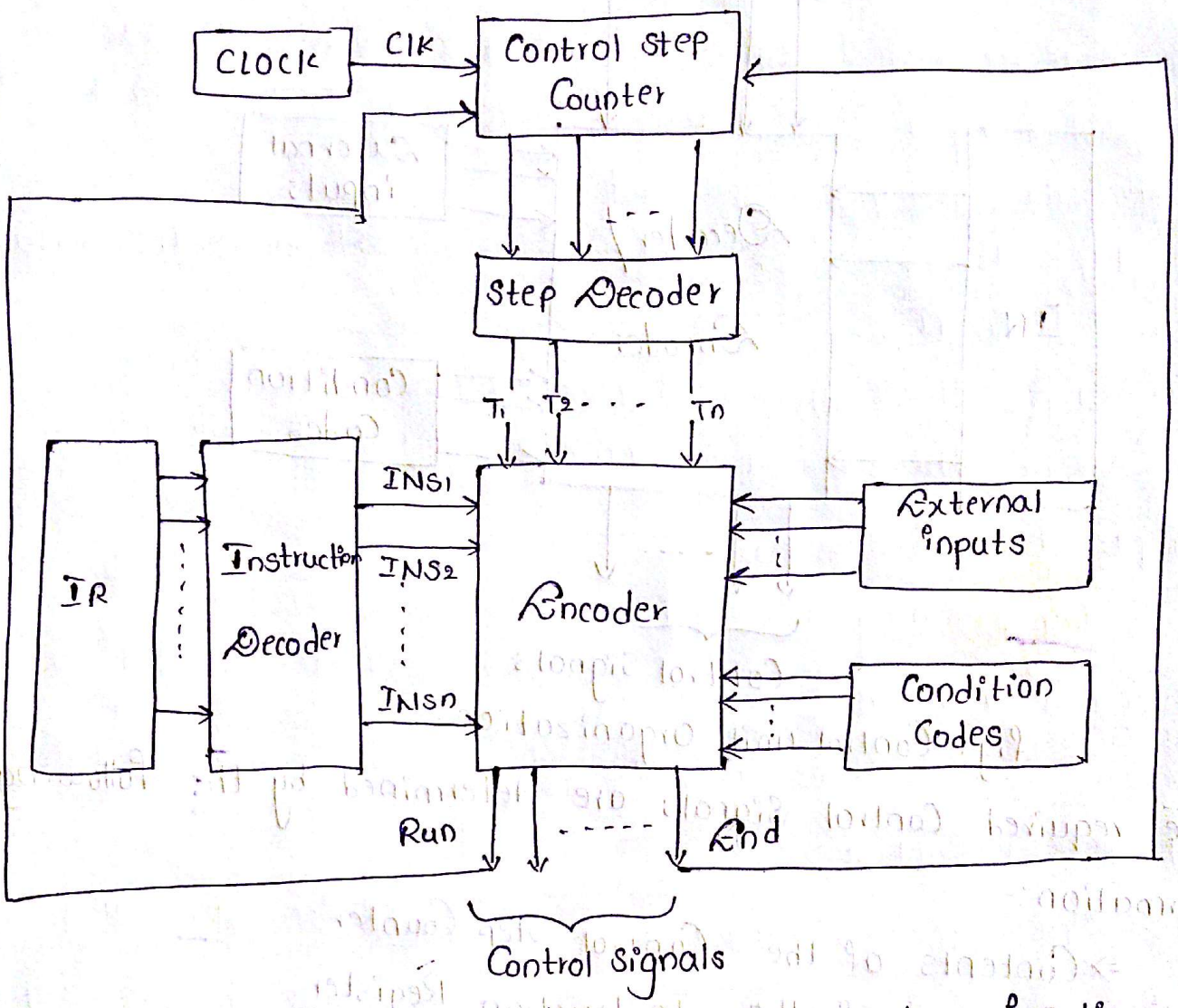


fig: Seperation of decoding and encoding functions

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 IR. A Controller that uses "hardwired Control" Can operate at high Speed.

A Complete Processor:-

A Complete processor can be designed using the following structure.

