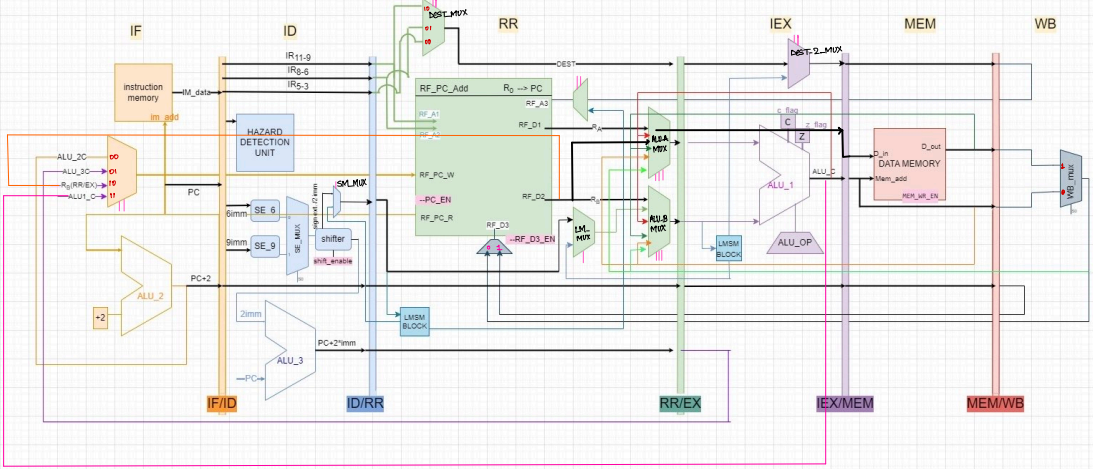
IIT B RISC PIPELINED

Github repository: https://github.com/shambhavii13/IITB-RISC-23

Deep Boliya
Shambhavi Shanker
Swadhin Dash
Scaria K.

instructions:

ADD/NAND	0001/0010	RA + RB ⟶ RC (Dest)
ADI	0000	RA + 6imm ⟶ RB (")
LLI	0011	0 + 9imm ⟶ RA (")
LW	0100	M (RB + 6imm) ⟶ RA (Dest)
SW	0101	M (RB + 6imm) ⟵ RA (Din)
BEQ	1000	if Branch PC' = PC + 2imm
BLT	1001	
BLE	1010	
JAL	1100	: Dest RA (PC+2) ; PC + 2imm
JLR	1101	: Dest RA (PC + 2) ; RB
JRI	1111	: to RA + 2imm


IF
ID
RR
IEX
MEM
WB
instruction memory
IM_data
im_add
HAZARD DETECTION UNIT
IR11-9
IR8-6
IR5-3
DEST_MUX
RF_PC_Add
R0 --> PC
RF_A3
RF_D1
RF_A1
RF_A2
RF_PC_W
--PC_EN
RF_PC_R
RF_D2
RF_D3
--RF_D3_EN
DEST
RA
RB
ALU_A MUX
ALU_B MUX
LM_MUX
DEST_2_MUX
c_flag
z_flag
C
Z
ALU_1
ALU_OP
ALU_C
D_in
D_out
DATA MEMORY
Mem_add
MEM_WR_EN
WB_mux
ALU_2C
ALU_3C
R0(RR/EX)
ALU1_C
PC
SM_MUX
6imm
9imm
SE_6
SE_9
SE_MUX
signext_2 imm
shifter
shift_enable
ALU_2
+2
PC+2
2imm
ALU_3
PC+2*imm
PC
LMSM BLOCK
IF/ID
ID/RR
RR/EX
IEX/MEM
MEM/WB

pipeline Registers

IF/ID

input : im_data
PC
PC+2

output : IR_{11-9} (RA_address)
IR_{8-6} (RB_address)
IR_{5-3} (RC_address)
IR_{5-0} (6_imm)
IR_{8-0} (9_imm)
PC, PC+2 , im_data_out

ID/RR

input : IR_{11-9} (RA_address)
IR_{8-6} (RB_address)
IR_{5-3} (RC_address)
sc imm or 2imm from SM mux
PC+2
PC+2imm , ix_in

output : IR_{11-9} (RA_address)
IR_{8-6} (RB_address)
IR_{5-3} (RC_address)
SM_mux_out
PC+2 , PC+2imm , ix_out

