Chapter 04:CPU Design

Hari K.C.

Department of Software Engineering

Gandaki College of Engineering and Science

Chapter-4

CPU Design

★ CPU - central processing unit.
ALU + Registers + control unit
- perform sequences of micro operations needed to perform fetch, decode & execute cycles of every instructions in CPU instruction set.

★ specifying a CPU

i) The first step in designing CPU is to determine its application. Matching the capabilities of CPU to tasks it will perform.

ii) Then, 2nd step is to design a instruction set architecture (ISA) which can handle the given tasks.

iii) 3rd step is to design State diagram for CPU. Here the different micro operations are shown which are performed during each state & the conditions that cause CPU to go from one state to another. CPU is a complex finite state machine.

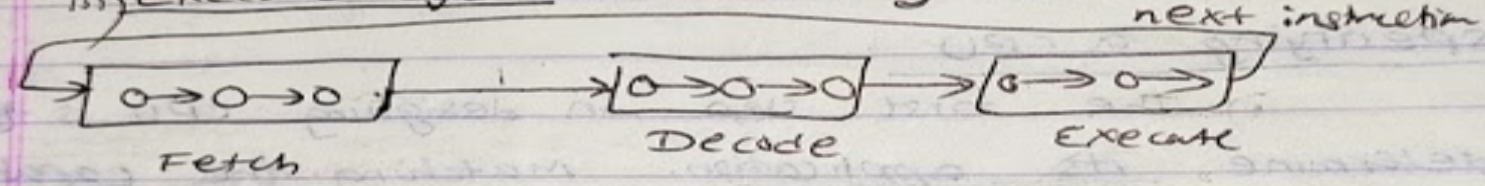
Then, specify the steps the CPU must perform inorder to fetch, decode & execute every instructions in its instruction set.

In general, a CPU performs following sequence of operations.

i) Fetch cycle: fetching an instruction from memory

ii) Decode cycle: Decoding an instruction that is determining which instruction has been fetched.

iii) Execute cycle: Executing the instruction.



* Design & implementation of very simple CPU.

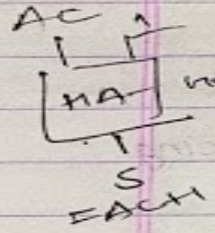
a) Specification for a simple CPU

Consider a CPU which can access 64 bytes of memory. (1 byte = 8 bits). The CPU does this by outputting a 6 bit address on its output ping A[5----0] & reading a 8 bit value from memory on its inputs D[7----0]

The CPU has one programmable register, 8 bit accumulator (AC). It has only four instructions in its instruction set.

A → Address AAAAAAA – 6 bit address

Instruction	Instruction code	operation
ADD	00 AAAAAA	AC ← AC + M[AAAAAA]
AND	01 AAAAAA	AC ← AC ^ M[AAAAAA]
JMP	10 AAAAAA	GOTO AAAAAA
INC	11 XXXXXX	AC ← AC + 1.



The CPU contains the following registers :-

Address register

AR , 6 bit . It supplies an address to memory through Address lines A[5---0]

Program counter

PC , 6bits . It contains the address of next instructions to be executed.

Data register

DR, 8bits . It receives instructions & data from memory through D[7---0].

Instruction register

IR , 2bit . It stores the opcode portion of the instruction code fetched from memory.

Fetching instructions from memory

For this CPU, to fetch instructions.

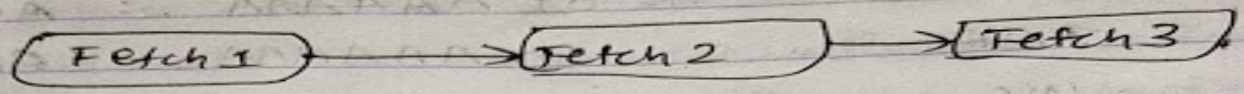


fig:- (fetch cycle)

the following actions are performed:-

a) Send address to memory by placing it on address pins A[5---0].

b) After retrieving the desired instruction, CPU reads data from D[7---0] pins of it

Fetch 1 → first stage of fetch cycle

fetch1 : AR ← PC.

Program counter stores the address of instruction to be fetched.

Fetch 2 : DR ← M

PC ← PC + 1.

CPU read & store data in data register from memory

PC is increment by 1 so that it holds addrs of next instruction to be fetched.

