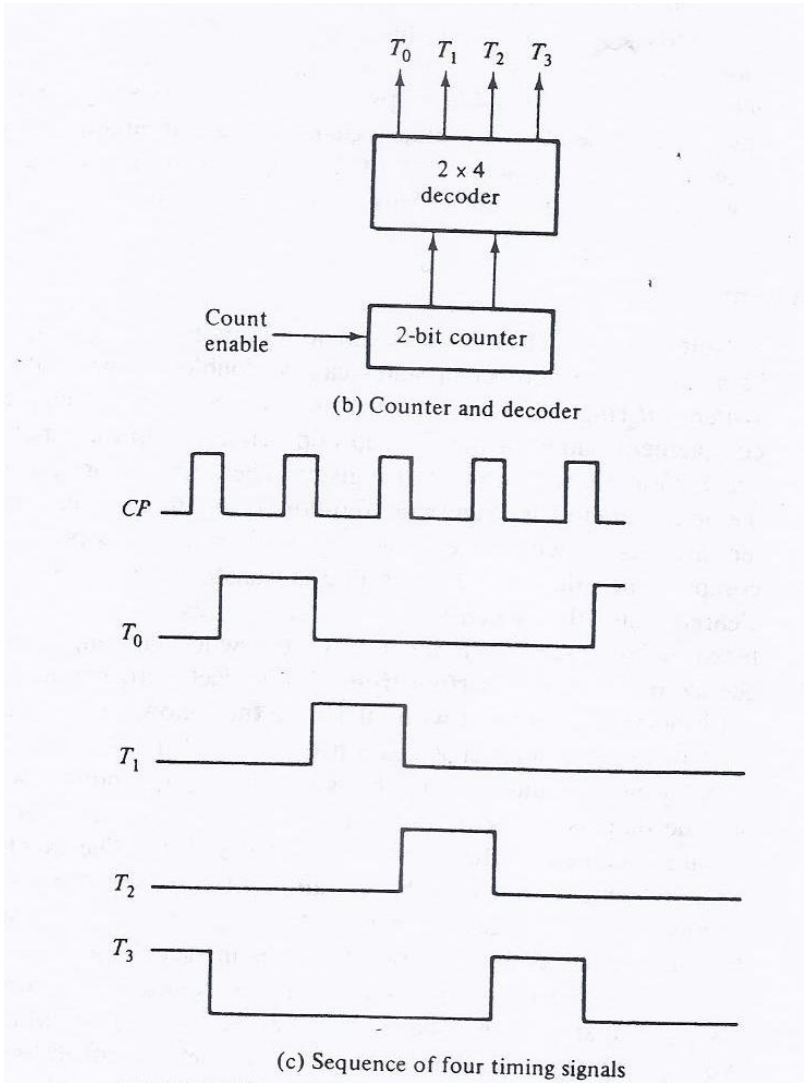
Hardwired Implementation (1)

- Clock
  - Repetitive sequence of pulses
  - Useful for measuring duration of micro-ops
  - Must be long enough to allow signal propagation
  - Need a counter with different control signals for t1, t2 etc.

Generation of Timing Signal



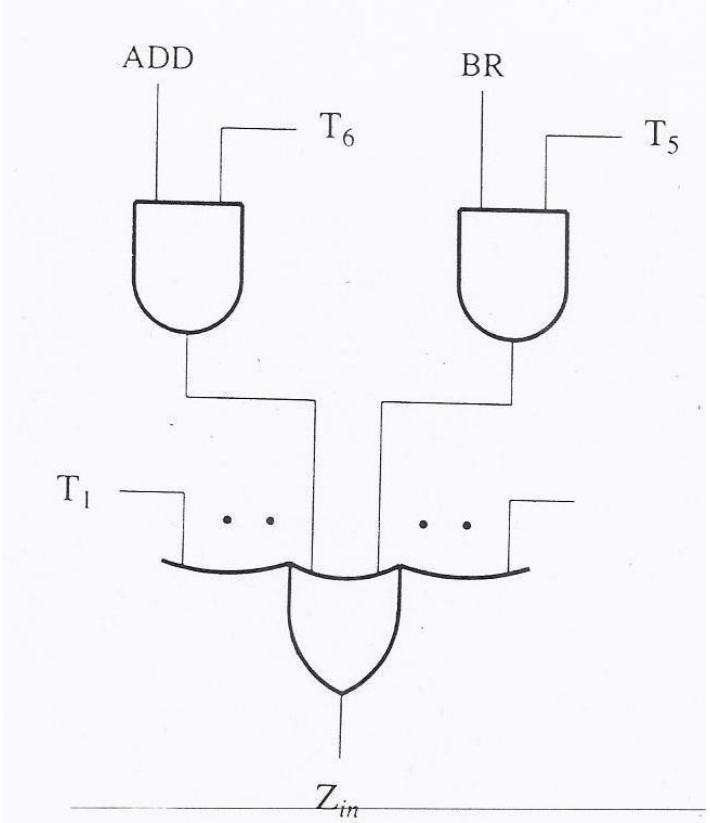
Instruction Fetch and Execute

- ADD (R3), R1
  - Add the content of Register R1 to the content of memory location whose memory address is in register R3 and store the result in R1

