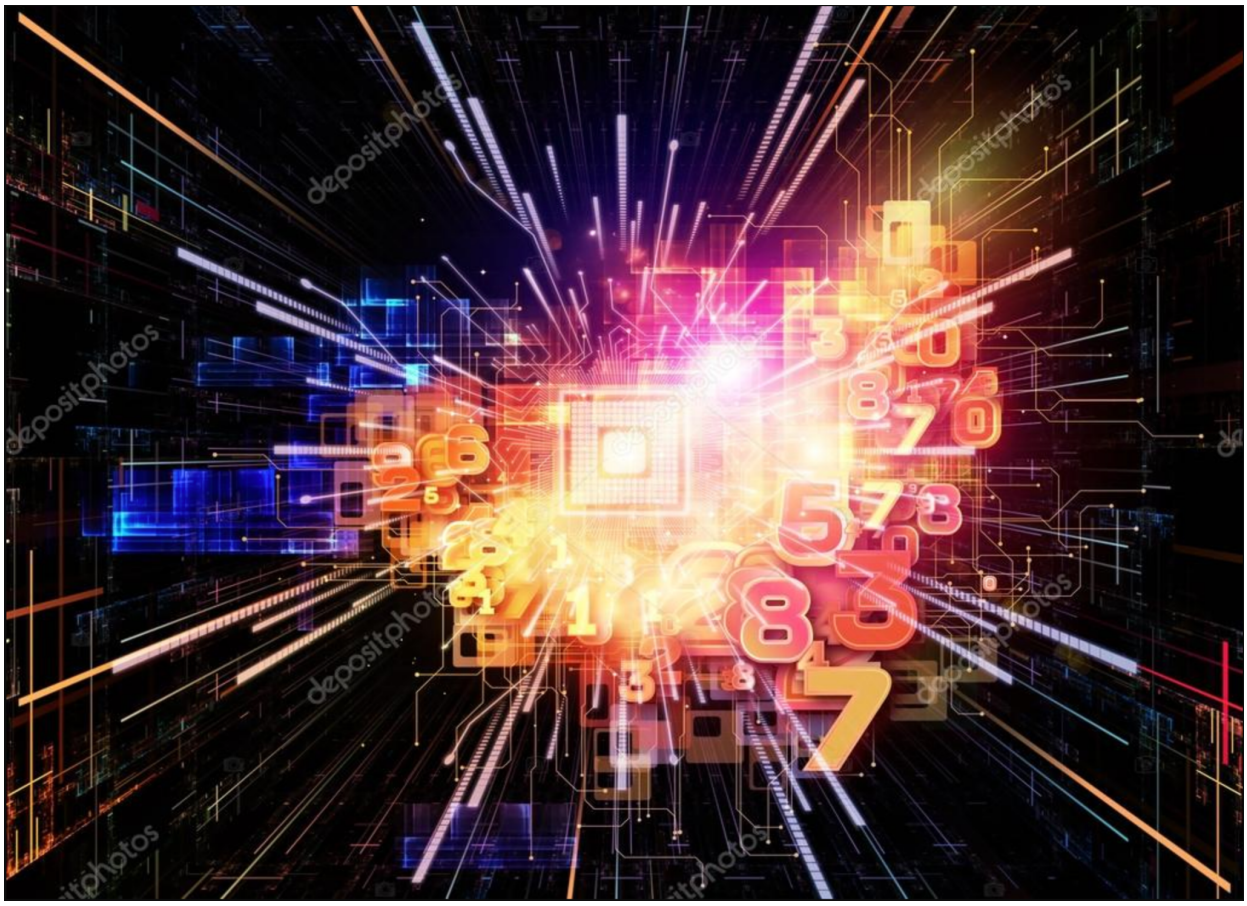


IITB_Proc

CS 226 Project Design Report

Aakriti
190050002

Abhinav Gupta
190050003

Sambit Behera
190050104

Tulip Pandey
190050125

AIM

The aim of this project is to use VHDL to implement a multi-cycle processor which is an 8-register, 16-bit computer system which uses point-to-point communication infrastructure (for e.g. A 16-bit very simple computer developed for the teaching purpose) .

The IITB-Proc 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.

