COA LAB-Risc Processor

Assignment Number : Verilog_Assgn_7

Problem Number : 04

Group Number : 63

Semester : AUTUMN 2023

Group Members :

- Parala Siva Sai Yeswanth (21CS30034)
- Nuthakki Varun (21CS10045)

Instructions Formats:

Arithmetic (Using Register Values)

opcode	rs	rt	rd	shamt	func
31-28	27-23	22-18	17-13	12-4	3-0

Arithmetic (Using immediate value as operand)

opcode	rs	rt	Imm1	func
31-28	27-23	22-18	17-4	3-0

Load and store

opcode	rs	rt	Imm1	func
31-28	27-23	22-18	17-4	3-0

Branch (without condition)

opcode	Imm2
31-28	27-0

Branch (with condition)

opcode	rs	Shamt1	func
31-28	27-23	22-4	3-0

Push and Pop

opcode	rs	No_val	func
31-28	27-23	22-4	3-0

Call

opcode	Imm2
31-28	27-0

Return

opcode	No_val
31-28	27-0

Move

opcode	rs	rt	No_val
31-28	27-23	22-18	17-0

Halt

opcode	No_val
31-28	27-0

Register Usage:

- Register bank consists of 16 general purpose registers each of size 32 bits.
- Zeroth register is especially for indication zero.
- A stack pointer of size 32 bits which is the 17th register.
- 5 bits are used to give a unique code for each register.

Reg_code (5 bits)	Register
00000	R0
00001	R1
00010	R2
00011	R3
00100	R4
00101	R5
00110	R6
00111	R7
01000	R8
01001	R9
01010	R10
01011	R11
01100	R12
01101	R13
01110	R14
01111	R15
10000	sp

Control Signals:

Present state	pc (re,we)	npc (re,we)	rgb (we,re1,re2, incdec)	sg (en)	muxsel (alu1,alu2)	aluout (re,we)	A (re,we)	B (re,we)	dmem (we)	lmd (re,we)	Reginmux sel	IR (we)	immmuxsel	alusel	cond	swap	select	nextstate
ms:4'd0 ss:4'd0	1,0	0,1										1						ms:4'd1 ss:4'd0
ms:4'd1 ss:4'd0	1,0	0,0										0					0	ms:4'd2 ss:4'd0
ms:4'd2 ss:4'd0	0,0																	ms: opcode + 3 ss:4'd0
ms:4'd3 ss:4'd0			0,1,1,00	1			0,1	0,1					10					ms:4'd3 ss:4'd1
ms:4'd3 ss:4'd1			0,0,0,00		0,0		1,0	1,0						func				ms:4'd3 ss:4'd2
ms:4'd3 ss:4'd2						0,1												ms:4'd3 ss:4'd3
ms:4'd3 ss:4'd3	0,1	1,0	1,0,0,00			1,0					1				000			ms:4'd3 ss:4'd4
ms:4'd3 ss:4'd4	0,0	0,0	0,0,0,00			0,0												ms:4'd0 ss:4'd0
ms:4'd4 ss:4'd0			0,1,0,00	1			0,1	0,0					00					ms:4'd4 ss:4'd1
ms:4'd4 ss:4'd1			0,0,0,00		0,1		1,0							func				ms:4'd4 ss:4'd2
ms:4'd4 ss:4'd2						0,1												ms:4'd4 ss:4'd3
ms:4'd4 ss:4'd3	0,1	1,0	1,0,0,00			1,0					1				000			ms:4'd4 ss:4'd4
ms:4'd4 ss:4'd4	0,0	0,0	0,0,0,00			0,0												ms:4'd0 ss:4'd0
ms:4'd5 ss:4'd0			0,1,1,00	1			0,1	0,1					00					ms:4'd5 ss:4'd1
ms:4'd5 ss:4'd1			0,0,0,00		0,1									0000				ms:4'd5 ss:4'd2
ms:4'd5 ss:4'd2			1,0,0,00															ms:4'd5 ss:4'd3

