Basic Processing Unit

1

Central Processing Unit (CPU)

- Examine internal structure of processor and how it performs the tasks of fetching, decoding and executing the instructions of a program

```
            SUB    R0, R0, R0; R0 has the value of index i
Loop_Begin: CMP    R0, N
            JEQ    Loop_End
            ADD    A[R0], B[R0], C[R0]
            INC    R0
            JMP    Loop_Begin
Loop_End:
```

- To execute a program, the processor fetches one instruction at a time and performs the operation specified
- Instructions are fetched from the successive memory locations until the branch instruction is encountered

2

2

Central Processing Unit (CPU)

- Program counter (PC):
  - Holds the next instruction to be fetched
  - Helps the processor to keep track of the address of the memory location of next instruction to be fetched
  - After fetching, the content of PC is updated to point to the next instruction in the sequence
  - Branch instruction may load different value into the PC
- Instruction register (IR):
  - Instruction to be executed is loaded into the IR

3

3

Execution of an Instruction

- Suppose each instruction is 4 bytes in length and stored in one memory word
- Assume that memory is byte addressable
- Steps to execute an instruction:
  - Step1: Fetch the content of the memory location pointed to by the PC and are load into IR

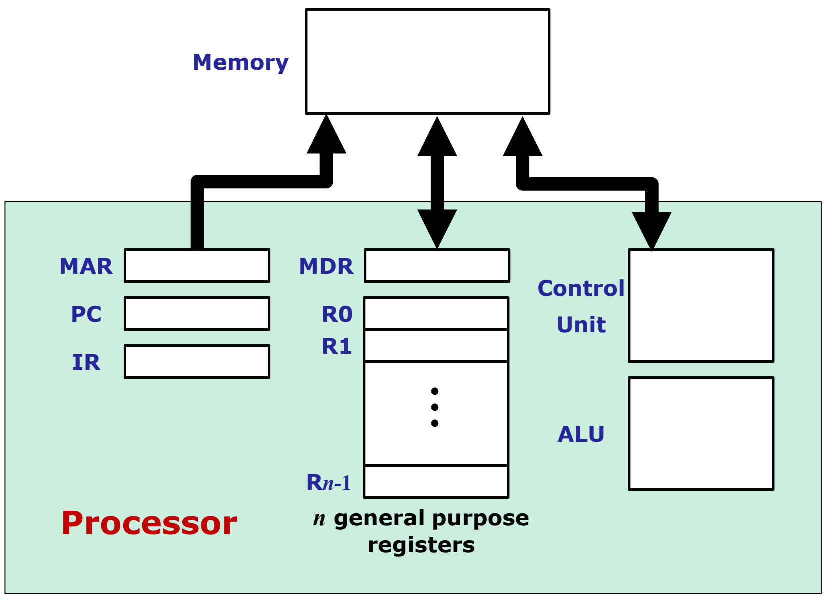
    $IR \leftarrow [[PC]]$
  - Step2: Increment the content of the PC by 4

    $PC \leftarrow [PC] + 4$
  - Step3: Carry out the actions specified by the instruction in the IR
- Step1 and Step2 repeated each time to fetch the instruction. They are referred to as Fetch Phase
- Step3 is referred as Execute Phase

