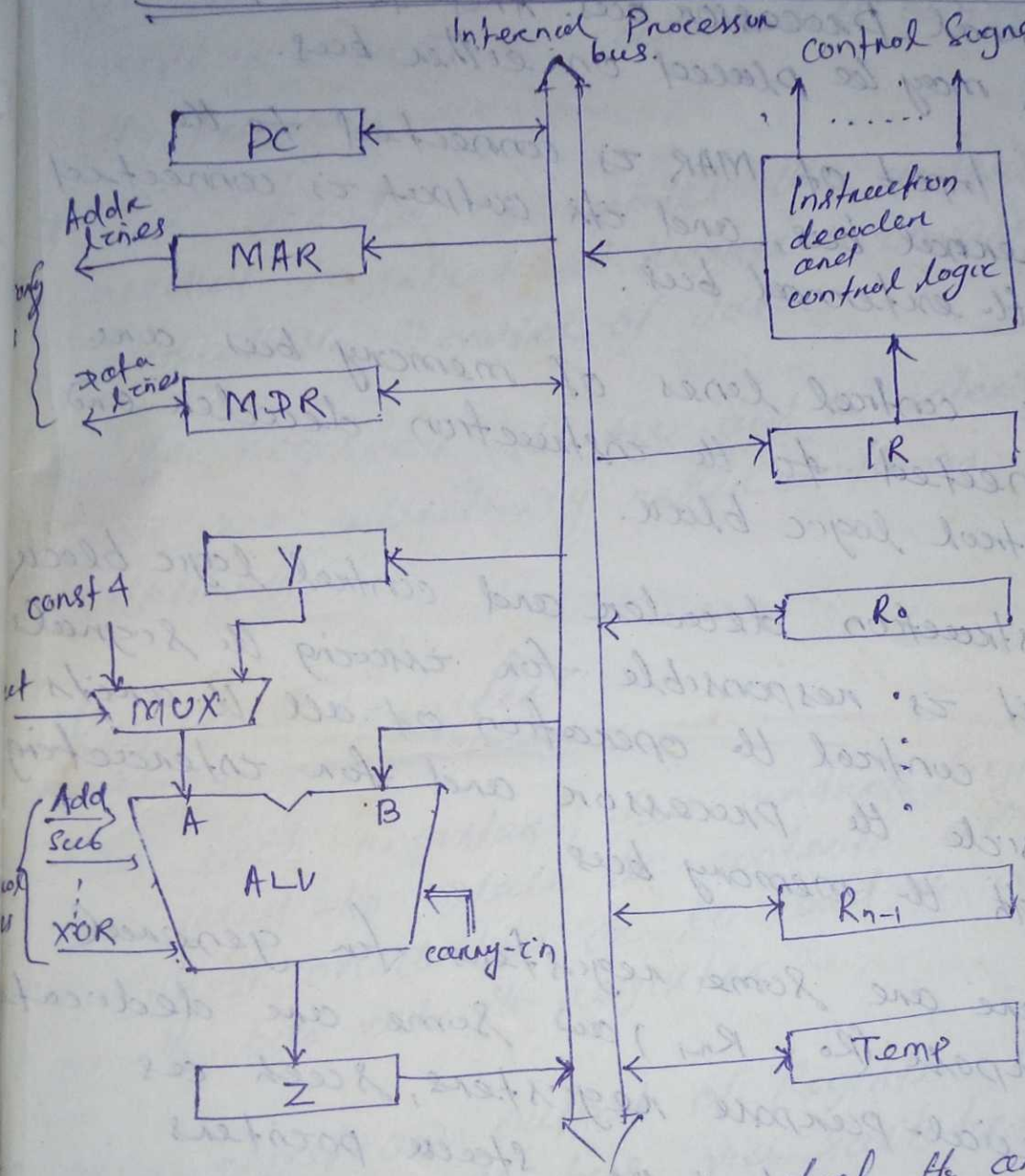
Single bus Organization of data path inside a Processor.



The above figure shows, in which the arithmetic and logic unit (ALU) and all the registers are interconnected via a single common bus. 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. Because data may be loaded into MDR either from the memory bus or from the internal Processor bus. And 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 memory bus are connected to the instruction decoder and control logic block.

* Instruction decoder and control logic block 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.

* There are some registers for general purpose ($R_0 - R_{n-1}$) and some are dedicated special-purpose registers, such as Index registers or stack pointers. Three registers Y, Z and Temp. are never referenced explicitly by any instruction. They are used by the processor for temporary storage during execution of some instructions.

* 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 Program counter.

* The decoder generates the control signals needed to select the registers involved and direct the transfer of data.

* The registers, the ALU, and the interconnecting bus are collectively referred to as the datapath.

Register Transfer:

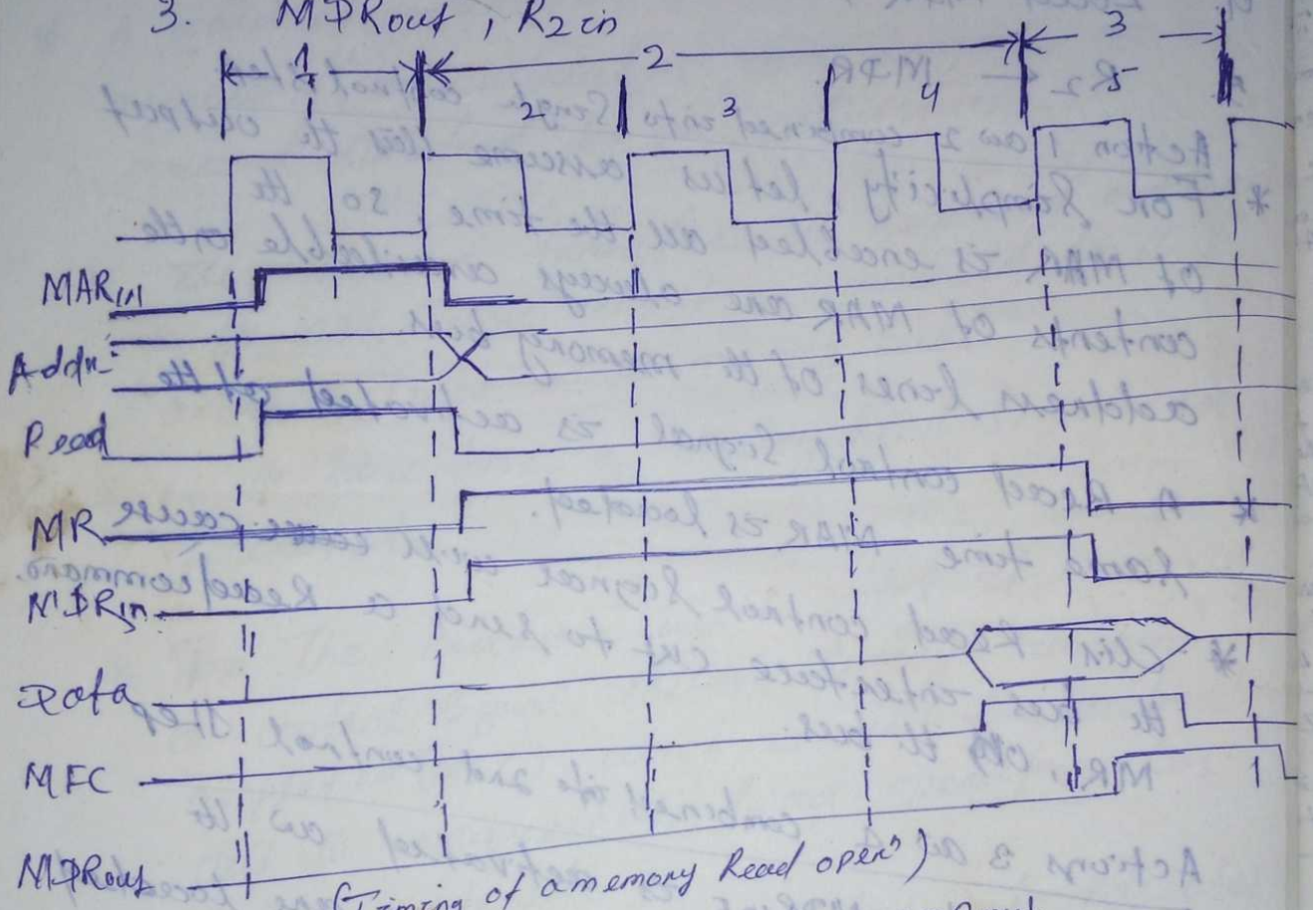
For each register, two control signals are used to place the contents of that register on the bus or to load the data on the bus into the register.

* The input and output of Register R_i are connected to the bus via switches controlled by the signals $R_i in$ and $R_i out$ respectively.

* When $R_i in$ is set to 1, the data on the bus are loaded into R_i. Similarly, when $R_i out$ is set to 1, the contents of Register R_i are placed on the bus.

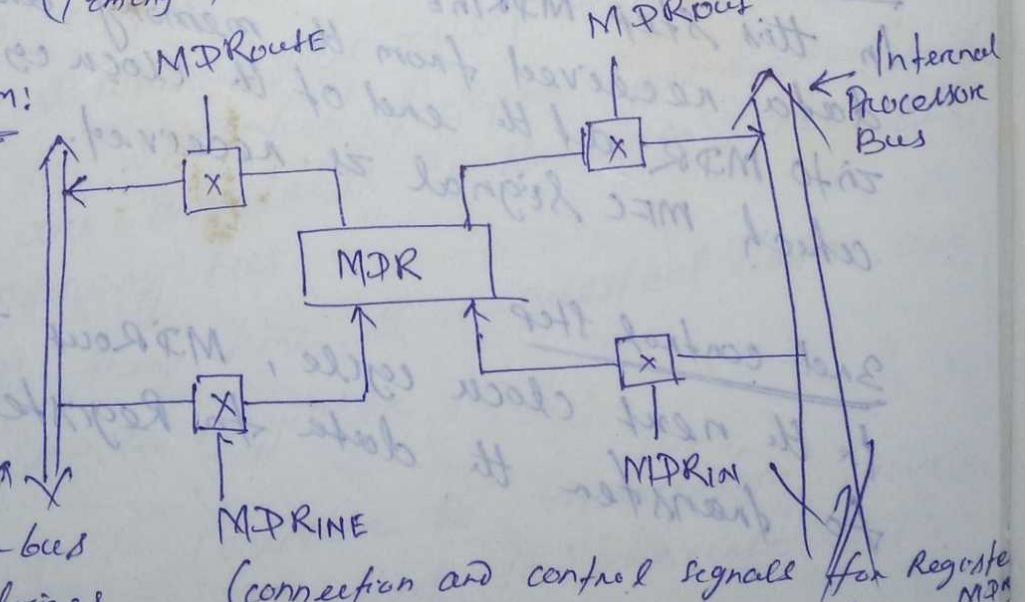
So the three control steps are:

1. R_{1out}, MAR_{IN}, Read
2. MDR_{inE}, WMFC (wait MFC)
3. MDR_{out}, R_{2in}



(Timing of a memory Read operation)

Diagram:



(connection and control signals for Register MDR)

It has four control signals: 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.

Storing A word In Memory

consider the instruction Move R_2, (R_1) requires the following sequence.

1. R_1 out, MAR in
2. R_2 out, MDR in, Write
3. MDR out E, WMFC

Execution of a complete Instruction

Let's 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.