Step	Action
1	PC_{out}, MAR_{in}, Read, Clear Y, Set carry-in to ALU, Add, Z_{in}
2	Z_{out}, PC_{in}, WMFC
3	MDR_{out}, IR_{in}
4	$R3_{out}$, MAR_{in}, Read
5	$R1_{out}$, Y_{in}, WMFC
6	MDR_{out}, Add, Z_{in}
7	Z_{out}, $R1_{in}$, End

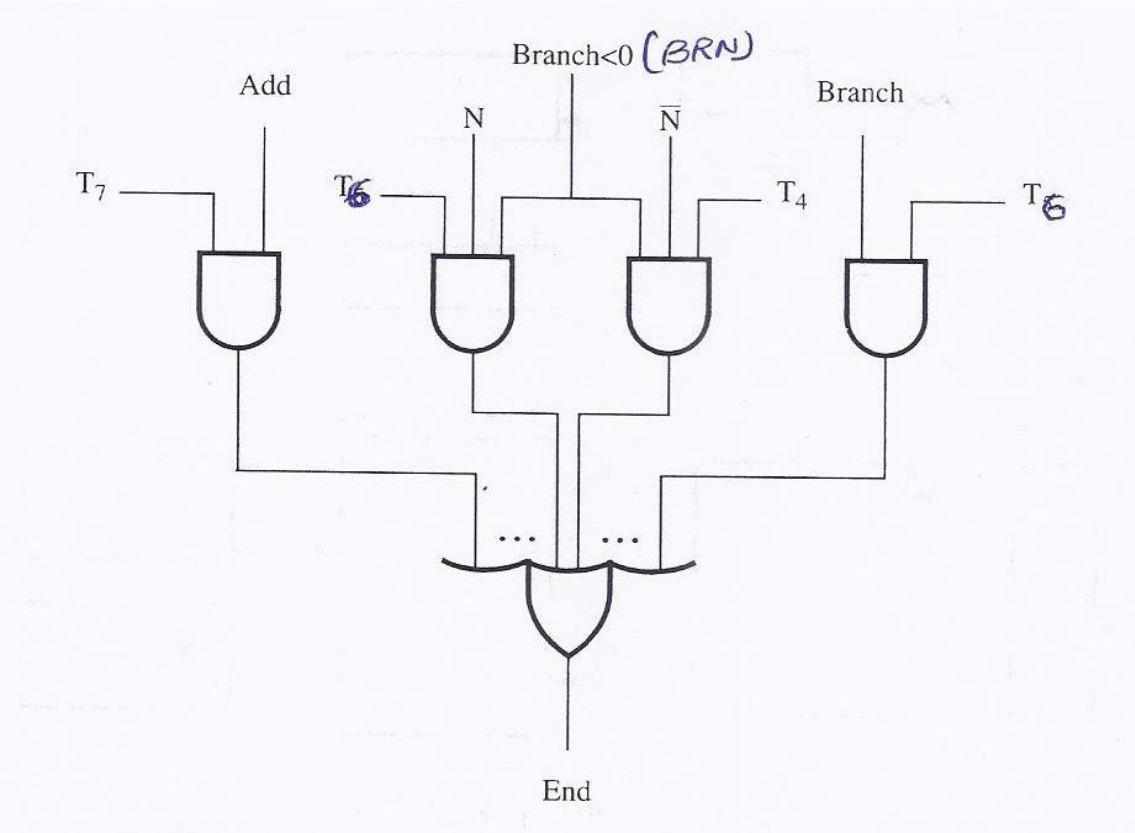
Control Signal

- Generation of signal Z_{in}
  - $Z_{in} = T_1 + T_6.ADD + T_5.BR + \ldots\ldots\ldots$



Control Signals

- Generation of signal: End
  - End = T_7.ADD + T_6.BR + (T_6.N + T_4.~N).BRN + …….



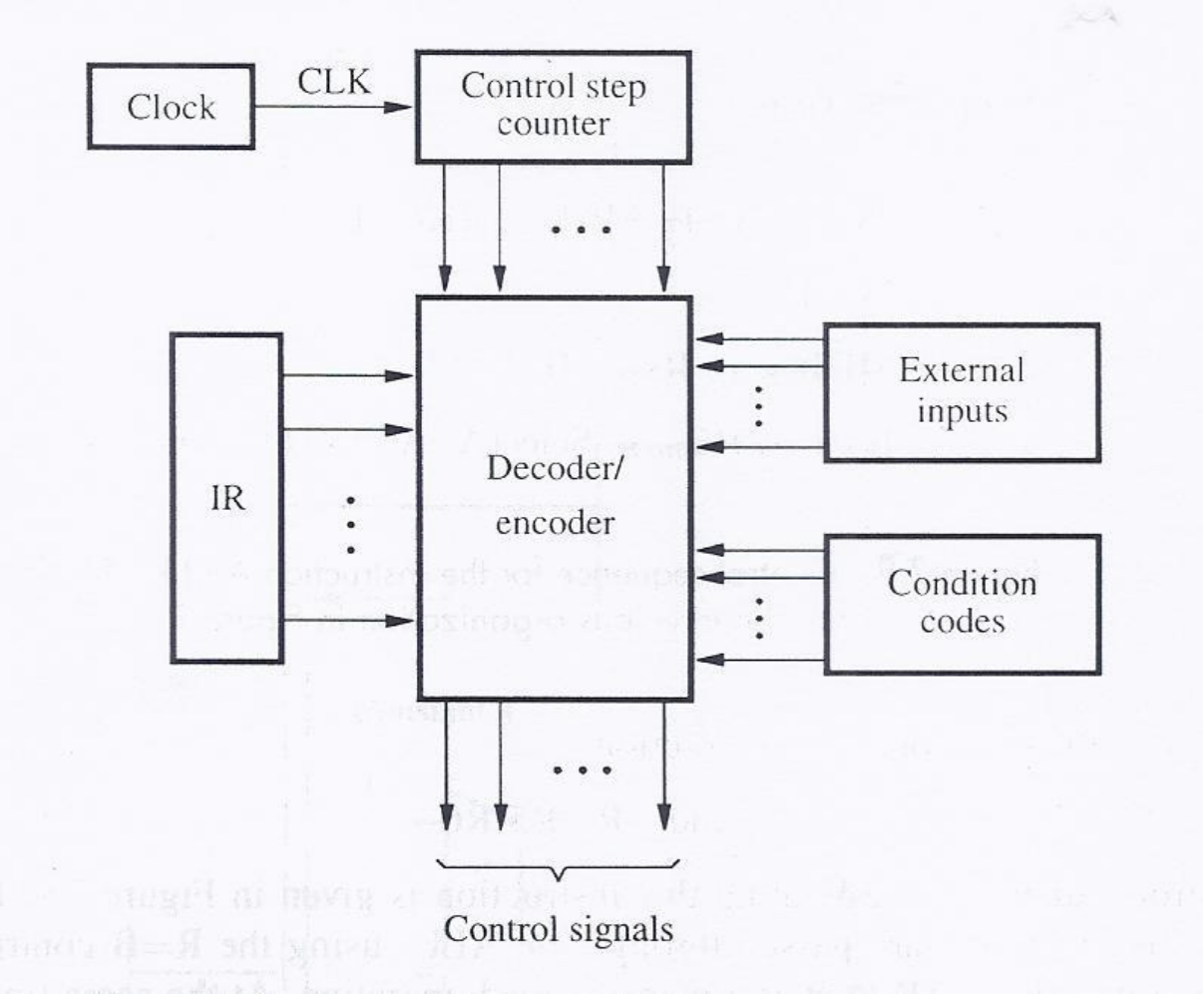
Control Signals

- Generation of signal: WMFC
  - $WMFC = T_2 + T_5.ADD + \ldots\ldots$
- Generation of signal: PC_{out}
  - $PC_{out} = T1 + T_4.BR + T_4.BRN + \ldots\ldots$
- Similarly, we need all the control signals
  - MAR_{in}, MDR_{in}, MDR_{out}, …..

