EE309

Microprocessor

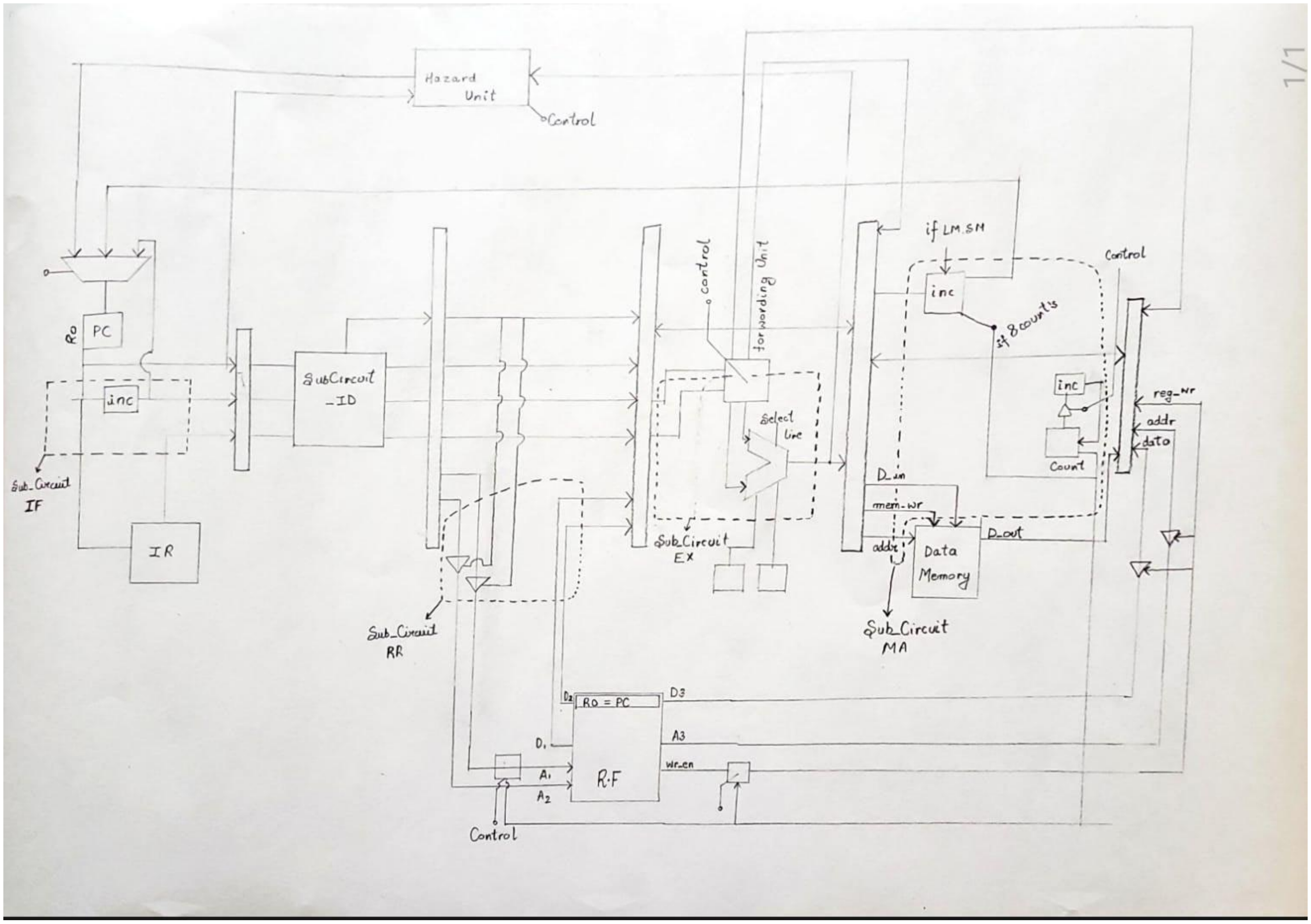
Pipelined Processor

Team Members:

Shounak Das	21D070068
Aditya Anand	21D070007
Animesh Kumar	21D070012
Abhijat Bharadwaj	210020002

Table Of Contents:

Datapath



Hazard Unit
Control
PC
R0
inc
IR
Sub_Circuit IF
SubCircuit _ID
Sub_Circuit RR
control
forwarding Unit
Select line
Sub_Circuit EX
if LM.SM
inc
if 8 count's
Control
inc
Count
reg_wr
addr
data
D_in
mem_wr
addr
D_out
Data Memory
Sub_Circuit MA
D2
R0 = PC
D3
D1
A3
A1
A2
R.F
wr_en
Control

Flow sheet of ideas

			FlowChart IITB-RISC-23				
		IF	ID	RR	EX	MA	WB
ADA	R	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-3) => addr_Rc IR(2)=> complementBit IR(1-0)=>CZ	addr_Ra => RF_A1 addr_Rb => RF_A2	RF_D1 =>alu_a RF_D2 => alu_b alu_out => data_out alu_cout => C alu_z=>Z	M_wr=0	addr_Rc => RF_A3 data_out => RF_D3
ADC	R	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-3) => addr_Rc IR(2)=> complementBit IR(1-0)=>CZ	addr_Ra => RF_A1 addr_Rb => RF_A2	RF_D1 =>alu_a RF_D2 => alu_b alu_out => data_out alu_cout => C alu_z=>Z	M_wr=0	addr_Rc => RF_A3 data_out => RF_D3
ADZ	R	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-3) => addr_Rc IR(2)=> complementBit IR(1-0)=>CZ	addr_Ra => RF_A1 addr_Rb => RF_A2	RF_D1 =>alu_a RF_D2 => alu_b alu_out => data_out alu_cout => C alu_z=>Z	M_wr=0	addr_Rc => RF_A3 data_out => RF_D3
AWC	R	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-3) => addr_Rc IR(2)=> complementBit IR(1-0)=>CZ	addr_Ra => RF_A1 addr_Rb => RF_A2	RF_D1 =>alu_a RF_D2 => alu_b C=> alu_cin alu_out=>data_out alu_cout=>C alu_z => Z	M_wr=0	addr_Rc => RF_A3 data_out => RF_D3
ACA	R	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-3) => addr_Rc IR(2)=> complementBit IR(1-0)=>CZ	addr_Ra => RF_A1 addr_Rb => RF_A2	RF_D1 =>alu_a not(RF_D2) => alu_b alu_out => data_out alu_cout => C alu_z=>Z	M_wr=0	addr_Rc => RF_A3 data_out => RF_D3
ACC	R	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-3) => addr_Rc IR(2)=> complementBit IR(1-0)=>CZ	addr_Ra => RF_A1 addr_Rb => RF_A2	RF_D1 =>alu_a not(RF_D2) => alu_b alu_out => data_out alu_cout => C alu_z=>Z	M_wr=0	addr_Rc => RF_A3 data_out => RF_D3
ACZ	R	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-3) => addr_Rc IR(2)=> complementBit IR(1-0)=>CZ	addr_Ra => RF_A1 addr_Rb => RF_A2	RF_D1 =>alu_a not(RF_D2) => alu_b alu_out => data_out alu_cout => C alu_z=>Z	M_wr=0	addr_Rc => RF_A3 data_out => RF_D3
ACW	R	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-3) => addr_Rc IR(2)=> complementBit IR(1-0)=>CZ	addr_Ra => RF_A1 addr_Rb => RF_A2	RF_D1 =>alu_a not(RF_D2) => alu_b C=> alu_cin alu_out=>data_out alu_cout=>C alu_z => Z	M_wr=0	addr_Rc => RF_A3 data_out => RF_D3
ADI	I	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-0) => imm6	addr_Ra => RF_A1	RF_A1=>alu_a imm6=>alu_b alu_out=>data_out alu_cout => C alu_z=>Z	M_wr=0	addr_Rb => RF_A3 data_out => RF_D3
NDU	R	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-3) => addr_Rc IR(2)=> complementBit IR(1-0)=>CZ	addr_Ra => RF_A1 addr_Rb => RF_A2	RF_A1=>alu_a RF_A2=>alu_b alu_out=>data_out alu_cout => C alu_z=>Z	M_wr=0	addr_Rc => RF_A3 data_out => RF_D3