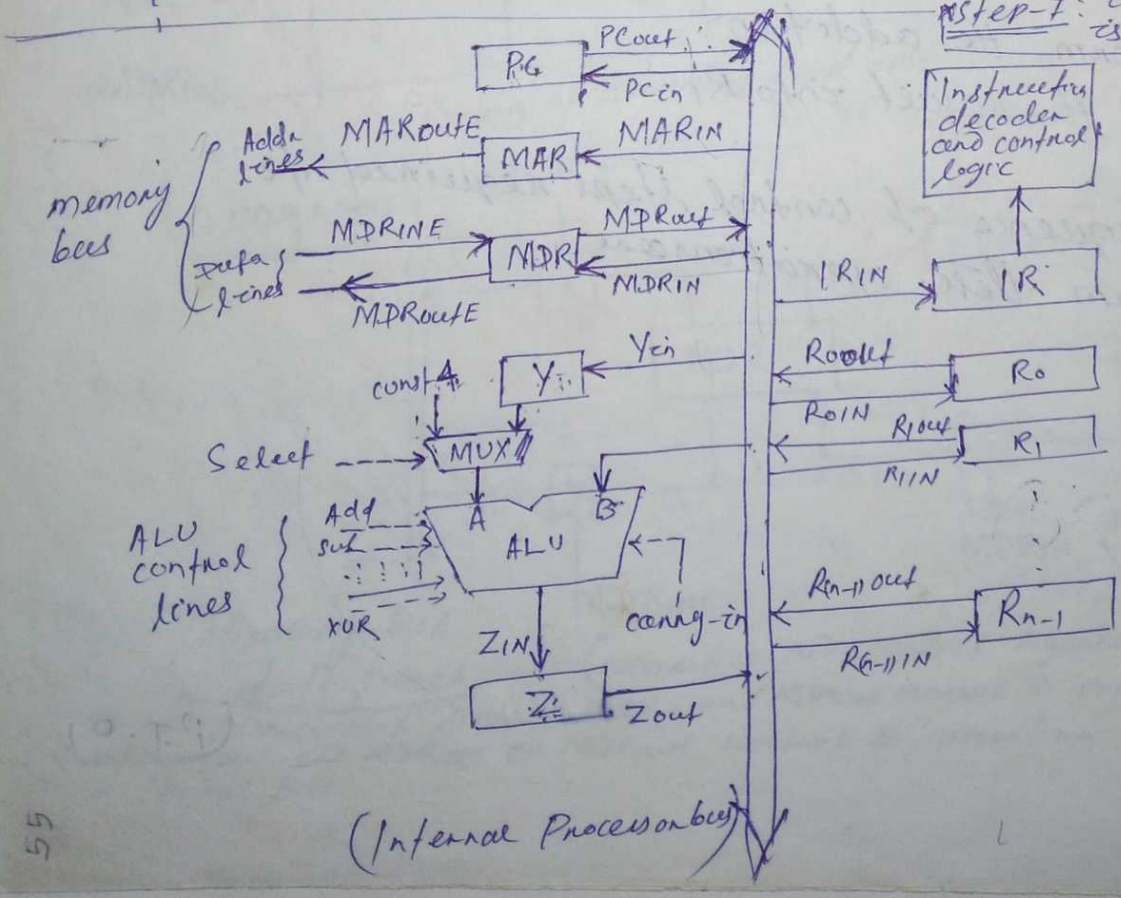
Executing this instruction requires the following actions.

1. Fetch the instruction. PC ← PC+1
2. Fetch the first operand (the contents of the memory location pointed to by R_3)
3. Perform the addition
4. Load the result into R_1.

The Sequence of control steps required to perform these operations are

(P.T.O)

Step	Action	comments
1.	PCout, MARin, Read, Select4, Add, Zin	Step-1: The addr of the instn is ~~sent to~~ PC, sent to MAR, Read signal is generated, 4 is selected to add to the content of PC to point to the next instn and put in Z register, which is sent to PC over the processor bus. The instn is loaded into MDR from memory.
2.	Zout, PCin, MDRINE, WMFC	
3.	MDRout, IRin	Step-3: Now the instn is available in MDR, will be sent to IR for decoding.
4.	R3out, MARIN, Read	Step-4: The address of the operand into MAR. Read signal is generated. At the beginning of Step 4 decoding ~~takes~~ takes place.
5.	Rlout, YIN, ~~WMFC~~ MDRINE, WMFC	Step-5: The second operand (content of R_1) is sent to ALU Register Y over the processor bus. The 1st operand is loaded into MDR from memory.
6.	MDRout, Select Y, Add, Z in	Step-6: The 1st operand is available at port A and 2nd operand is at B. After addition by the ALU Result is placed in Z register
7.	Zout, RIN, END	Step-7: The result in Z is loaded into



* Step 1 through 3 constitute Instruction fetch phase.
* Decoding is done at the beginning of the step 4.
* Steps 4 through 6 constitute the operand fetch and execution phase.
* step-7 is the result write stage.

Branch Instructions : unconditional branch

* A branch instruction replaces the contents of the PC with the branch target address.
* This target address is obtained by adding an offset X, to the updated value of PC.
* The offset value is available in the decoding stage.
* The updated PC value is available at point A through Y and the offset is available at point B through processor bus from IR.

The sequence of control steps for unconditional branch Instruction.

Step	Action
1.	PCout, MARIN, Read, Select 4, Add; Zin
2.	Zout, PCin, Yin, MDRINE, WMFC
3.	MDRout, IRIN Select Y
4.	Offset-field-of-IRout, Add, Zin
5.	Zout, PCIN, END.

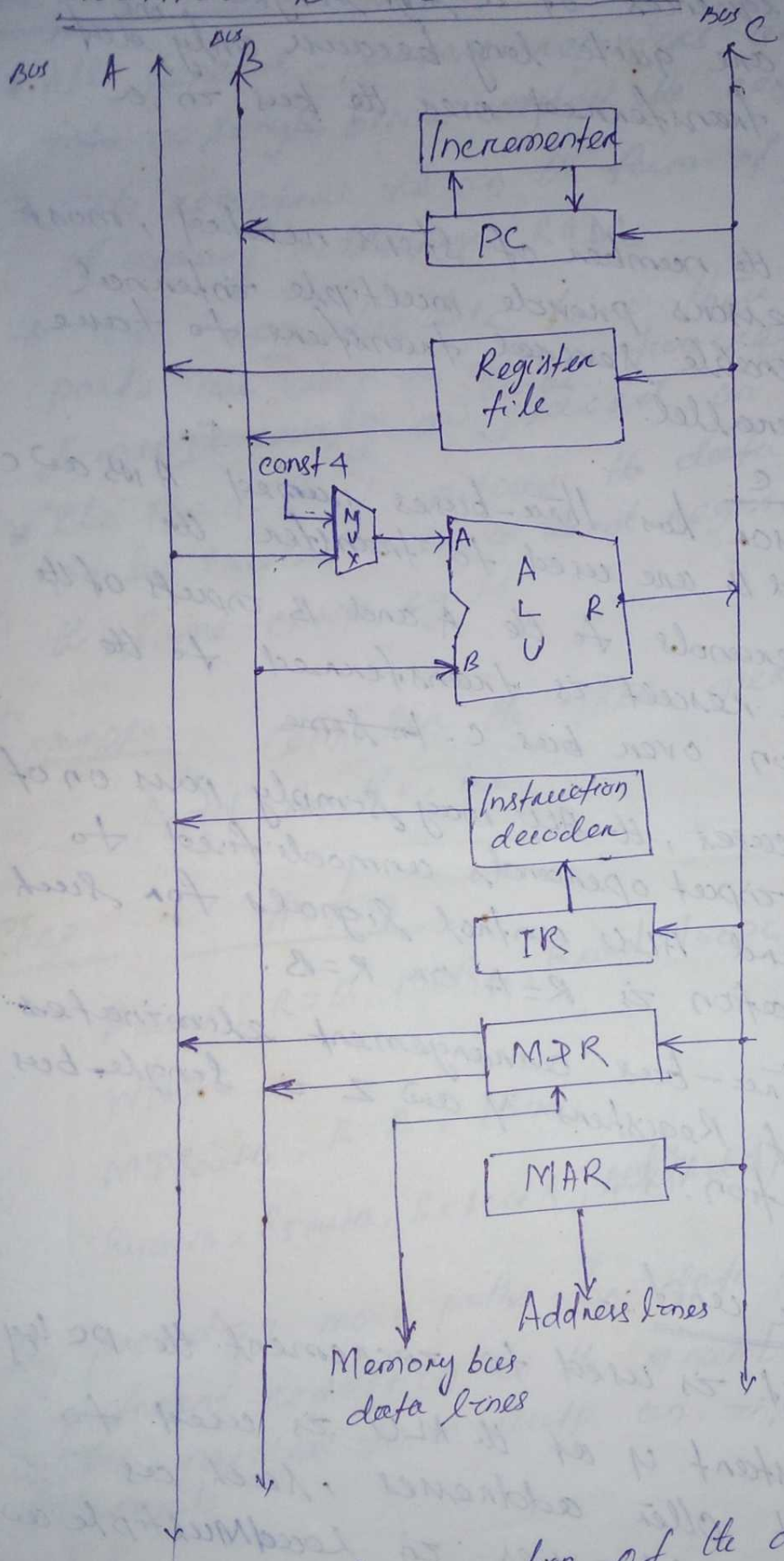
In case of conditional branch Instruction step-4 is modified to offset-field-of-IRout, Add, if N=0 then END
if N=0 the processor returns to step 1 immediately after