RType Instruction format

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

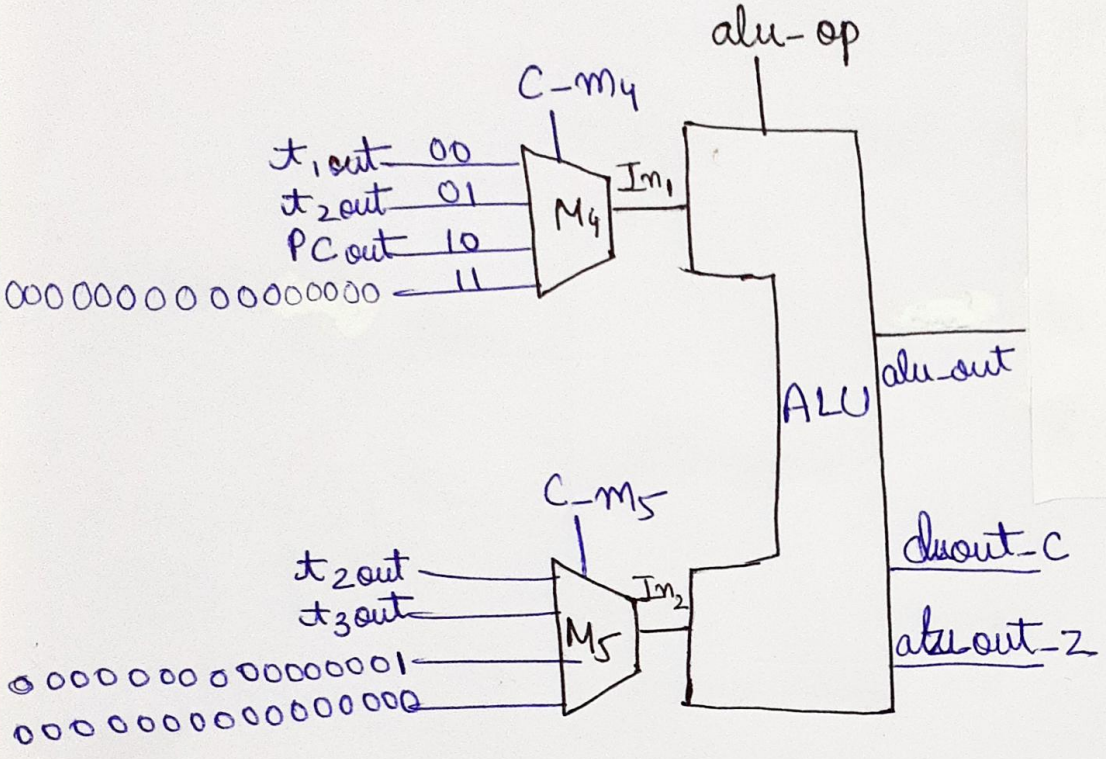
I Type Instruction format

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

JType Instruction format

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

Arithmetic Logic Unit (ALU)



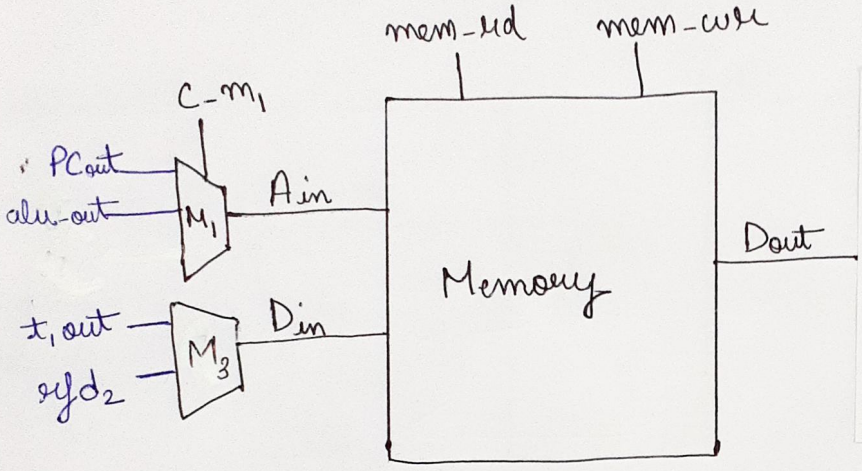
Glossary :-

alu-op :- 0 if SUM, 1 if NAND

M4 :- 16 bit 4x1 mux to choose In_1

M5 :- 16 bit 4x1 mux to choose In_2

Memory



Glossary

Ain :— Address to be read from or written to

Din :— Data to be written

Dout :— Read out data

M_1 :— 2×1 16 bit mux selecting reading of instruction or loading of data

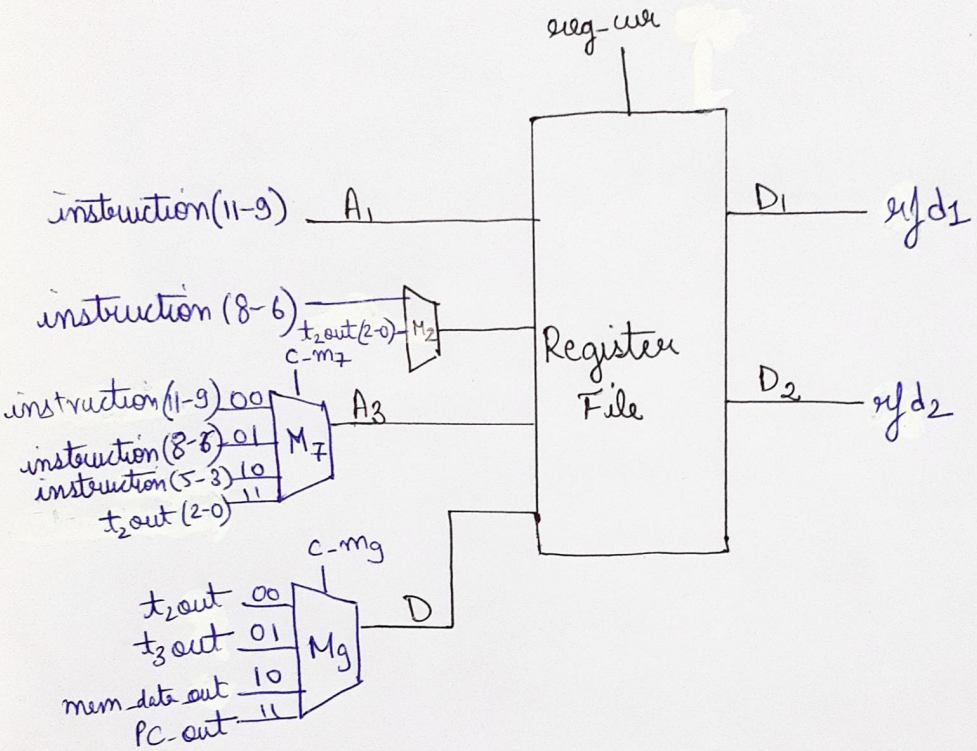
D_1 :— 1×4 16bit demux directing Dout to Instruction / t5 / gbg-16 depending on value of C-d1.

mem_wr :— 1 if writing to memory, else 0

mem_rd :— 1 if reading from memory, else 0.

