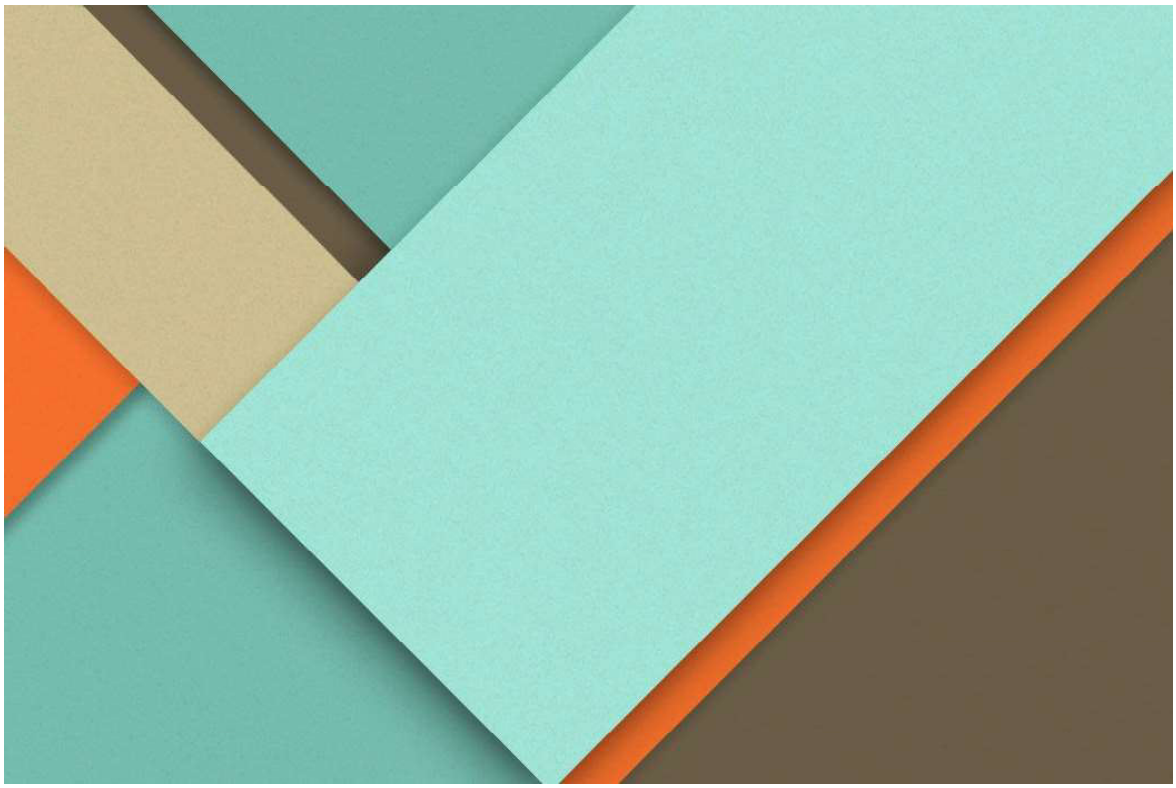


PIPELINED RISC PROCESSOR DESIGN

EE309

ABHISHEK KONALE	14D070010
ASHWIN LELE	14D070013
SRAVAN PATCHALA	14D070012
JAINAM SHAH	14D070014

INDEX

OBJECTIVE:

To design a 6 stage pipelined processor, **IITB-RISC**

IITB-RISC is a 16-bit very simple computer developed for the teaching purpose.

The IITB-RISC is an 8-register,16-bit computer system.

It should follow the standard 6 stage pipelines:

- Instruction fetch
- Instruction decode
- Register read
- Execute
- Memory access
- Write back

The architecture should be optimized for performance, i.e.,should include hazard mitigation techniques.Hence, it should have at least data forwarding mechanism.

