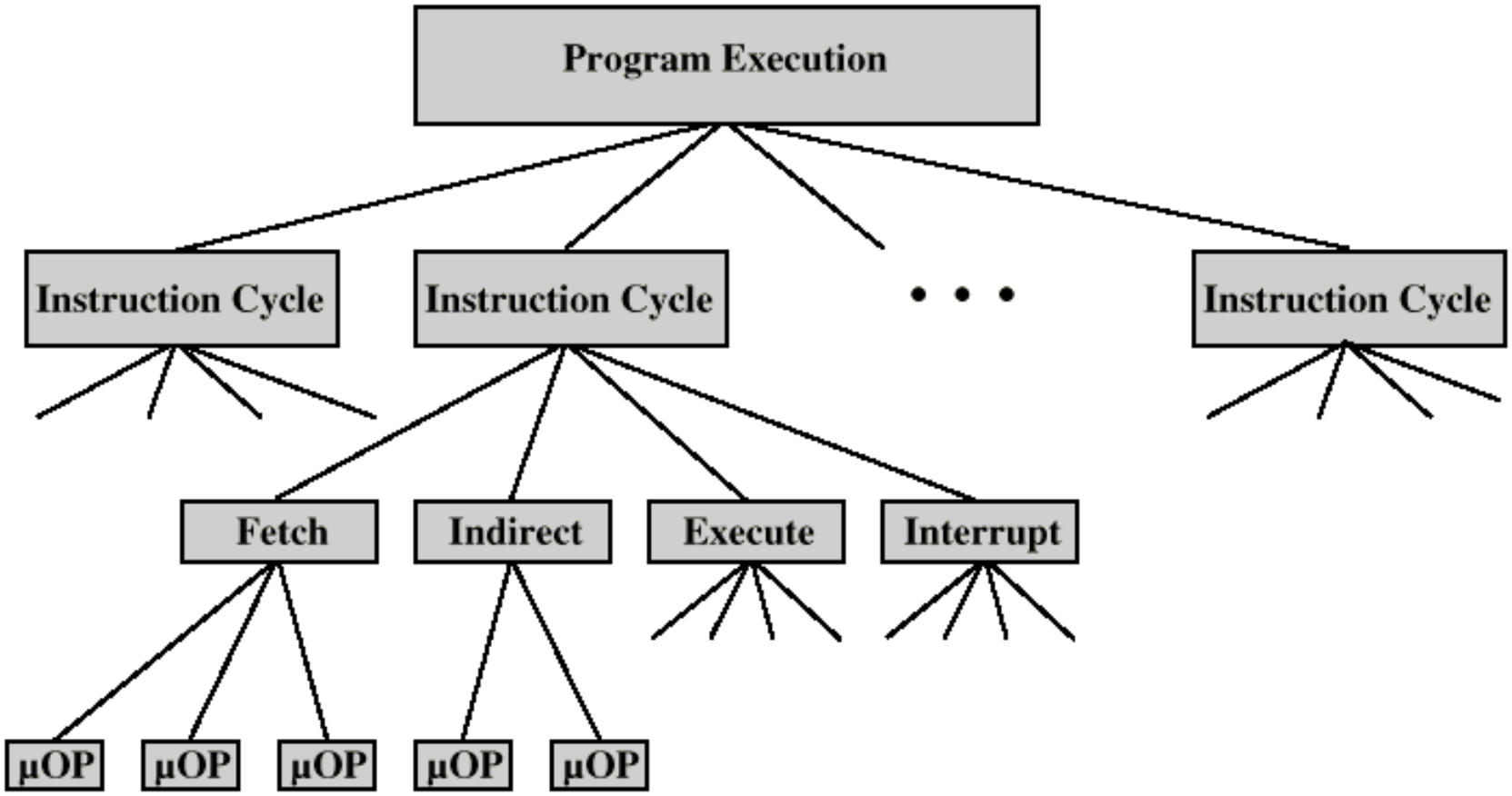
Control Unit Operation

Micro-Operations

- A computer executes a program
- Fetch/execute cycle
- Each cycle has a number of steps
- Called micro-operations
- Each step does very little

Constituent Elements of Program Execution



Fetch - 4 Registers

- Memory Address Register (MAR)
  - —Connected to address bus
  - —Specifies address for read or write op
- Memory Buffer Register (MBR)
  - —Connected to data bus
  - —Holds data to write or last data read
- Program Counter (PC)
  - —Holds address of next instruction to be fetched
- Instruction Register (IR)
  - —Holds last instruction fetched

Fetch Sequence

- Address of next instruction is in PC
- Address (MAR) is placed on address bus
- Control unit issues READ command
- Result (data from memory) appears on data bus
- Data from data bus copied into MBR
- PC incremented by 1 (in parallel with data fetch from memory)
- Data (instruction) moved from MBR to IR
- MBR is now free for further data fetches

Fetch Sequence (symbolic)

- t1: MAR <- (PC)
- t2: MBR <- (memory)
  PC <- (PC) +1
- t3: IR <- (MBR)
  - (tx = time unit/clock cycle)

- t1: MAR <- (PC)
- t2: MBR <- (memory)
- t3: PC <- (PC) +1
  IR <- (MBR)

Rules for Clock Cycle Grouping

- Proper sequence must be followed
  - —MAR <- (PC) must precede MBR <- (memory)
- Conflicts must be avoided
  - —Must not read & write same register at same time
  - —MBR <- (memory) & IR <- (MBR) must not be in same cycle
- Also: PC <- (PC) +1 involves addition
  - —Use ALU
  - —May need additional micro-operations

Indirect Cycle

- MAR <- ($IR_{address}$) - address field of IR
- MBR <- (memory)
- $IR_{address}$ <- ($MBR_{address}$)

- MBR contains an address
- IR is now in same state as if direct addressing had been used