			FlowChart IITB-RISC-23				
		IF	ID	RR	EX	MA	WB
NDC	R	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-3) => addr_Rc IR(2)=> complementBit IR(1-0)=>CZ	addr_Ra => RF_A1 addr_Rb => RF_A2	RF_A1=>alu_a RF_A2=>alu_b alu_out=>data_out alu_cout => C alu_z=>Z	M_wr=0	addr_Rc => RF_A3 data_out => RF_D3
NDZ	R	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-3) => addr_Rc IR(2)=> complementBit IR(1-0)=>CZ	addr_Ra => RF_A1 addr_Rb => RF_A2	RF_A1=>alu_a RF_A2=>alu_b alu_out=>data_out alu_cout => C alu_z=>Z	M_wr=0	addr_Rc => RF_A3 data_out => RF_D3
NCU	R	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-3) => addr_Rc IR(2)=> complementBit IR(1-0)=>CZ	addr_Ra => RF_A1 addr_Rb => RF_A2	RF_A1=>alu_a not(RF_A2)=>alu_b alu_out=>data_out alu_cout => C alu_z=>Z	M_wr=0	addr_Rc => RF_A3 data_out => RF_D3
NCC	R	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-3) => addr_Rc IR(2)=> complementBit IR(1-0)=>CZ	addr_Ra => RF_A1 addr_Rb => RF_A2	RF_A1=>alu_a not(RF_A2)=>alu_b alu_out=>data_out alu_cout => C alu_z=>Z	M_wr=0	addr_Rc => RF_A3 data_out => RF_D3
NCZ	R	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-3) => addr_Rc IR(2)=> complementBit IR(1-0)=>CZ	addr_Ra => RF_A1 addr_Rb => RF_A2	RF_A1=>alu_a not(RF_A2)=>alu_b alu_out=>data_out alu_cout => C alu_z=>Z	M_wr=0	addr_Rc => RF_A3 data_out => RF_D3
LLI	J	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-0) => imm9		imm9=>Ra(lower 9bit) 00_0=>Ra(higher 7bit)	M_wr=0	
LW	I	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-0) => imm6	addr_Rb => RF_A2	RF_D2=>alu_a imm6=>SE6=>alu_b alu_out=>data_out	data_out=>DMem_a	addr_Ra=> RF_A3 DMem_d => RF_D3
SW	I	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-0) => imm6	addr_Rb => RF_A2	imm6=>SE6=> alu_a RF_D2 => alu_b alu_out=>data_out	data_out=>DMem_a	addr_Ra=> RF_A1 RF_D1=>DMem_d

LM	J	pc=>inc,IMem_A
		inc=>PC
		IMem_D=>IR
			IR(15-12) => opCode
			IR(11-9) => addr_Ra
			IR(8-0) => imm9
				add_Rb=>RF_A1
				RF_D1=>TR_D
						i=0
						while(i<8)
						if IR(i)=1 then
						(i)=>RF_A3
						TR_D+i=>DMem_a_
						DMem_dout=>RF_D3
						i++	

SM	J	pc=>inc,IMem_A inc=>PC IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-0) => imm9	addr_Ra => RF_A1 RF_D1=> TR_D i=0 while (i<8) if IR(i)=1 i=>RF_A2 TR_D+i=>data_out		data_out=>DMem_a	RF_D2=>DMem_d
BEQ	I	pc=>inc,IMem_A inc=>pc IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-0) => imm6	addr_Ra => RF_A1 addr_Rb => RF_A2	if reg A=reg B imm6 => alu1_a 2 => alu1_b alu1_out => alu2_a PC => alu2_b alu2_out=>PC	M_wr=0	NONE

		IF	ID	RR	EX	MA	WB
BLT	I	pc=>inc,IMem_A inc=>pc IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-0) => imm6	addr_Ra => RF_A1 addr_Rb => RF_A2	if reg A<reg B imm6 => alu1_a 2 => alu1_b alu1_out => alu2_a PC => alu2_b alu2_out=>PC	M_wr=0	NONE
BLE	I	pc=>inc,IMem_A inc=>pc IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Rb IR(5-0) => imm6	addr_Ra => RF_A1 addr_Rb => RF_A2	if reg A(</=)reg B imm6 => alu1_a 2 => alu1_b alu1_out => alu2_a PC => alu2_b alu2_out=>PC	M_wr=0	NONE
JAL	J	pc=>inc,IMem_A inc=>pc,pc_init IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-0) => imm9	addr_Ra => RF_A1	imm9 => alu1_a 2 => alu1_b alu1_out => alu2_a PC => alu2_b alu2_out=>pc	M_wr=0	addr_Ra=>RF_A3 pc_init => RF_D3
JLR	I	pc=>inc,IMem_A inc=>pc,pc_init IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-6) => addr_Ra IR(5-0)=>000000	addr_Ra => RF_A1	pc=>alu_a 2=>alu_b alu_out=>pc	M_wr=0	addr_Ra=>RF_A3 pc_init => RF_D3
JRI	J	pc=>inc,IMem_A inc=>pc IMem_D=>IR	IR(15-12) => opCode IR(11-9) => addr_Ra IR(8-0) => imm9	addr_Ra => RF_A1	imm9 => alu1_a 2 => alu1_b alu1_out => alu2_a RF_A1 => alu2_b alu2_out=>pc	M_wr=0	NONE