M_3 :— 2×1 16 bit mux selecting reading of

Register File



Glossary :-

A_1 :- first address to read from

A_2 :- second address to read from

A_3 :- address to write in

D :- data to be written

D_1 :- content of register at address A_1

D_2 :- content of register at address A_2

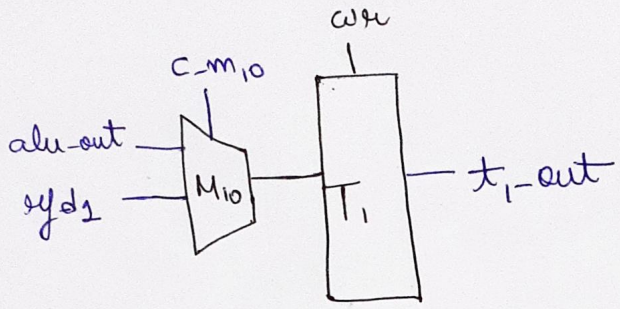
M_7 :- 3 bit 4x1 mux to choose A_3 using $c-m_7$

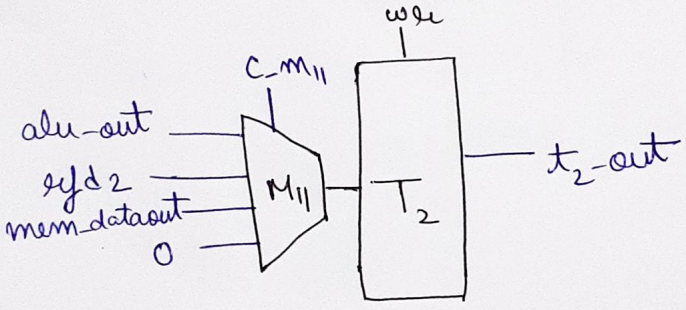
M_g :- 16 bit 4x1 mux to choose D using $c-m_g$

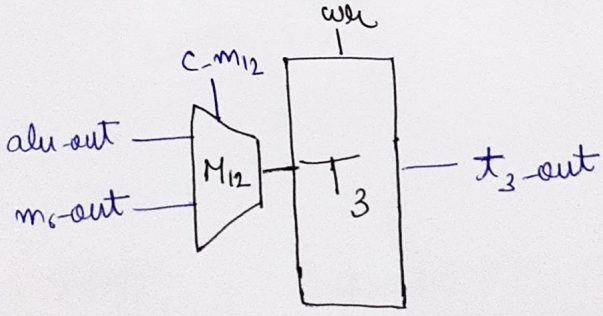
reg_wr :- controls writing into the register file

M_2 :- 3 bit 4x1 mux to choose A_2 using $c-m_2$

T_1, T_2, T_3 :- dff-register



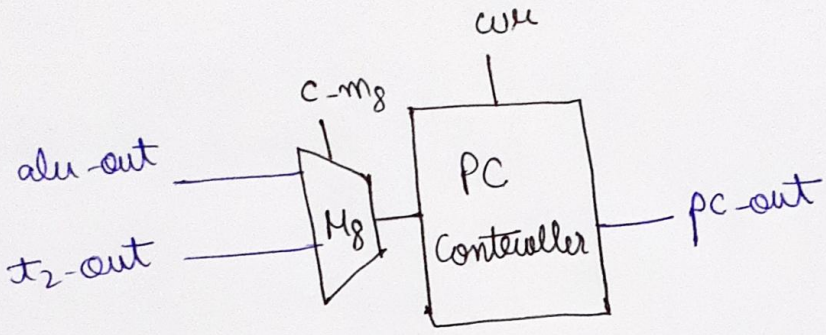




Glossary :-

dff-register :- D Flip Flop Register

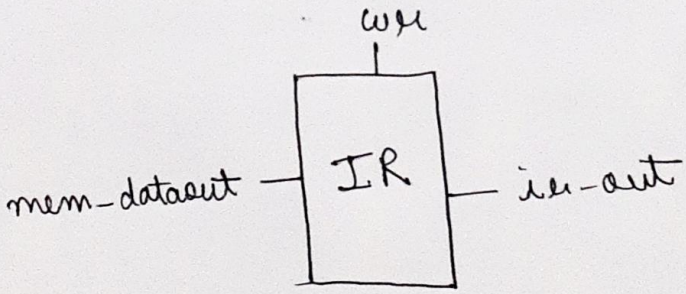
PC Controller :- dff-register



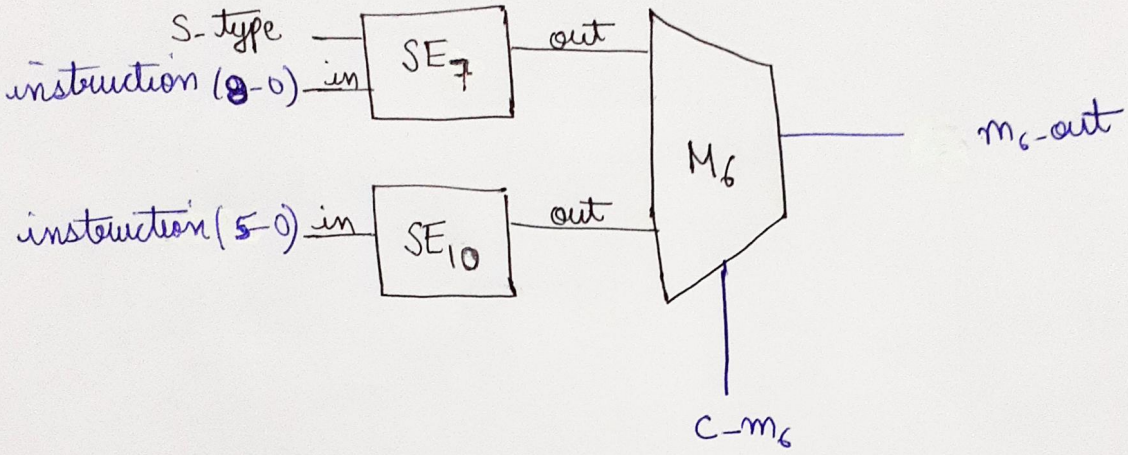
Glossary

M_8 :- 16 bit 2x1 mux to choose one value for PC out of alu-out (otherwise) & t_2-out (JLR)

