



CHAPTER-4

# CPU Design

- =>In order to illustrate CPU design process, consider small and somewhat impractical CPU.
- =>This CPU will have only one programmable accessible register called accumulator(AC).
- =>It has only 4 instruction in its instruction set as shown in table:

Instruction set for the Very Simple CPU

Instruction	Instruction Code	Operation
ADD	00AAAAAA	$AC \leftarrow AC + M[AAAAAA]$
AND	01AAAAAA	$AC \leftarrow AC \wedge M[AAAAAA]$
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	$AC \leftarrow AC + 1$

## Contd..

In addition to AC, this CPU needs several additional registers to perform the internal operations necessary to fetch, decode, and execute instructions. The registers in this CPU are fairly standard and are found in many CPUs; their sizes vary depending on the CPU in which they are used. This CPU contains the following registers:

- A 6-bit **address register**, *AR*, which supplies an address to memory via A[5..0]
- A 6-bit **program counter**, *PC*, which contains the address of the next instruction to be executed
- An 8-bit **data register**, *DR*, which receives instructions and data from memory via D[7..0]
- A 2-bit **instruction register**, *IR*, which stores the opcode portion of the instruction **code** fetched from memory

# Fetching instruction from memory

- The address of instruction to be fetched is stored in the PC. Since, address register(AR) supplies an address to memory through A[5.....0] pins, firstly copy the content of PC to AR.

Fetch 1:  $AR \leftarrow PC$

- The CPU must read the instructions from the memory and store in DR.

Fetch 2:  $DR \leftarrow M, PC \leftarrow PC + 1$

- Finally, two higher order bits of DR are copied into IR; these two bit indicate which instruction is to be executed, and the CPU copies the lower order 6-bit of DR into AR.

Fetch 3:  $IR \leftarrow DR[7,6], AR \leftarrow DR[5....0]$



# Decoding instructions

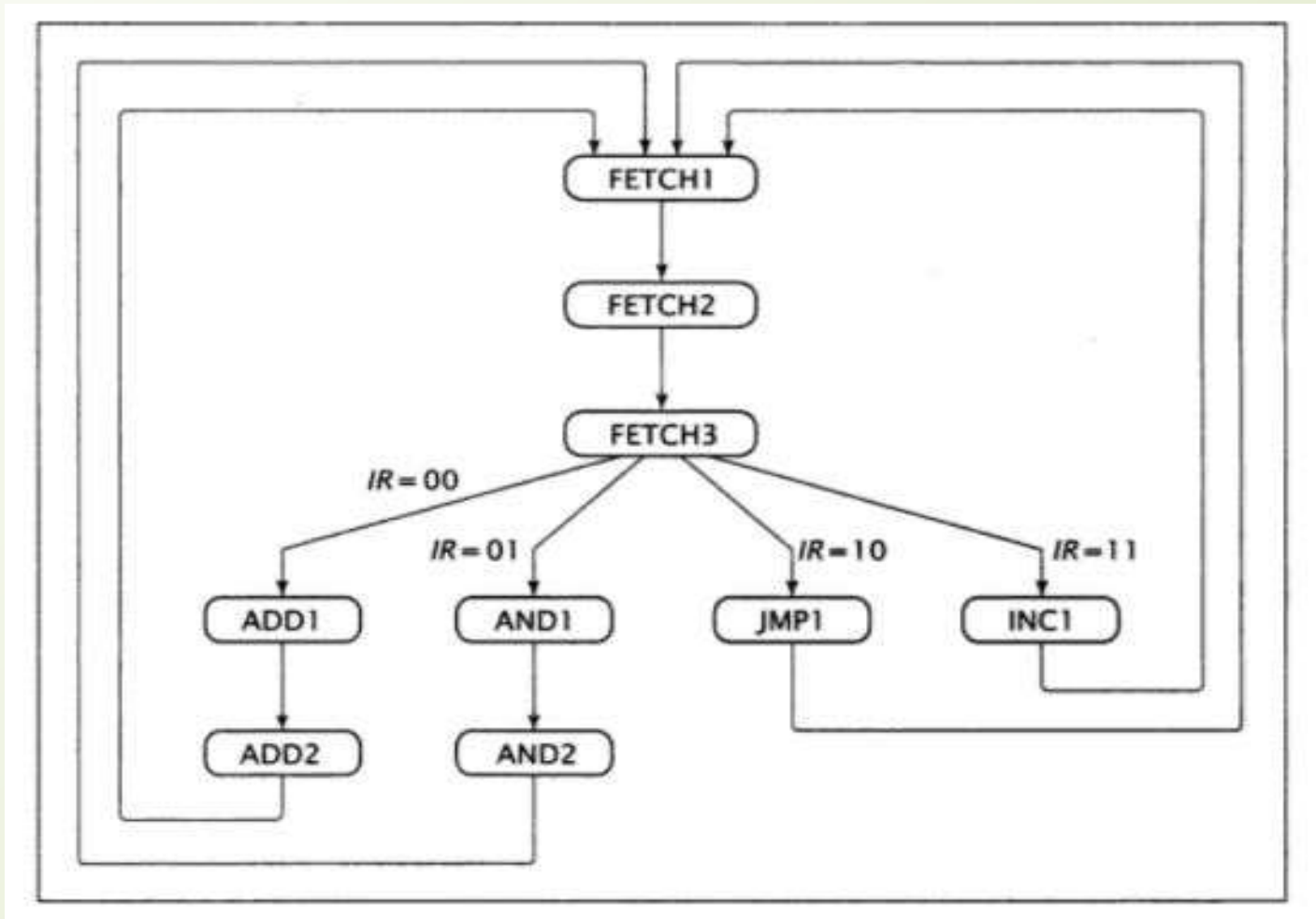


- ❑ The CPU must determine which instruction it has fetched so as to invoke the current routine.
- ❑ The value of IR i.e. 00, 01, 10 and 11 determine which routine is to be invoked.