IITB-RISC14 Instruction Set Architecture

IITB-RISC is a 16-bit very simple computer developed for the teaching that is based on the Little Computer Architecture. The *IITB-RISC* is an 8-register, 16-bit computer system. It has 8 general-purpose registers (R0 to R7). Register R7 is always stores Program Counter. All addresses are short word addresses (i.e. address 0 corresponds to the first two bytes of main memory, address 1 corresponds to the second two bytes of main memory, etc.). This architecture uses condition code register which has two flags Carry flag (c) and Zero flag (z). The *IITB-RISC* is very simple, but it is general enough to solve complex problems. The architecture allows predicated instruction execution and multiple load and store execution. There are three machine-code instruction formats (R, I, and J type) and a total of 14 instructions. They are illustrated in the figure below.

R Type Instruction format

Opcode (4 bit)	Register A (RA) (3 bit)	Register B (RB) (3-bit)	Register B (RB) (3-bit)	Unused (1 bit)	Condition (CZ) (2 bit)

I Type Instruction format

Opcode (4 bit)	Register A (RA) (3 bit)	Register C (RC) (3-bit)	Immediate (6 bits signed)

J Type Instruction format

Opcode (4 bit)	Register A (RA) (3 bit)	Immediate (9 bits signed)

Instructions Encoding:

ADD:	00_00	RA	RB	RC	0	00
ADC:	00_00	RA	RB	RC	0	10
ADZ:	00_00	RA	RB	RC	0	01
ADI:	00_01	RA	RB	6 bit Immediate		
NDU:	00_10	RA	RB	RC	0	00
NDC:	00_10	RA	RB	RC	0	10
NDZ:	00_10	RA	RB	RC	0	01
LHI:	00_11	RA	9 bit Immediate			
LW:	01_00	RA	RB	6 bit Immediate		
SW:	01_01	RA	RB	6 bit Immediate		
LM:	01_10	RA	0 + 8 bits corresponding to Reg R7 to R0			
SM:	01_11	RA	0 + 8 bits corresponding to Reg R7 to R0			
BEQ:	11_00	RA	RB	6 bit Immediate		
JAL:	10_00	RA	9 bit Immediate offset			
JLR:	10_01	RA	RB	000_000		