Step 4.

* If N=1 step: 5 is performed to load a value into the PC, thus performing the branch operation.

(5)

MULTIPLE-BUS ORGANIZATION



Three-bus Organization of the datapath.

The control sequences of a single-bus organization are quite long because only one data item can be transferred over the bus in a clock cycle.

* To reduce the number of steps needed, most of the processors provide multiple internal paths that enable several transfers to take place in parallel.

BUS A, B and C

* This processor has three-buses named A, B and C. Buses A and B are used to transfer the source operands to the A and B inputs of the ALU and the result is transferred to the destination over bus C. ~~In some~~
* In some cases, the ALU may simply pass on of the two input operands unmodified to bus C, and ALU control signals for such an operation is R=A or R=B.
* The three-bus arrangement eliminates the use of Registers Y and Z in single-bus organization.

Incrementer unit:-

* This unit is used to increment the PC by 4
* The constant 4 at the ALU is used to increment other addresses, such as the memory addresses in LoadMultiple and StoreMultiple instructions.

Register file

* All general-purpose registers are combined into a single block called the register file.
* The registers is in the form of an array of memory cells like RAMs.
* The register-file have three ports. Two ports are used to access two registers simultaneously and placed on buses A and B. can be
* The third-port allows the data on bus C to be loaded into a third register during the same clock cycle.

Example: Consider the ~~three~~ operand instruction

Add R_4, R_5, R_6

Step	Action
1.	PCout, R=B, MAR_{IN}, Read, IncPC
2.	WMFC
3.	MDRoutB, R=B, IR_{IN}
4.	R_4outA, R_5outB, SelectA, ~~Add~~ Add, R_{6IN}, End.

By providing more paths for data transfer a significant reduction in the number of clock cycles needed to execute an instruction is achieved.

(6)

To execute instructions, the processor must have some means of generating the control signals needed in proper sequence.

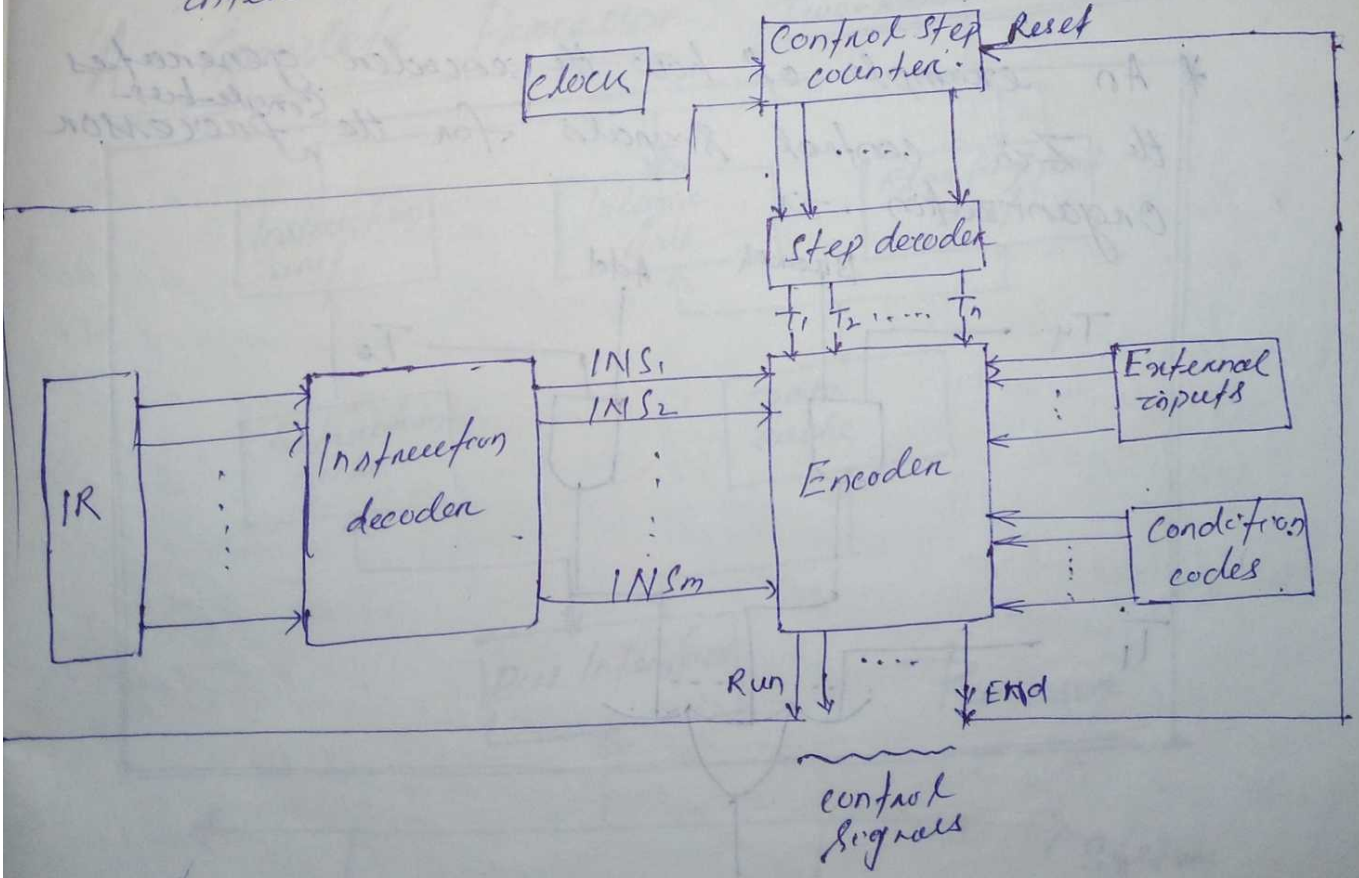
Two approaches are used for the purpose.