# Executing instructions:


- ADD instruction:  
=> First it must fetch one operand from the memory.  
=> Then, it must add this operand to the current content of accumulator and store the result back into the accumulator.  
ADD1: DR <- M  
ADD2: AC <- AC + DR
- AND instruction:  
AND1: DR <- M  
AND2: AC <- AC ^ DR
- JMP instruction:  
=> The address to which the CPU must jump is copied into the PC.  
=> Since, the result is already stored in DR[5...0], we simply copy this value to PC.  
JMP1: PC <- DR[5...0]
- INC instruction:  
INC1: AC <- AC + 1

# Complete state diagram for very simple CPU





# Establishing required data path

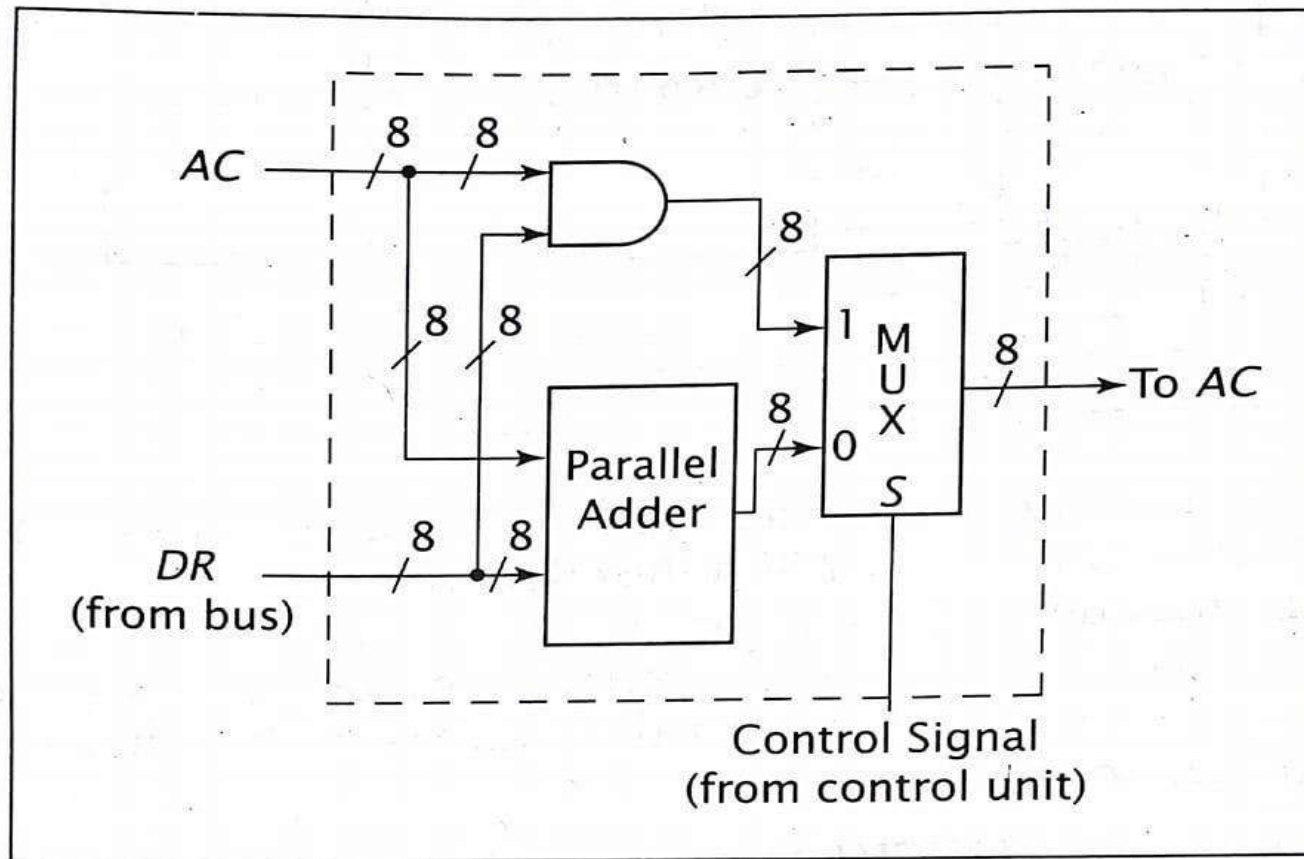


Fetch 1:  $AR \leftarrow PC$   
Fetch 2:  $DR \leftarrow M, PC \leftarrow PC + 1$   
Fetch 3:  $IR \leftarrow DR[7,6], AR \leftarrow DR[5\dots 0]$   
ADD1:  $DR \leftarrow M$   
ADD2:  $AC \leftarrow AC + DR$   
AND1:  $DR \leftarrow M$   
AND2:  $AC \leftarrow AC \wedge DR$   
JMP1:  $PC \leftarrow DR[5\dots 0]$   
INC1:  $AC \leftarrow AC + 1$