RR/EX :

i/p : from Dest mux
from ALU_A_mux
from ALU-B_mux
RA
PC+2 , PC+2imm , ix_in

o/p : Dest
alu-a-mux
alu-b_mux
RA
PC+2, PC+2imm , ix_out

EX/MEM

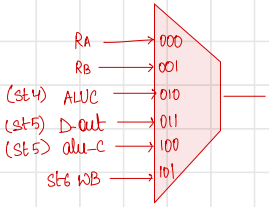
i/p : from dest_2mux
RA
alu-c
PC+2 , ix-in

output : dest_2mux
RA (Din)
alu-c (mem_add)
PC+2 , ix_out

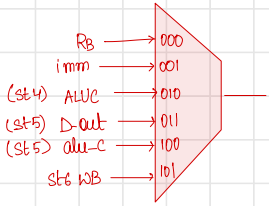
MEM/WB

input : Dest
D_out
mem_add (alu_c)
PC+2 , ix-in

output : Dest
D_out
PC+2
alu-c (mem_add), ix_out



alu_mux_A

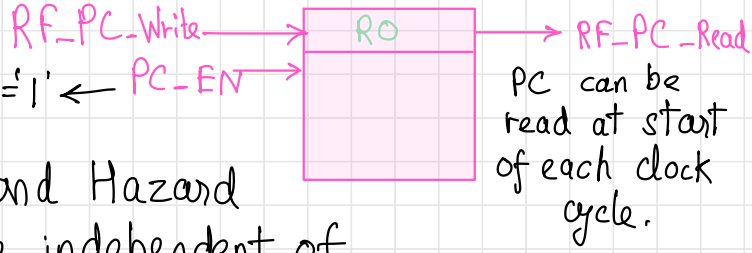


alu_mux_B

Our Modifications and Assumptions

→ Register File has 3 more ports

PC can be written using this port if PC_EN='1'



→ The controller and Hazard Detection unit are independent of the pipeline i.e they effect all pipelines at once.

→ Since the pointer counter (PC) is stored in R0 in the register file, this might lead to branching in other instructions such as ADA if R0 is the destination, hence causing hazards.
We leave it upto the programmers decision

→ We have used an LMSM block for the instruction LM-SM, whose working is explained later.

0001 { ADD, ADC, ADZ, ACA, ACC, ACZ (0000 0000 0000 0010 0010 0010); AWC, ACW (0001 0001) }

alu_op

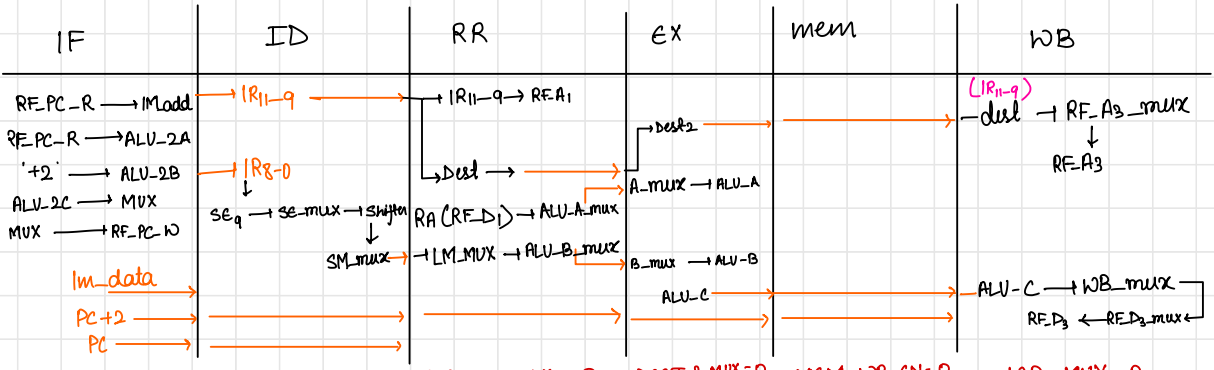
NDU, NDC, NDZ (0100); NCU, NCC, NCZ (0101) } 0010