(1) Hardwire Control
(2) Microprogrammed control.

Hardwire Control

The required control ~~Sequence~~ 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.



* A counter is used to keep track of the control steps.

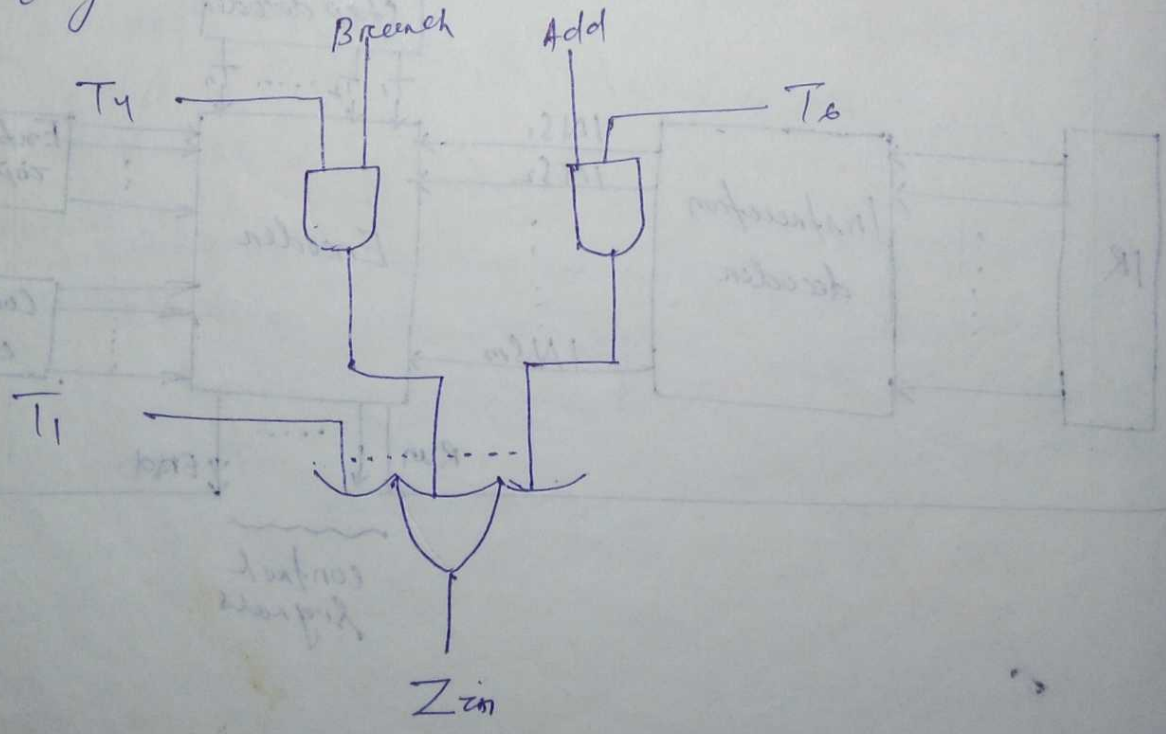
* The step decoder provides a separate signal line for each step, or time slot, in the control sequence.

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

* For any instruction ~~decoder~~ loaded in the IR, one of the output lines INS, through INSm is set to 1, and all other lines are set to 0.

* All the input signals to the encoder block are combined to generate the individual control signals Yin, PCout, ~~etc~~ Add, End, and so on.

* An example of how the encoder generates the Zin control signals for the single-bus-organization is



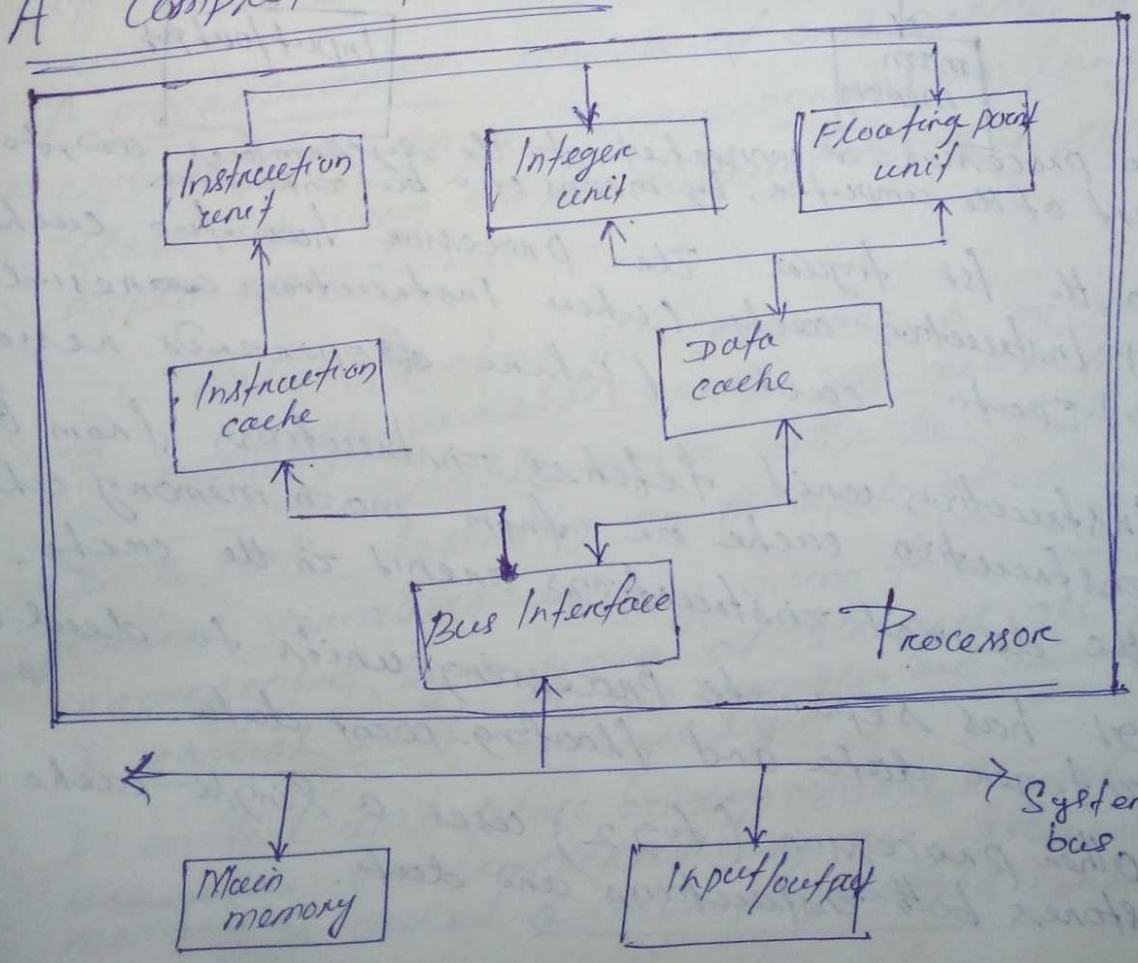
$$Z_{in} = T_1 + T_6 \cdot ADD + T_4 \cdot BR + \ldots$$