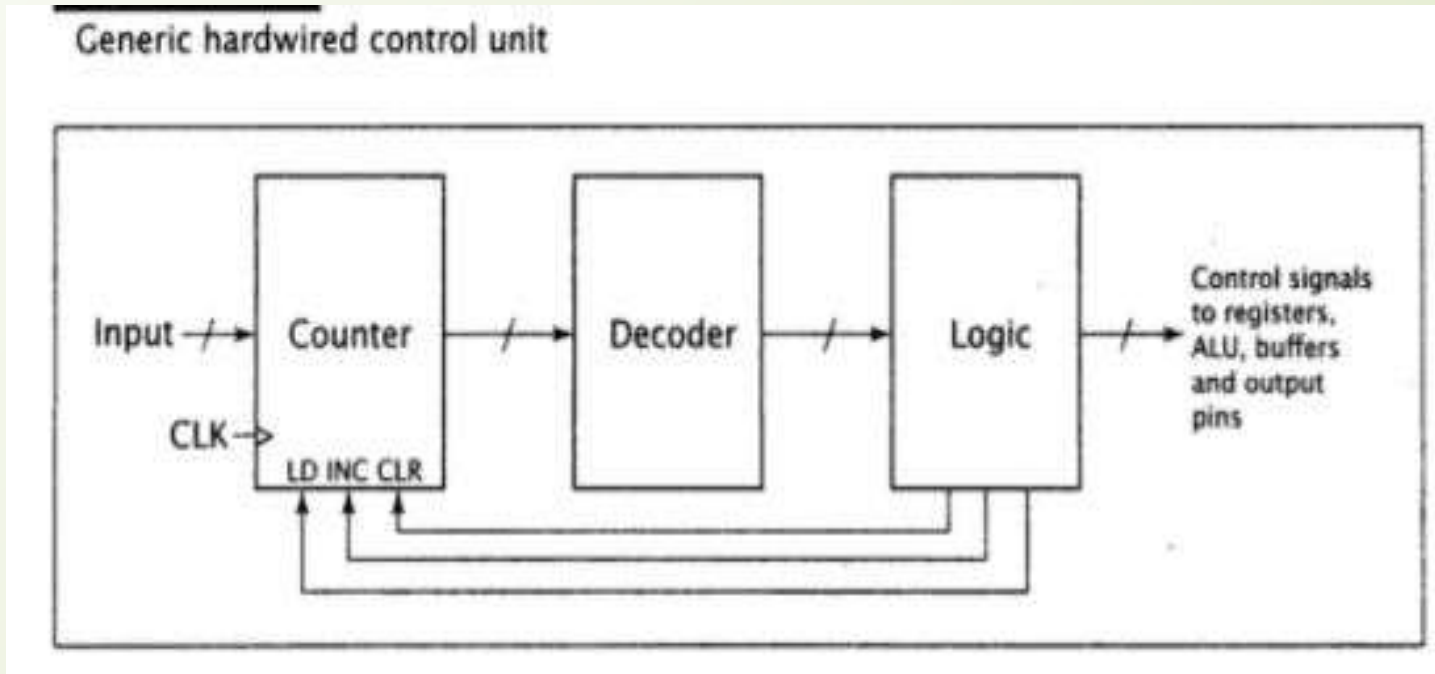
# Design of very simple ALU

Figure 8.1  
A Very Simple ALU



# Designing Hardwired Control Unit:

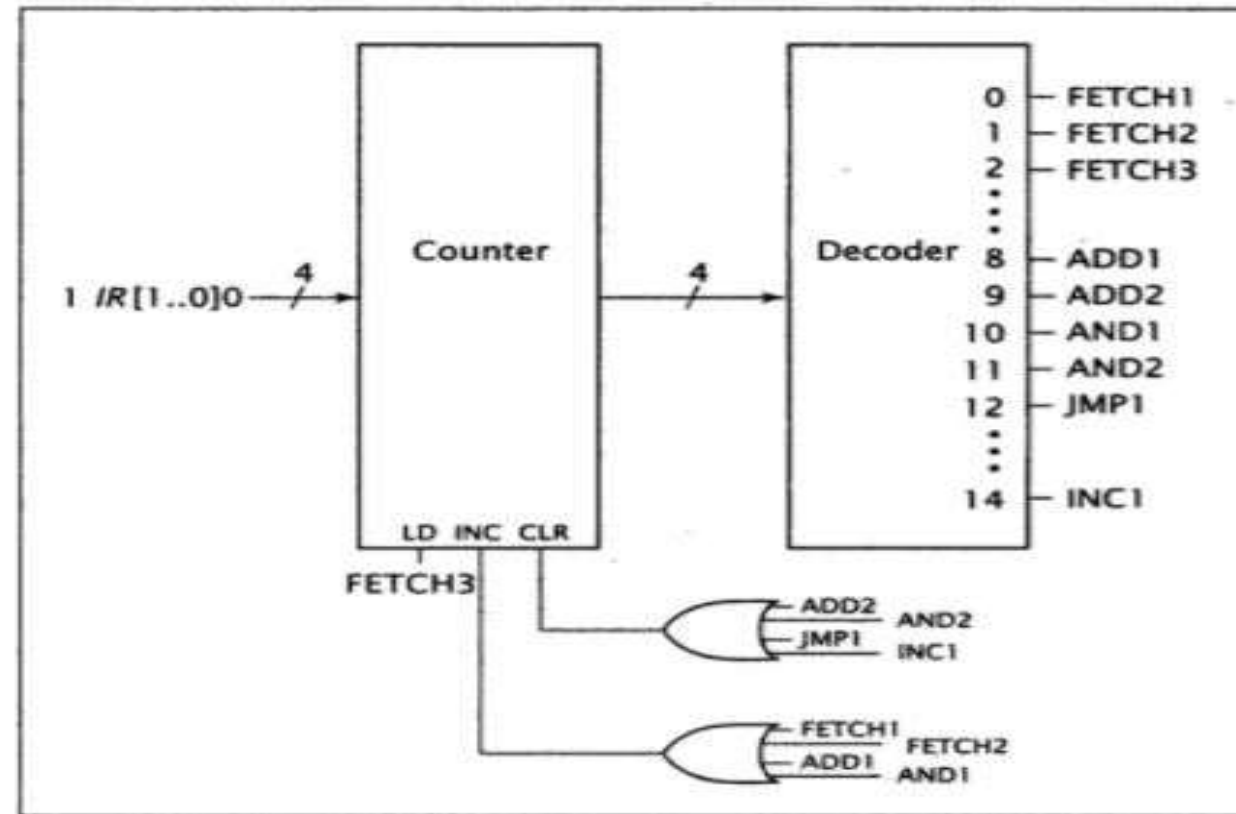
(This very simple CPU requires only a ordinary control unit which has three components: counter, decoder and some combinational logic)