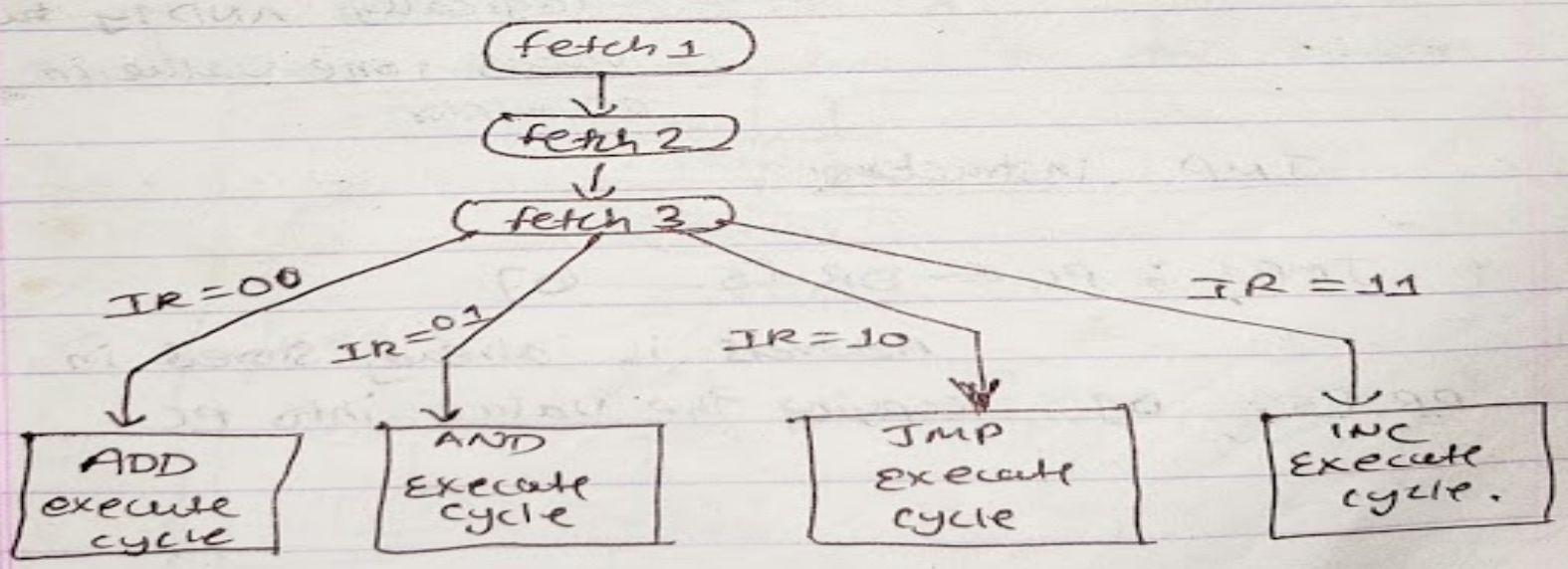
Fetch 3 : IR ← DR[7---6]

AR ← DR[5---0]

In 3rd steps of fetch cycle, the two high order bits of DR is copied to IR. These two bits indicate which instruction is to be executed. Also CPU copies the six low order of DR to AR during fetch routine.

Decoding Instructions:

After CPU has fetched an instruction from memory, it must determine which instruction it has fetched so that it may invoke the correct execute routine. For this CPU, there are four instructions & thus four execute routines. The value in IR (00, 01, 10 or 11) determines which execute routine is involved.



Executing instructions

ADD instructions

ADD 1 : DR ← M — fetch one operand from memory & store in DR.

ADD 2 : AC ← AC + DR — Add this operand to current contents of accumulator & store in accumulator

AND instructions

AND1 : DR ← M — fetching one operand from memory & store in DR.

AND2 : AC ← AC ^ DR — logically AND ify two values, one value in accumulator

JMP instructions

JMP1 : PC ← DR [5---0]

Address is already stored in DR [5---0] . copying the value into PC

INC instructions

INC_1 : AC ← AC+1.