RA: Register A

RB: Register B

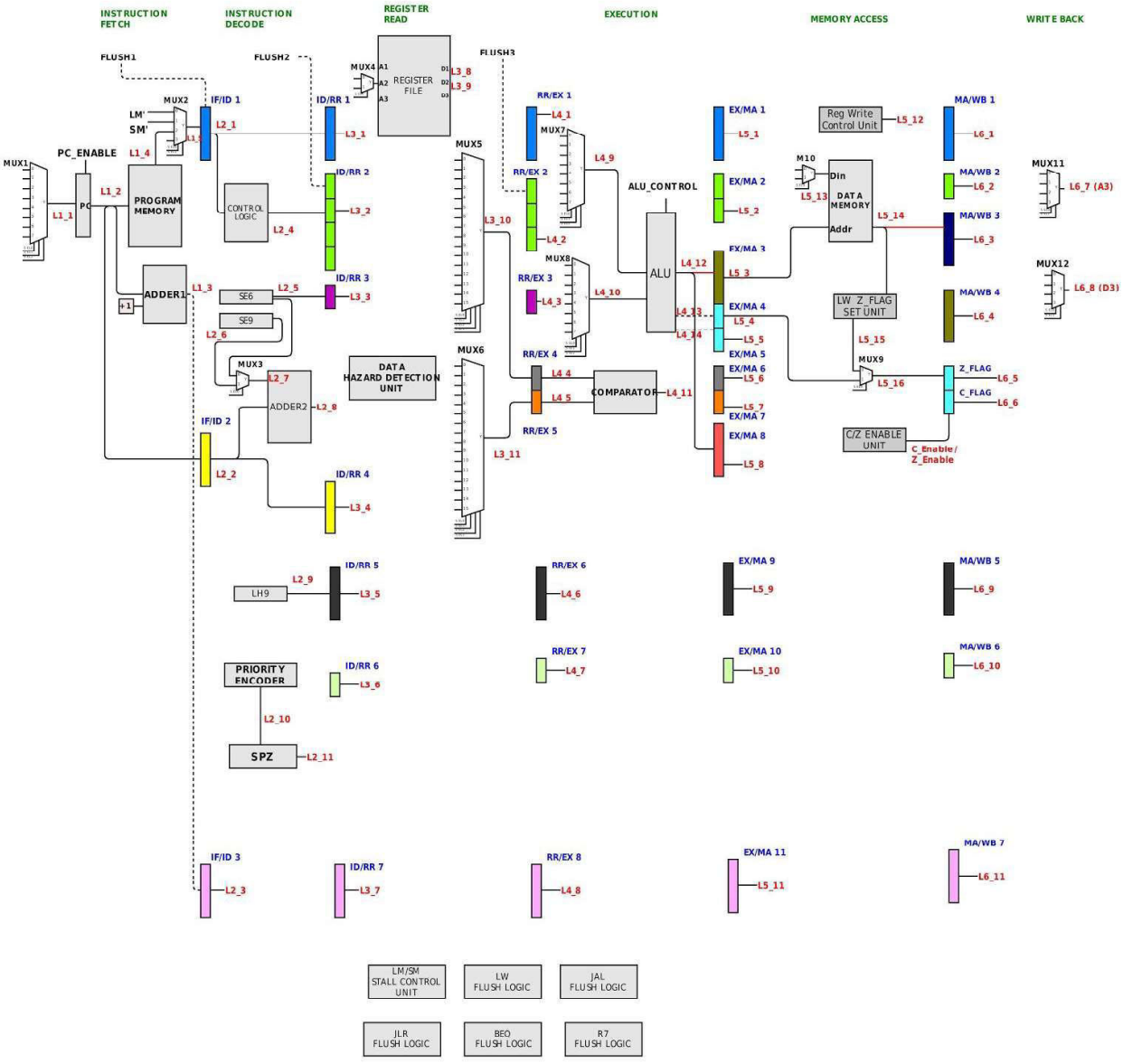
RC: Register C

Instruction Description

Mnemonic	Name & Format	Assembly	Action
ADD	ADD (R)	add rc, ra, rb	Add content of regB to regA and store result in regC. *It set C and Z flags*
ADC	Add if carry set (R)	adc rc, ra, rb	Add content of regB to regA and store result in regC, if carry flaf is set. *It sets C & Z flags*
ADZ	Add if zero set (R)	adz rc, ra, rb	Add content of regB to regA and store result in regC, if zero flag is set. *It sets C & Z flags*
ADI	Add immediate (I)	adi rb, ra, imm6	Add content of regA with Imm (sign extended) and store result in regB. *It sets C and Z flags*
NDU	Nand (R)	ndu rc, ra, rb	NAND the content of regB to regA and store result in regC. *It sets Z flag*
NDC	Nand if carry set (R)	ndc rc, ra, rb	NAND the content of regB to regA and store result in regC if carry flag is set. *It sets Z flag*
NDZ	Nand if zero set (R)	ndc rc, ra, rb	NAND the content of regB to regA and store result in regC if zero flag is set. *It sets Z flag*
LHI	Load higher immediate (J)	lhi ra, Imm	Place 9 bits immediate into most significant 9 bits of register A (RA) and lower 7 bits are assigned to zero.
LW	Load (I)	lw ra, rb, Imm	Load value from memory into reg A. Memory address is computed by adding immediate 6 bits with content of reg B. *It sets zero flag*.

SW	Store (I)	sw ra, rb, Imm	Store value from reg A into memory. Memory address is formed by adding immediate 6 bits with content of red B.
LM	Load multiple (J)	lw ra, Imm	Load multiple registers whose address is given in the immediate field (one bit per register, R7 to R0) in order from right to left, i.e, registers from R0 to R7 if corresponding bit is set. Memory address is given in reg A. Registers are loaded from consecutive addresses.
SM	Store multiple (J)	sm, ra, Imm	Store multiple registers whose address is given in the immediate field (one bit per register, R7 to R0) in order from right to left, i.e, registers from R0 to R7 if corresponding bit is set. Memory address is given in reg A. Registers are stored to consecutive addresses.
BEQ	Branch on Equality (I)	beq ra, rb, Imm	If content of reg A and regB are the same, branch to PC+Imm, where PC is the address of beq instruction
JAL	Jump and Link (I)	jalr ra, Imm	Branch to the address PC+ Imm. Store PC+1 into regA, where PC is the address of the jalr instruction
JLR	Jump and Link to Register (I)	jalr ra, rb	Branch to the address in regB. Store PC+1 into regA, where PC is the address of the jalr instruction