* This signal is asserted during time slot T_1 for all instructions, during T_6 for an Add instruction, during T_4 for an unconditional branch instruction and so on.

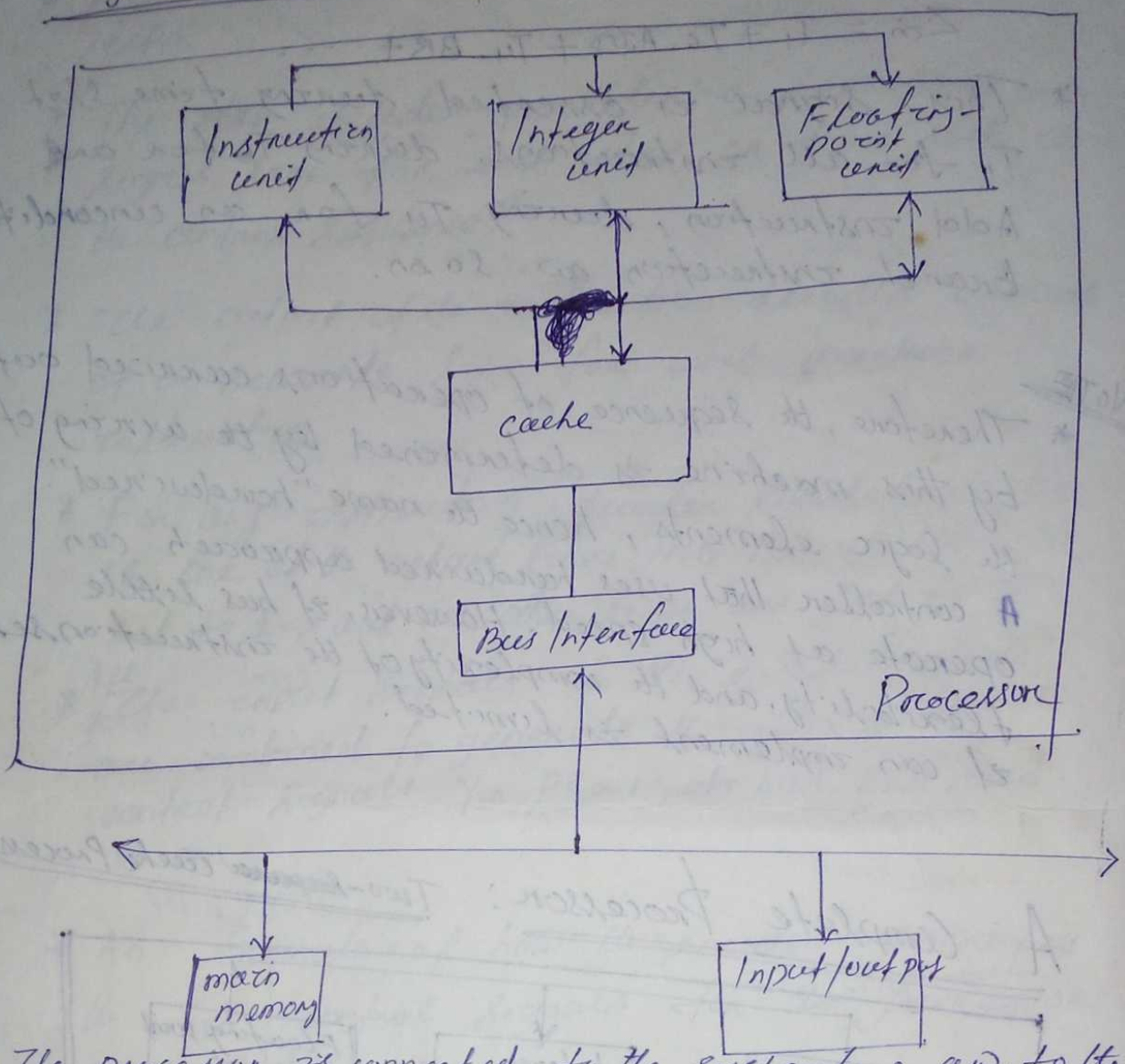
NOTE

* Therefore, the sequence of operations carried out by this machine is determined by the wiring of the logic elements, hence the name "hardwired". A controller that uses hardwired approach can operate at high speed. However, it has little flexibility, and the complexity of the instruction set it can implement is limited.

A Complete Processor: Two-~~Level~~ cache Processor



Single cache Processor



* The processor is connected to the system bus and, to the rest of the computer, by means of a bus interface.
* In the 1st figure the processor has two caches
  (i) Instruction cache (where Instructions reside)
  (ii) Data cache (where operands reside)
* Instruction unit fetches instructions from the instruction cache or from main memory when the desired instructions aren't in the cache.
* It has separate Processing units to deal with integer data and floating-point data.
* Other processor (fig 2) uses a single cache that stores both instructions and data.