# Contd..

(For this CPU, there are total number of 9 states and therefore a 4-bit counter and 4-to-16-bit decoder is needed)

Hardwired control unit for the Very Simple CPU



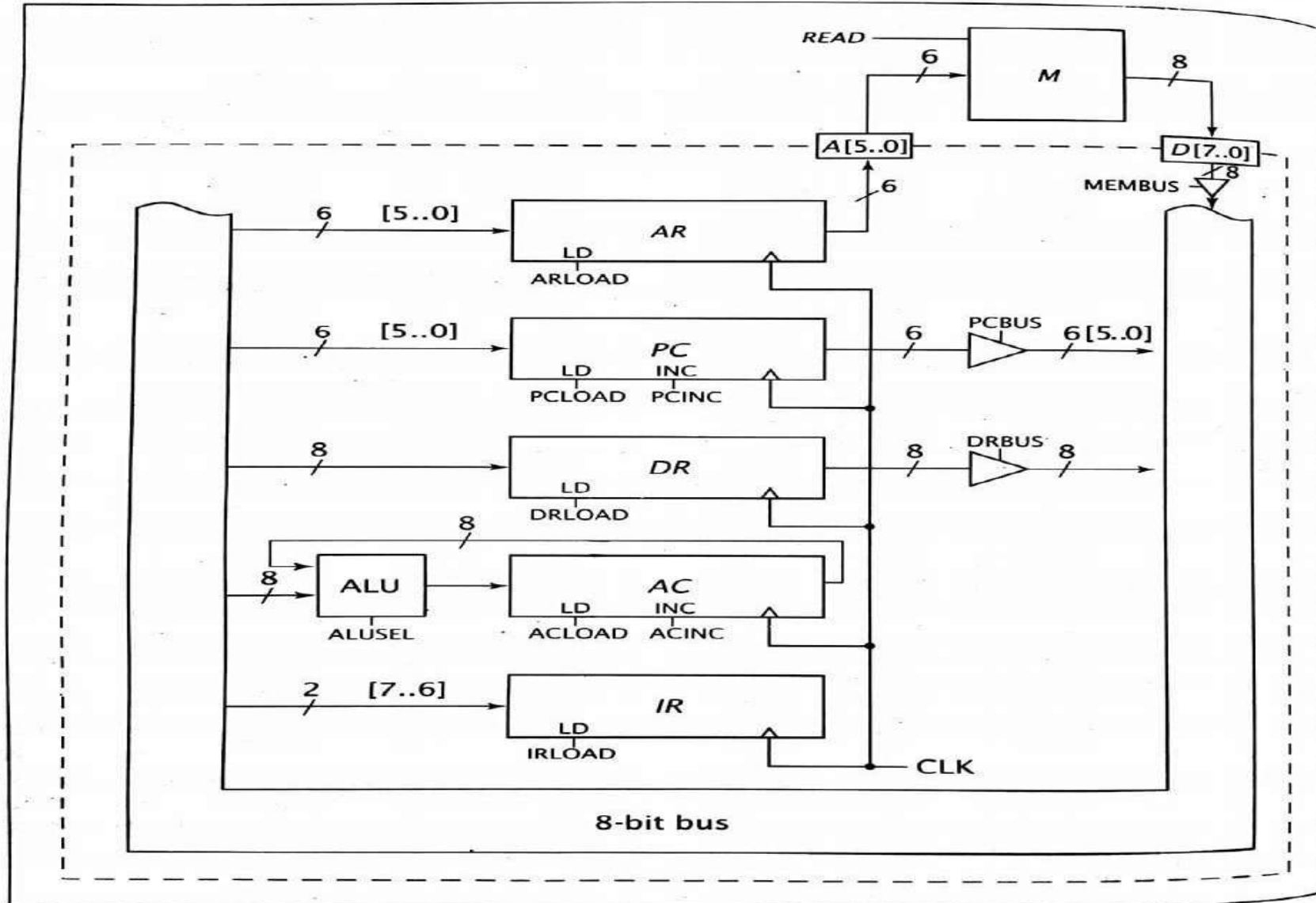


## Contd..

- While working on the actual transfer, we note that AR only supplies its data to memory, but not to other components. Therefore, it is not necessary to connect its output to other internal bus.
- IR does not supply data to any other components through the internal bus. So, its output connection can be removed.
- AC actually does not supply data to any components. Therefore, its connection to the internal bus can be removed.
- AC must be able to load the sum of AC and DR as well as logical AND of AC and DR. Therefore, the CPU needs to include ALU that can generate these results.