IR Controller :- dff-register



Signed Extender



Glossary

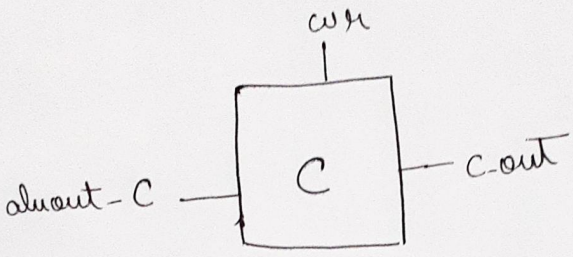
SE_7 :- Signed extender that extends 9 bits to 16 bits

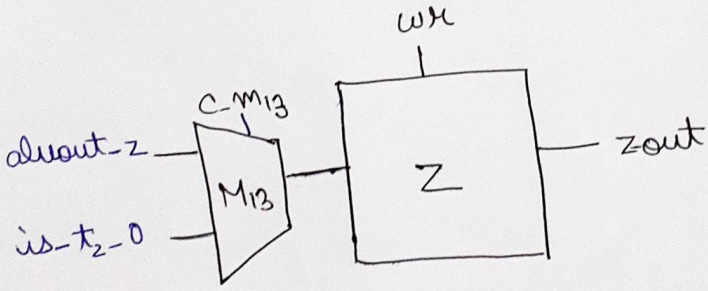
SE_{10} :- Signed extender that extends 6 bits to 16 bits

S-type :- 1 if instruction is JAL, else 0 to switch b/w signed extension and MSB placement.

M_6 :- 2×1 16 bit mux that chooses SE_7 or SE_{10} based on C-m_6.

D-Flip Flops :- C, Z

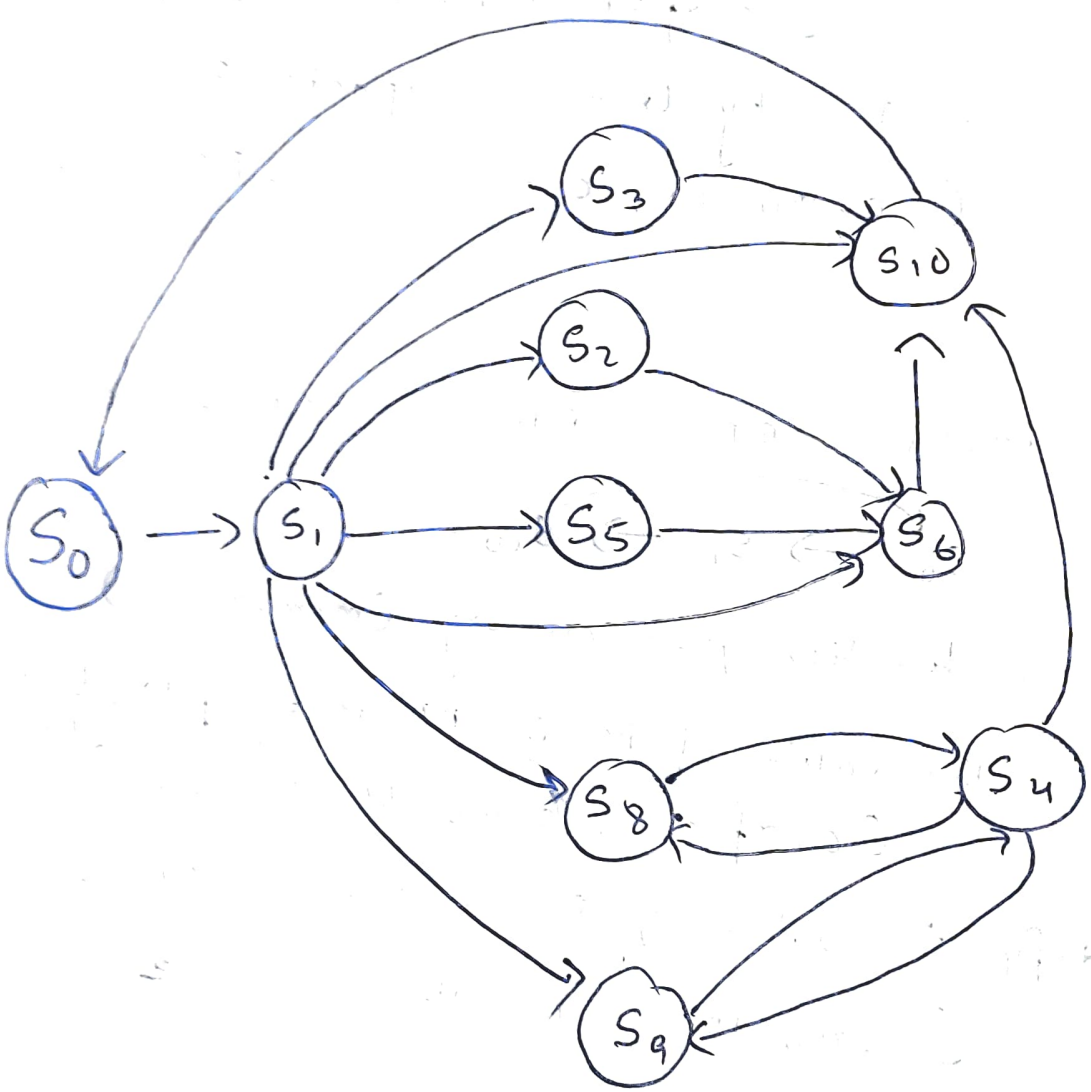




Glossary :—

M_{13} :— 1 bit 2×1 mux to choose if z_out (z flag) is equal to value of $is\_t_2\_0$ (LW case) or alu_out_z (otherwise)