7)

Microprogrammed Control

* The function of the control unit in a digital computer is to initiate sequences of microoperations.
* When the control signals are generated by hardware using conventional logic design techniques, the control unit is said to be hardwired. It is difficult to design and test such a piece of hardware. Furthermore, the design is relatively inflexible, i.e. it is difficult to change the design if one wishes to add a new machine instruction.

* Microprogramming is a second alternative for designing the control unit of a digital computer, in which the logic of the control unit is specified by a microprogram.

A microprogram consists of a sequence of micro-instructions in a microprogramming language. These are very simple instructions that specify micro-operations.

* The control function that specifies a microoperation is a binary variable (0 or 1). The active state of a control variable (control signals) may be either the 1 state or the 0 state, depending on the application. The control variables at any given time can be represented by a string of 1's and 0's called a control word. [control variables are known as control signals]

* A control unit whose binary control variables (microoperations) are stored in memory is called a microprogrammed control unit.

* A memory that is part of a control unit and stores the set of microinstructions is referred to as a control memory / control store.

* The control memory can be a read-only memory (ROM). ROM words are made permanent during the hardware production of the unit. The content of the word in ROM at a given address specifies a microinstruction.

* Reading a microinstruction (a word) from the control memory is the same as executing that microinstruction.

* A more advanced development known as dynamic microprogramming permits a microprogram to be loaded initially from an auxiliary memory such as a magnetic disk. Control units that use dynamic microprogramming employ a writable control memory. This type of memory can be used for writing (to change the microprogram) but used mostly for reading.

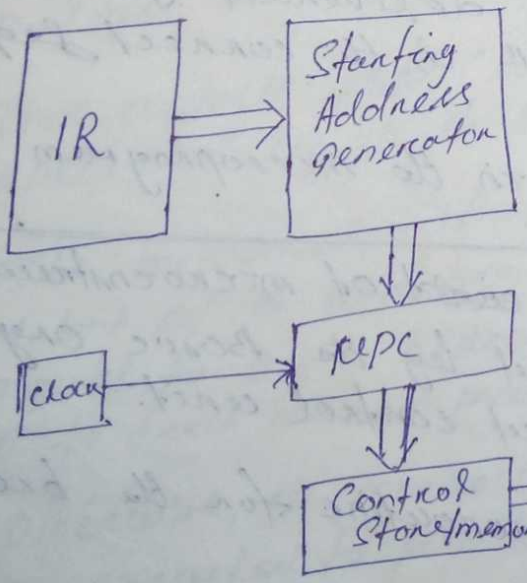
* A control word (CW) is a word whose individual bits represent the various control signals.

* A sequence of CWs corresponding to the control sequence of a machine instruction constitutes the microroutine for that instruction, and the individual control words in this microroutine are referred to as microinstructions.

An Example of microinstructions for machine instruction Add $(R_2), R_1$

micro instruction	...	PCin	PCout	MARin	Read	MDRout	IRin	Yin	Select	Add	Zin	Zout	R1out	R1in	R3out	WMFC	END
1																	

Basic Organization of a Microprogrammed control unit



Basic Organization of a microprogrammed control unit

* The microroutines for all instructions in the instruction set of a computer are stored in a special memory called the control memory/control Store.

* The control unit can generate the control signals for any instruction by sequentially reading the CWs of the corresponding microroutine from the control store/memory.

* To read the control words sequentially from the control store/control memory, a microprogram counter ~~PC~~ (μPC) is used.

* Every time a new instruction is loaded into the IR, the output of "Starting Address Generator" is loaded into μPC. The μPC is then automatically incremented by the clock, causing successive microinstructions to be read from the control store. Hence the control signals are delivered to various parts of the processor in the correct sequence.

Conditional branching in the microprogram

Conditional branching ~~exists~~ of microinstruction could be implemented by the Basic organisation of a microprogrammed control unit.

Example: of a microroutine for the branch instruction.

(8)

Address	Action / microinstruction
0	PCout, MARin, Read, Select4, Add, Zin
1	Zout, PCin, Yin, WMFC
2	MDRout, IRin
3	Branch to starting address of appropriate microroutine.
...	...
25	If N=0, then branch to microinstruction 0
26	offset-field-of-IRout, Select Y, Add, Zin
27	Zout, PCin, End

* For a conditional branch to take place, the microinstructions should specify the branch address, external inputs, condition codes, or, bits of the instruction register.

* After loading this branch instruction into IR, a branch microinstruction transfers control to the corresponding microroutine, which is assumed to start at location 25 in the control memory.

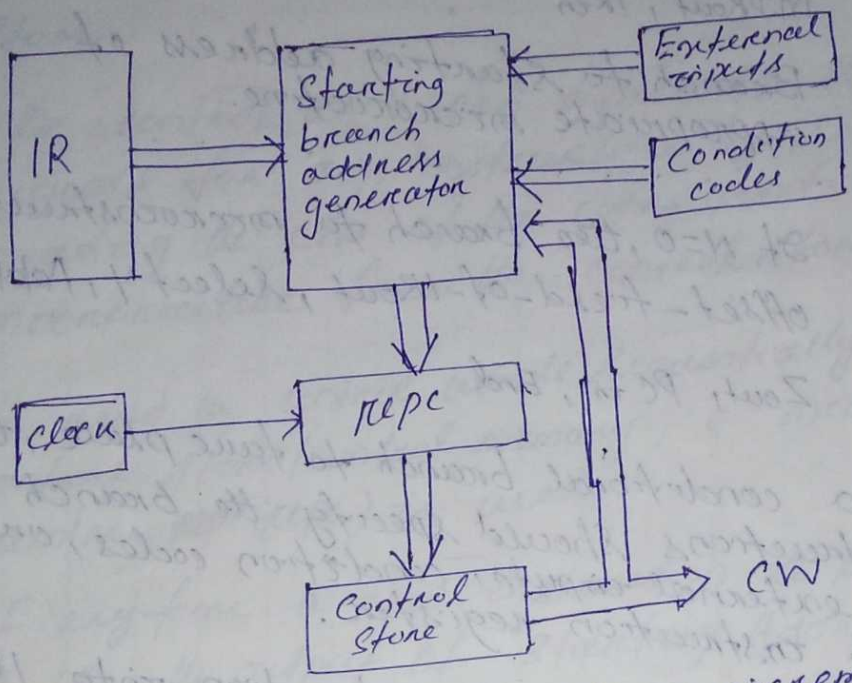
* This address (Location 25) is ~~defer~~ specified by the the starting address generator.

* The microinstruction at Location 25 tests the N bit of the condition codes. If this bit is equal to 0, a branch takes place to location 0 to fetch a new machine instruction.

* Otherwise, the microinstruction at Location 26 is executed to put the branch target address into register Z.

* The microinstruction in Location 27 loads this address into PC.

The modified organization of the control unit to support microprogramming branching.



In this control unit, the μPC is incremented every time a new microinstruction is fetch from the microprogram memory, except in the following situations.

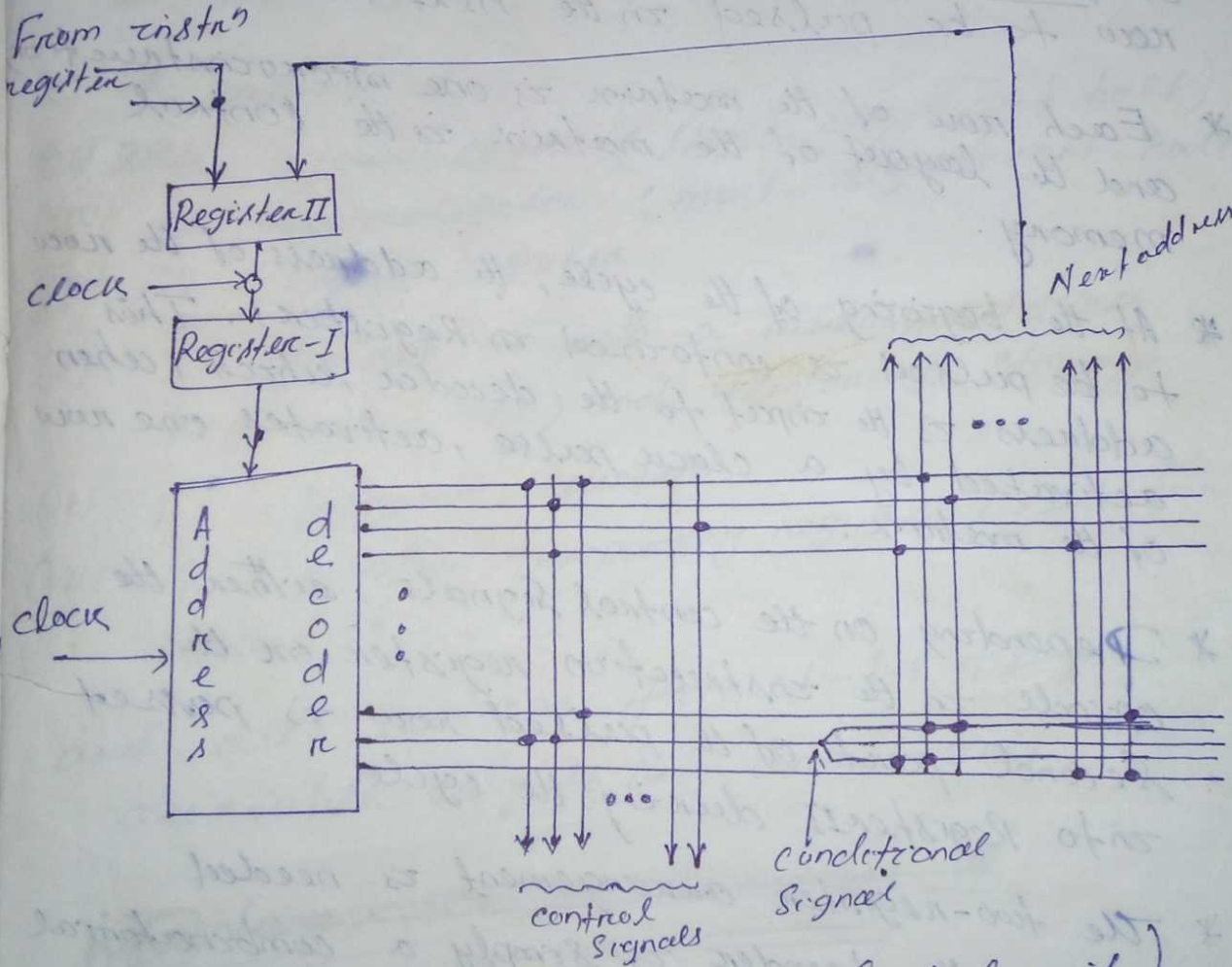
1. When a new instruction is loaded into the IR, the μpc is loaded with the starting address of the microroutine for the instruction.
2. When a branch microinstruction is encountered and branch condition is satisfied, the μpc is loaded with the branch address
3. When an End microinstruction is encountered,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

1(9)
4(10)

Wilkes Control cenit

The term microprogram was first coined by M.V. Wilkes in the early 1950s. Wilkes Proposed an approach to control unit design that was systematic and organized and avoided the complexities of a hardwired implementation.

The first computer, ie, IBM's System/360 ~~was~~ control cenet was based on microprogramming.



(Wilkes's Microprogrammed Control cenit)

* The heart of this system is a matrix partially filled with diodes.

* During a machine cycle, one row of the matrix is activated with a pulse. This generates signals at those points where a diode is present (indicated by a dot in the diagram).

* The 1st part of the row generates the control signals that control the operation of the processor. The second part generates the address of the row to be pulsed in the next machine cycle.

* Each row of the matrix is one microinstruction, and the layout of the matrix is the control memory.

* At the beginning of the cycle, the address of the row to be pulsed is contained in Register I. This address is the input to the decoder, which, when activated by a clock pulse, activates one row of the matrix.

* Depending on the control signals, either the op-code in the instruction register or the second part of the pulsed row is passed into RegisterII during the cycle.

* The two-register arrangement is needed because the decoder is simply a combinational circuit, with only one register, the output would become the input during a cycle, causing an unstable condition.

* In the basic-organization of a control unit, the μpc is incremented by one to get the next address. In Wilkes Scheme, the next address is contained in the microinstruction itself.

* To permit branching, a row must contain two address parts, controlled by a conditional signal.

* The Processor of the hypothetical machine includes the following registers.

(i) A multiplicand
(ii) B accumulator (least-significant half)
(iii) C accumulator (most-significant half)
(iv) D Shift register
(v) E Serves as both a memory address registers (MAR) and temporary storage.
(vi) F Program counter
(vii) G Another temporary register used for counting.

and also contains two 1-bit flags accessible only to the control unit.

~~Advantages and Disadvantages of~~