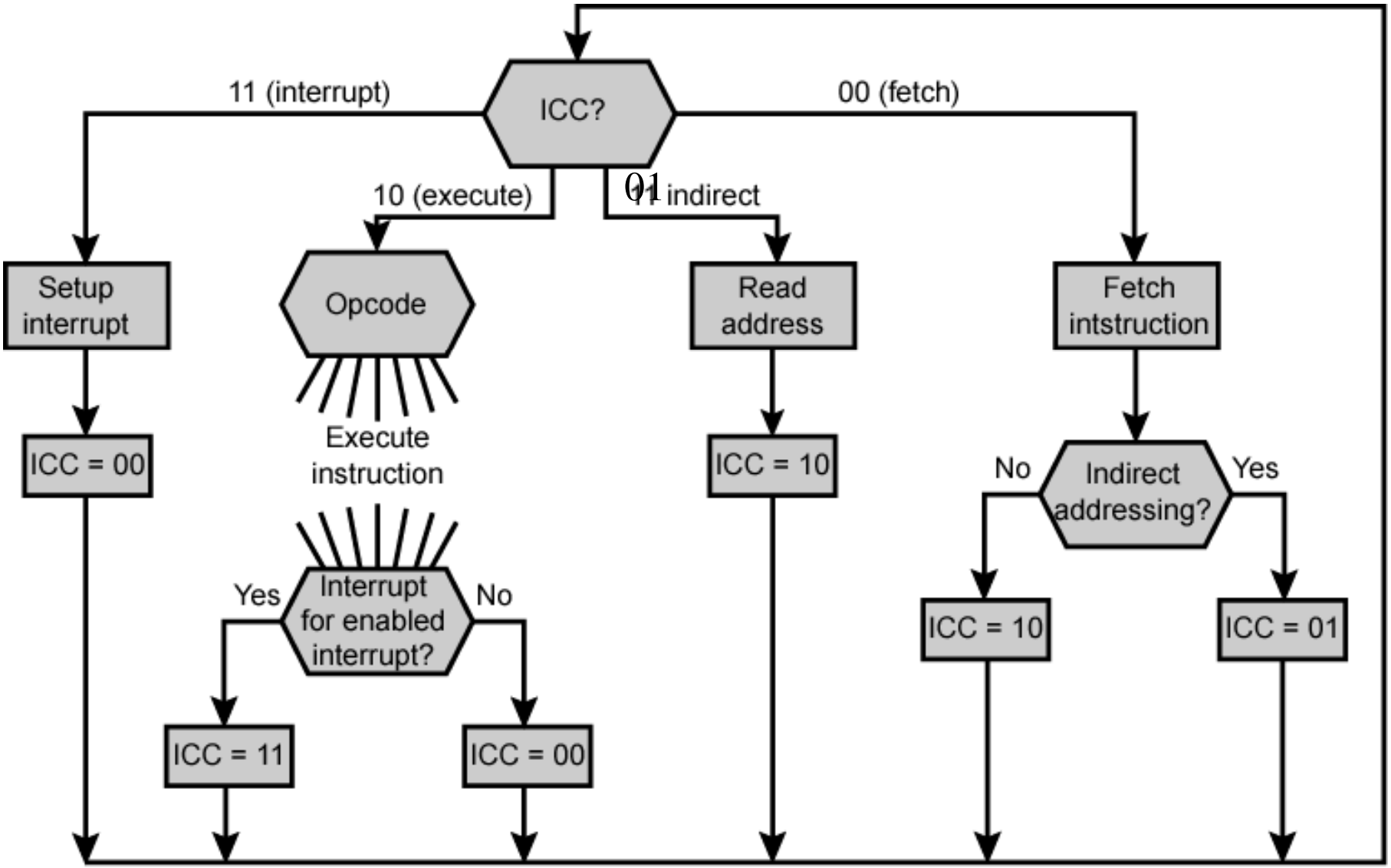
Execute Cycle (ADD)

- Different for each instruction
- e.g. ADD R1,X - add the contents of location X to Register R1 , result in R1
  - t1: MAR <- ($IR_{address}$)
  - t2: MBR <- (memory)
  - t3: R1 <- R1 + (MBR)

Instruction Cycle

- Each phase decomposed into sequence of elementary micro-operations
- E.g. fetch, indirect, and interrupt cycles
- Execute cycle
  - One sequence of micro-operations for each opcode
- Need to tie sequences together
- Assume new 2-bit register
  - Instruction cycle code (ICC) designates which part of cycle processor is in
    - 00: Fetch
    - 01: Indirect
    - 10: Execute
    - 11: Interrupt

Flowchart for Instruction Cycle



Functional Requirements

- Define basic elements of processor
- Describe micro-operations that processor performs
- Determine functions that control unit must perform

Basic Elements of Processor

- ALU
- Registers
- Internal data paths
- External data paths
- Control Unit

Types of Micro-operation

- Transfer data between registers
- Transfer data from register to external
- Transfer data from external to register
- Perform arithmetic or logical ops

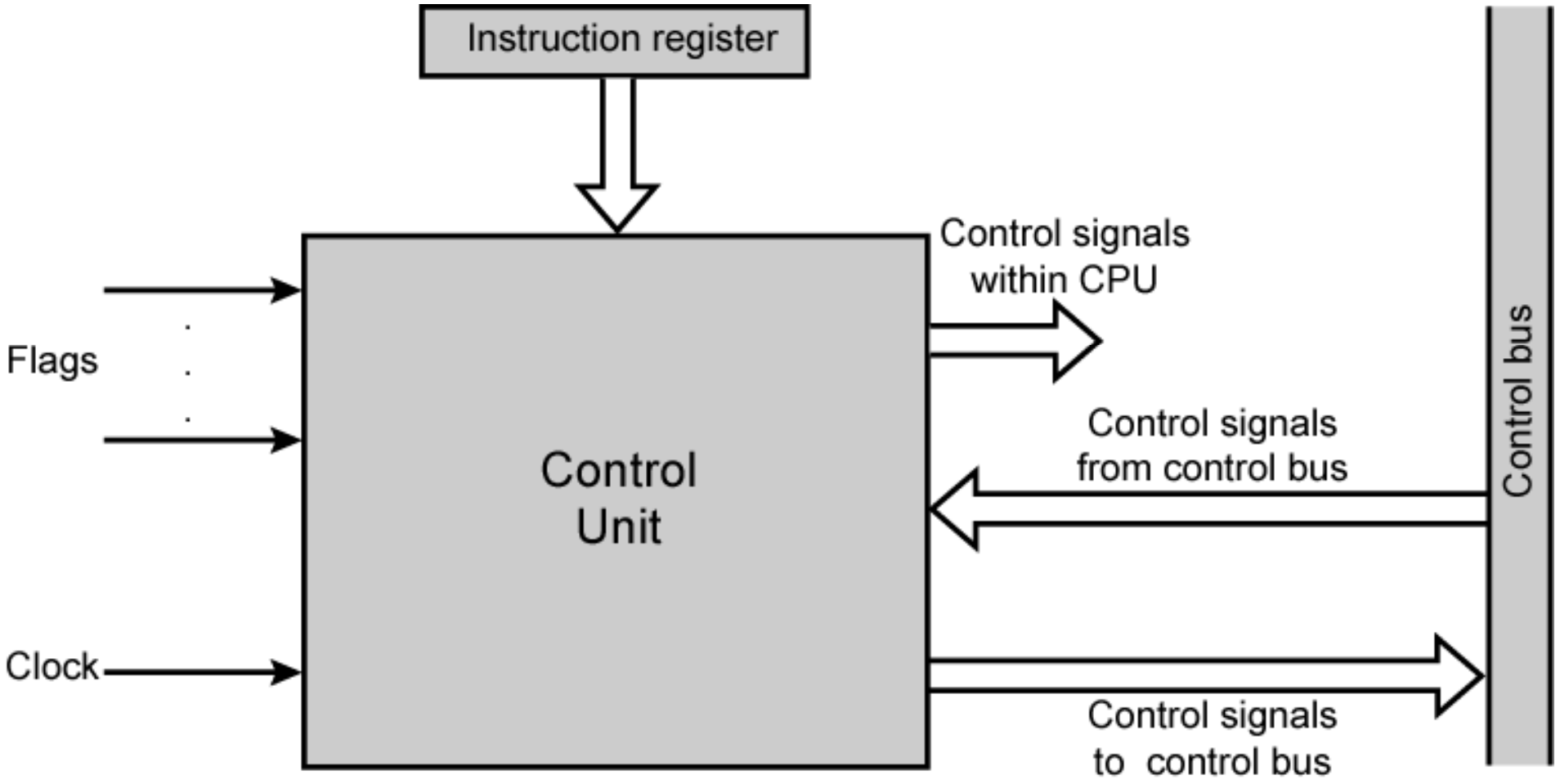
Functions of Control Unit

- Sequencing
  - —Causing the CPU to step through a series of micro-operations
- Execution
  - —Causing the performance of each micro-op
- This is done using Control Signals