IF	ID	RR	EX	mem	WB
RF_PC_R → IMaddr RF_PC_R → ALU_2A '+2' → ALU_2B ALU_2C → MUX MUX → RF_PC_W Im_data PC+2 PC	IR11-9 IR8-6 IR5-3	IR11-9 → RF_A1 IR8-6 → RF_A2 → Dest → RA (RF_D1) → ALU_A_mux RB (RF_D2) → ALU_B_mux	Dest2 A_mux → ALU_A B_mux → ALU_B ALU_C		(IR5-3) dest → RF_A3_mux → RF_A3 ALU_C → WB_mux RF_D3 ← RF_D3_mux
MUX: 00 PC_EN=1	SH_EN=0 SM_MUX=0 SE_MUX=0	DEST_MUX=00 LM_MUX=0 ALU_A_MUX=000 ALU_B_MUX=000	DEST_2_MUX=0 ALU_C flag = 1 ALU_2 flag = 1	MEM_WR_EN=0	WB_MUX=0 RF_D3_MUX=0 RF_D3_EN=1 RF_A3_MUX=0

0000 ADI ALU_op: 0000 (add) (Ra + Imm6) → Rb

IF	ID	RR	EX	mem	WB
RF_PC_R → IMaddr RF_PC_R → ALU_2A '+2' → ALU_2B ALU_2C → MUX MUX → RF_PC_W Im_data PC+2 PC	IR11-9 IR8-6 IR5-0 SE6 → se_mux → shifter → SM_mux	IR11-9 → RF_A1 → Dest → RA (RF_D1) → ALU_A_mux → LM_MUX → ALU_B_mux	Dest2 A_mux → ALU_A B_mux → ALU_B ALU_C		(IR8-6) dest → RF_A3_mux → RF_A3 ALU_C → WB_mux RF_D3 ← RF_D3_mux
MUX: 00 PC_EN=1	SH_EN=0 SM_MUX=0 SE_MUX=0	DEST_MUX=01 LM_MUX=0 ALU_A_MUX=000 ALU_B_MUX=001	DEST_2_MUX=0 ALU_C flag = 1 ALU_2 flag = 1	MEM_WR_EN=0	WB_MUX=0 RF_D3_MUX=0 RF_D3_EN=1 RF_A3_mux=0

0011 LLI ALU_op : 1001 (LLI) : just returns ALU_B LLI ra, Imm9



MUX : 00
PC_EN=1

SH_EN=0
SM_MUX =0
SE_MUX = 1

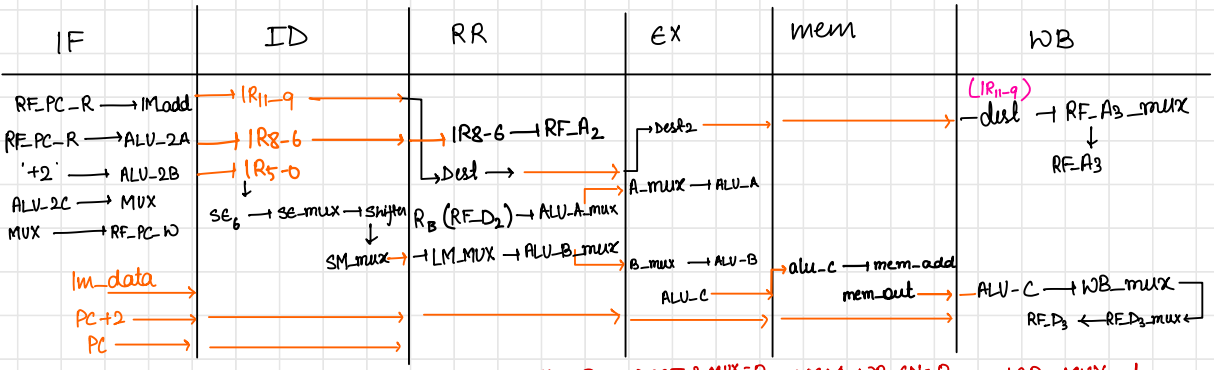
DEST_MUX=10
LM_MUX =0
ALU_A_MUX = 000
ALU_B_MUX=001