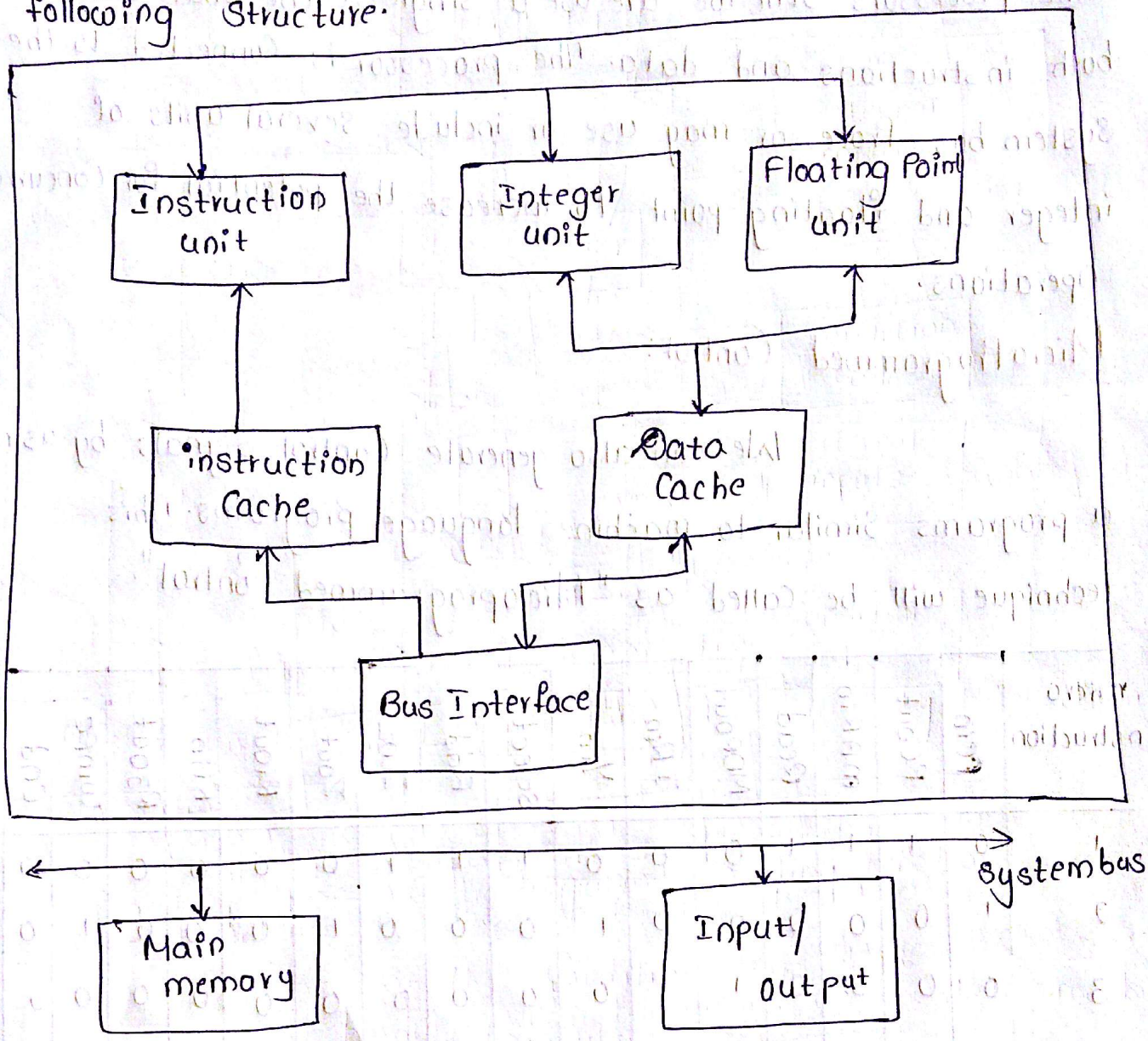


fig: Block Diagram of Complete processor.

This structure has an instruction unit that fetches instruction from an instruction Cache or from the main memory which when desired instructions are not available in Cache. It has seperate processing units to deal with integer data and floating point data. A data Cache is inserted b/w these units and the main memory.

Now a days a many processors Commonly we are using seperate Caches for instructions and data. We are using seperate Caches Some processors Still we are use a single Cache that stores both instructions and data. The processor is Connected to the System bus. Processor may use or include several units of integer and floating point to increase the potential for Concurrent Operations.

Micro Programmed Control :-

We can also generate Control Signals by using a programs Similar to machine language programs. This technique will be called as "Microprogrammed Control".