4

4

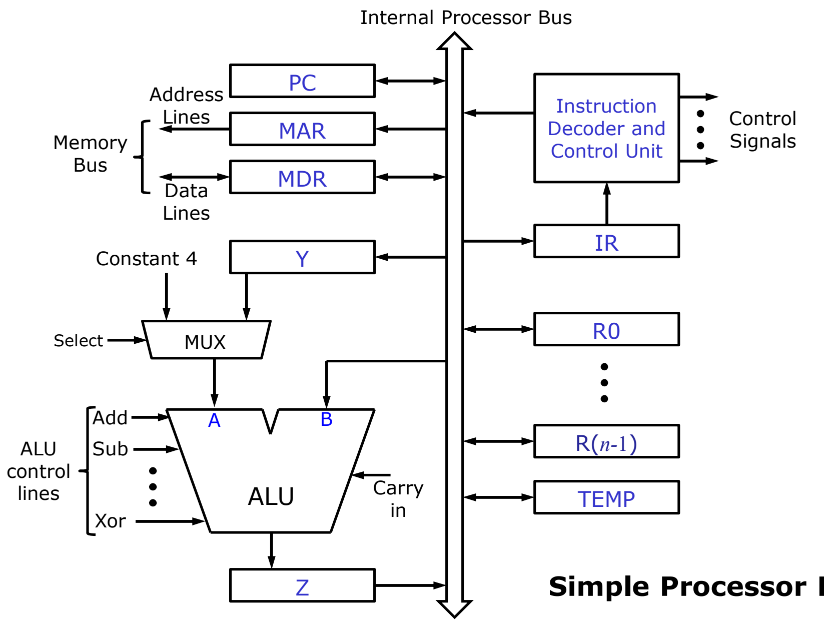
Building Blocks of a Processor



5

5

Single Bus Organization of the Data Path inside Processor



Simple Processor Model

6

6

Operations During Instruction Execution

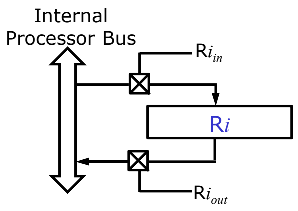
- Instruction are executed by performing one or more of the following operations:
  - Transfer a word of data from one processor register to another or to ALU
  - Perform arithmetic or logic operation and store the results in the processor register
  - Fetch the content of a given memory location and load them into a processor register
  - Store a word of data from a processor register into a given memory location
- All the operations and data transfer within processor take place within a time period defined by processor clock
- Control signals that govern a particular transfer are asserted at the start of the clock cycle
- Each action is performed in one or more clock cycles

7

7

Register Transfers

- Transfer a word of data from one processor register to another or to ALU
- Example: MOV R1, R4 Data is transferred from R4 to R1



- For each register, two control signals are used
  - One to place the content of the register on the bus
  - Another to load the data on the bus into the register
- Input and output of register Ri are connected to bus via switches controlled by the signals Ri_{in} and Ri_{out}
  - $Ri_{in} = 1$: Data on the bus loaded into Ri
  - $Ri_{out} = 1$: Dta in Ri palce on the bus

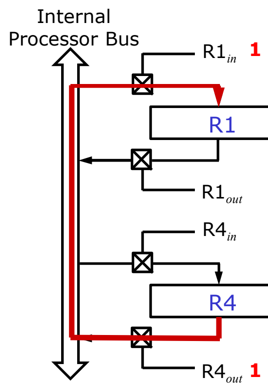
8

8

Register Transfers

- Example: MOV R1, R4

$R1 \leftarrow R4$ }

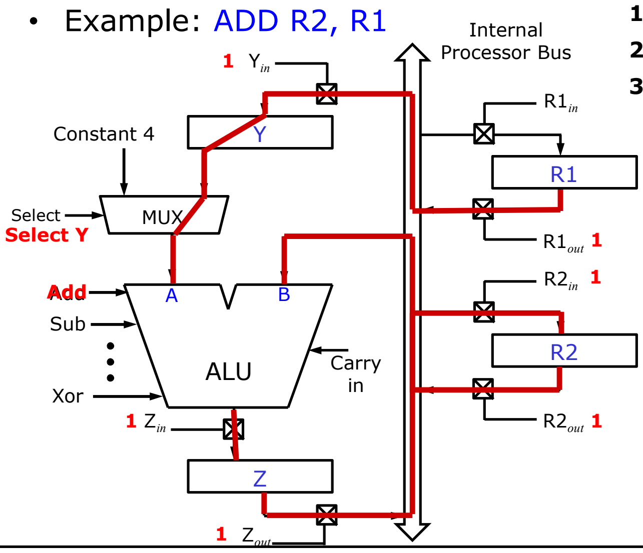


- $R4_{out}$, $R1_{in}$
- Processor completes its internal data transfer in 1 clock cycle

Arithmetic or Logic Operation

- ALU is a combinational circuit that has no internal storage
- Example: ADD R2, R1