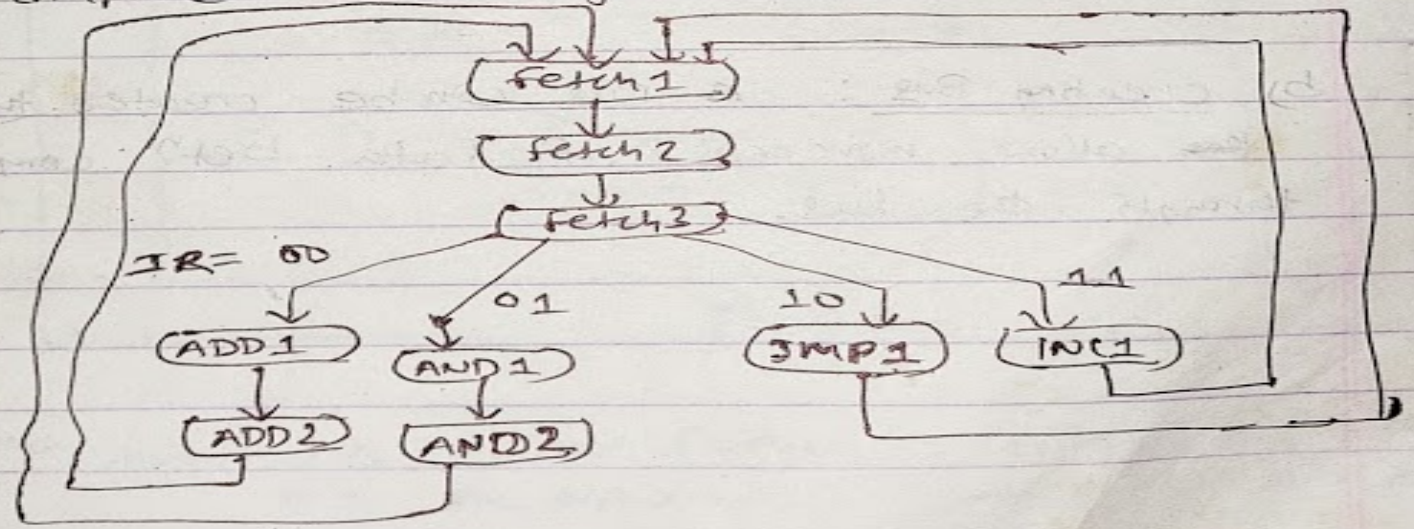
CPU adds 1 to the contents of AC & goes to fetch routine.

Thus concluding :-

Fetch 1 : AR ← PC
Fetch 2 : DR ← M , PC ← PC+1
Fetch 3 : IR ← DR[7---6] , AR ← DR[5---0]
ADD1 : DR ← M
ADD2 : AC ← AC+DR
AND1 : DR ← M
AND2 : AC ← AC ^ DR
JMP1 : PC ← DR[5---0]
INC1 : AC ← AC+1.

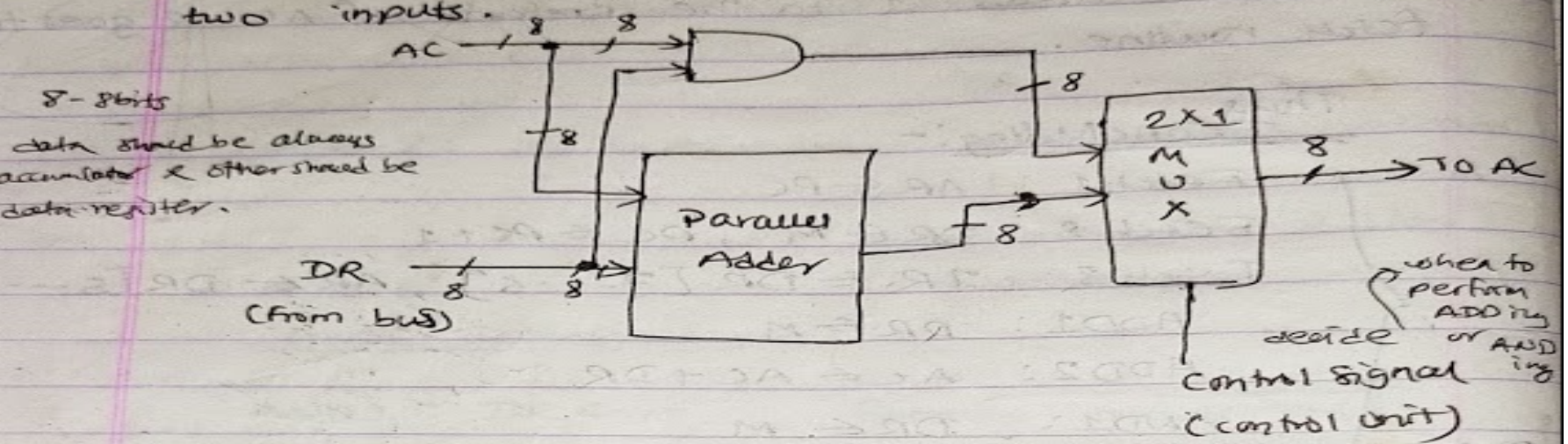
Total 9 States

The complete state diagram for this simple CPU



An ALU for relatively simple CPU.