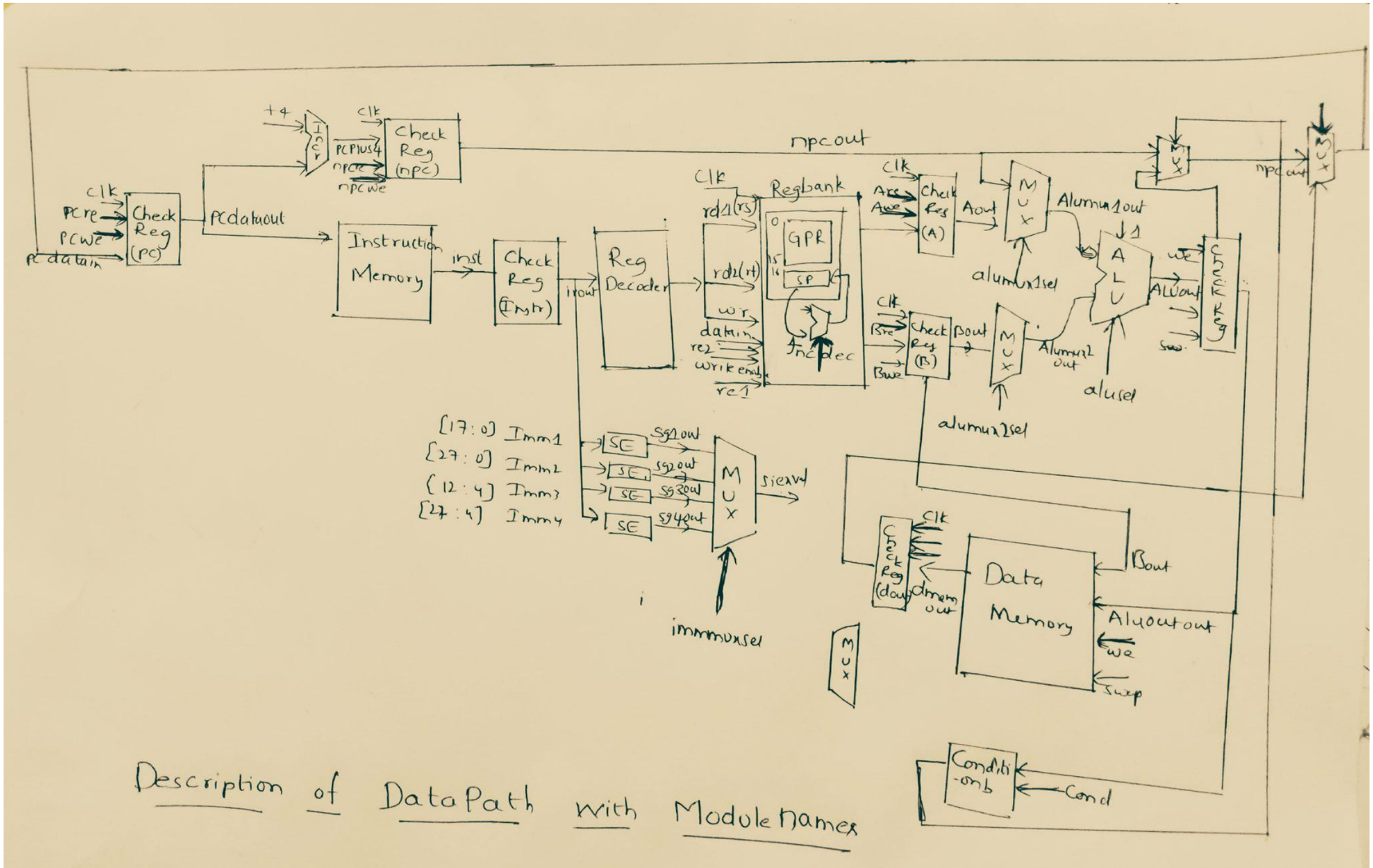
ms:4’d5 ss:4’d3	0,1	1,0	0,0,0,00						if(store) 1	if(load) 1,1 else 0,0	if(load) 0							ms:4’d5 if(load) ss:4’d4 else ss:4’d5
ms:4’d5 ss:4’d4			1,0,0,00							1,0								ms:4’d5 ss:4’d6
ms:4’d5 ss:4’d5			0,0,0,00						0									ms:4’d5 ss:4’d6
ms:4’d5 ss:4’d6	0,0	0,0	0,0,0,00							0,0								ms:4’d0 ss:4’d0
ms:4’d6 ss:4’d0		1,0	0,0,0,00	1	1,1	0,1							01	0000				ms:4’d6 ss:4’d1
ms:4’d6 ss:4’d1	0,1					1,0									100			ms:4’d6 ss:4’d2
ms:4’d6 ss:4’d2	0,0	0,0													000			ms:4’d0 ss:4’d0
ms:4’d7 ss:4’d0			0,1,0,00	1			0,1						11					ms:4’d7 ss:4’d1
ms:4’d7 ss:4’d1		1,0	0,0,0,00		1,1	0,1	1,0							0000	func + 1			ms:4’d7 ss:4’d2
ms:4’d7 ss:4’d2	0,1					1,0												ms:4’d7 ss:4’d3
ms:4’d7 ss:4’d3	0,0	0,0				0,0									000			ms:4’d0 ss:4’d0
ms:4’d8 ss:4’d0			if(push) 0,0,1,10 if(pop) 0,0,1,01					0,1										ms:4’d8 if(push) ss:4’d1 if(pop) ss:4’d2
ms:4’d8 ss:4’d1			0,1,0,00		1,0		1,1	0,0						1001				ms:4’d8 ss:4’d3
ms:4’d8 ss:4’d3			0,0,0,00			1,1	1,0											ms:4’d8 ss:4’d4
ms:4’d8 ss:4’d4	0,1	1,0						1,0	1						000			ms:4’d8 ss:4’d5
ms:4’d8 ss:4’d5	0,0	0,0				0,0			0						000			ms:4’d0 ss:4’d0
ms:4’d8 ss:4’d2			0,1,0,00		0,0		1,1							1011				ms:4’d8 ss:4’d6
ms:4’d8 ss:4’d6			0,0,0,00			0,1	1,0											ms:4’d8 ss:4’d7