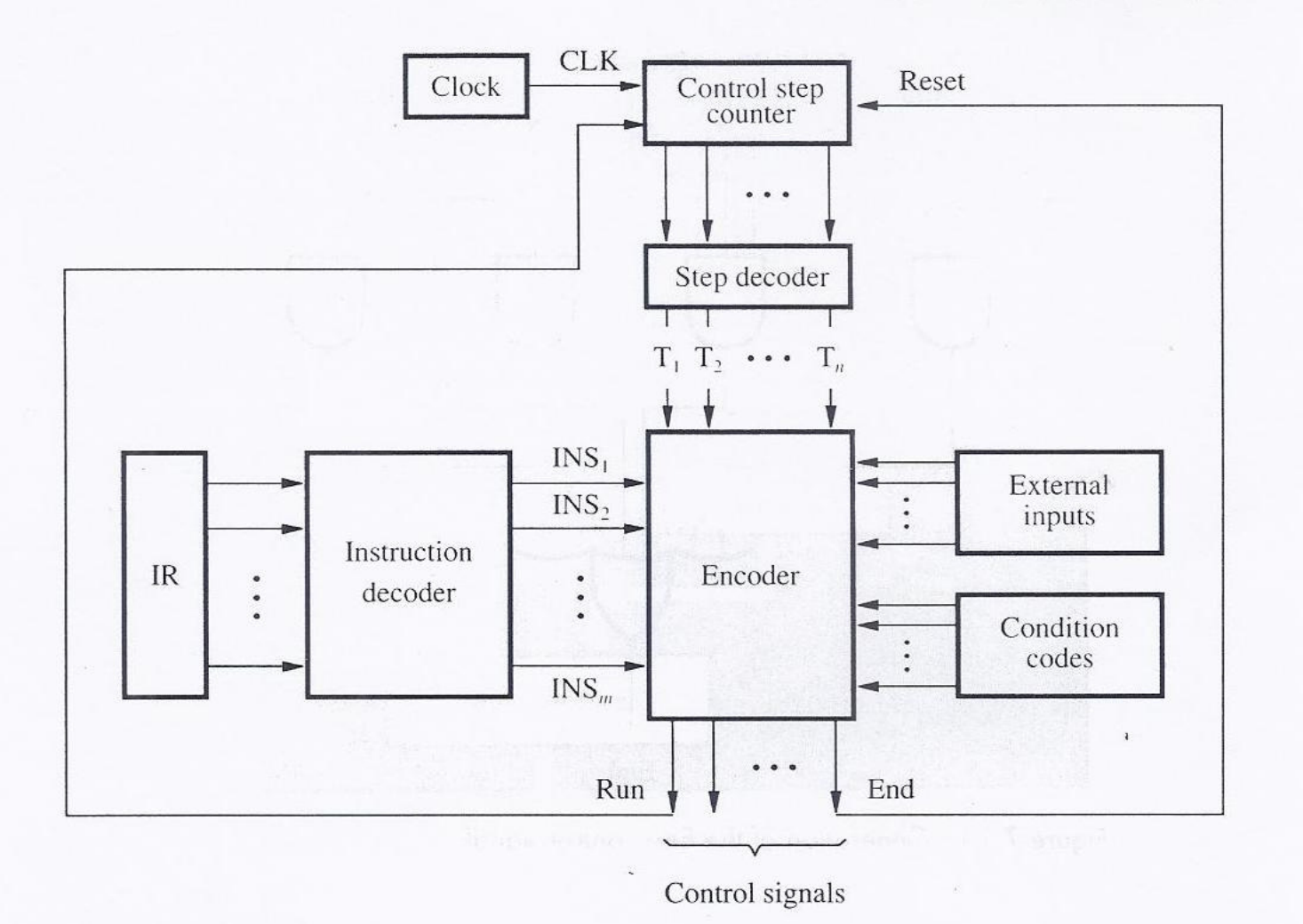
Hardwired Implementation (2)

- Control unit inputs
- Flags and control bus
  - —Each bit means something
- Instruction register
  - —Op-code causes different control signals for each different instruction
  - —Unique logic for each op-code
  - —Decoder takes encoded input and produces single output
  - —*n* binary inputs and 2^n outputs