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	0	1	0	1	0	0	1

fig: Micro Instruction For Control Sequence of Add $(R_3), R_1$

The Common terms used in microprogrammed Control are Control Word (CW), microroutine, microinstruction.... etc.,

⇒ A "Control Word" is a word whose individual bits represents the various Control Signals.

⇒ A Sequence of CW's Corresponding to the Control Sequence of a machine instruction Constitutes the "micro Routine" for that instruction.

⇒ The individual Control Words in this microroutine are reffered to as "microinstructions".

The microroutines for all instructions in the instruction set of a Computer are stored in a special menu called the "Control Store". The Control unit Can generate the Control Signals for any instruction by Sequentially reading the CW's of microroutine from the Control Store. To read the Control Words Sequentially from the Control store a "micro program Counter (μPC)" is used.

When a new instruction is loaded into IR the Storing address is generated by "Starting address generator". This will be loaded into μPC and then incremented by 1 while exceeding Successive micro instructions.

When the Control unit is required to check the status of the Condition Codes or external inputs we require an appropriate logic function. The microprogrammed Control uses an approach of "Conditional Branch Microinstructions".

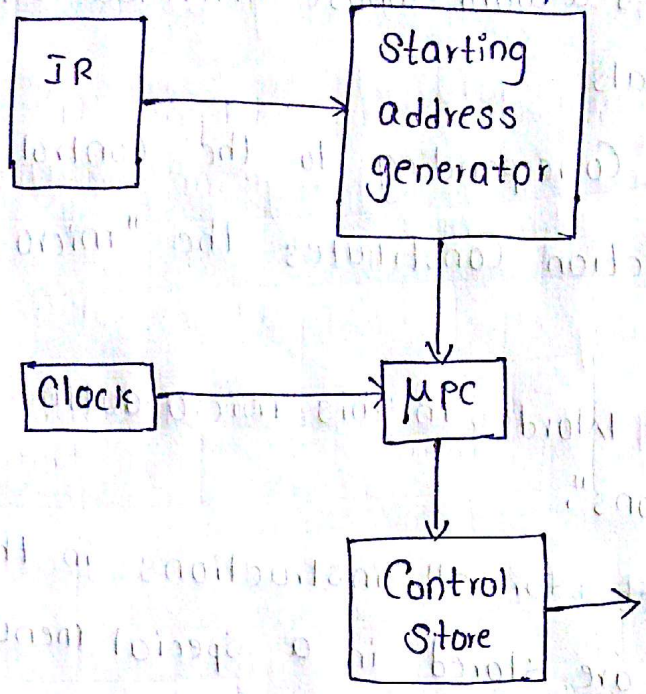


fig: Basic Organization of microprogrammed Control unit.

Address	Microinstructions
0	PCout, MARin, Read, Select4, Add, Zin
1	Zout, PCin, Yin, WMFC
2	MDRout, IRin
3	Branch to starting address of appropriate micro routine
- - - -	- - - -
25	If N=0, then branch to microinstruction 0
26	Offset-field-of-IRout, SelectY, Add Zin
27	Zout, PCin, End

fig: Micro Routine for the instruction Branch<0

In addition to branch address, the microinstruction specify which external inputs, Condition Codes, should be check as a Condition for branching to take place.