Design ALU of a CPU that performs only two functions : add its two inputs or logically ANDs its two inputs.



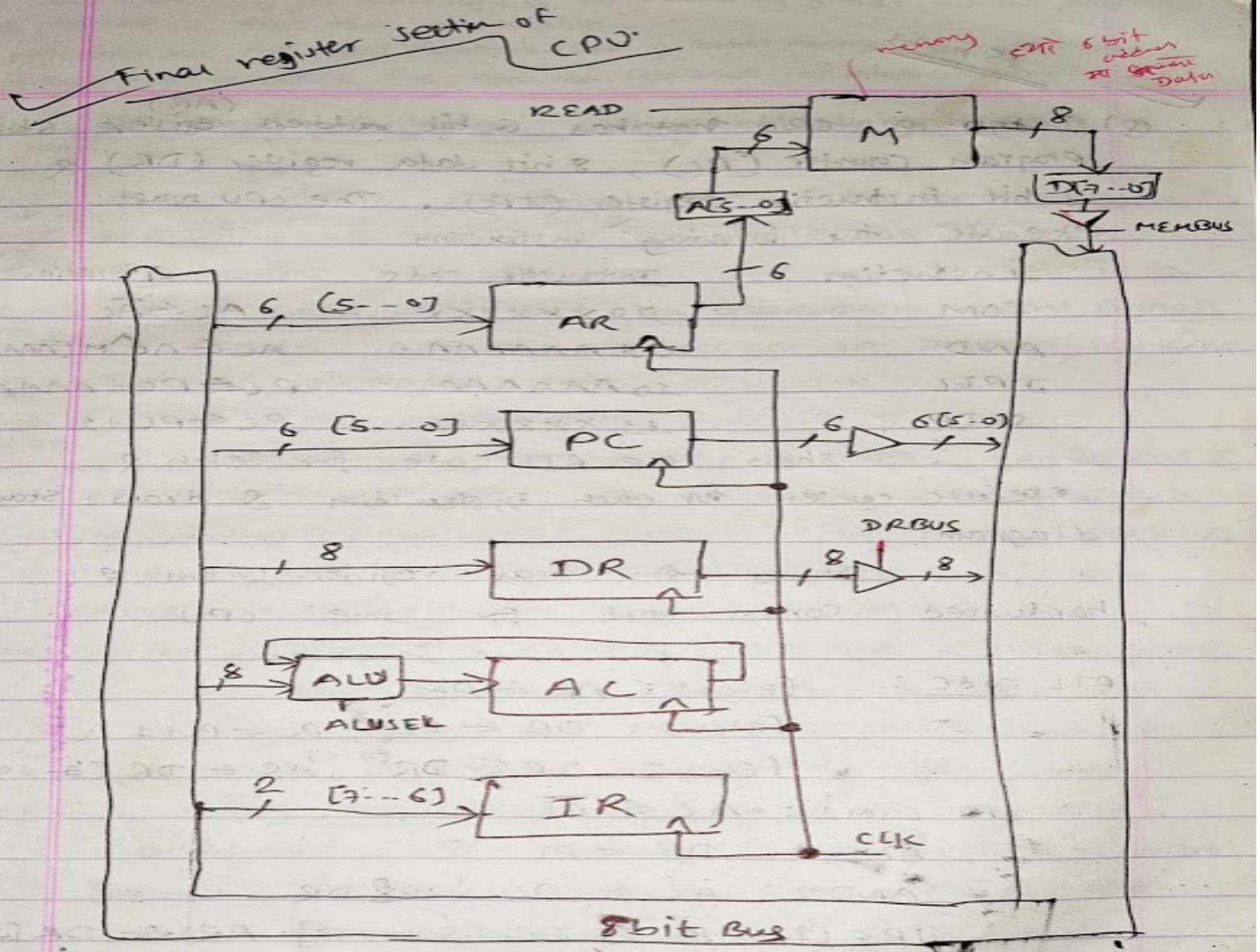
* Data path : Data path can be design in two ways :-

a) Direct path b) creating Bus.

a) Direct path : To transfer data, direct path can be created beth components

b) creating Bus : A Bus can be created that allow movement of data beth components through the bus.