Component Documentation

master_register

This is a variable bit register, we can set the value of resize to change the number of bits

```
entity master_reg is

generic (

        regsiZe : integer := 16;
    );

    port(
  clock,reset,wr: in std_logic;
  inp: in std_logic (regsiZe-1 downto 0);
  outp: out std_logic (regsiZe-1 downto 0)
  );
end master_reg;
```

SubCircuit_IF

This is used to fetch instruction from the instruction memory according to the adress in program counter, it updates the pc_write only if the istructions are not BEQ,BLT,BLE,JAL,LRI,LM,SM

```
entity subCircuit_IF is
 port(
  clk,reset: in std_logic;
  pc_read: in std_logic_vector(15 downto 0);
  pc_write: out std_logic_vector(15 downto 0);
  pc_wr:out std_logic;
  IR: out std_logic_vector(15 downto 0)
 );
end subCircuit_IF;
```

SubCircuit_ID

This decodes the instruction according to the op code and modifies z_write,c_write, ID_Mem_Write accordingl

```
entity subCircuit_ID is
 port(

  clk,reset : in std_logic;
  instr_in : in std_logic_vector(15 downto 0);
  reg_write: OUT STD_LOGIC;
          reg_write_add: OUT STD_LOGIC_VECTOR(2 downto 0);
      reg_read_1: OUT STD_LOGIC;
      reg_read_2: OUT STD_LOGIC;
      read_c: OUT STD_LOGIC;
      read_z: OUT STD_LOGIC;
          z_write: OUT STD_LOGIC;

          c_write: OUT STD_LOGIC;
     ID_Mem_Write: OUT STD_LOGIC
 );
end subCircuit_ID;
```

SubCircuit_RR

Read data from the register file according to the selected address

```
entity subCircuit_RR is
  port ( instr: in std_logic_vector(15 downto 0);
   clock,mem_wr_in: in std_logic;
   mem_wr_out: out std_logic;
   reg_read_1: in STD_LOGIC;
   reg_read_2: in STD_LOGIC;
   rf_a1,rf_a2:out std_logic_vector(2 downto 0);
   rf_d1,rf_d2:in std_logic_vector(15 downto 0);
     data_reg1, data_reg2 :out std_logic_vector(15 downto 0)
    );

end subCircuit_RR;
```

SubCircuit_EX

This is used whenever an ALU operation is needed

```
entity subCircuit_EX is
  port ( instr: in std_logic_vector(15 downto 0);
     data_reg1, data_reg2,pc_in :in std_logic_vector(15 downto 0);
   imm_6:in std_logic_vector(5 down to 0);
   imm_9:in std_logic_vector(8 down to 0);
         clk :in std_logic;
   c,z:in std_logic_vector;
   c_out,z_out : out std_logic_vector;
   ex_out,pc_out:out std_logic_vector(15 downto 0)

    );
```

SubCircuit_MA

This is used for the instruction which require data memory access LW, SW LM, SM.

```
entity subCircuit_MA is
  port ( instr,addr, d_in : in std_logic_vector(15 downto 0);
   clk,reset,mem_wr : in std_logic;
   d_out : out std_logic_vector(15 downto 0)

    );

end subCircuit_MA;
```

SubCircuit_WB

The final values are written back to the register file accordingly

```
entity subCircuit_WB is
 port (
      clk,reset : in std_logic;
  reg_write: in std_logic;

  D3_in : out std_logic_vector(15 downto 0);
  reg_write_add: OUT STD_LOGIC_VECTOR(2 downto 0);
    A3 : out std_logic_vector(2 downto 0);
  D3_out : out std_logic_vector(15 downto 0)

   );
 end entity;
```