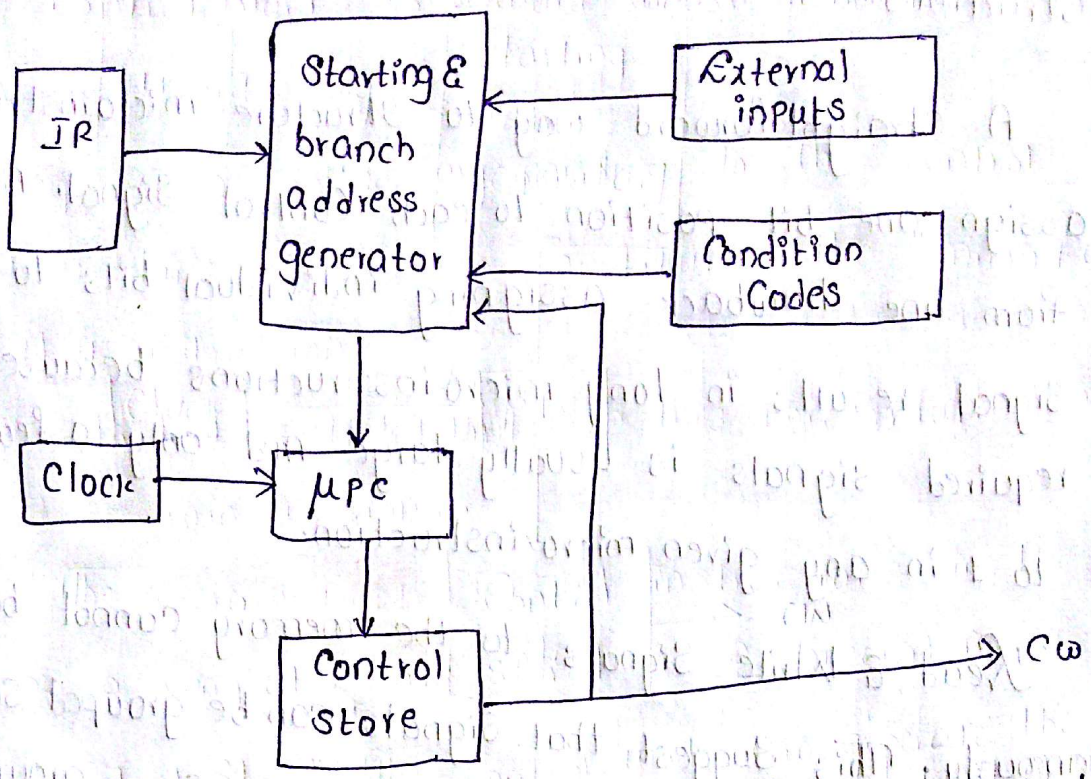


fig: Organization of the Control unit to allow Conditional branching in the microprogram.

After loading instruction into IR, a branch micro instruction transfers control to the Corresponding microroutine which is assumed to start at location 2/6 in the Control Sequence.

① When a new instruction is loaded into to IR, the μpc is loaded with the starting address of the microroutine for the instruction.

② When a branch microinstruction is encountered and the branch condition is satisfied the μpc is loaded with the branch address.

③ When an End micro instruction is encounted, the μpc is load with the address of the first CW in the microroutine for the instruction fetch cycle.

Micro Instructions :-

A Straight forward way to Structure microinstructions is to assign one bit position to each Control Signal. But it suffers from one drawback - assigning individual bits to each Control Signal results in long microinstructions because the no. of required signals is usually large. And only a few bits are set to 1 in any given micro instruction.

Read & Write Signals to the memory cannot be active Simultaneously. This Suggest that Signals can be grouped so that all mutually exclusive Signal are placed in the same group. So atmost one microoperation per group is specified in any microinstruction. Then binary Coding Scheme can be used to represent the Signals within a group.

The idea of grouping and encoding only mutually exclusive Control signals can be extended by enumerating the patterns of required Signals in all possible micro instructions. Each meaningfull combination of active Control Signals Can be assigned a distinct code that represents the microinstruction. This encoding will reduces the length of microwords but also increases the Complexity of the required decoder Circuit.

Highly encoded schemes that use Compact Codes to Specify only a small no. of Control functions in each micro instruction are referred to as "Vertical Organization".

The minimally encoded scheme which has seen previously called as "Horizontal Organization".

F_1	F_2	F_3	F_4
F_1 (4 bits)	F_2 (3 bits)	F_3 (3 bits)	F_4 (4 bits)
0000: No Transfer	000: No Transfer	000: No Transfer	0000: Add
0001: PCout	001: PCin	001: MARin	0001: SUB
0010: MDRout	010: IRin	010: MDRin	⋮
0011: Zout	011: Zin	011: TEMPin	⋮
0100: R0out	100: R0in	100: Yin	1111: XOR
0101: R1out	101: R1in		16 ALU functions.
0110: R2out	110: R2in		
0111: R3out	111: R3in		
1010: TEMPout			
1011: Offsetout			

F_5	F_6	F_7	F_8	- - -
F_5 (2 bits)	F_6 (1 bit)	F_7 (1 bit)	F_8 (1 bit)	
00: No action	0: Select y	0: No action	0: Continue	
01: Read	1: Select 4	1: WMFC	1: End	
10: write				