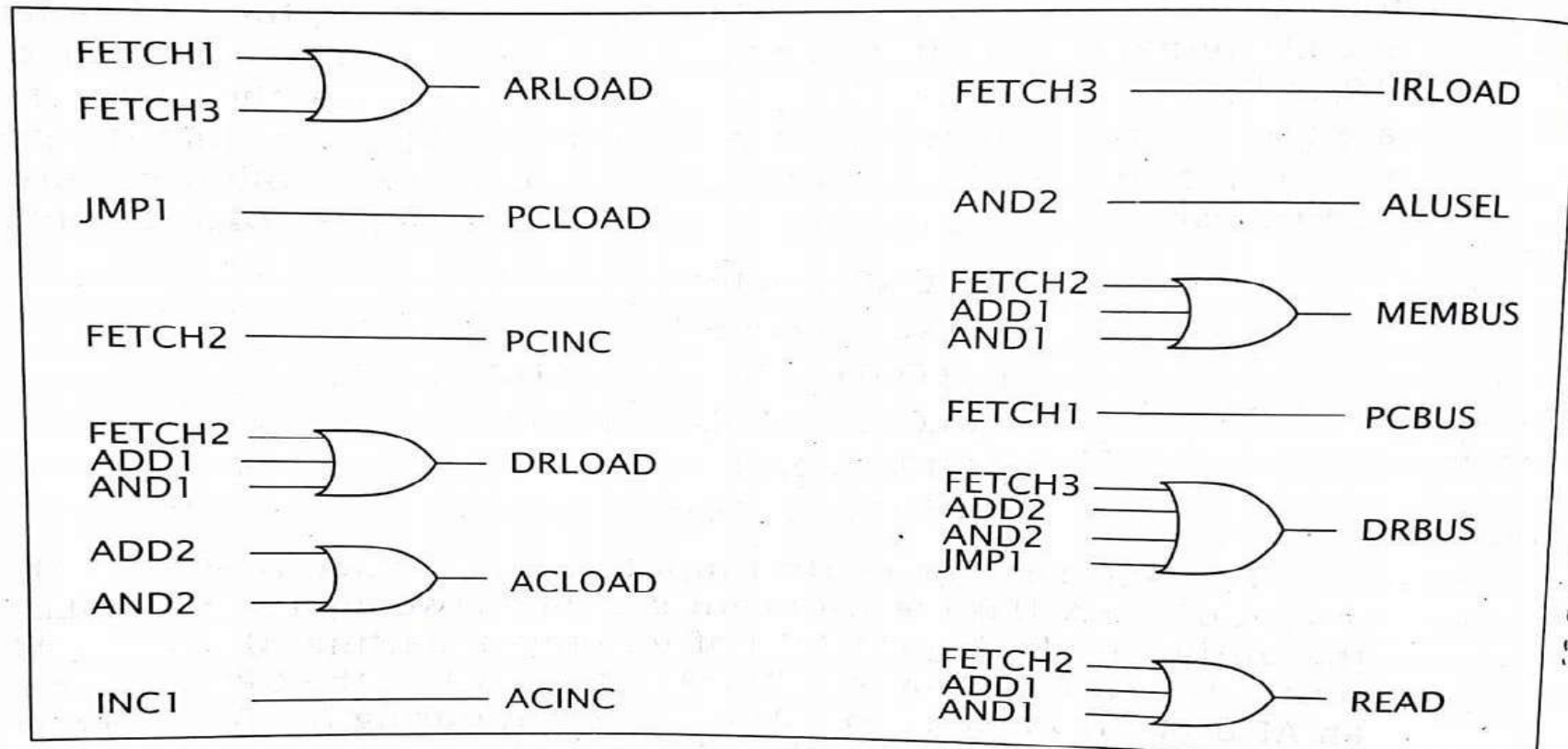
# Contd...