Control Unit Organization

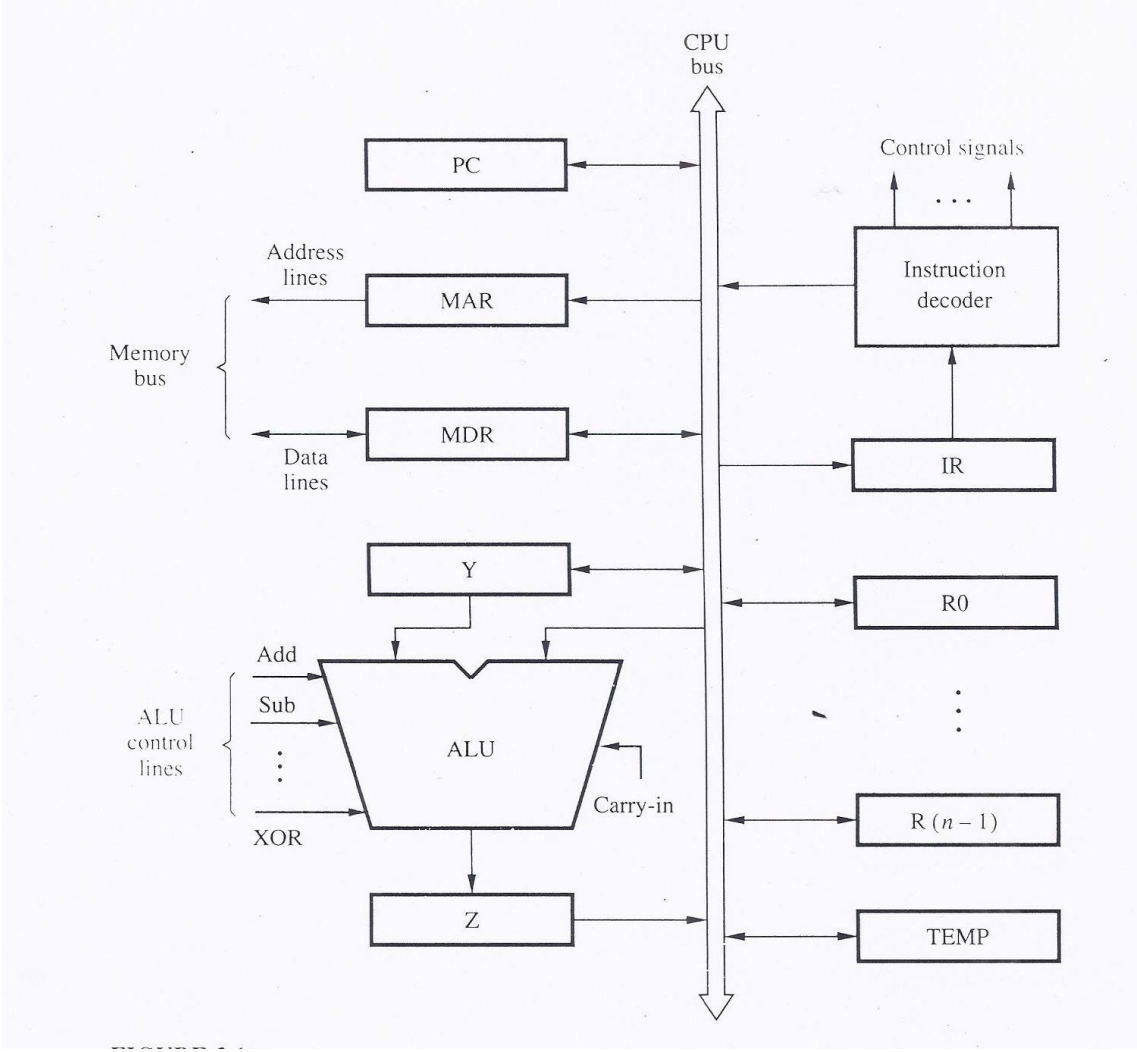


Decoder and encoder

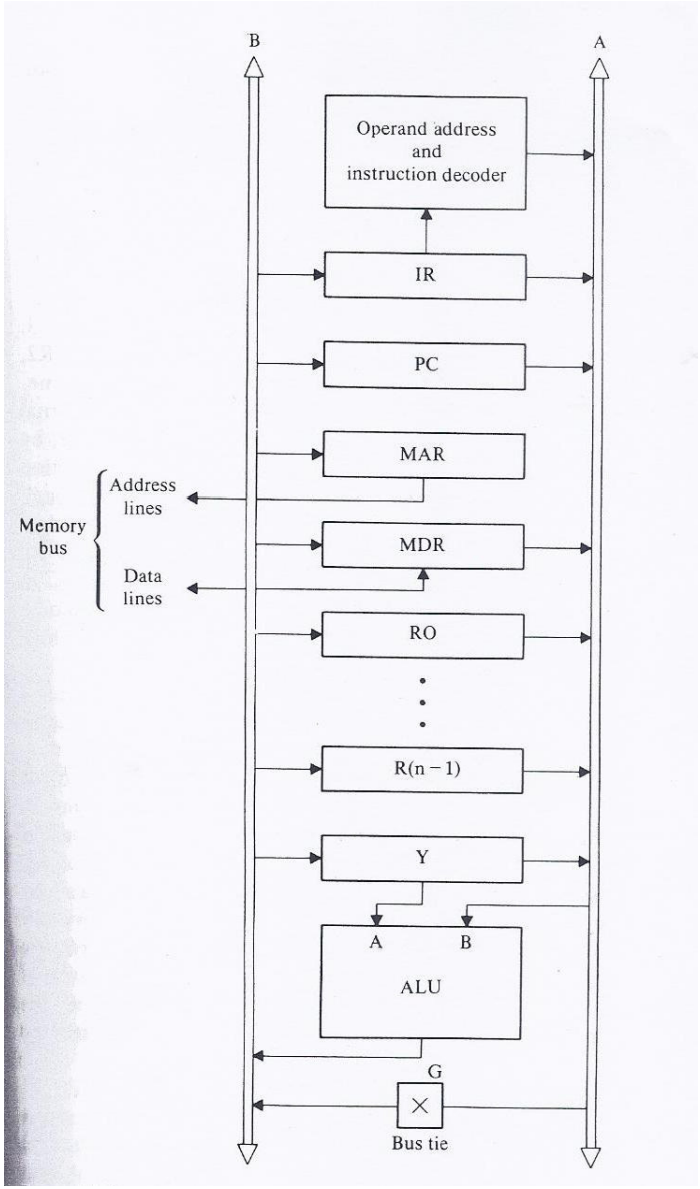


Some Other Organizations

- Already discussed



CPU Organization: Two Internal Buses



Control Step for Execution

- ADD R1, R2, R3
  - Add the contents of Register R1 and R2 and store the result in R3

Step	Action
1	$R1_{out}$, G_{enable}, Y_{in}
2	$R2_{out}$, ADD, ALU_{out}, $R3_{in}$

Control Step for Execution

- ADD R1, R2, R3
  - Add the contents of Register R1 and R2 and store the result in R3