Control Signals

- Clock
  - One micro-instruction (or set of parallel micro-instructions) per clock cycle
- Instruction register
  - Op-code for current instruction
  - Determines which micro-instructions are performed
- Flags
  - State of CPU
  - Results of previous operations
- From control bus
  - Interrupts
  - Acknowledgements

Model of Control Unit



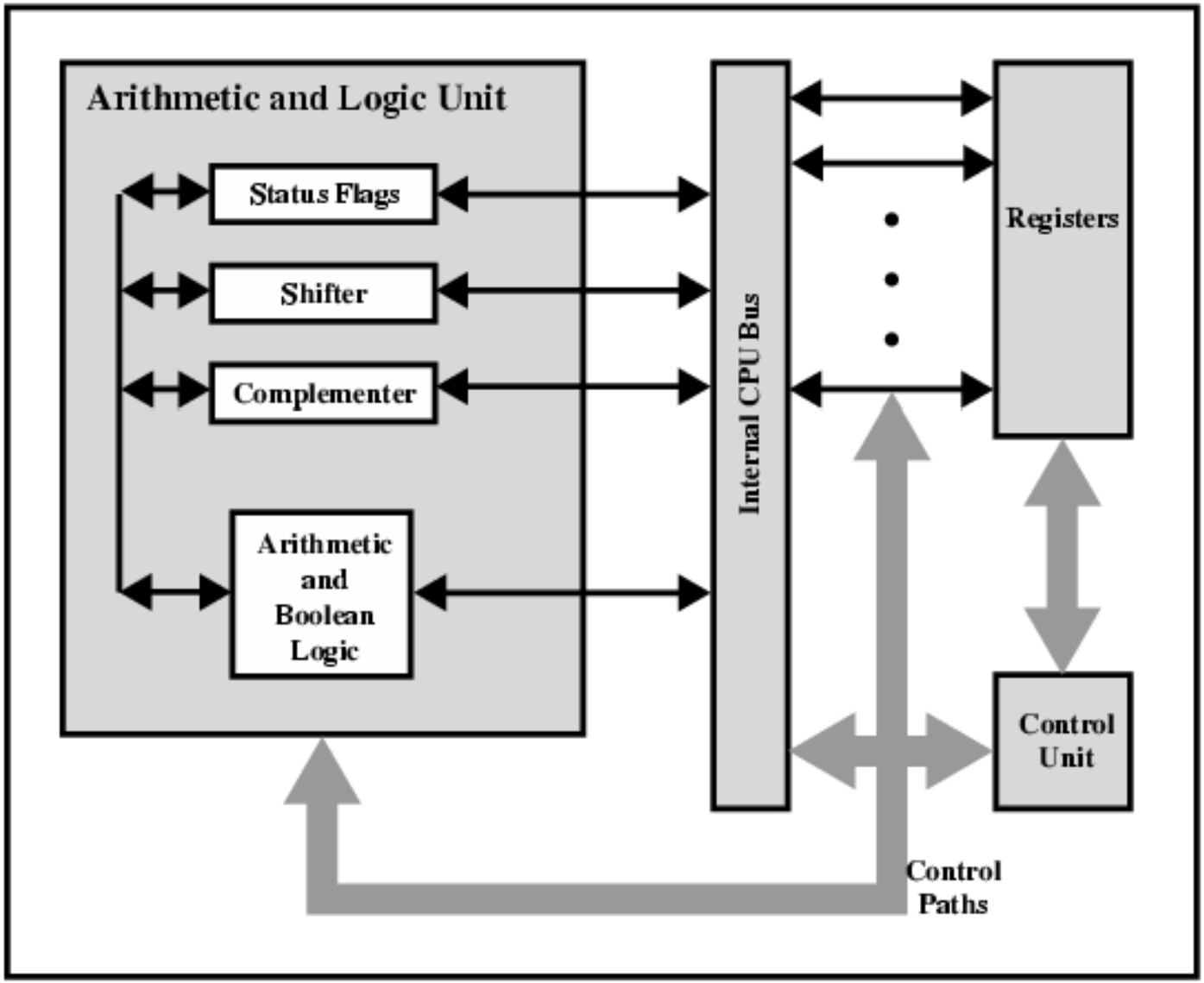
Control Signals - output

- Within CPU
  - Cause data movement
  - Activate specific functions
- Via control bus
  - To memory
  - To I/O modules