DATAPATH:


INSTRUCTION FETCH
INSTRUCTION DECODE
REGISTER READ
EXECUTION
MEMORY ACCESS
WRITE BACK
FLUSH1
FLUSH2
FLUSH3
MUX1
PC_ENABLE
PC
PROGRAM MEMORY
ADDER1
MUX2
LM'
SM'
IF/ID 1
CONTROL LOGIC
ID/RR 1
ID/RR 2
ID/RR 3
ID/RR 4
ID/RR 5
ID/RR 6
ID/RR 7
SE6
SE9
MUX3
ADDER2
DATA HAZARD DETECTION UNIT
IF/ID 2
IF/ID 3
LH9
PRIORITY ENCODER
SPZ
MUX4
REGISTER FILE
MUX5
MUX6
MUX7
MUX8
RR/EX 1
RR/EX 2
RR/EX 3
RR/EX 4
RR/EX 5
RR/EX 6
RR/EX 7
RR/EX 8
ALU_CONTROL
ALU
COMPARATOR
EX/MA 1
EX/MA 2
EX/MA 3
EX/MA 4
EX/MA 5
EX/MA 6
EX/MA 7
EX/MA 8
EX/MA 9
EX/MA 10
EX/MA 11
Reg Write Control Unit
M10
DATA MEMORY
Din
Addr
LW Z_FLAG SET UNIT
MUX9
C/Z ENABLE UNIT
C_Enable / Z_Enable
Z_FLAG
C_FLAG
MA/WB 1
MA/WB 2
MA/WB 3
MA/WB 4
MA/WB 5
MA/WB 6
MA/WB 7
MUX11
MUX12
L6_7 (A3)
L6_8 (D3)
LM/SM STALL CONTROL UNIT
LW FLUSH LOGIC
JAL FLUSH LOGIC
JLR FLUSH LOGIC
BEQ FLUSH LOGIC
R7 FLUSH LOGIC

CONTROL WORD FORMAT:

BIT	ID
1-0	MUX11_Sel
3-2	MUX12_Sel
4	Reg_Write
5	mem_read
6	mem_write
7	MUX9_Sel
8	MUX10_Sel
9	ALU_CONTROL
12-10	MUX7_Sel
15-13	MUX8_Sel
16	MUX4_Sel

CONTROL WORDS:

Instruction	MUX4 _sel	Mux8 _sel	MUX7 _sel	ALU_control	MUX10 _sel	Mux9 _sel	mem_write	mem_read	reg_write	Mux12 _sel	Mux11 _sel
ADD	1	_000	_000	1	0	0	0	0	1	_00	10
ADZ	1	_000	_000	1	0	0	0	0	1	_00	10
ADC	1	_000	_000	1	0	0	0	0	1	_00	10
NDU	1	_000	_000	0	0	0	0	0	1	_00	10
NDZ	1	_000	_000	0	0	0	0	0	1	_00	10
NDC	1	_000	_000	0	0	0	0	0	1	_00	10
ADI	1	_001	_000	1	0	0	0	0	1	_00	_01
LHI	0	_000	_000	0	0	0	0	0	1	11	_00
LW	1	_000	_001	1	0	1	0	1	1	_01	_00
SW	1	_000	_001	1	0	0	1	0	0	_00	_00
LM	0	_010	_000	1	1	0	0	L2_1(7-0)!=0	L2_1(7-0)!=0	_01	11
SM	0	_010	_000	1	1	0	L2_1(7-0)!=0	0	0	_00	_00
LM'	0	_011	_010	1	1	0	0	L2_1(7-0)!=0	L2_1(7-0)!=0	_01	11
SM'	0	_011	_010	1	1	0	L2_1(7-0)!=0	0	0	_00	_00
BEQ	1	_000	_000	0	0		0	0	1	_00	_00
JAL	0	_000	_000	0	0	0	0	0	1	10	_00
JLR	1	_000	_000	0	0	0	0	0	1	10	_00

RESOLVING DATA HAZARDS:

- Data dependency is resolved during Register Read Stage with MUX 5 and MUX 6.
- Higher priority must be given to initial stage if multiple data dependencies are encountered.

REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
DEPENDENT INSTRUCTION	ADD/ADI/ NDU	X	X

FORWARD DATA FROM : L4_12

REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
DEPENDENT INSTRUCTION	X	ADD/ADI/ NDU	X

FORWARD DATA FROM : L5_3

REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
DEPENDENT INSTRUCTION	X	X	ADD/ADI/ NDU

FORWARD DATA FROM : L6_4

REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
DEPENDENT INSTRUCTION	JAL/JLR	X	X

FORWARD DATA FROM : L4_8

REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
DEPENDENT INSTRUCTION	X	JAL/JLR	X

FORWARD DATA FROM : L5_11

REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
DEPENDENT INSTRUCTION	X	X	JAL/JLR

FORWARD DATA FROM : L6_11

REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
DEPENDENT INSTRUCTION	LHI	X	X

FORWARD DATA FROM : L4_6

REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
DEPENDENT INSTRUCTION	X	LHI	X

FORWARD DATA FROM : L5_9

REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
DEPENDENT INSTRUCTION	X	X	LHI

FORWARD DATA FROM : L6_9

-For LM check validity from control word if Write Address is 000.

-Insert a NOP after LW instruction.

REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
DEPENDENT INSTRUCTION	X	LW/LM	X

FORWARD DATA FROM : L5_14

REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
DEPENDENT INSTRUCTION	X	X	LW/LM

FORWARD DATA FROM : L6_3

REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
DEPENDENT INSTRUCTION	ADC/ADZ NDC/NDZ	X	X

STALL IF,ID,RR For 1 Cycle

REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
DEPENDENT INSTRUCTION	X	ADC/ADZ NDC/NDZ	X

FORWARD DATA FROM : L5_3 (Check Carry/Zero Flags)

REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
DEPENDENT INSTRUCTION	X	X	ADC/ADZ NDC/NDZ

FORWARD FROM L6_4 (Check Control Signal L6_2(4))

RESOLVING CONTROL HAZARDS:

Control Hazards are resolved via MUX 1,FLUSH 1,FLUSH 2 units.

- **JAL**

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+1]	JAL	X	X	X	X

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+IMM]	NOP	JAL	X	X	X

- **JLR**

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+1]	JLR	X	X	X	X

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+2]	NOP	JLR	X	X	X

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+IMM]	NOP	NOP	JLR	X	X

- ADD/ADI/NDU : DESTINATION R7

DESTINATION :R7

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+1]	ADD/ADI/NDU	X	X	X	X

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+2]	NOP	ADD/ADI/NDU	X	X	X

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+3]	NOP	NOP	ADD/ADI/NDU	X	X

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+ALU_RESULT]	NOP	NOP	NOP	ADD/ADI/NDU	X

- ADZ/ADC/NDZ/NDC DESTINATION : R7

DESTINATION :R7 Check C/Z Flags

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+1]	ADC/ADZ/NDC/NDZ	X	X	X	X

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+2]	NOP	ADC/ADZ/NDC/NDZ	X	X	X

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+2]	NOP	NOP	ADC/ADZ/NDC/NDZ	X	X

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+4]	NOP	NOP	NOP	ADC/ADZ/NDC/NDZ	X

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+ALU_RESULT]	NOP	NOP	NOP	NOP	ADC/ADZ/NDC/NDZ

- **BEQ**

DESTINATION :R7 Check C/Z Flags

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+1]	BEQ	X	X	X	X

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+IMM]	NOP	BEQ	X	X	X

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+IMM+1]	I[PC+IMM]	NOP	BEQ	X	X