DEST_2_MUX=0
ALU_C_flag = 0
ALU_2_flag = 0

MEM_WR_EN=0

WB_MUX = 0
RF_D3_MUX = 0
RF_D3_EN= 1
RF_A3_mux=0

0100 LW alu_op : 0000 (add) M(Rb + Imm6) → Ra



MUX : 00
PC_EN=1

SH_EN=0
SM_MUX =0
SE_MUX = 0

DEST_MUX=10
LM_MUX =0
ALU_A_MUX = 001
ALU_B_MUX=001

DEST_2_MUX=0
ALU_C_flag = 0
ALU_2_flag = 0

MEM_WR_EN=0

WB_MUX = 1
RF_D3_MUX = 0
RF_D3_EN= 1
RF_A3_mux = 0

SW 0101 alu_op = add (0000) SW ra, rb, 9mm

IF	ID	RR	EX	mem	WB
RF_PC_R → IMadd RF_PC_R → ALU_2A '+2' → ALU_2B ALU_2C → MUX MUX → RF_PC_W Im_data PC+2 PC	IR11-9 IR8-6 IR5-0 SE_6 → se_mux → shifter SM_mux	IR11-9 → RF_A1 IR8-6 → RF_A2 RA(RF_D1) R_B (RF_D2) → ALU_A_mux LM_MUX → ALU_B_mux	A_mux → ALU_A B_mux → ALU_B ALU_C	RA → Din alu_c → mem_add	
MUX: 00 PC_EN=	SH_EN=0 SM_MUX=0 SE_MUX=0	DEST_MUX=00 LM_MUX=0 ALU_A_MUX=001 ALU_B_MUX=001	DEST_2_MUX=0 ALU_C flag=0 ALU_2 flag=0	MEM_WR_EN=1	WB_MUX=0 RF_D3_MUX=0 RF_D3_EN=0 RF_A3_mux=0

BEQ 1000 BLT 1001 BLE 1010

alu_op 0110 0111 1000

IF	ID	RR	EX	mem	WB
RF_PC_R → IMadd RF_PC_R → ALU_2A '+2' → ALU_2B ALU_2C → MUX MUX → RF_PC_W Im_data PC+2 PC	IR11-9 IR8-6 IR5-0 SE_6 → se_mux → shifter from ALU3, PC+2imm	IR11-9 → RF_A1 IR8-6 → RF_A2 R_A (RF_D1) → ALU_A_mux RB(RF_D2) → ALU_B_mux	A_mux → ALU_A B_mux → ALU_B ALU_C (sbit) if sbit = 1 : ALU_3C → MUX MUX → RF_PC_W		
MUX: 00 PC_EN=1	SH_EN= 1 SM_MUX=0 SE_MUX=0	DEST_MUX=00 LM_MUX=0 ALU_A_MUX=000 ALU_B_MUX=000	DEST_2_MUX=0 ALU_C flag=0 ALU_2 flag=0 MUX=00 if sbit=0 =01 if sbit=1 PC_EN=1	MEM_WR_EN=0	WB_MUX=0 RF_D3_MUX=0 RF_D3_EN=0 RF_A3_mux=0

JAL 1100

alu-op : nop (1111)

IF	ID	RR	EX	mem	WB
RF_PC_R → IMadd RF_PC_R → ALU_2A '+2' → ALU_2B ALU_2C → MUX MUX → RF_PC_W Im_data PC+2 PC	IR11-9 IR8-0 → SE_9 → se_mux → shifter from ALU3, PC+2imm	Dest	Dest2 ALU_2C → mux mux → RF_PC_W		Dest → RF_A3_mux → RF_A3 PC+2 → RF_D3_mux → RF_{D_3}
MUX : 00 PC_EN=1	SH_EN=1 SM_MUX=0 SE_MUX=1	DEST_MUX=10 LM_MUX=0 ALU_A_MUX=000 ALU_B_MUX=000	DEST_2_MUX=0 ALU_C_flag=0 ALU_2_flag=0 MUX=01 PC_EN=1	MEM_WR_EN=0	WB_MUX=0 RF_D3_MUX=1 RF_D3_EN=1 RF_A3_mux=0