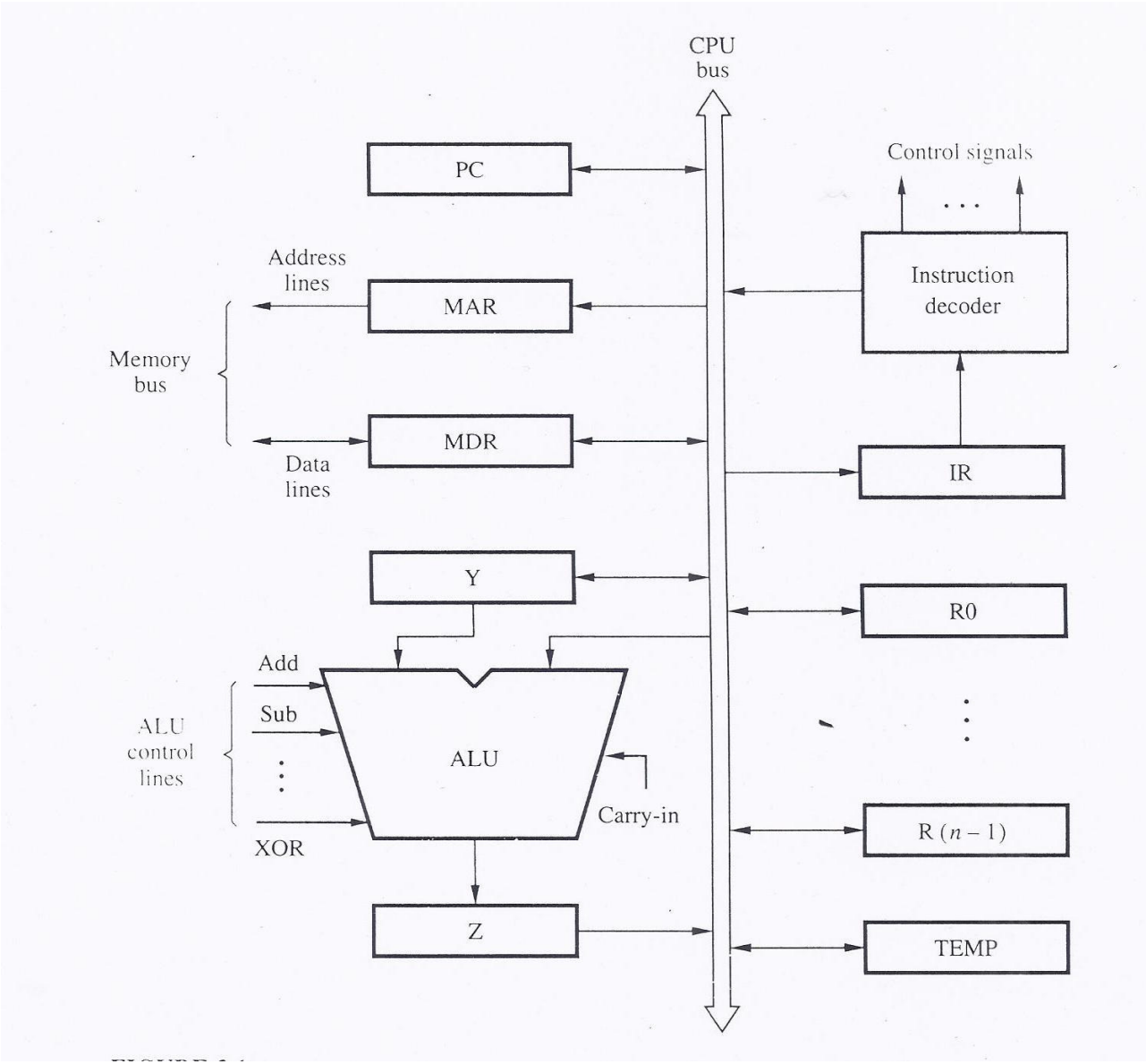
Example Control Signal Sequence - Fetch

- MAR <- (PC)
  - Control unit activates signal to open gates between PC and MAR
- MBR <- (memory)
  - Open gates between MAR and address bus
  - Memory read control signal
  - Open gates between data bus and MBR