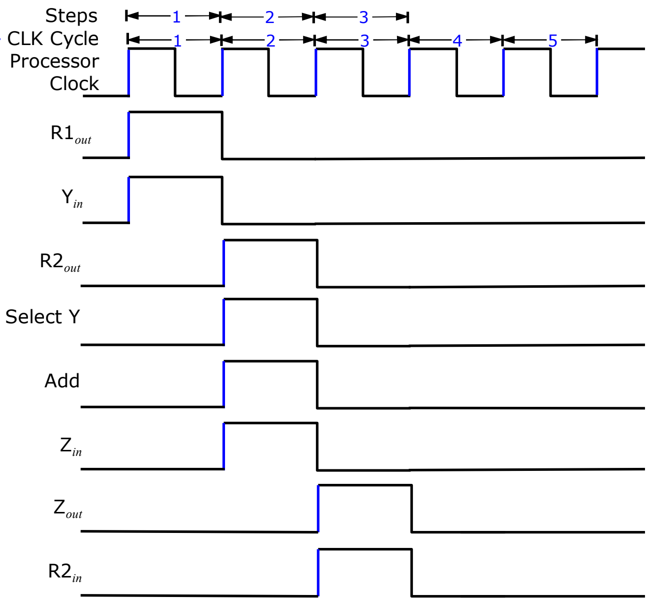
1. $R1_{out}$, Y_{in}
2. $R2_{out}$, Select Y, Add, Z_{in}
3. Z_{out} , $R2_{in}$



Timing of Arithmetic Operation

- Example: ADD R2, R1
- Control Sequence:

1. **$R1_{out}$, Y_{in}**
2. **$R2_{out}$, Select Y, Add, Z_{in}**
3. **Z_{out} , $R2_{in}$**



11

11

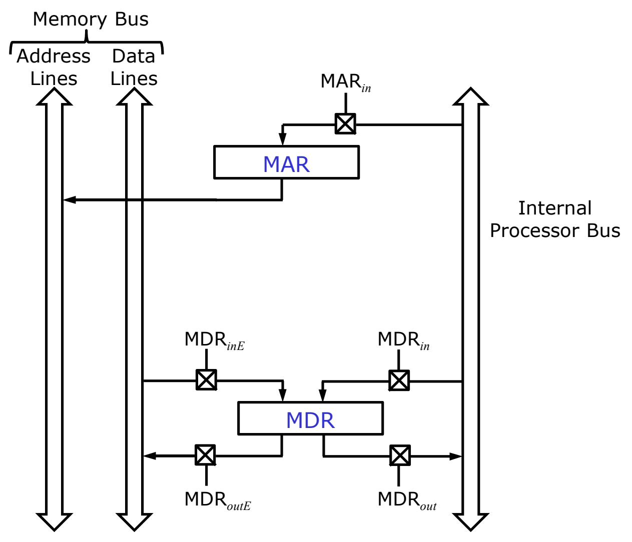
Fetching a Word from Memory

- To fetch a word of information from memory,
  - The processor has to specify the address of memory location where the information is stored
  - Request a read operation
- This applies whether information to be fetched is an instruction of program or an operand specified by the instruction
- Processor transfer address to MAR, whose output connected to address lines
  - At the same time processor uses control lines of memory bus to indicate Read operation
- Requested data received are stored in MDR
  - From MDR, the data is transferred to processor register

12

12

Fetching a Word from Memory



13

13

Fetching a Word from Memory

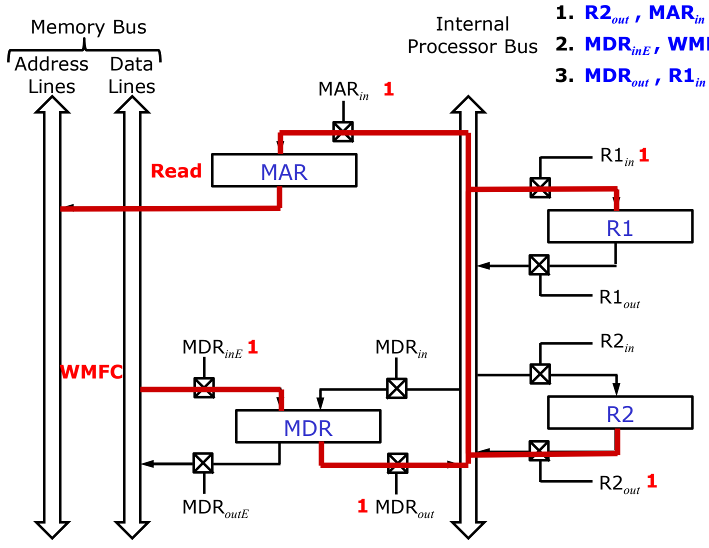
- Example: MOV R1, [R2] $\longleftrightarrow$ $\boxed{\text{MOV R1, R2}}$
  - Transfer the content of location addressed by R2 to R1
- Set of operations involved in executing this instruction
  1. MAR ← [R2]
  2. Start Read operation on memory bus
  3. Wait for memory function complete (MFC) signal from memory
     - It is an indication that data is in memory bus
  4. Load MDR from memory bus
  5. R1 ← [MDR]
- Each action is completed in one clock cycle, except the third operation
- Control sequence:
  1. **$R2_{out}$, MAR_{in} , Read**
  2. **MDR_{inE} , WMFC**
  3. **MDR_{out} , $R2_{in}$**