Final register section of CPU.



when ALUSEL = 0, the o/p of ALU is arithmetic sum of its input
= 1 the o/p of ALU is logical AND of its input

15 marks

Q). Design a CPU that has 6 bit address register (AR), 6bit program counter (PC), 8 bit data register (DR) & 2-bit instruction register (IR). The CPU must execute the following instructions

Instruction	Instruction code	Operation
COM	00XXXXXX	$AC \leftarrow \overline{AC}$
AND	01AAAAAA	$AC \leftarrow AC \wedge M[AAAAAA]$
JREL	10AAAAAA	$PC \leftarrow PC + AAAAAA$
SKIP	11XXXXXX	$PC \leftarrow PC + 1$

Show the RTL code for fetch & execute cycles for each instruction & draw state diagram.

Show the final register section & hardwired control unit for same CPU.

RTL code:

fetch1 : $AR \leftarrow PC$

fetch2 : $DR \leftarrow M$, $PC \leftarrow PC+1$

fetch3 : $IR \leftarrow DR[7-6]$, $AR \leftarrow DR[5--0]$

COM1 : $AC \leftarrow \overline{AC}$

AND1 : $DR \leftarrow M$

AND2 : $AC \leftarrow AC \wedge DR$

JREL1: ~~$PC \leftarrow DR[5---0]$~~ $AR \leftarrow DR[5--0]$

JREL2: $PC \leftarrow PC + AR$

SKIP1 : $PC \leftarrow PC+1$

In HLW control unit, the control logic is implemented by gates, flip flops, decoders
It produce a fast mode of operation
In MLP control unit, the control information is stored in control memory. The control memory is programmed to initiate the required sequence of micro operations.

Designing control unit using Hardwired control.

A HLW control unit requires changes in wiring if design has to be modified.

In MLP control unit, require changes or modification can be done by updating the microprograms in control memory.

After designing the CPU for every operation necessary to fetch, decode & execute the entire instruction set. The next step is to design the circuit to generate control signals to cause operation to occur in proper sequence. This is control unit.

Two methods:-

Hardwired control: It uses sequential & combinational logic to generate control signals

Microprogrammed control: It uses a lookup memory to o/p the control signals.

Here, we focus on hardwired control.

The simple control unit has 3 components.

i) Counter: It contain current state.

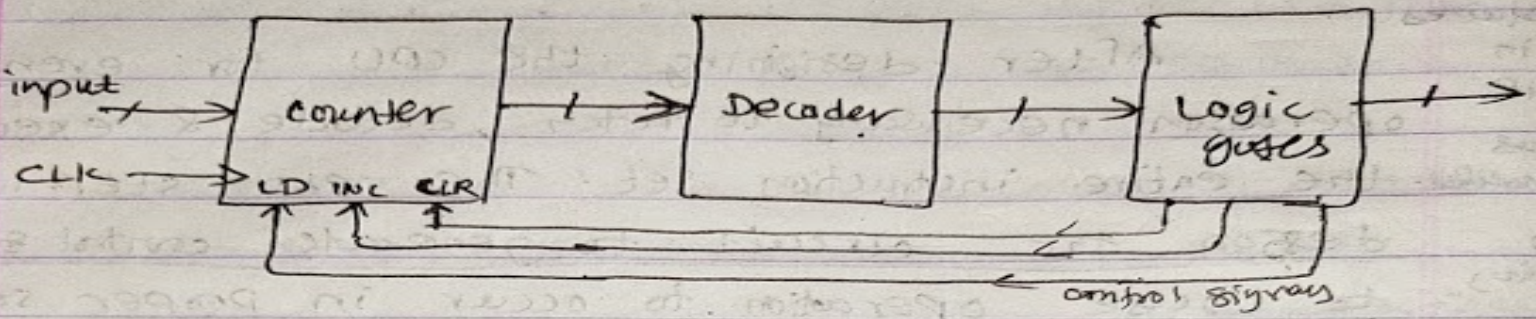
ii) decoder: It takes current state & generates individual signals for each state

iii) combinational logic: It takes the individual state signals & generate the control signals for each component as well as the signals to control the counter.

These signals cause control unit to traverse the states in proper order.

1 00 0
1 01 1
1 10 1
11

General hardwired control unit.



The CPU that we have design has 9 total states: So, a 4 bit counter & 4 to 16 decoder are ~~needeed~~ needed. only 9 of the o/p of decoder is used, other 7 o/p's are not used.

input → In input, states are provided.

CLK → a clk signal

CLR → CLR input is used to reach the state needed.