Step Action

1 $R1_{out}$, G_{enable}, Y_{in}

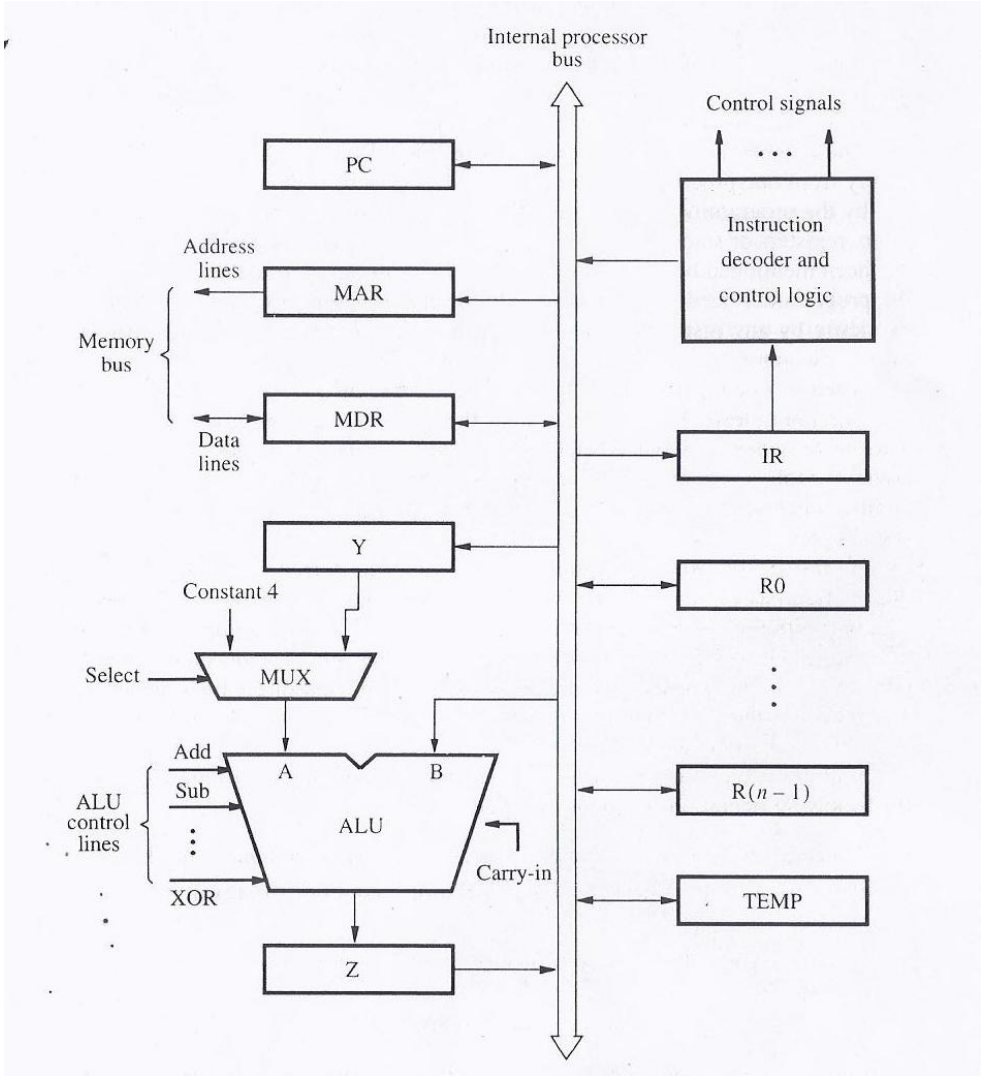
2 $R2_{out}$, ADD, ALU_{out}, $R3_{in}$

Two Bus

Step	Action
1.	$R1_{out}$, Y_{in}
2.	$R2_{out}$, Add, Z_{in}
3.	Z_{out}, $R3_{in}$

Single Bus

Single Bus Organization: another version



Fetch and Execute Instruction

Add (R3),R1

