IITB-CPU Design

Submitted in partial fulfillment of the requirements of the course EE224

Shristi Shrivastava (21D070069)   Devesh Soni(21D070025)
Dhrumil Lotiya(21D070026)   Rudraksh Sangore(21D070060)

November 30, 2022

Abstract

The goal of this project is to implement a RISC processor, being able to perform all the below given instructions(implementation described in the later sections).

1 Major Components used in our DataPath

1.1 Memory

- Stores the instruction word(16bit) as per the format specified.
- Outputs the word whenever the corresponding address location is given with help of program counter.
- Register 7 of register file is chosen as Program counter.
- Also stores data in location as per the given input , also enabled to read data from memory corresponding to the given address.

1.2 Instruction Register

- Decodes output from Instruction word to operand/destination registers/immediate value with corresponding sign extensions as required.

1.3 Register File

- Contains 8 registers with 2 set of input and output read pins 1 set of input and output write pin.
- Another port present to directly read from R7 , which stores the Program counter.

1.4 ALU

- Used to increment PC value.
- Performs addition, subtraction and bitwise NAND operation according to Opcode and alu selectline.
- Also modifies the Carry-flag, Zero-flag and Equal-flag.

1.5 Temporary Registers

- Temporary registers are used to cause some delays, so that there is no case of reading and writing simultaneously for a component.
- 4 temporary registers are used , each can store 16 bits when corresponding writes are enabled , else read is enabled .

1.6 Flag Registers

- Temporary register which can store 1 bit , used to store Carry-flag and Zero-flag outputs.
- 2 flag-registers are used which can store data when write is enabled .

2 Port Maps for different components

2.1 Memory

- **mem_d1 :** port for loading data in memory
- **mem_a1 :** port for address in/from which data is loaded/read
- **mem_d2 :** port for data out
- **mem_write :** memory write enable

2.2 Instruction Register

- **instr_d1 :** port for data in
- **instr_d2 :** port for data out
- **instr_write :** memory write enable

2.3 Register File

- **rf_d1 :** port for loading data in register A
- **rf_a1 :** port for address of register A in/from which data is loaded/read
- **rf_d2 :** port for loading data in register B
- **rf_a2 :** port for address of register B in/from which data is loaded/read
- **rf_d3 :** port for loading data in register C
- **rf_a3 :** port for address of register C in/from which data is loaded/read
- **pc :** port for Program counter signal
- **rf_write :** port for write enable

2.4 ALU

- **alu_a :** port for data in A
- **alu_b :** port for data in B
- **alu_c :** port for data out C
- **flag_op :** port for selecting add or NAND operation
- **flag_carry :** port for indicating carry flag
- **flag_zero :** port for indicating zero flag
- **flag_equal :** port for indicating equal flag

2.5 Temporary Register

4 such registers, namely t1,t2,t3 and t4 are used .

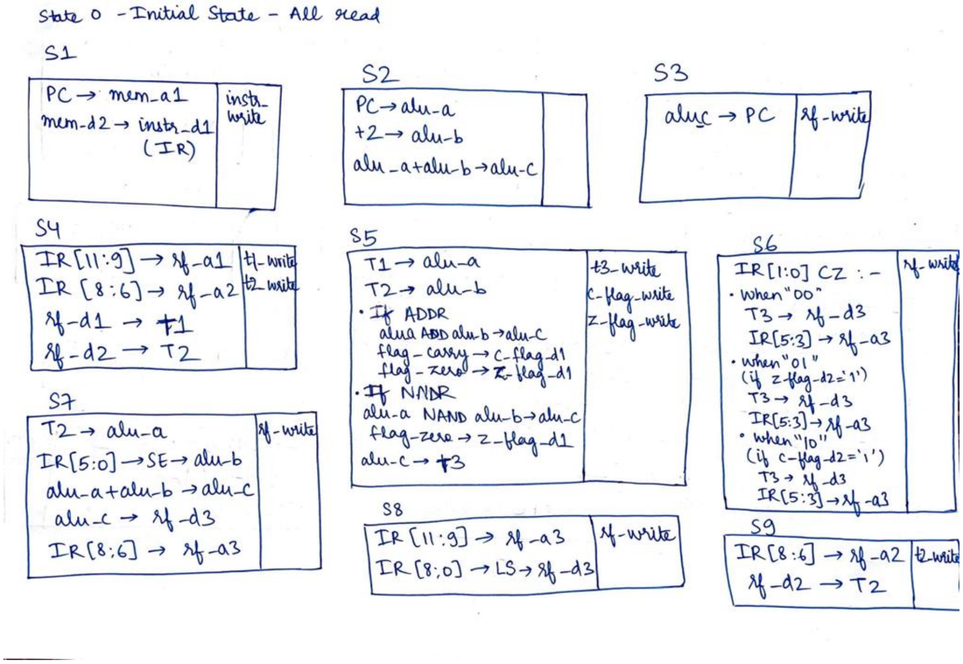
- **temp16_d1 :** port for loading data in register
- **temp16_d2:** port for data out
- **temp16_write :** port for write enable

2.6 Flag Register

2 such registers, namely carry_flag and zero_flag are used .