MISPREDICTION

INSTRUCTION FETCH	INSTRUCTION DECODE	REGISTER READ	EXECUTION	MEMORY ACCESS	WRITE BACK
I[PC+1]	NOP	NOP	NOP	BEQ	X

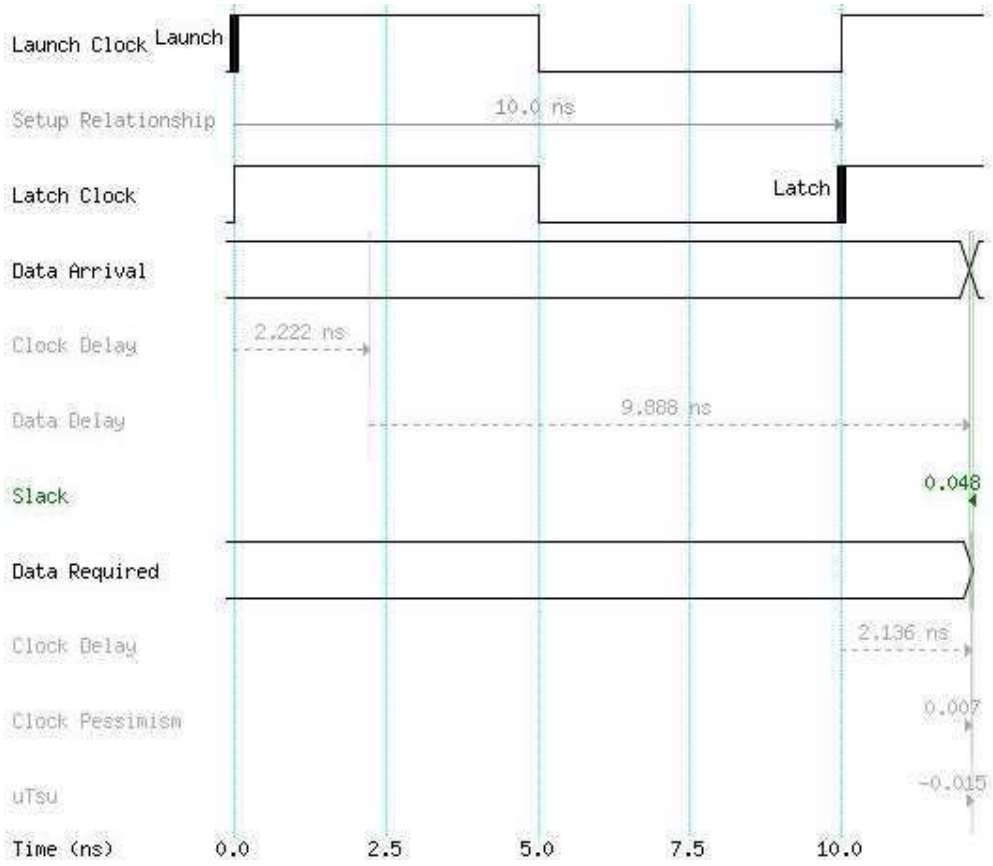
- **LHI**

DESTINATION :R7

(HANDLED LIKE JAL)

TIMING REPORT:

Cycle Time : 10 ns



PERFORMANCE:

Instruction	Fraction of total instructions
LW	α
ADD/ADI/NDU	β
JAL	γ
JLR	ϕ
BEQ	δ (ε fraction of it is predicted correctly)
ADZ/ADC/NDZ/NDU	ε (κ fraction of these instructions give rise to data dependency)
LHI	ζ
SW	η
LM/SM	λ (μ registers accessed on average)

CPI $= 1 + \alpha + \gamma + 2\phi + \delta\varepsilon + 2\delta(1-\varepsilon) + \lambda\mu + \varepsilon\kappa$

(neglecting Instructions where destination is R7)

Cycle Time = 10 ns

TPI = $10^{-8} * CPI$ sec

Throughput = $10^8 / (1 + \alpha + \gamma + 2\phi + \delta\varepsilon + 2\delta(1-\varepsilon) + \lambda\mu + \varepsilon\kappa)$