which adds the contents of a memory location pointed to by R3 to register R1. Executing this instruction requires the following actions:

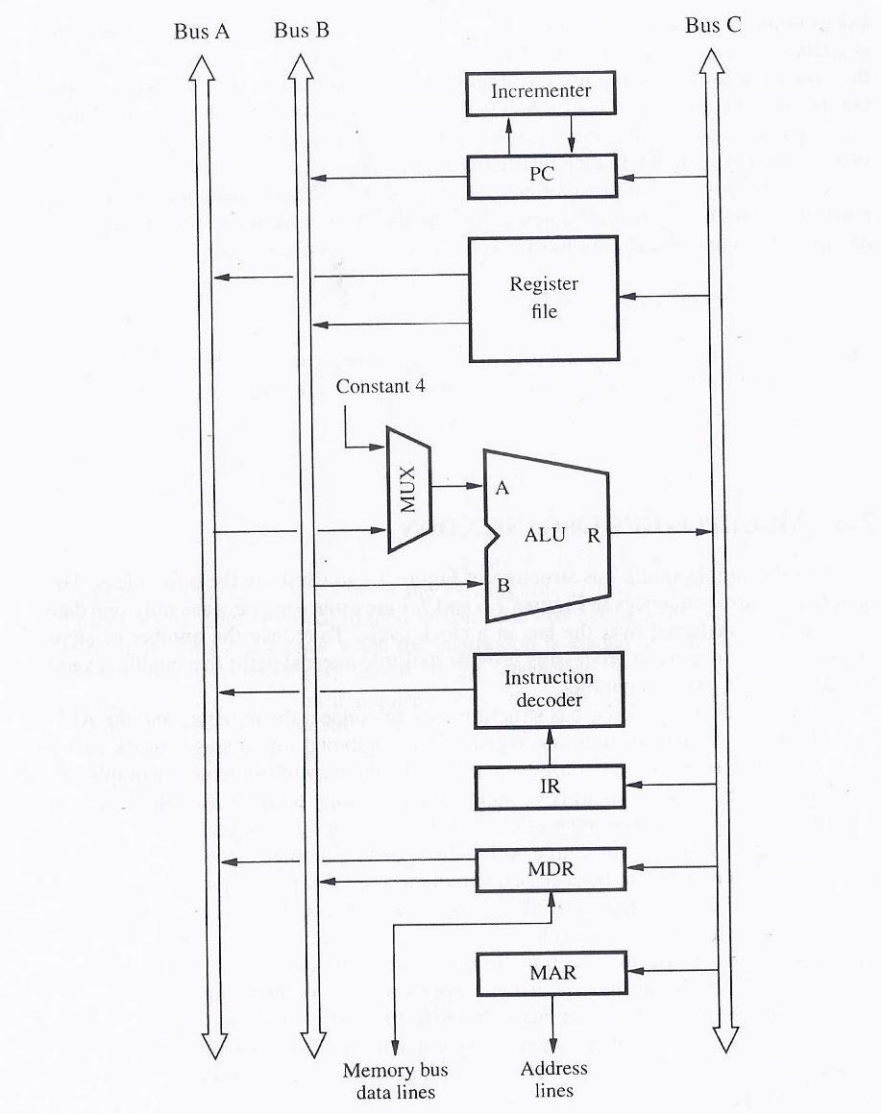
1. Fetch the instruction.
2. Fetch the first operand (the contents of the memory location pointed to by R3).
3. Perform the addition.
4. Load the result into R1.