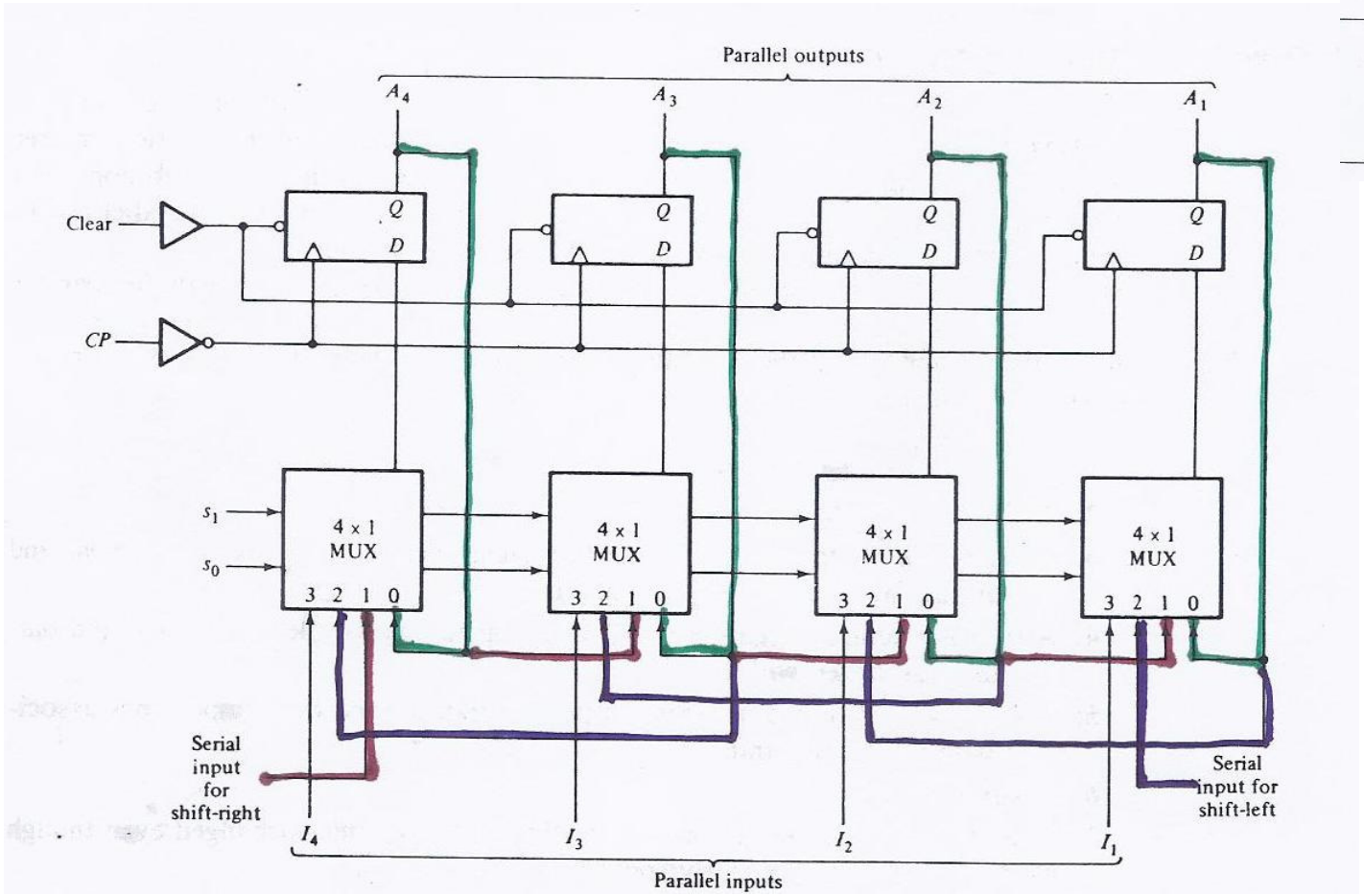
CPU Internal Structure



Single Bus Organization of CPU

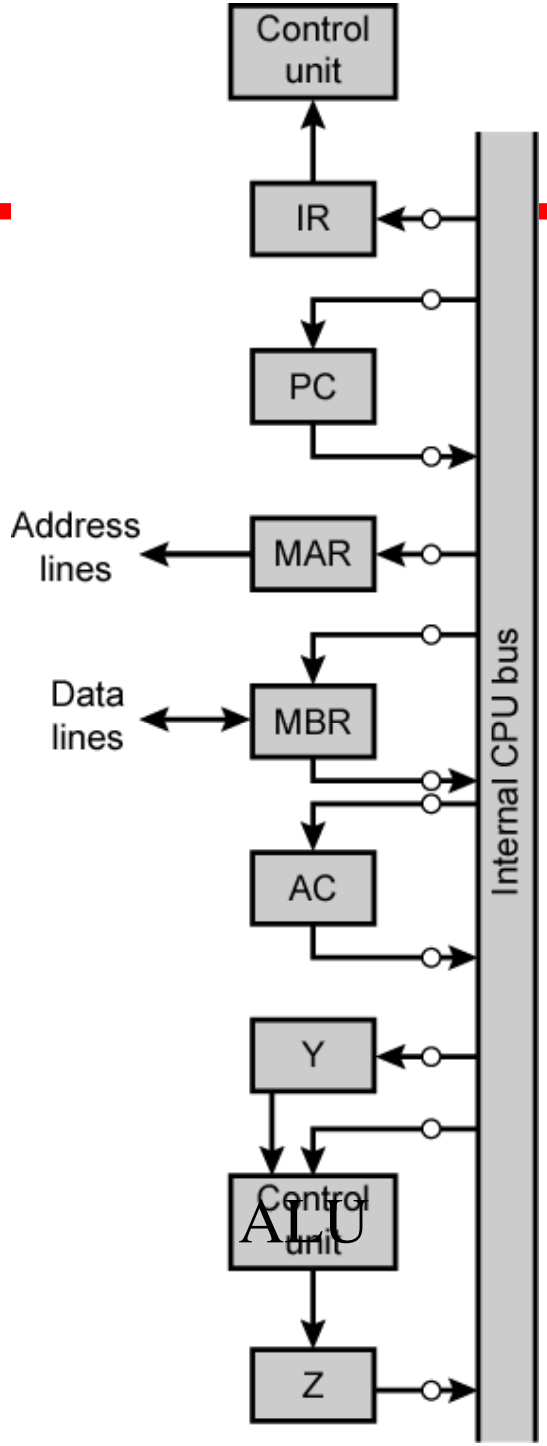


4-bit Register (Parallel Load & Shift)

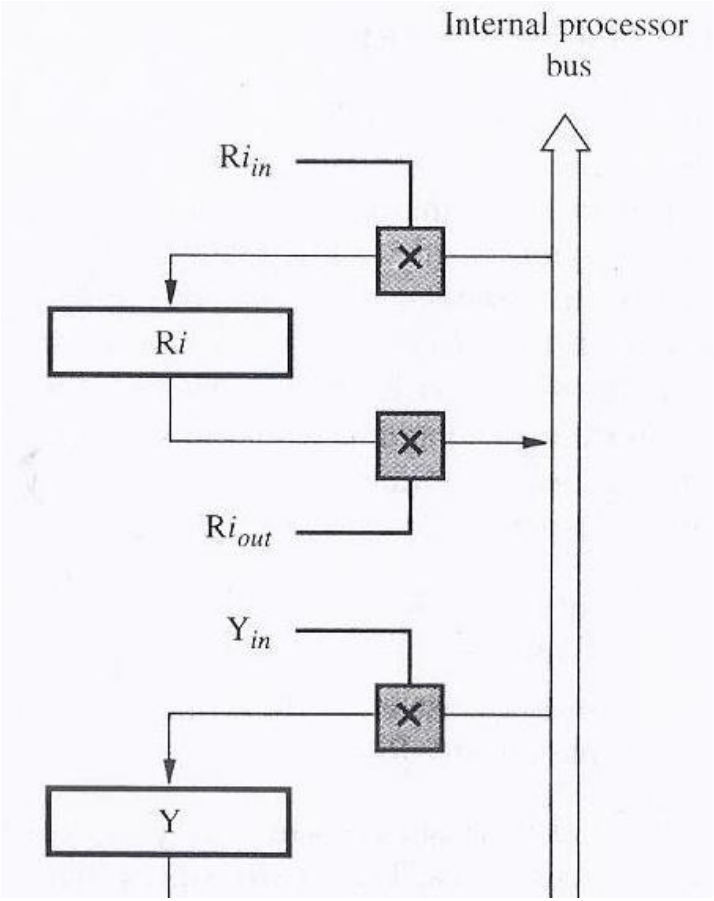


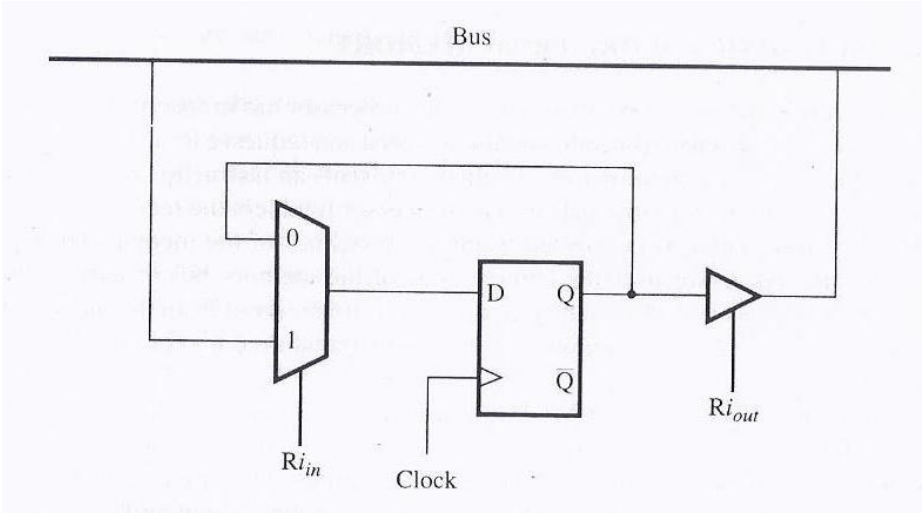
Mode Control		
S_1	S_0	Register Operation
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