JLR 1101

alu-op : NOP (1111)

jlr ra, rb

IF	ID	RR	EX	mem	WB
RF_PC_R → IMadd RF_PC_R → ALU_2A '+2' → ALU_2B ALU_2C → MUX MUX → RF_PC_W Im_data PC+2 PC	IR11-9 IR8-6	Dest IR8-6 → RF_A2 R_B (RF_D_2) → R_B → mux mux → RF_PC_W	Dest2		Dest → RF_A3_mux → RF_A3 PC+2 → RF_D3_mux → RF_{D_3}
MUX : 00 PC_EN=1	SH_EN=0 SM_MUX=0 SE_MUX=0	DEST_MUX=10 LM_MUX=0 ALU_A_MUX=000 ALU_B_MUX=000 MUX=10 PC_EN=1	DEST_2_MUX=0 ALU_C_flag=0 ALU_2_flag=0	MEM_WR_EN=0	WB_MUX=0 RF_D3_MUX=1 RF_D3_EN=1 RF_A3_mux=0

JRI 1111 alu_op = 0000 (add) jri, ra, Imm9

IF	ID	RR	EX	mem	WB
RF_PC_R → IMadd RF_PC_R → ALU_2A '+2' → ALU_2B ALU_2C → MUX MUX → RF_PC_W Im_data PC+2 PC	IR11-9 IR8-0 Se9 → se_mux → shifter → SM_mux 2imm	IR11-9 → RFA1 RA (RF_D1) → ALU_A_mux 2imm → LM_MUX → ALU_B_mux	Ra → alu1-a 2imm → alu_b ALU_C → mux mux → RF_PC_W		
MUX: 00 PC_EN=1	SH_EN=1 SM_MUX=0 SE_MUX=1	DEST_MUX=00 LM_MUX=0 ALU_A_MUX=000 ALU_B_MUX=001	DEST_2_MUX=0 ALU_C_flag=0 ALU_Z_flag=0 MUX=11 PC_EN=1	MEM_WR_EN=0	WB_MUX=0 RF_D3_MUX=0 RF_D3_EN=0 RF_A3_mux=0

LM and SM

→ We are using an LMSM block that does the following:

Say Imm : 1000 1001 (R0, R4, R7) → LM-SM → 111

This is RF_A3 address

This over-writes the Immediate

1000 1000 (R0, R4, R7)

ra → A, Imm → B, ALU → C

ALU_OP = 1110

Mem-Address (ra + i)

Here i is the position of the first bit set in Immediate from the right

This loop repeats until Immediate equals zero.

LM 0110 alu_op 1110

	IF	ID	RR	EX	mem	WB
1.	I_1					
2.	I_2	I_1				
3.	I_2	I_2	I_1			
4.	I_2	I_2	I_2	I_{17}		
5.	I_2	I_2	I_2	I_{14}	I_{17}	
6.	I_2	I_2	I_2	I_{10}	I_{14}	I_{17}
7.	I_2	I_2	I_2	$I_{imm=0}$	I_{10}	I_{14}
8.	I_2	I_2	I_2	$I_{imm=0}$	$I_{imm=0}$	I_{10}
9.	I_n	I_2	I_2	I_2	$I_{imm=0}$	$I_{imm=0}$

↳ new instruction loaded.

in (2). PC_EN=0, ∴ new instructions dont get loaded

in (4) we stop write enable to RR|EX

in (7), when immediate = 0, we turn off all controls i.e we disable ($I_{imm=0}$)

in (9) we set PC_EN=1 and RR_EX_EN=1

SM 0111 alu_op 1110

	IF	ID	RR	EX	mem	WB
1.	I_1					
2.	I_2	I_1				
3.	I_2	I_2	I_{17}			
4.	I_2	I_2	I_{14}	I_{17}		
5.	I_2	I_2	I_{10}	I_{14}	I_{17}	
6.	I_2	I_2	$I_{imm=0}$	I_{10}	I_{14}	I_{17}
7.	I_3	I_2	$I_{imm=0}$	$I_{imm=0}$	I_{10}	I_{14}
8.	I_4	I_3	I_2	$I_{imm=0}$	$I_{imm=0}$	I_{10}

in (2) PC-EN=0

in (3) ID-RR disabled

in (6) when immediate = 0 ($I_{imm=0}$) gets disabled

PC-EN=1

in (7), ID_RR is enabled again