Control Steps: Fetch and Execute

ADD (R3), R1: Add the content of register R1 and memory location pointed by R3; and store the result in R1

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

CPU organization: Three Internal Buses

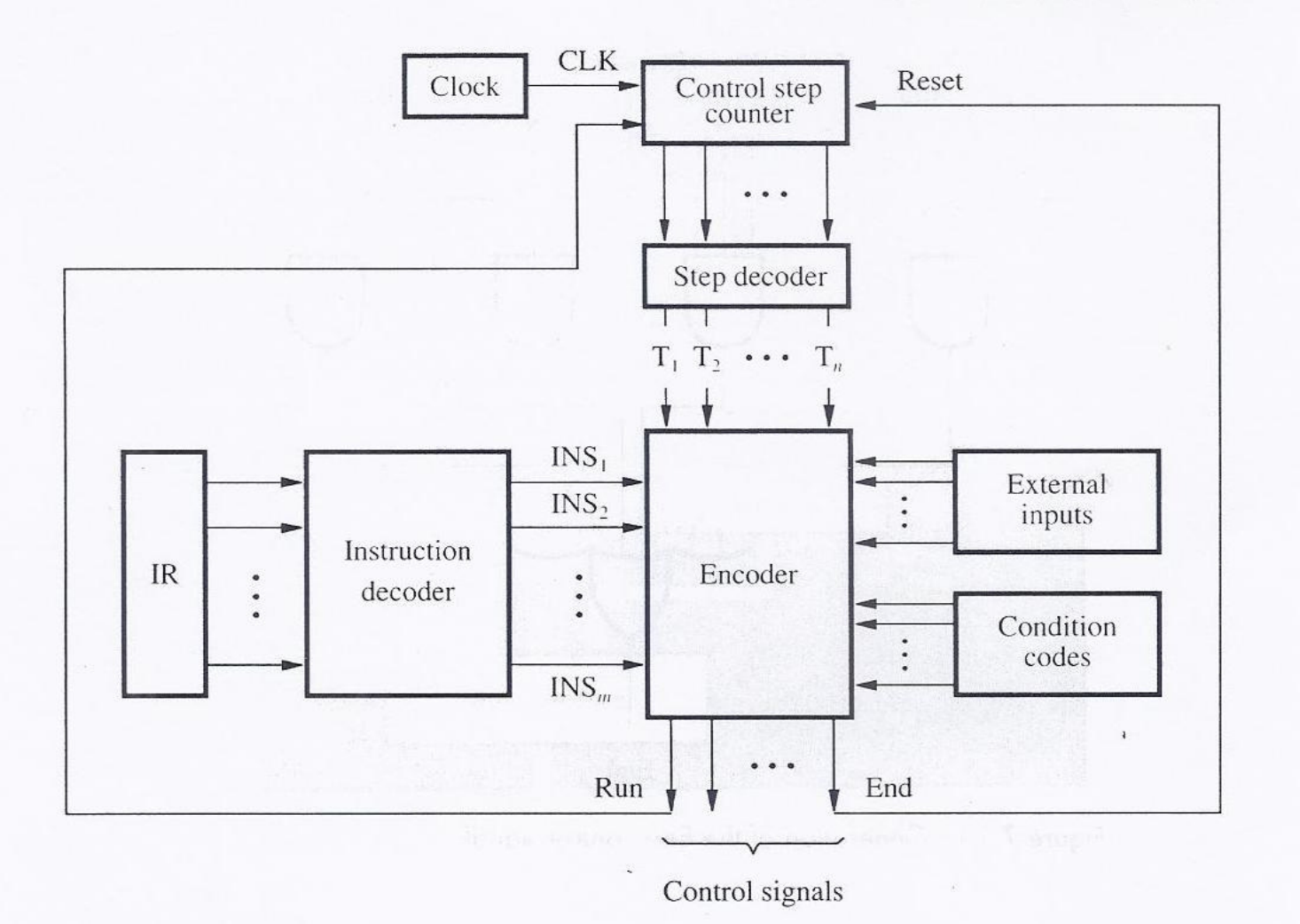


- One special circuit to increment the PC: Incrementer
- IncPC increments the value of PC by 4
  - Need timing control to load the new value to PC
- Special arrangement to pass the value through the ALU
  - R = B control signal to pass the value of bus B to bus C through ALU

ADD R4, R5, R6: Add the content of register R4 and R5; and store the result in R6

Step	Action
1	PC_{out}, R=B, MAR_{in}, Read, IncPC
2	WMFC
3	MDR_{outB}, R=B, IR_{in}
4	$R4_{outA}$, $R5_{outB}$, SelectA, Add, $R6_{in}$, End

Control Unit: Hardwired Control



Problems With Hard Wired Designs

- Complex sequencing & micro-operation logic
- Difficult to design and test
- Inflexible design
- Difficult to add new instructions