Final register section for the Very Simple CPU



# Control signal generation for very simple CPU

Control signal generation for the Very Simple CPU

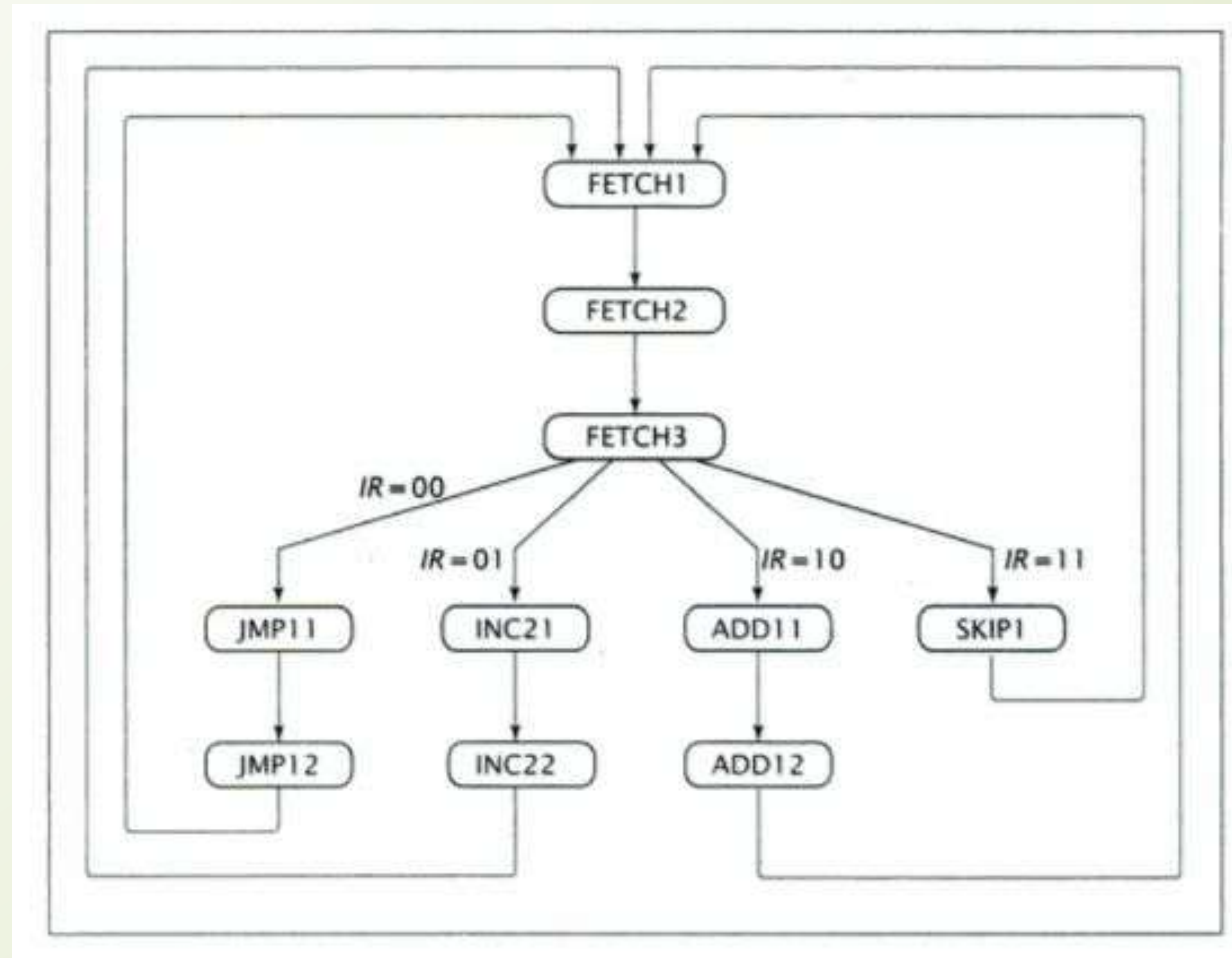




# Example:

Instruction	Instruction Code	Operation
JMP1	00AAAAAA	$PC \leftarrow AAAAAA + 1$
INC2	01XXXXXX	$AC \leftarrow AC + 2$
ADD1	10AAAAAA	$AC \leftarrow AC + M[AAAAAA] + 1$
SKIP	11XXXXXX	$PC \leftarrow PC + 1$