STATE DIAGRAM



INSTRUCTION :

ADD : 00-00 RA RB RC 0 00

NDU : 00-10 RA RB RC 0 00

FSM path : $S_0 \rightarrow S_1 \rightarrow S_5 \rightarrow S_6 \rightarrow S_{10}$

S_0 : Read instruction from memory (mem-rd = 1)

S_1 : Read content of RA into t_1 and RB into t_2

(c-t_1 = 1, c-t_2 = 1, c-m_{10} = 1, c-m_{11} = 01, c-m_2 = 0)

S_5 : Add/Nand t_1 and t_2 in ALU and store in t_3

(C-alu = 0/1, c-m_4 = 00, c-m_5 = 00, c-m_{12} = 0)

S_6 : Read the value of t_3 into RF at RC

(c-rf = 1, c-m_7 = 10, c-m_9 = 01)

S_{10} : Add 1 to current PC value and store in PC

(c-m_4 = 10, c-m_5 = 10, c-m_8 = 0, c-PC = 1, c-ALU = 0)

INSTRUCTION :

ADC - 00-00 RA RB RC 0 10

NDC 00-10 RA RB RC 0 10

FSM path - $S_0 \rightarrow S_1 \rightarrow S_5 \rightarrow S_6 \rightarrow S_{10}$

S_0 - Read instructions from memory (mem-rd = 1)

S_1 - Read content of RA into t_1 and RB into t_2

(c-t_1 = 1, c-t_2 = 1, c-m_{10} = 1, c-m_{11} = 01, c-m_2 = 0)

S_5 - Add/nand t_1 and t_2 and store in t_3 if c flag is set 1

(c-C = 1/0 c-m_4 = 00, c-m_5 = 00, c-m_{12} = 0, c-2 = 1, c-ALU = 0/1)

S_6 - Read the value of t_3 into RC (c-rf = 1, c-m_3 = 10, c-m_9 = 0

S_{10}: Add 1 to current PC value and store in PC

(c-alu = 0, c-m_4 = 10, c-m_5 = 10, c-m_8 = 0, c-PC = 1)

INSTRUCTION:

ADZ	00-00	RA	RB	RC	0	01
NDZ	00-10	RA	RB	RC	0	01

FSM Path: $S_0 \rightarrow S_1 \rightarrow S_5 \rightarrow S_6 \rightarrow S_{10}$

S_0: Read instructions from memory (mem-rd = 1)

S_1: Read the content of RA into t_1 and R_B into t_2

(c-t_1 = 1, c-t_2 = 1, c-m_{10} = 1, c-m_{11} = 01, c-m_2 = 0)

S_5: Add/Nand t_1 and t_2 and store in t_3 if z flag is set 1. (c-C = 1/0, c-m_4 = 00, c-m_5 = 00, c-m_{12} = 0, c-Z = 1, c-ALU = 0/1)

~~S_6~~

S_6: Read the value of t_3 into RC (c-rf = 1, c-m_7 = 10, c-m_9 = 01)

S_{10}: Add 1 to current PC value and store in PC

(c-alu = 0, c-m_4 = 10, c-m_5 = 10, c-m_8 = 0, c-PC = 1)

INSTRUCTION

ADI:	00-01	RA	RB	6-bit immediate

FSM path - $S_0 \rightarrow S_1 \rightarrow S_5 \rightarrow S_6 \rightarrow S_{10}$

S_0: Read instructions from memory (mem-rd = 1)

S_1: Read the content of RA into t_1 and ~~RB into t2~~ immediate into t_2

(c-t_1 = 1, c-t_2 = 1, c-m_{10} = 1, c-m_{11} = 01, c-m_2 = 0)

S_5: Adds t_1 and t_2 and stores value in t_3

(c-C = 1, c-Z = 1, c-m_4 = 00, c-m_5 = 01, c-m_{12} = 0, c-alu = 0, c-t_3 = 1)

S_6: Read the value of t_3 into register B (c-rf = 1, c-m_7 = 00, c-m_9 = 01)

S_{10}: Add 1 to current PC and store in PC (c-alu = 0, c-m_4 = 10, c-m_5 = 10, c-m_8 = 0)

INSTRUCTION :

LHI 00-11 RA 8 bit immediate

FSM path : $S_0 \rightarrow S_1 \rightarrow S_6 \rightarrow S_{10}$

S_0 : Read instruction from memory (mem-rd = 1)

S_1 : Read content from RA into t_1 and use sign extender to extend the 8 bits, and store in t_3

(c-t_1 = 1, c-m_2 = 0, c-m_{10} = 1, c-m_{11} 01, c-m_{12} = 1, c-m_6 = 0, c-sext8 = 1)

S_6 : Load value in t_3 into RA

(c-m_7 = 00, c-m_9 = 01, c-rf = 1)

S_{10} : Add 1 to PC and store in PC

(c-m_4 = 10, c-PC = 1, c-alu = 0, c-m_5 = 10, c-m_8 = 0)

Instruction: LW 01-00 RA RB 6 bit immediate

FSM path : $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_6 \rightarrow S_{10}$

S_0 : Read instruction from memory (mem-rd = 1)

S_1 : Read content from RA into t_1 and RB into t_2 and 6 bit immediate in t_3

(c-t_1 = 1, c-t_2 = 1, c-m_2 = 0, c-m_{10} = 1, c-m_{11} = 01 c-t_3 = 1, c-m_{12} = 1)

S_2 : Add t_2 and t_3 and read the memory from the address and store in t_2

(c-alu = 0, c-m_4 = 01, c-m_5 = 01, c-m_1 = 1, c-t_2 = 1, c-m_{11} = 10)