Supplementary Material:-

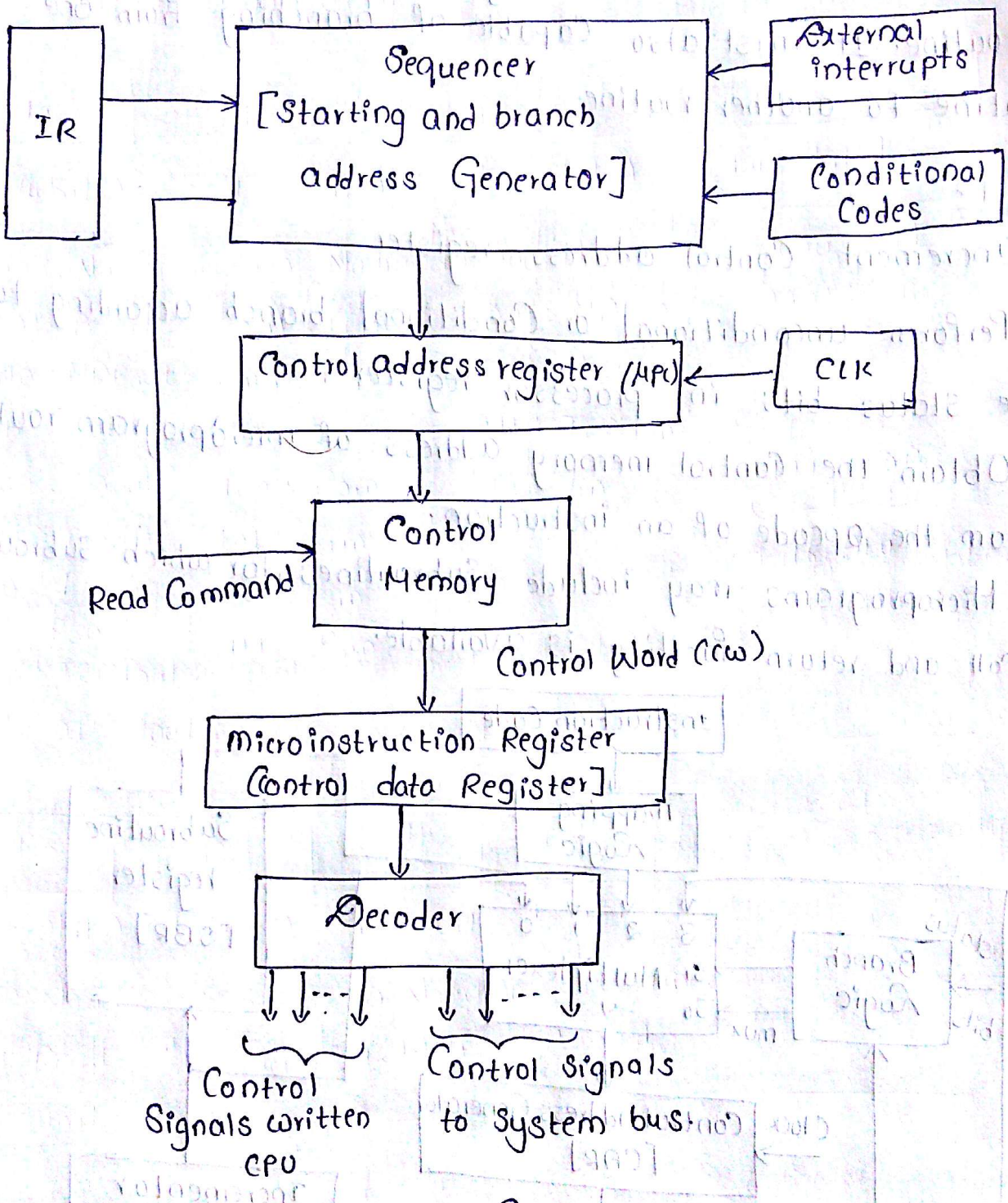


fig: Micro programmed Control unit.

Address Sequencing:-

Grouping technique is used to reduce the no. of bits in the microinstruction. Micro Instructions are stored in Control memory in groups which each group specifying a routine

The hardware Controls the address Sequencing of the Control memo It must be Capable of Sequencing the microinstructions within a routine. It must also Capable of branching from one routine to another routine.

Steps:-

* Increment Control address register
* Perform unconditional or Conditional branch according to the status bits in processor register.
* Obtain the Control memory address of microprogram routine from the opcode of an instruction.
* Microprograms may include Subroutines, for which Subroutine Call and return facility is available.