ms:4’d8 ss:4’d7	0,1	1,0				1,0				0,1					000			ms:4’d8 ss:4’d8
ms:4’d8 ss:4’d8	0,0	0,0	1,0,0,00							1,0	0							ms:4’d8 ss:4’d9
ms:4’d8 ss:4’d9										0,0								ms:4’d0 ss:4’d0
ms:4’d9 ss:4’d0		1,0	0,0,0,00	1														ms:4’d9 ss:4’d1
ms:4’d9 ss:4’d1			0,0,1,00		1,0		1,1	1,1						1011				ms:4’d9 ss:4’d2
ms:4’d9 ss:4’d2						1,1	1,0	1,0										ms:4’d9 ss:4’d3
ms:4’d9 ss:4’d3						1,0			1							1		ms:4’d9 ss:4’d4
ms:4’d9 ss:4’d4		1,0				1,0												ms:4’d9 ss:4’d5
ms:4’d9 ss:4’d5		1,0	0,0,0,10		1,1	1,1			0				01	0000		0		ms:4’d9 ss:4’d6
ms:4’d9 ss:4’d6	1,0		0,0,0,00			1,0	1,0	1,0							100			ms:4’d9 ss:4’d7
ms:4’d9 ss:4’d7	0,0														000			ms:4’d0 ss:4’d0
ms:4’d10 ss:4’d0			0,1,0,00		0,1		0,1	0,1						1001				ms:4’d10 ss:4’d1
ms:4’d10 ss:4’d1			0,0,0,01			1,1	0,0	0,0										ms:4’d10 ss:4’d2
ms:4’d10 ss:4’d2			0,0,0,00			1,0				1,0								ms:4’d10 ss:4’d3
ms:4’d10 ss:4’d3	0,1	1,0				0,0				0,1	0						1	ms:4’d10 ss:4’d4
ms:4’d10 ss:4’d4	0,0									0,0							0	ms:4’d0 ss:4’d0
ms:4’d12 ss:4’d0																		ms:4’d12 ss:4’d0
ms:4’d13 ss:4’d0	0,1	1,0													000			ms:4’d13 ss:4’d1
ms:4’d13 ss:4’d1	0,0	0,0																ms:4’d0 ss:4’d0

Control Signals Description:

- Opcode,store,load,func,push,pop can be known from the given instruction and the blanks spaces in the table represents that the control signal is same as it is in its previous state
- ms: Main State
- ss: Sub State
- re: Read Enable (signal which decides about loading)
- We: Write enable (signal which decides about storing)
- Re1: For rs register
- Re2: For rt register
- Alu1 & Alu2: For the muxes
- Alusel: Selection for operation of ALU.
- Cond: To decide branch instruction like br,bz,bmi,bpl.

Data Path detailed Design:



Description of DataPath with Module Names