# Contd...







# Contd...

Fetch 1:  $AR \leftarrow PC$

Fetch 2:  $DR \leftarrow M, PC \leftarrow PC + 1$

Fetch 3:  $IR \leftarrow DR[7,6], AR \leftarrow DR[5....0]$

JMP11:  $PC \leftarrow AR$

JMP12:  $PC \leftarrow PC + 1$

INC21:  $AC \leftarrow AC + 1$

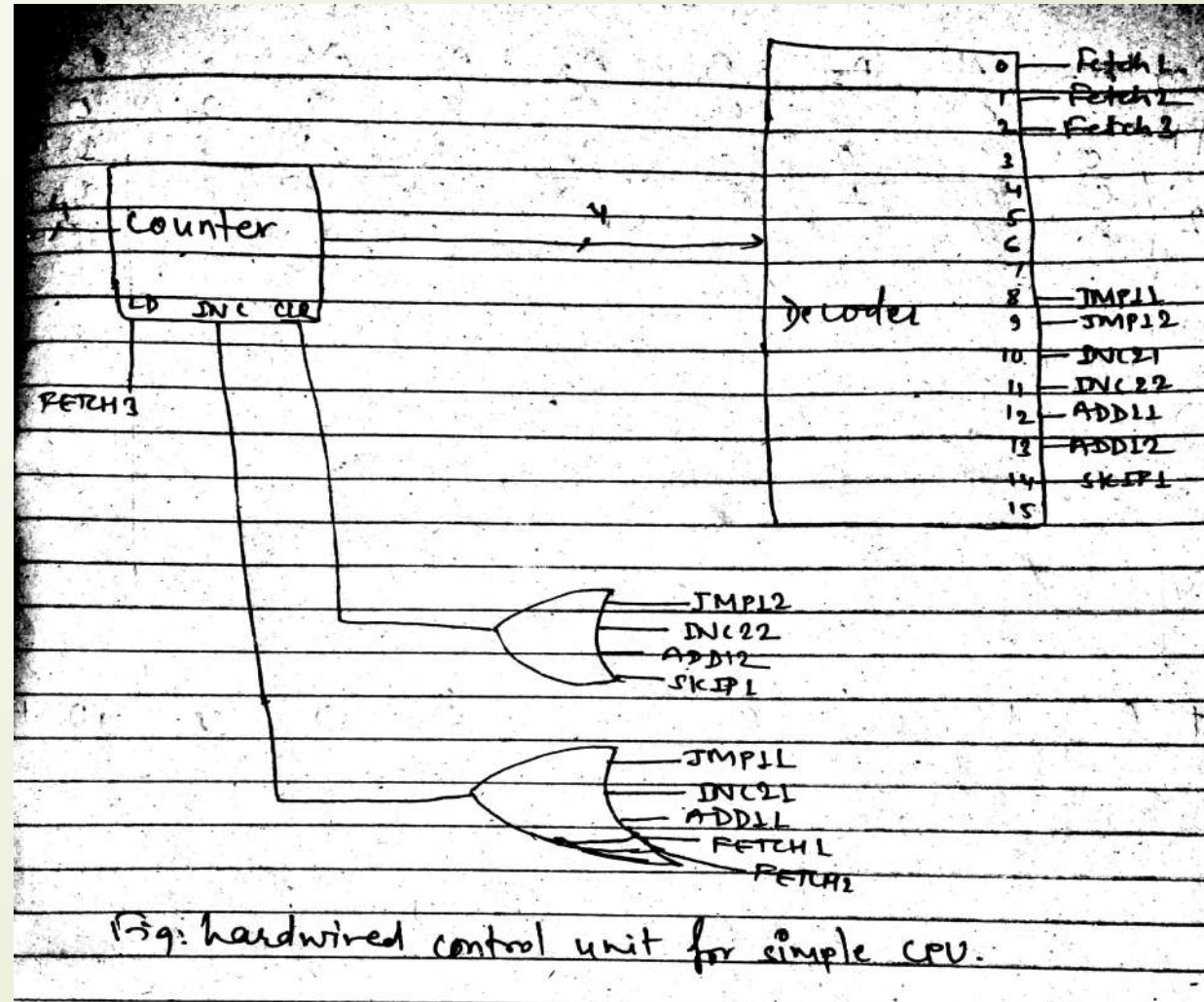
INC22:  $AC \leftarrow AC + 1$

ADD11:  $DR \leftarrow M, AC \leftarrow AC + 1$

ADD12:  $AC \leftarrow AC + DR$

SKIP1:  $PC \leftarrow PC + 1$

Contd..



## Contd...

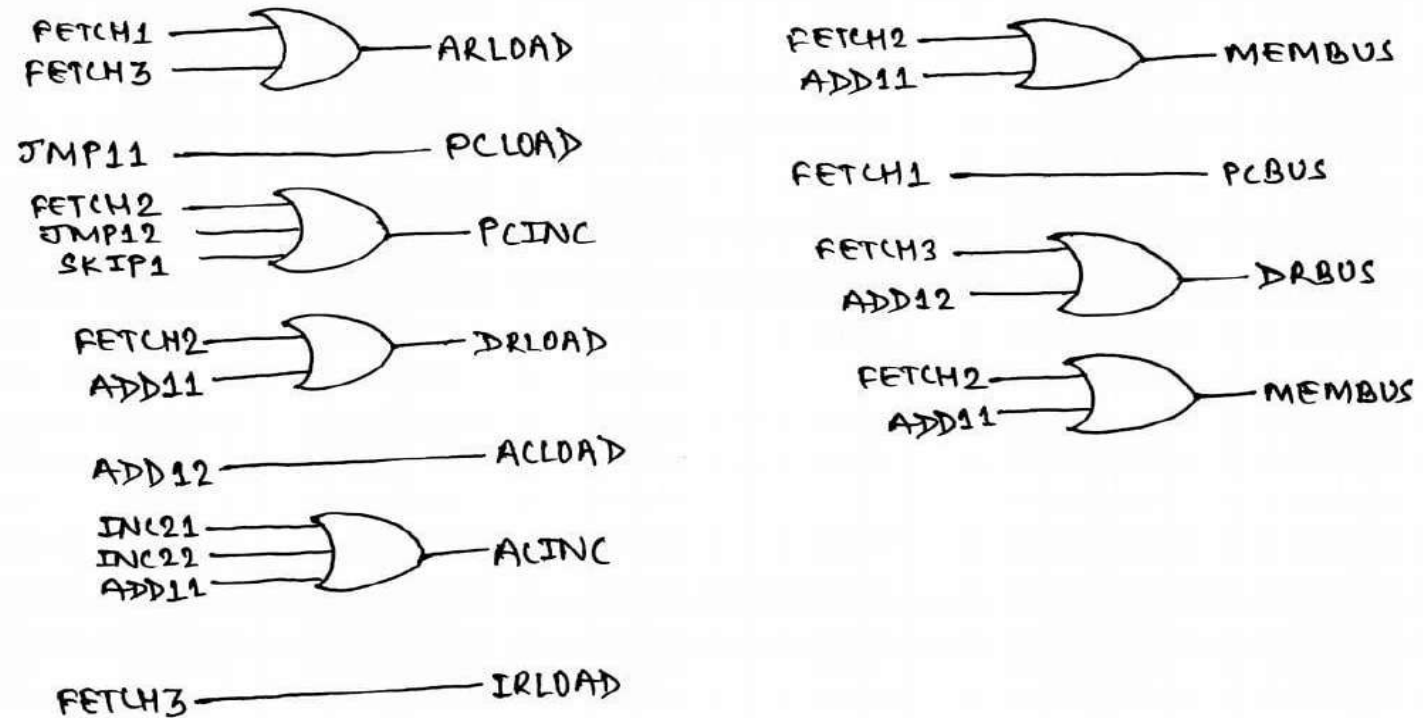


Fig: control signal generation for the very simple CPU.

# Example

Design a CPU that meets the following specifications.