CPU with Internal Bus

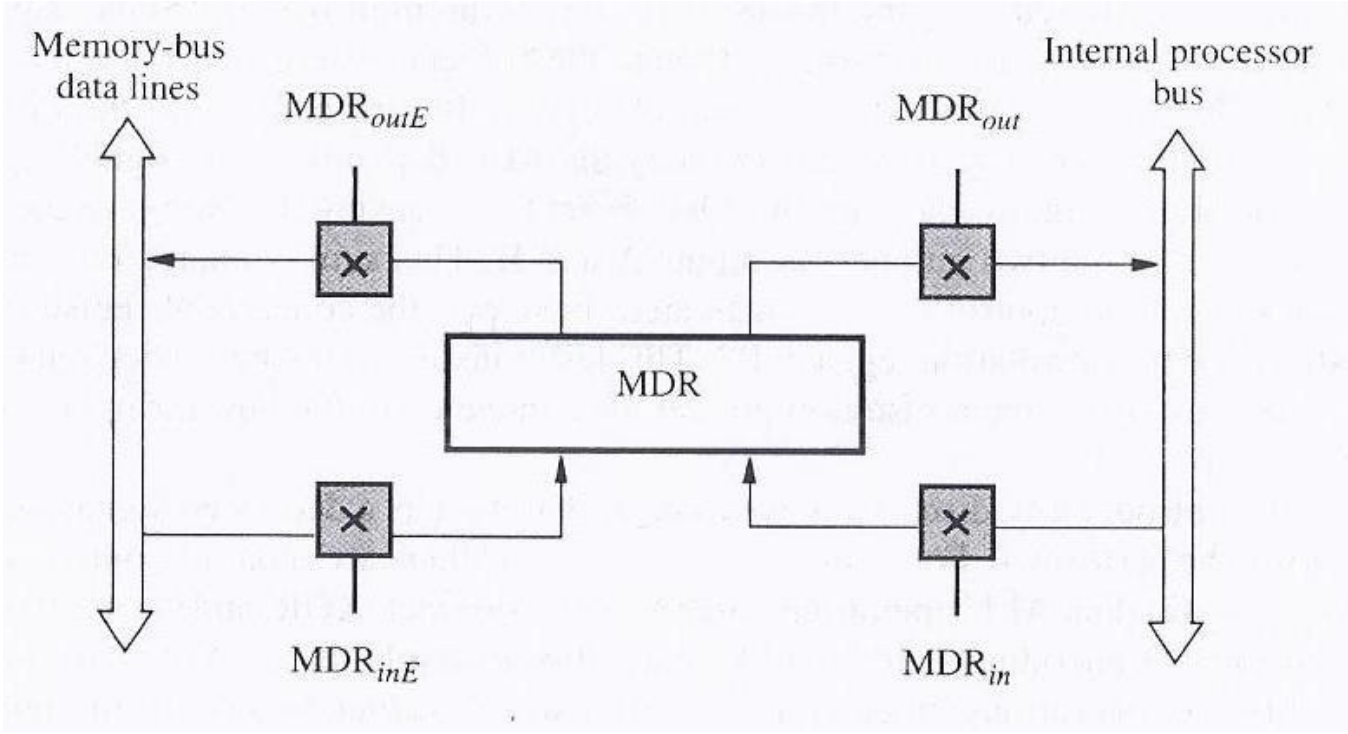


Register and Bus Connection

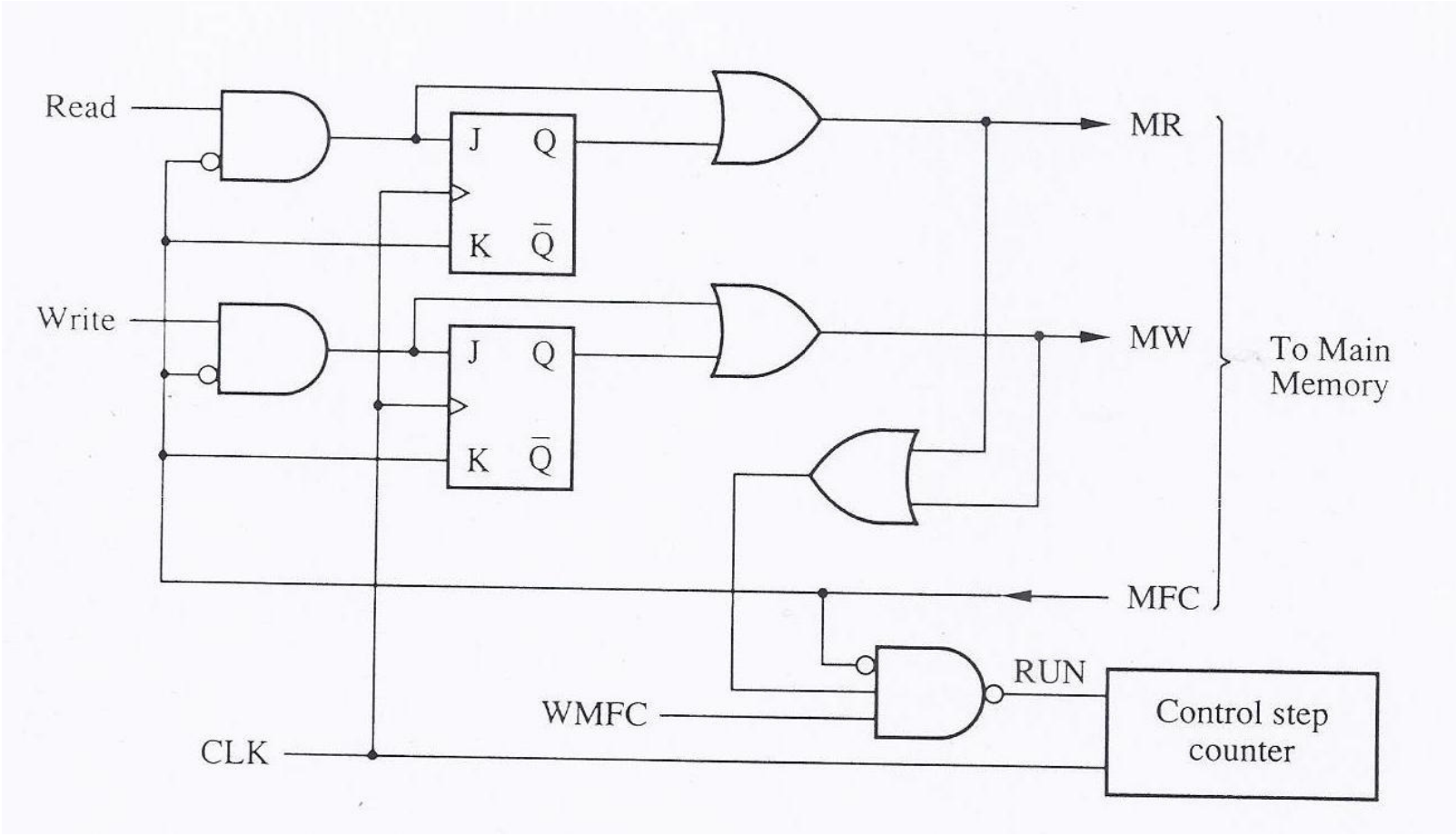




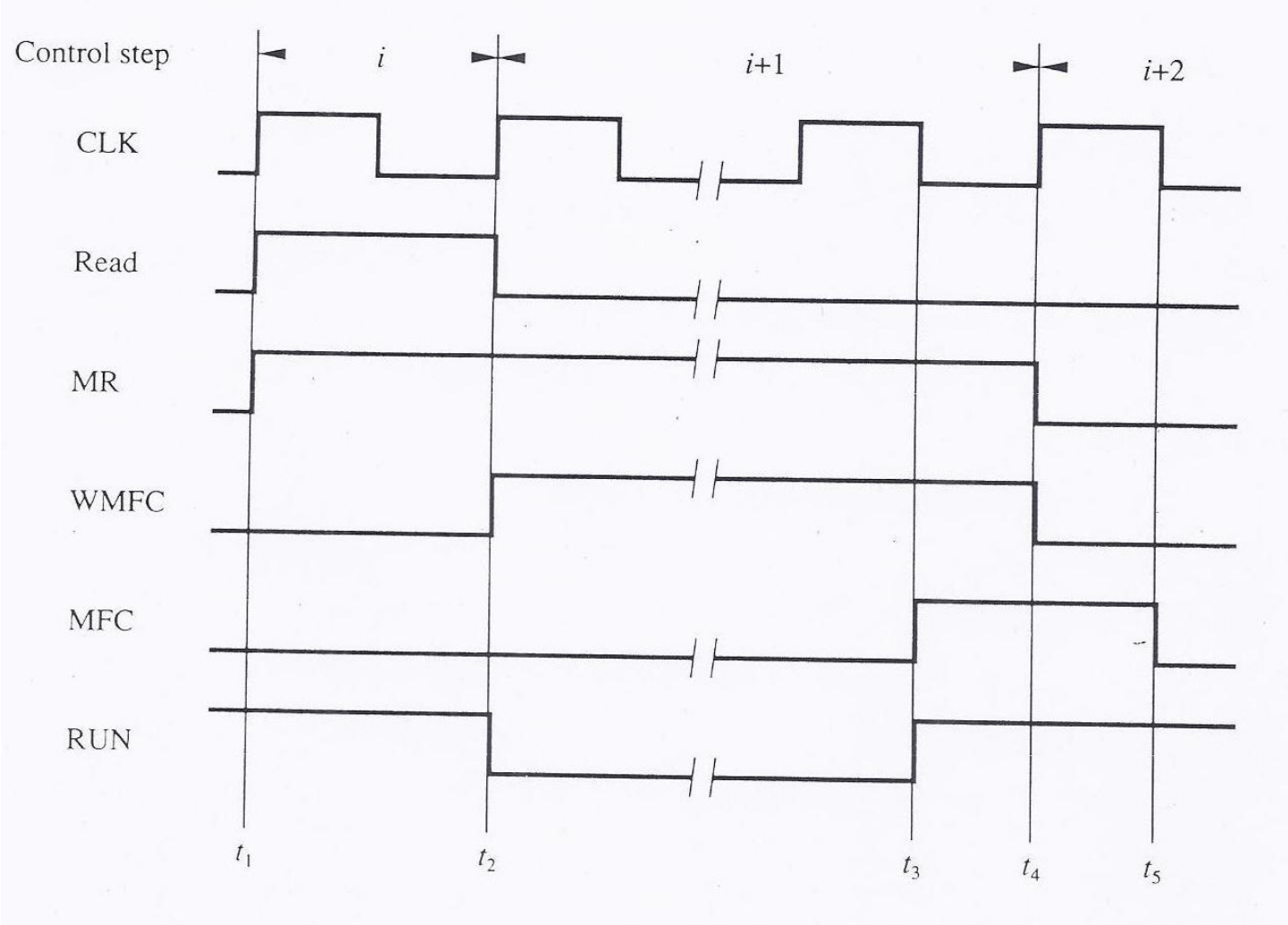
Internal and External Bus



Read and Write Signal



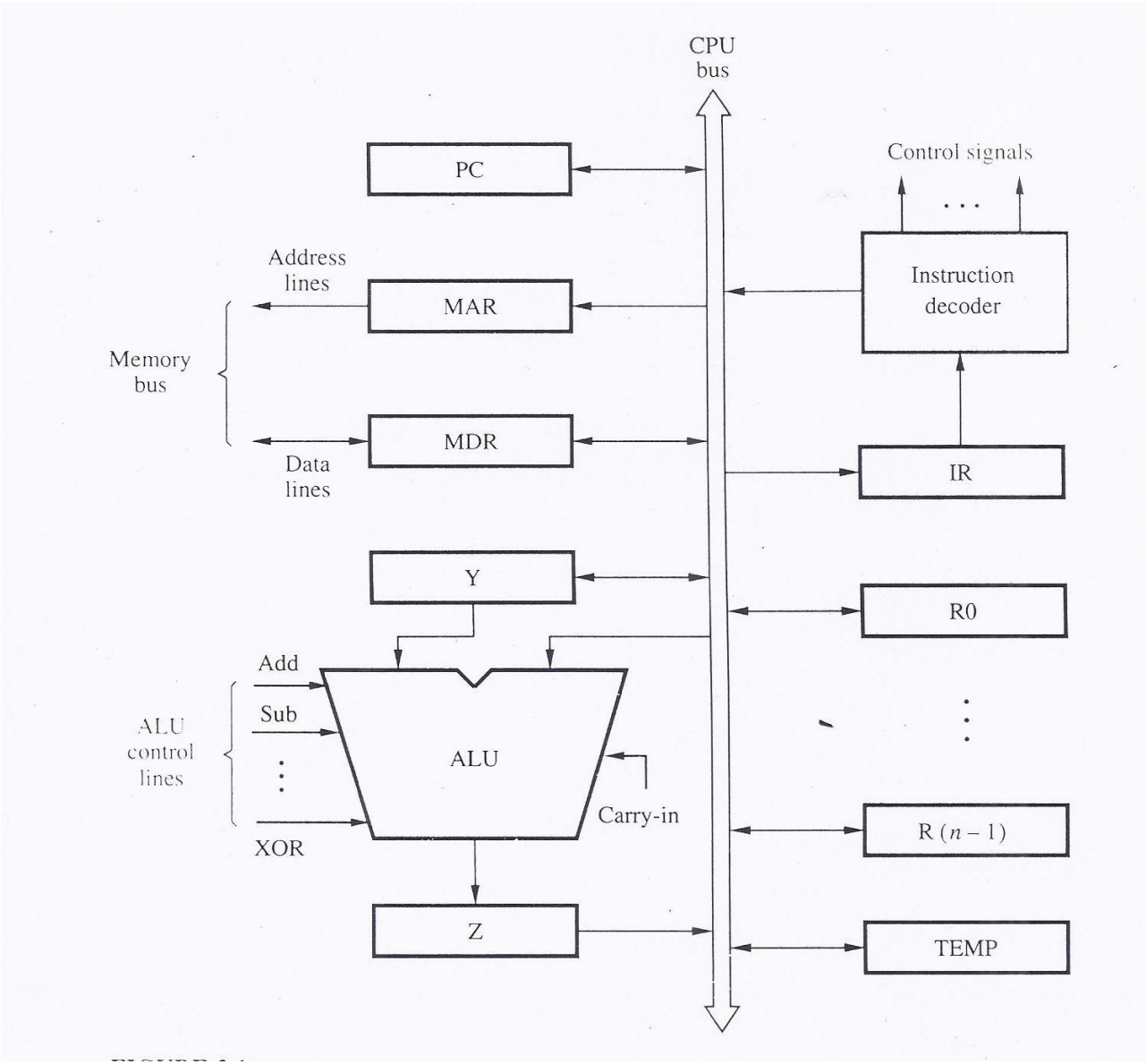
Timing Diagram



Control Step for Execution

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

Single Bus Organization of CPU



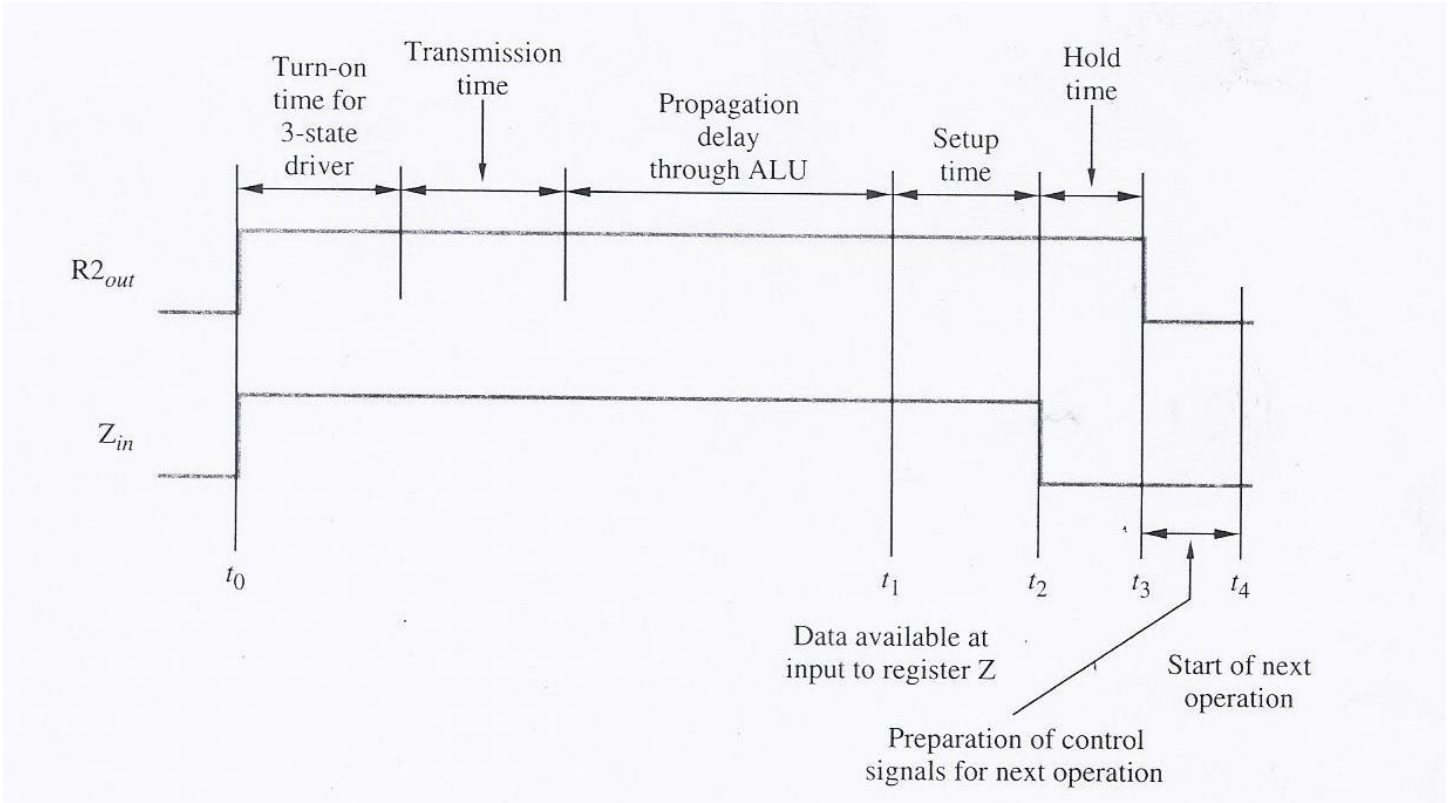
Control Step for Execution

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