14

14

Fetching a Word from Memory

- Example: MOV R1, [R2]
- Control sequence:
  1. **R2$_{out}$, MAR$_{in}$, Read**
  2. **MDR$_{inE}$, WMFC**
  3. **MDR$_{out}$, R1$_{in}$**



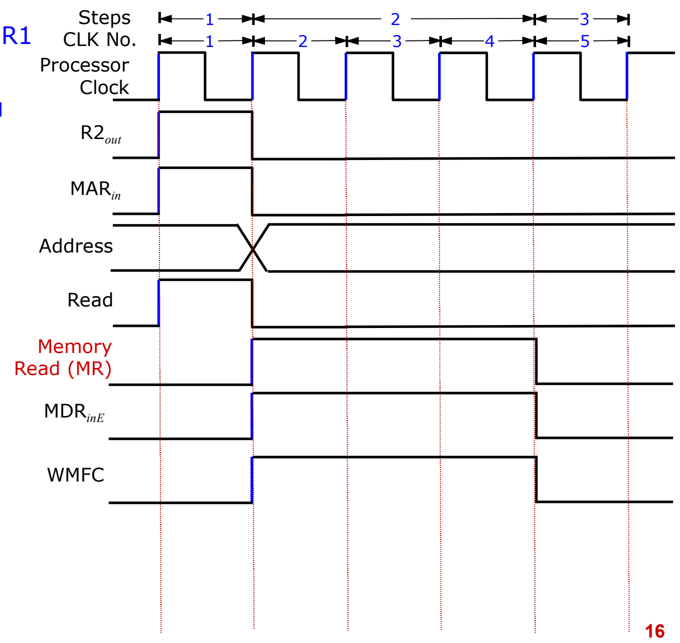
15

15

Timing of Fetching Word from Memory

- Example: ADD R2, R1
- Control Sequence:
  1. **R2$_{out}$, MAR$_{in}$, Read**
  2. **MDR$_{inE}$, WMFC**
  3. **MDR$_{out}$, R1$_{in}$**

Control signal generated in memory unit:



16

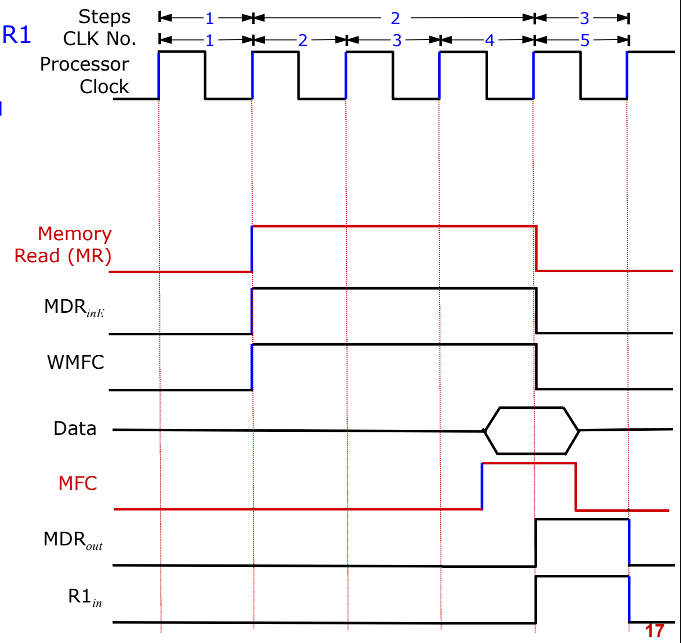
16

Timing of Fetching Word from Memory

- Example: ADD R2, R1
- Control Sequence:
  1. $R2_{out}$, MAR_{in} , Read
  2. MDR_{inE} , WMFC
  3. MDR_{out} , $R1_{in}$

Control signal generated in memory unit:

Control signal generated in memory unit:



17

17

Storing a Word into Memory

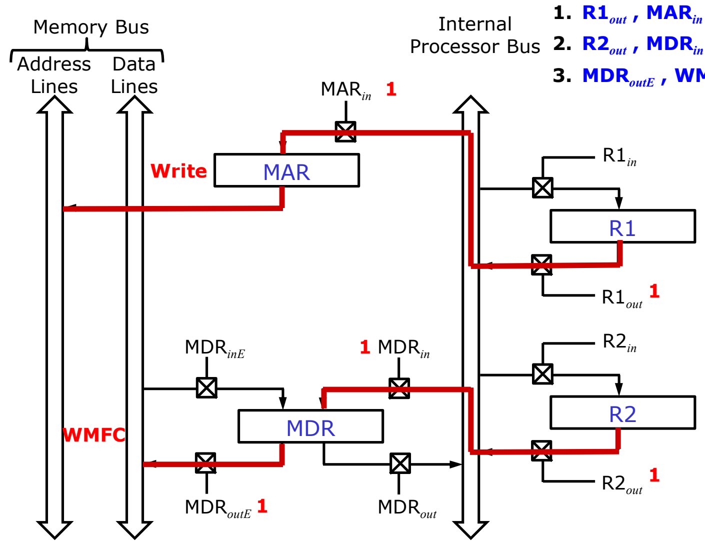
- Address is loaded into MAR
- Data to be written are loaded into MDR
- Example: MOV [R1], R2
  - Transfer/Store the content of R2 to the location addressed by R1
- Set of operations involved in executing this instruction
  1. MAR ← [R1]
  2. Load MDR from R2
  3. Start Write operation on memory bus
  4. Keep the data from MDR onto memory bus
  5. Wait for memory function complete (MFC) signal from memory
- Control sequence:
  1. $R1_{out}$, MAR_{in}
  2. $R2_{out}$, MDR_{in} , Write
  3. MDR_{outE} , WMFC

18

18

Storing a Word into Memory

- Example: MOV [R1], R2
- Control sequence:
  1. $R1_{out}$, MAR_{in}
  2. $R2_{out}$, MDR_{in} , Write
  3. MDR_{outE} , WMFC



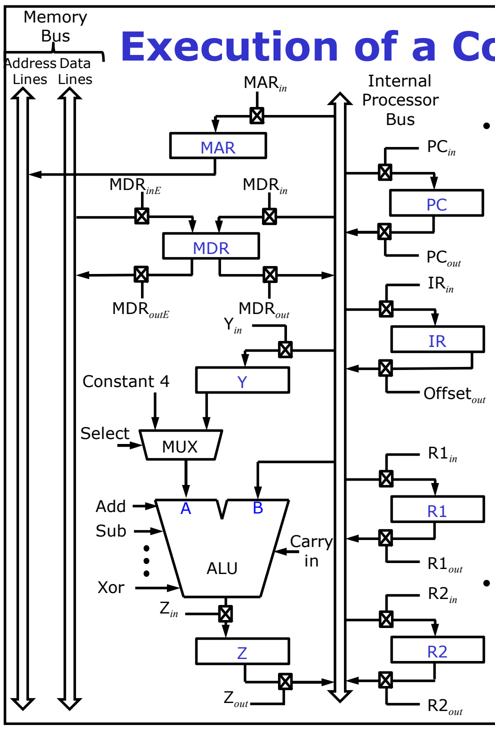
19

Execution of a Complete Instruction

- Example: ADD R1, [R2]
  - Add the content of a memory location pointed by R2 to the register R1 and store the result in R1
- Actions involved in execution:
  - Fetch the instruction
  - Fetch the first operand (content of the memory location pointed to by R2)
  - Perform the addition
  - Load the result on to R1

20

Execution of a Complete Instruction



- Control sequence for execution of ADD R1, [R2]:

1. PC_{out} , MAR_{in} , Read, Select 4, Add, Z_{in}
2. Z_{out} , PC_{in} , Y_{in} , MDR_{inE} , WMFC
3. MDR_{out} , IR_{in}

4. $R2_{out}$, MAR_{in} , Read
5. $R1_{out}$, Y_{in} , MDR_{inE} , WMFC
6. MDR_{out} , Select Y, Add, Z_{in}
7. Z_{out} , $R1_{in}$, End

- Y_{in} in the instruction fetch phase is used for speed up during branch instruction

21

21

Branch Instruction Execution

- Branch instruction replaces the contents of the PC with the branch target address

R0 > N, R0 == N

EQ	Z	LT	GT	LE	GE

```
 000010:              SUB R0, R0, R0
→000011: Loop_Begin:  CMP R0, N
 000012:              JEQ Loop_End ✓
→000013:              ADD A[R0], B[R0], C[R0]
 000014:              INC R0
→000015:              JMP Loop_Begin ✓
→000016: Loop_End:
```

PC ← 000011

- Branch target address is obtained by adding an offset given in branch instruction to the updated value of PC

OPCODE	TARGET

00010	000011

X

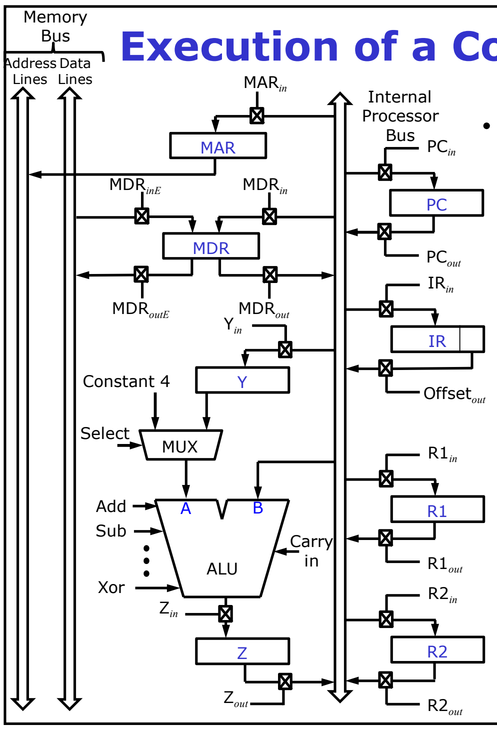
00010	offset -

Present value of PC ⇔ Target.

22

22

Execution of a Complete Instruction



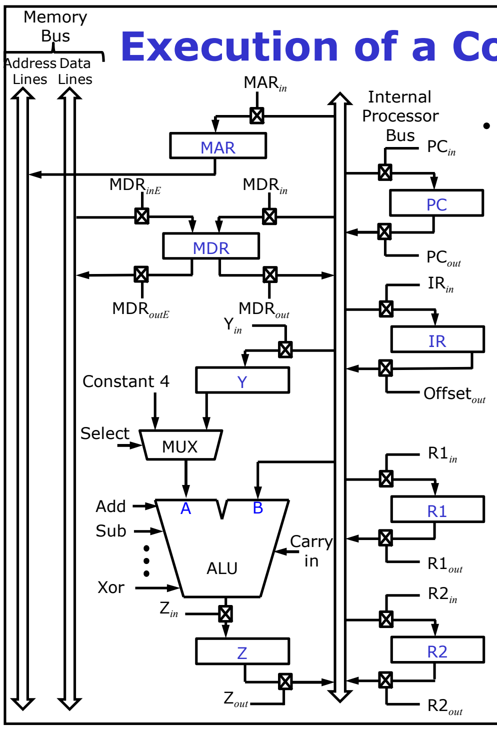
- Control sequence for execution of JMP Loop_Begin:

1. **PC$_{out}$, MAR$_{in}$, Read, Select 4, Add, Z$_{in}$**
2. **Z$_{out}$, PC$_{in}$, Y$_{in}$, MDR$_{inE}$, WMFC**
3. **MDR$_{out}$, IR$_{in}$**
4. **Offset_field_of_IR$_{out}$, Select Y, Add, Z$_{in}$**
5. **Z$_{out}$, PC$_{in}$, End**

23

23

Execution of a Complete Instruction



- Control sequence for execution of conditional branch - JN Loop_Begin (branch on negative):

1. **PC$_{out}$, MAR$_{in}$, Read, Select 4, Add, Z$_{in}$**
2. **Z$_{out}$, PC$_{in}$, Y$_{in}$, MDR$_{inE}$, WMFC**
3. **MDR$_{out}$, IR$_{in}$**
4. **Offset_field_of_IR$_{out}$, Select Y, Add, Z$_{in}$, If N=0 then End**
5. **Z$_{out}$, PC$_{in}$, End**

24

24