INC → USE INC input of counter to traverse the states.

LD → USE LD input to reach the proper execute routine.

[mapping Logic]

mapping logic

To make the function simple as possible :-

Consider the possible mapping 10 IR[1---0]

i-e if IR = 00, the input to counter is 1000

for IR = 01, the input is 1001 & so on.

So, the counter values will be.

IR	counter value	State
00	1000 (8)	ADD1
01	1001 (9)	AND1
10	1010 (10)	JMP1
11	1011 (11)	INC1.

Here, the problem arises. since state ADD1 has a counter value of 8 & AND1, a counter value of 9 : There is no counter value that has to be assign to ADD2. (not best soln)

Now, if we use mapping function 1IR[1---0]0, which results in counter values of

00 → 1000 → 8 ADD1

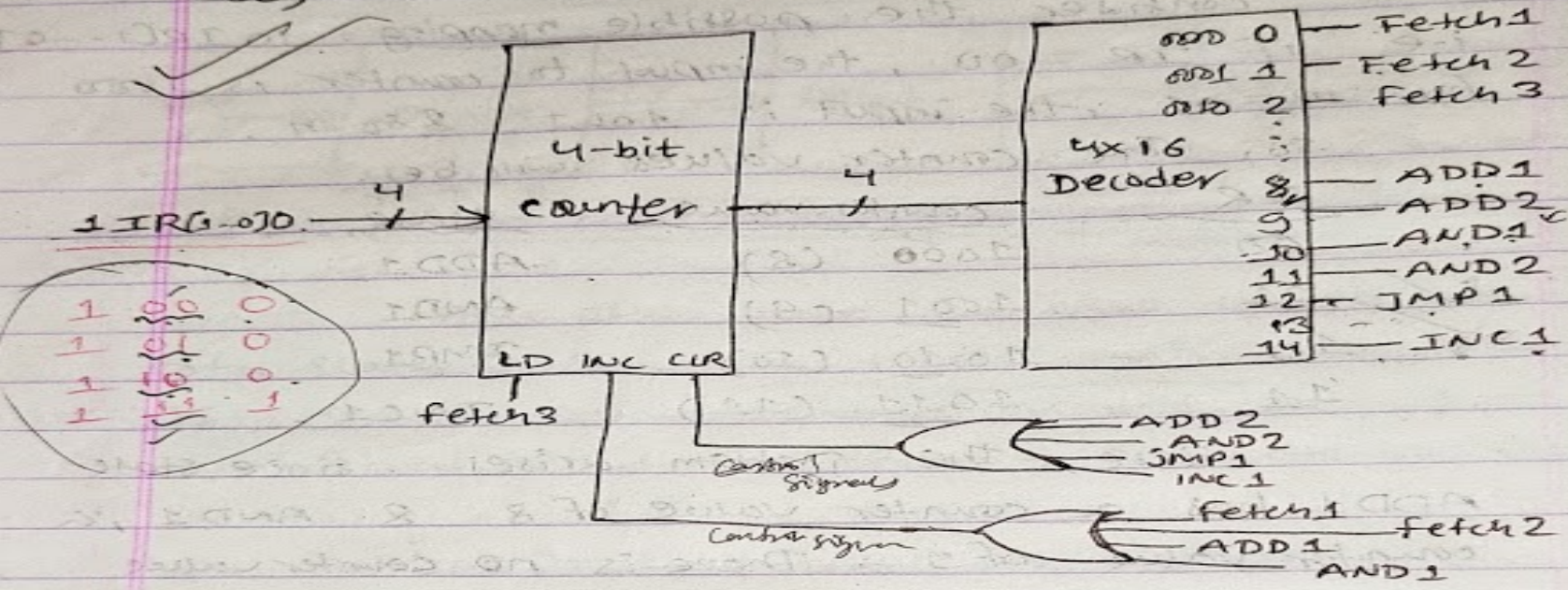
01 → 1010 → 10 AND1

10 → 1100 → 12 JMP1

11 → 1110 → 14 INC1.

Now, we can assign counter value 9 to ADD2 & 11 to AND2

So, a complete hardwired control unit is.



For counter, INC, CLR & LD signals are needed.
INC is asserted when control unit is traversing sequential states, during fetch1, fetch2, JMP1 & INC1. (next state ni other)

LD is asserted at end of fetch cycle during state fetch3

CLR is asserted at end of each execute cycle to return to fetch cycle. This happens during ADD 2, AND 2, JMP1 & INC1.

The End