S_6 : Read the value of t_2 into register file in RA, and change z flag (c-m_7 = 00, c-m_9 = 00, c-z = 1, c-m_{13} = 1)

S_{10} : Add 1 to PC and store in PC

(c-alu = 0, c-m_4 = 10, c-m_5 = 10, c-PC = 1, c-m_8 = 0)

Instruction

SW : 01-01 RA RB 6 bit immediate

FSM path : $S_0 \rightarrow S_1 \rightarrow S_3 \rightarrow S_{10}$

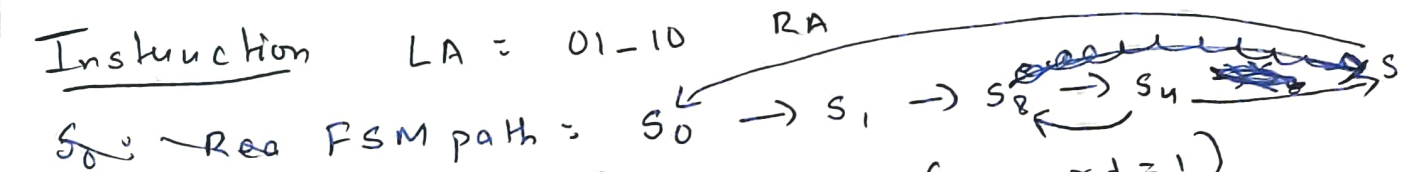
S_0 : Read instruction from memory (mem-rd = 1)

S_1 : Read content of R_A into t_1 and R_B into t_2 and use sign extender on 6 bit immediate and store in t_3
(c-t_1 = 1, c-t_2 = 1, c-t_3 = 1, c-m_6 = 1, c-sext9 = 0)

S_3 : Add t_3 and t_2 and write RA into that memory address (c-m_4 = 01, c-m_5 = 01, mem-wr = 1, c-m_1 = 1, c-alu = 0, c-m_3 = 0)

S_{10} : Add 1 to PC and store in PC
(c-alu = 0, c-m_4 = 10, c-m_5 = 10, c-PC = 1, c-m_8 = 0)

Instruction LA : 01-10



FSM path : $S_0 \rightarrow S_1 \rightarrow S_8 \rightarrow S_4 \rightarrow S_{10}$

S_0 : Read instruction from memory (mem-rd = 1)

S_1 : t_1 stores the memory address and t_2 is set to 0
(c-t_1 = 1, c-m_{11} = 11, c-t_2 = 1)

S_8 : Add t_1 and t_2 and read memory from the resulting address and load the corresponding register
(c-rf = 1, c-m_4 = 00, c-m_5 = 00, c-m_1 = 1, c-m_9 = 10, c-alu = 0)

S_4 : Increment t_2's value and return to S_8 if $t_2 < 7$, else go to S_{10}
(c-m_4 = 01, c-m_5 = 10, c-alu = 0, c-t_2, c-m_{11} = 00)

S_{10} : Add 1 to PC and store in PC
(c-alu = 0, c-m_4 = 10, c-m_5 = 10, c-PC = 1, c-m_8 = 0)

Instruction :

SA : 01 - 11 RA

FSM path : $S_0 \rightarrow S_1 \rightarrow S_9 \rightarrow S_4 \rightarrow S_{10}$

S_0 : Read instruction from memory (mem_rd = 1)

S_1 : t_1 stores address of initial R_0 and t_2 is set to 0
(c_m11 = 11, c_t1 = 1, c_t2 = 1)

S_9 : ~~Add t_1 and t_2 and~~
S_9 : Read value from RF into t_2 + t_1 and write into memory
(c_m2 = 1, c_m4 = 00, c_m5 = 00, mem_wr = 1, c_m1 = 1, c_alu = 0, c_m3 = 1)

S_4 : Increment t_2 and move to S_9 if $t_2 < 7$, else move to S_{10} (c_m4 = 01, c_m5 = 10, c_t2 = 1, c_m11 = 00, c_alu = 0)

S_{10} : Increment PC by 1 and update it
(c_alu = 0, c_m4 = 10, c_m5 = 10, c_m8 = 0, c_PC = 1)

Instruction :

BEQ : 11 - 00 RA RB 6 bit immediate

FSM path : $S_0 \rightarrow S_1 \rightarrow S_{10}$

S_0 : Read instruction from memory (mem_rd = 1)

S_1 : Read content from RA into t_1 and RB into t_2
(c_t1 = 1, c_t2 = 1, c_m2 = 0, c_m10 = 1, c_m11 = 01, c_m12 = 1)

S_{10} : If $t_1 = t_2$, increment PC by Imm, else by 1
(c_m4 = 10, c_PC = 1, c_m5 = 01 / 10, c_m8 = 0)

Instruction

JAL : 10-00 RA A bit immediate offset

FSM path : $S_0 \rightarrow S_1 \rightarrow S_{10}$

S_0 : Read instruction from memory (mem-rd = 1)

S_1 : Write value of PC into RA in Register file
(c-rf = 1, c-m_6 = 0, c-m_7 = 00, c-m_9 = 11)

S_{10} : Increment PC by 1mm and update PC
(c-m_4 = 10, c-m_5 = 01, c-PC = 1, c-m_8 = 0)

Instruction :

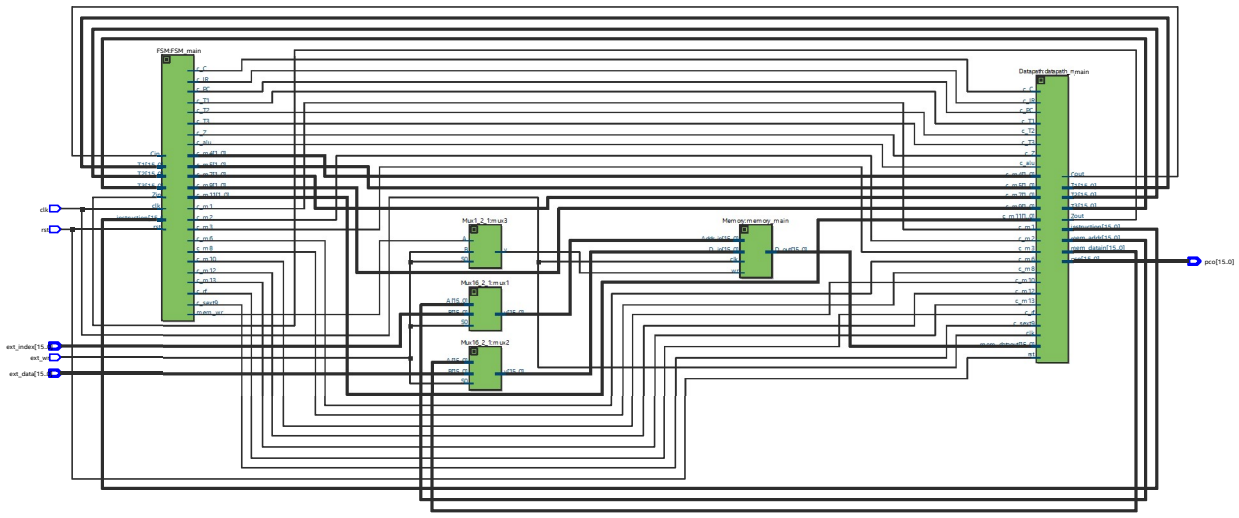
JLR : 10-01 RA RB 000-000

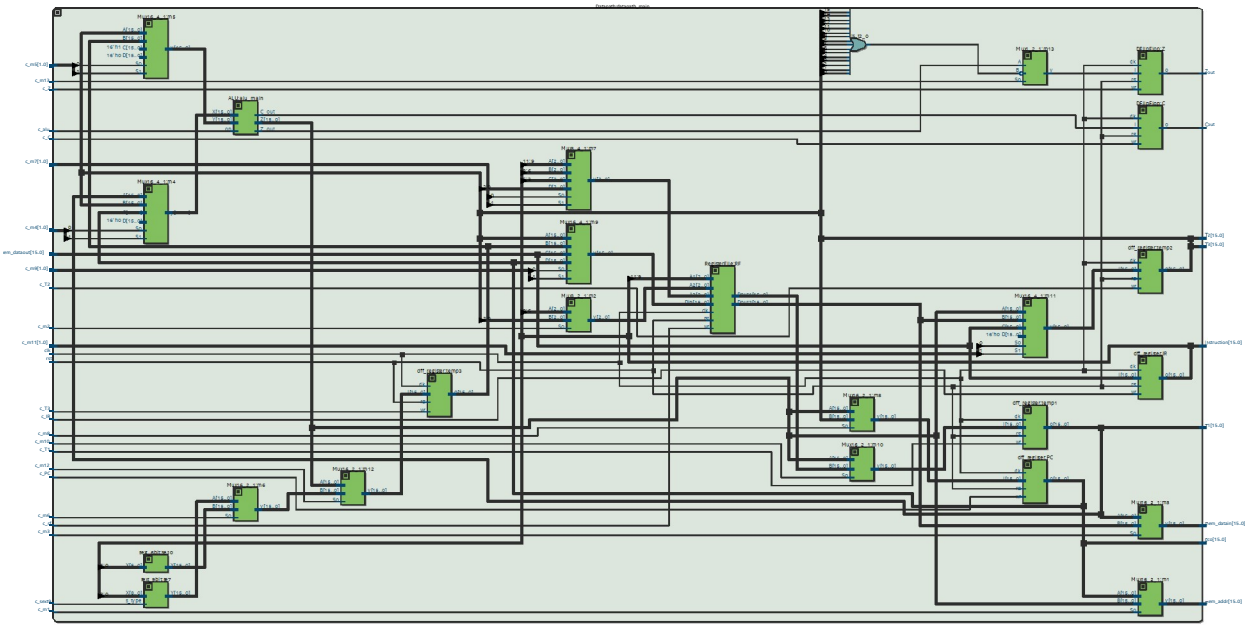
FSM path : $S_0 \rightarrow S_1 \rightarrow S_{10}$

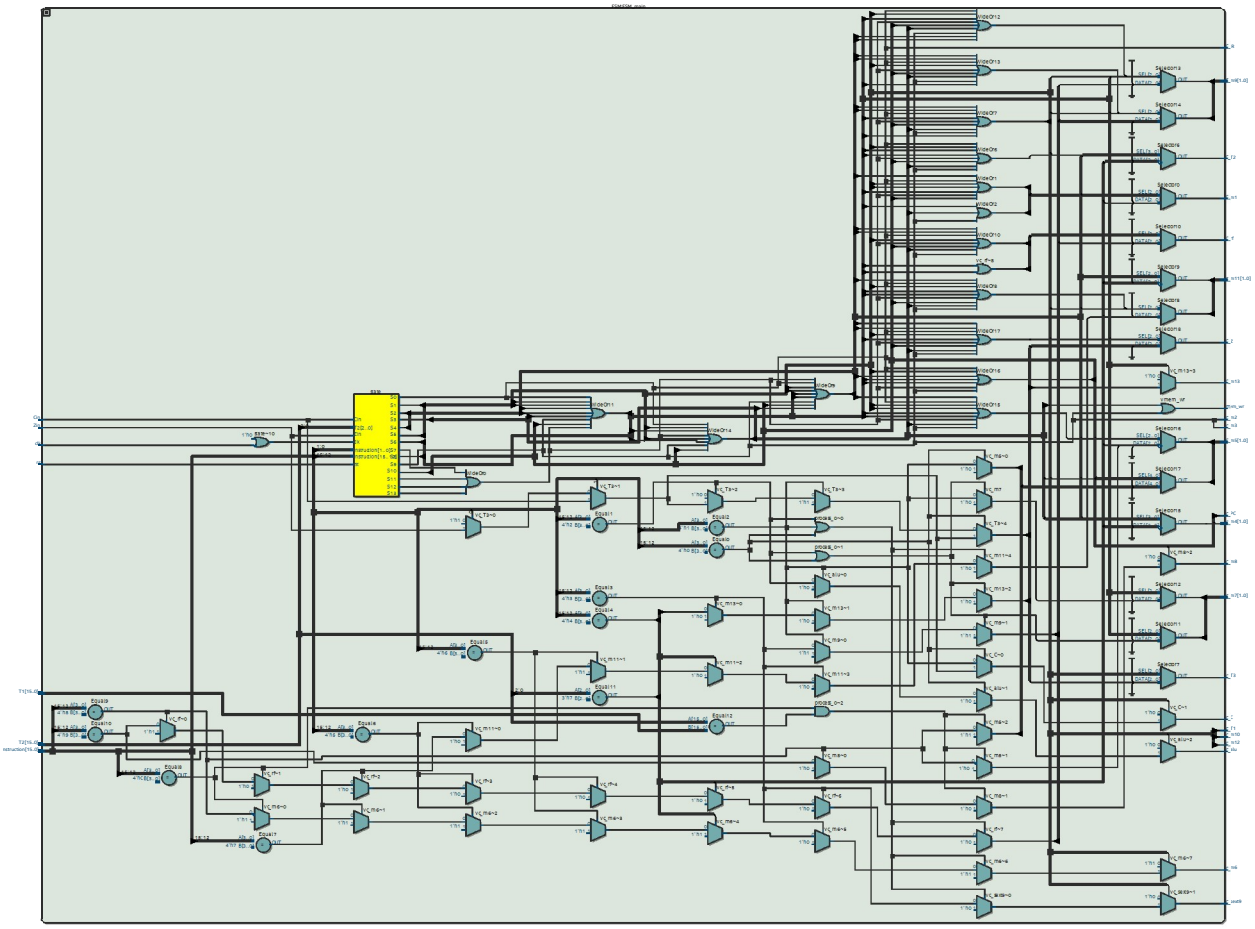
S_0 : Read instruction from memory (mem-rd = 1)

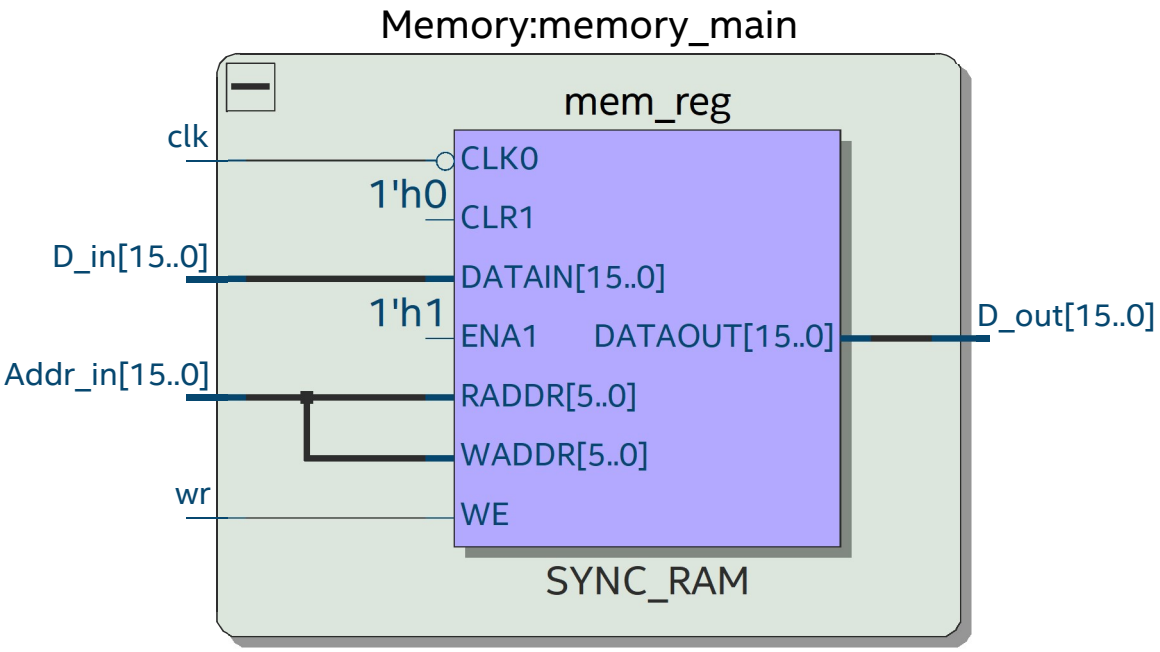
S_1 : Store the value of PC in RA in reg. file, t_1, t_2 have RA and RB values
(c-rf = 1, c-m_7 = 00, c-m_9 = 11)

S_{10} : Modify the value of PC to value in RB and update
(c-m_4 = 10, c-m_5 = 10, c-alu = 0, c-m_8 = 1)


FSM:FSM_main
Datapath:datapath_1:main
Memory:memory_main
Mux1_2_1:mux3
Mux16_2_1:mux1
Mux16_2_1:mux2
clk
rst
pco[15:0]






Memory:memory_main
mem_reg
clk
CLK0
1'h0
CLR1
D_in[15..0]
DATAIN[15..0]
1'h1
ENA1
DATAOUT[15..0]
D_out[15..0]
Addr_in[15..0]
RADDR[5..0]
WADDR[5..0]
wr
WE
SYNC_RAM