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

Clock Timing

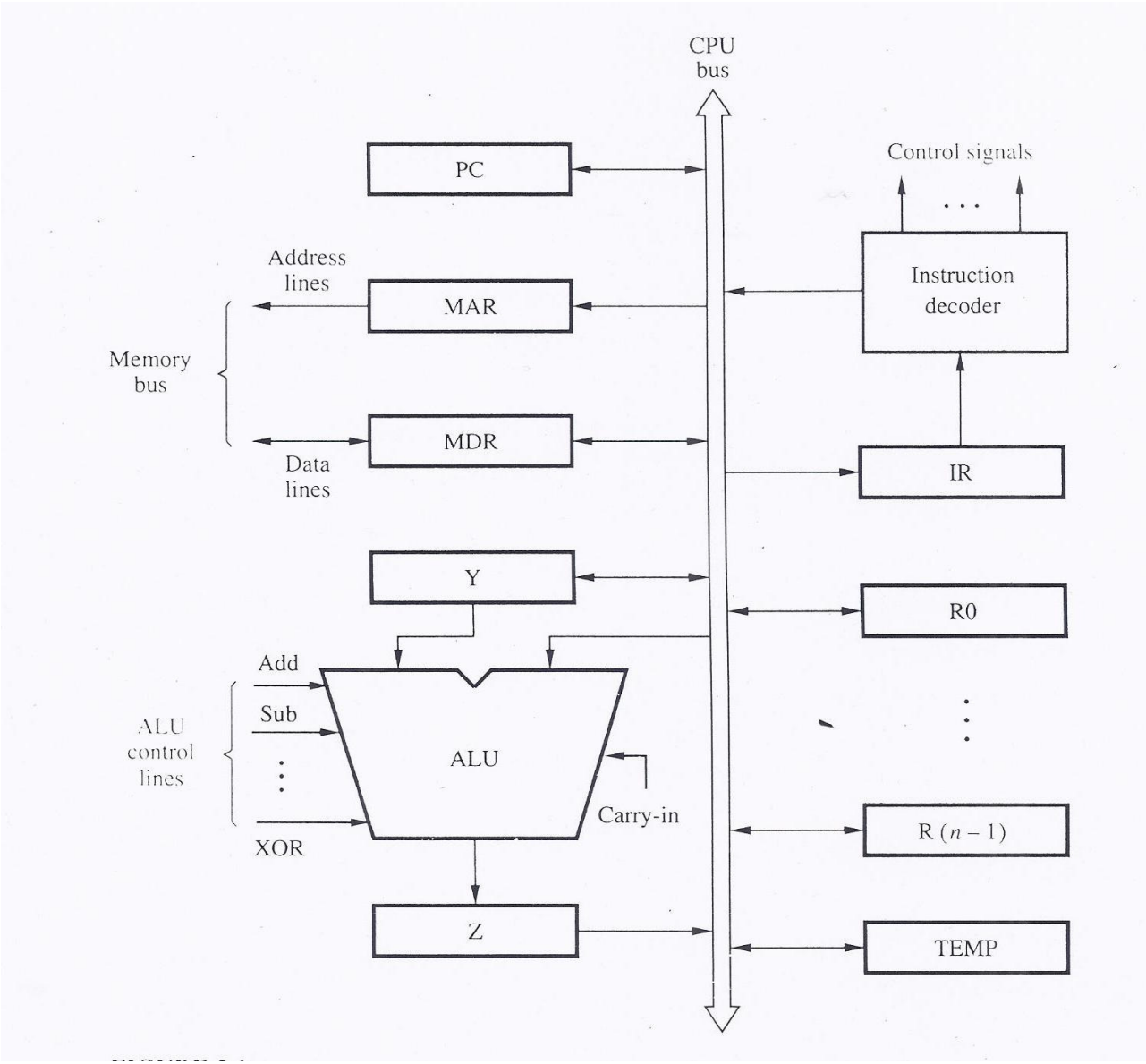
- Time needed for micro-operation 2
  - $R2_{out}$, ADD, Z_{in}



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

Single Bus Organization of CPU



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

Unconditional Branch

- Control sequence for an unconditional Branch Instruction

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	PC_{out}, Y_{in}
5	Offset-field-of-IR_{out}, Add, Z_{in}
6	Z_{out}, PC_{in}, End

Conditional Branch

- Branch on Negative: BRN
  - Use of condition codes/flags
  - depends on flag bit: N – Negative
  - N: set to 0 if the ALU result is not negative
  - N: set to 1 if the ALU result is negative

Conditional Branch

- Branch on Negative: BRN
  - Step 4 is replaced by:
  - PC_{out}, Y_{in}, If N == 0 then End

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	PC_{out}, Y_{in}
5	Offset-field-of-IR_{out}, Add, Z_{in}
6	Z_{out}, PC_{in}, End