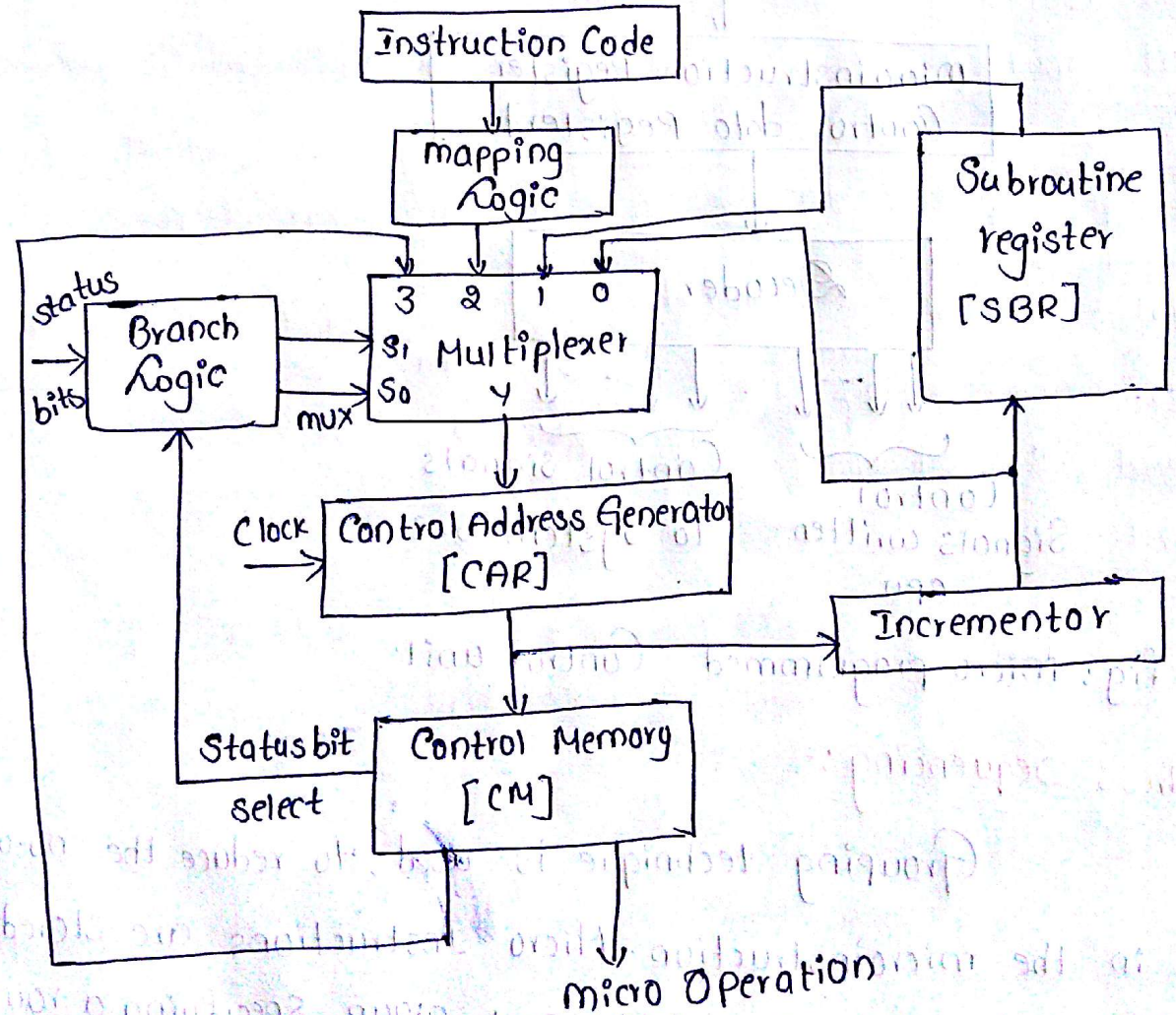
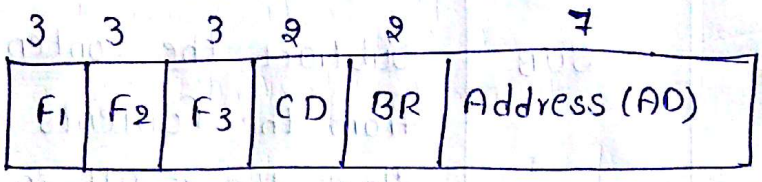


fig: Selection of address for Control memory.