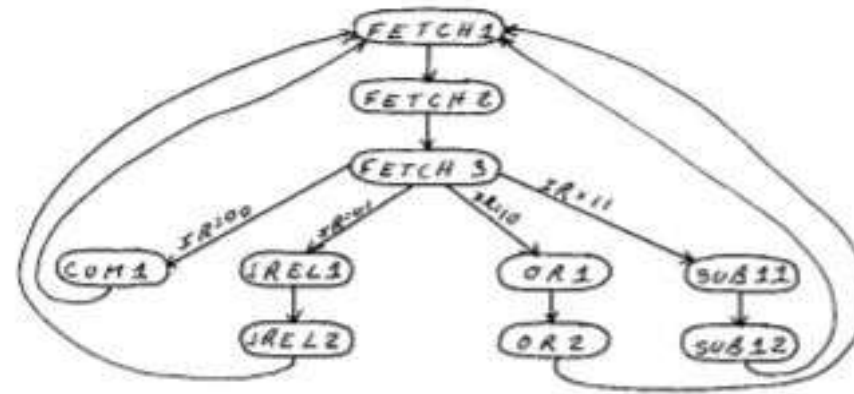
- It can access 64 words of memory, each word being 8 bits wide. The CPU does this by outputting a 6-bit address on its output pins A[5..0] and reading in the 8-bit value from memory on its inputs D[7..0].
- The CPU contains a 6-bit address register (AR) and program counter (PC); an 8-bit accumulator (AC) and data register (DR); and a 2-bit instruction register (IR).
- The CPU must realize the following instruction set.

Instruction	Instruction Code	Operation
COM	00XXXXXX	$AC \leftarrow AC'$
JREL	01AAAAAA	$PC \leftarrow PC + 00AAAAAA$
OR	10AAAAAA	$AC \leftarrow AC \vee M[00AAAAAA]$
SUB1	11AAAAAA	$AC \leftarrow AC - M[00AAAAAA] - 1$

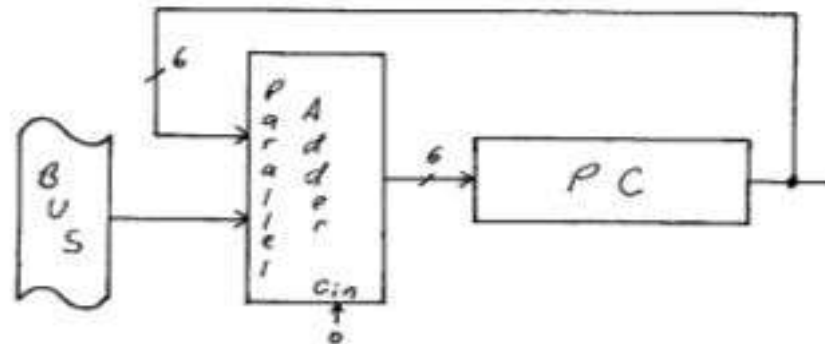
# Contd...

State diagram and 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:  $AC \leftarrow AC'$   
JREL1:  $DR \leftarrow M$   
JREL2:  $PC \leftarrow PC + DR[5..0]$   
OR1:  $DR \leftarrow M$   
OR2:  $AC \leftarrow AC \vee DR$   
SUB11:  $DR \leftarrow M$   
SUB12:  $AC \leftarrow AC + DR'$



The register section is the same as Figure 6.6, except for the data input to PC, shown below.





# Contd..

## Control signals:

$ARLOAD = FETCH1 \vee FETCH3$

$PCLOAD = JREL2$

$PCINC = FETCH2$

$PCBUS = FETCH1$

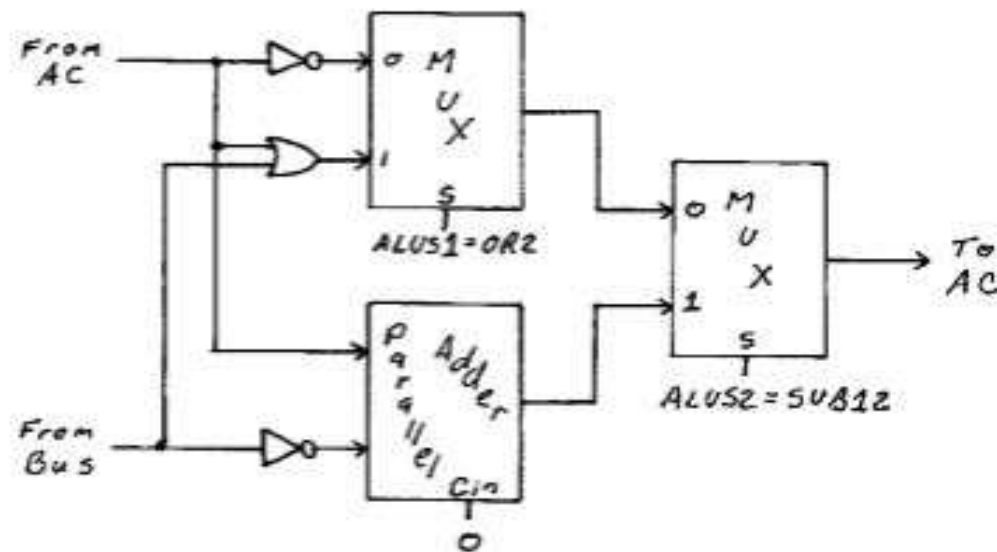
$DRLOAD = MEMBUS = READ = FETCH2 \vee JREL1 \vee OR1 \vee SUB11$

$DRBUS = FETCH3 \vee JREL2 \vee OR2 \vee SUB12$

$ACLOAD = COM1 \vee OR2 \vee SUB12$

$IRLOAD = FETCH3$

## ALU:



# Contd..

Control unit:

