# lMicroprocessors Project 1: RISC Design

The microprocessor designed IITB – RISC is an 8 – register, 16 bit system. It has eight general purpose registers (R0 – R7). R7 acts as the program counter (PC). All addresses are short word addresses. This architecture uses two conditional flags, carry flag and zero flag. It uses point-to-point communication.

There are three machine – code instruction formats (R, I and J type) and a total of 14 instructions.

## Flowcharts and Control Signals

1. **ADD /ADC /ADZ**

1

4

3

2

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

**2) ADI**

5

4

2

1

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

**3) NDU / NDC / NDZ**

3

2

1

4

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

**4) LOAD HIGHER**

6

1

|  |  |
| --- | --- |
|  |  |
|  |  |

**5) LOAD**

8

9

7

2

1

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**6) STORE**

10

2

7

1

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

**BRANCH ON EQUALITY**

11

3

2

1

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
| \*\* | , add, |

**8) JUMP AND LINK**

12

1

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | , add, |

11

**9) JUMP AND LINK TO REGISTER**

13

12

2

1

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  | , add, |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | , |
|  | , |

**10) LOAD MULTIPLE**

1

2

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | , |
|  | , |

**12) STORE MULTIPLE**

1

2

# Datapath

# Datapath Components

1. REGISTER
2. REGISTER BANK
3. MEMORY
4. ALU
5. ADDER
6. SUBTRACTOR
7. NAND LOGIC CIRCUIT
8. PRIORITY ENCODER
9. MULTIPLEXERS
10. GENERIC MUX
11. PROGRAM COUNTER (P.C.)
12. INSTRUCTION REGISTER
13. LOAD HIGHER
14. ALU\_REG\_OUT
15. SIGN EXTENDERS
16. 6 TO 16 BIT
17. 9 TO 16 BIT

DEFAULT VALUE OF SIGNALS

Pc\_reg\_ctrl = 0  
Store\_ctrl = 1  
Load\_ctrl = 1  
address\_ctrl = 01  
WREN =  
RDEN =   
counter\_enable = 0  
ir\_ctrl = 0  
mem\_data\_ctrl = 0  
reg\_data\_ctrl = 00  
reg\_sel\_ctrl = 01  
regWrite =   
R7\_select =   
mem\_data\_in\_mux\_ctrl = 0  
reg\_B\_ctrl = 0  
Reg\_A\_sel = 0  
Reg\_A\_ctrl = 0  
alu\_a\_sel = 01 -- input from reg A  
alu\_b\_sel= 00 -- input from reg B  
ctrl\_FSM =   
alu\_reg\_ctrl = 0  
PC\_source\_control=0