Micro Instruction Format :-

Microinstruction format will be like below



* F_1, F_2, F_3 are microoperation fields. Each field is of 3 bits. They specifies micro-operations for the Computer. Each field encoded to specify seven distinct micro-operations.

*CD: A two-bit field specifies status bit condition for branch operation.

*BR: A two-bit field specifies the type of branch to be used. Branch type includes unconditional branch, branch if zero, branch if negative and so on...

*AD: This is an address field which contains a branch address. This field is seven bits since Control memory has 128 words.

F1 3 bit	Associated micro-operation	Symbol	Description
000	-	NOP	No Operation
001	AC ← AC + DR	ADD	Add DR & AC Store result in AC
010	AC ← 0	CLRAC	Clear AC
011	AC ← AC + 1	INC AC	Increment AC
100	AC ← DR	DRTAC	Copy Contents of DR in AC
101	AR ← DR(0-10)	DRTAR	Copy Contents of DR in AR
110	AR ← DC	PCTAR	Copy Contents of PC in AR
111	M[AR] ← DR	INRITE	Copy the Contents of DR into memory location addressed by AR.

F_2 3-bit	Associated Micro Operation	Symbol	Description
000	-	NOP	No Operation
001	$AC \leftarrow AC - DR$	SUB	Subtract the Contents of DR from the Contents of AC and Store the result in AC.
010	$AC \leftarrow AC \vee DR$	OR	Logically OR the Contents of DR with the Contents of AC and Store the result in AC
011	$AC \leftarrow AC \wedge DR$	AND	Logically AND the Contents of DR with the Contents of AC and Store the result in AC.
100	$DR \leftarrow M[AR]$	READ	Read the Contents of memory location addressed by AR in DR
101	$DR \leftarrow AC$	ACTOR	Copy the Contents of AC in DR
110	$DR \leftarrow DR+1$	INCDR	Increment the Contents of DR
111	$DR(0-10) \leftarrow PC$	PCTDR	Copy the Contents of PC in DR

F_3 3-bits	Associated Micro Operation	Symbol	Description
000	-	NOP	No Operation
001	$AC \leftarrow AC \oplus DR$	XOR	Logically XOR the Contents of DR and AC and Store result in AC
010	$AC \leftarrow \overline{AC}$	COM	Complement the Contents of AC
011	$AC \leftarrow shl AC$	SHL	Left shift the Contents of AC by 1-bit
100	$AC \leftarrow SHR AC$	SHR	Right-Shift the Contents of AC by 1
101	$PC \leftarrow PC+1$	INCPC	Increment the Contents of PC
110	$PC \leftarrow AC$	ARTPC	Copy the Contents of AC into PC

Condition Field :-

CD 2-bit	Symbol	Status Condition	Description
00	U	(1) always	Represents Conditional branch
01	I	DR(15) 15th bit of DR	Indirect address bit
10	S	AC(15) 15th bit of AC	Represents sign bit of AC
11	Z	AC = 0	Indicates zero value in AC

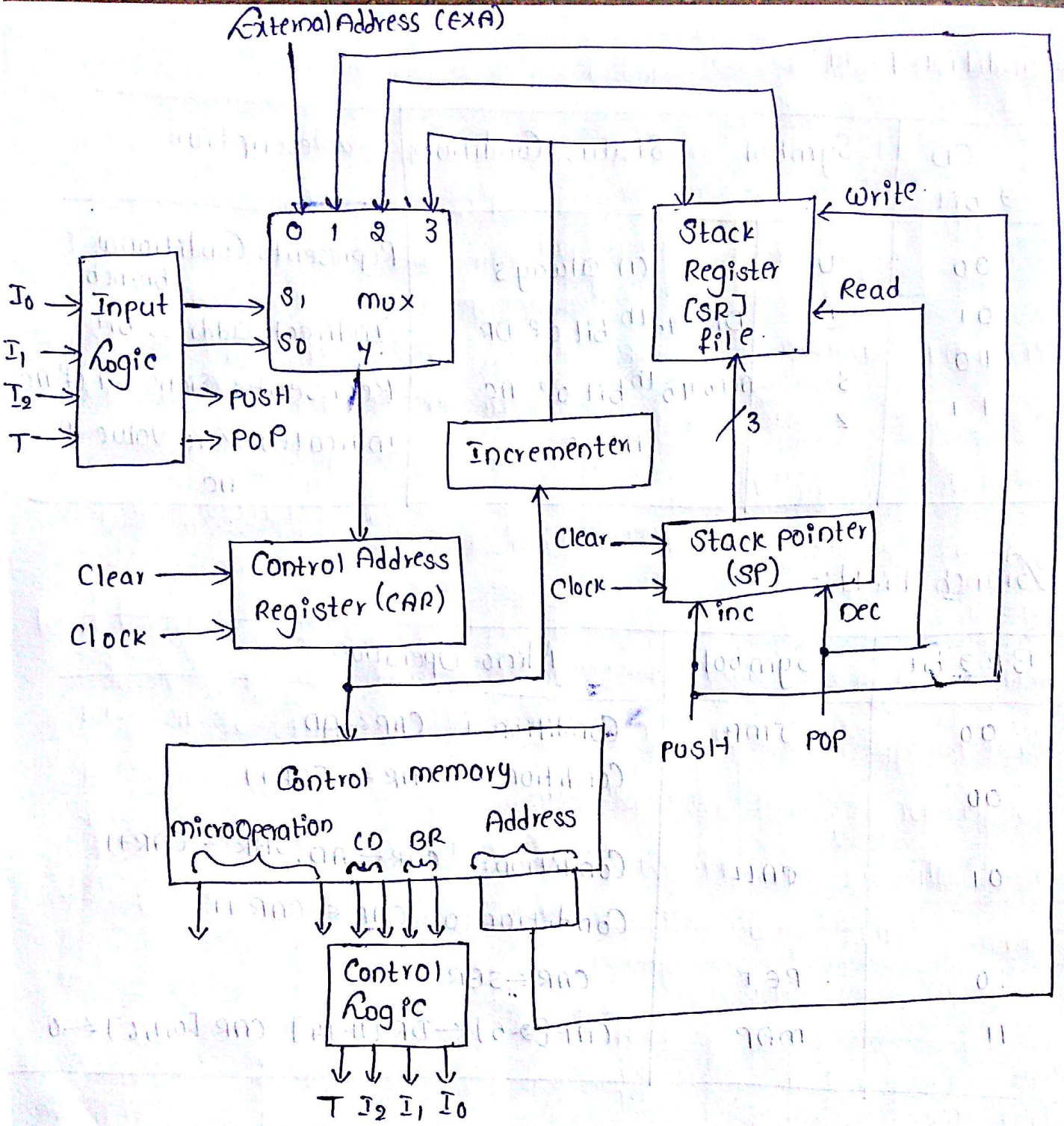
Branch Field :-

BR 2-bit	Symbol	Micro-Operation
00	JMP	Condition=1 CAR ← AD
00		Condition=0 CAR ← CAR+1
01	CALL	Condition=1 CAR ← AD, SBR ← CAR+1
		Condition=0 CAR ← CAR+1
10	RET	CAR ← SBR
11	MAP	CAR[2-5] ← DR[11-14] CAR[0,1,6] ← 0

Micro Program Sequencer :-

The submit of microprogrammed Control unit which presents an address to the Control memory is called "microprogram Sequencer". The next address logic of the Sequencer determines the specific address source to be loaded into the Control address register.

The following diagram gives the microprogram Sequencer



* It Contains a multiplexer that selects an address from four four sources and routes it into a Control address register. The Output from CAR provides the address for the Control memory.

* The Contents of CAR incremented and applied to multiplexer and stack register. Which register Selected will be indicated by Stack pointer.

* Inputs I_0, I_1, I_2 and T derived from CD and BR fields of microinstruction specify the operation for the Sequencer.

When we want to work with Subroutines then PUSH, POP Signals will come in use. During Subroutine Call the incremented address is stored in the stack. This address also called as "Return Address".

The function table of microprogram Sequencer is like below

I_2	I_1	I_0	T	S_1	S_0	Operation	Description
X	0	0	X	0	0	CAR ← EXA	Transfer external address.
1	0	1	1	0	1	CAR ← BRA SAR ← CAR+1	Branch to Subroutine. and store the next instruction address [stack].
0	0	1	1	0	1	CAR ← BRA	Transfer Branch address.
X	1	0	X	1	0	CAR ← SR	Transfer from Stack register
0	1	1	0	1	1	CAR ← CAR+1	Increment Address.