- **flag_d1 :** port for loading data in register
- **flag_d2 :** port for data out
- **falg_write :** port for write enable

3 Simulation



State 0 - Initial State - All read

S1

PC → mem_a1
mem-d2 → instr_d1
(IR)

instr_write

S2

PC → alu-a
+2 → alu-b
alu_a+alu-b → alu-C

S3

aluc → PC

rf-write

S4

IR[11:9] → rf-a1
IR [8:6] → rf-a2
rf-d1 → T1
rf-d2 → T2

t1-write
t2-write

S5

T1 → alu-a
T2 → alu-b
• If ADDR
alua ADD alu-b → alu-C
flag-carry → C-flag-d1
flag-zero → Z-flag-d1
• If NNDR
alu-a NAND alu-b → alu-c
flag-zero → z-flag-d1
alu-c → T3

t3-write
C-flag-write
Z-flag-write

S6

IR[1:0] CZ :-
• When "00"
T3 → rf-d3
IR[5:3] → rf-a3
• when "01"
(if z-flag-d2='1')
T3 → rf-d3
IR[5:3] → rf-a3
• when "10"
(if c-flag-d2='1')
T3 → rf-d3
IR[5:3] → rf-a3

rf-write

S7

T2 → alu-a
IR[5:0] → SE → alu-b
alu-a+alu-b → alu-c
alu-c → rf-d3
IR[8:6] → rf-a3

rf-write

S8

IR [11:9] → rf-a3
IR [8:0] → LS → rf-d3

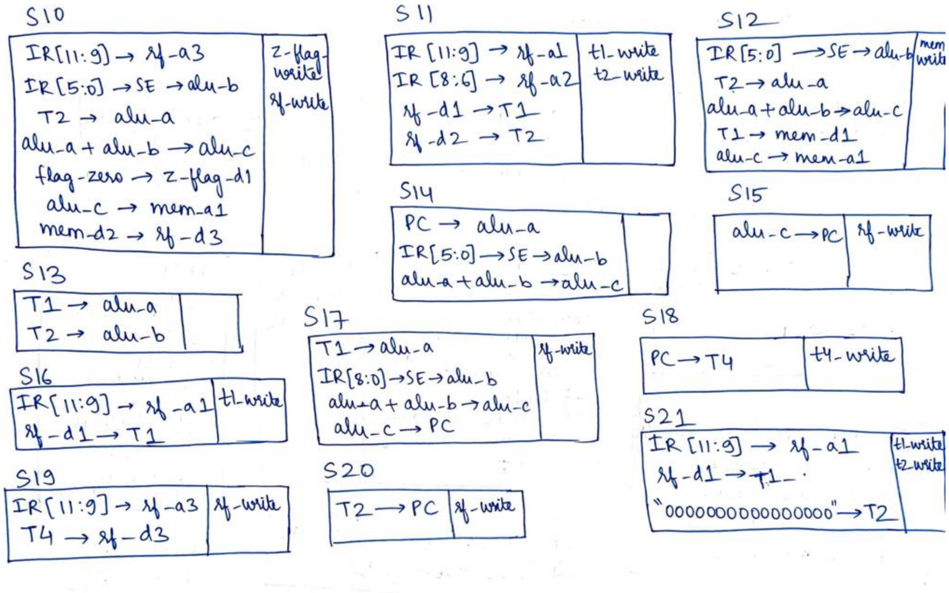
rf-write

S9

IR[8:6] → rf-a2
rf-d2 → T2

t2-write

Figure 1: 1



S10

IR[11:9] → rf-a3
IR[5:0] → SE → alu-b
T2 → alu-a
alu-a + alu-b → alu-c
flag-zero → z-flag-d1
alu-c → mem-a1
mem-d2 → rf-d3

z-flag-write
rf-write

S11

IR [11:9] → rf-a1
IR [8:6] → rf-a2
rf-d1 → T1
rf-d2 → T2

t1-write
t2-write

S12

IR[5:0] → SE → alu-b
T2 → alu-a
alu-a + alu-b → alu-c
T1 → mem-d1
alu-c → mem-a1

mem write

S13

T1 → alu-a
T2 → alu-b

S14

PC → alu-a
IR[5:0] → SE → alu-b
alu-a + alu-b → alu-c

S15

alu-c → PC

rf-write

S16

IR[11:9] → rf-a1
rf-d1 → T1

t1-write

S17

T1 → alu-a
IR[8:0] → SE → alu-b
alu-a + alu-b → alu-c
alu-c → PC

rf-write

S18

PC → T4

t4-write

S19

IR[11:9] → rf-a3
T4 → rf-d3

rf-write

S20

T2 → PC

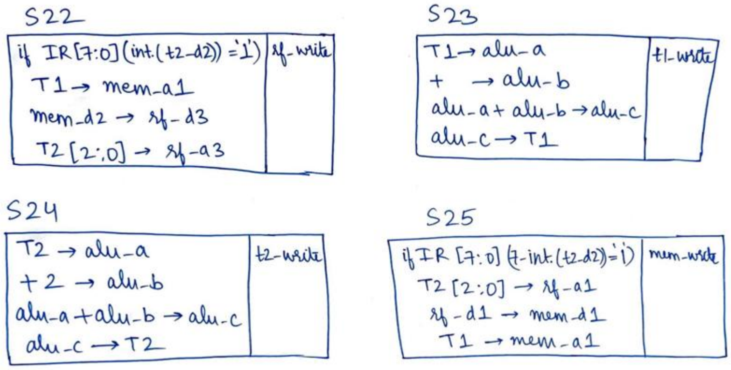
rf-write

S21

IR [11:9] → rf-a1
rf-d1 → T1
"0000000000000000" → T2

t1-write
t2-write

Figure 2: 2



S22

if IR[7:0](int.(t2-d2)) ='1') T1→ mem-a1 mem-d2 → rf-d3 T2[2:0] → rf-a3	rf-write

S23

T1→ alu-a + → alu-b alu-a+ alu-b → alu-c alu-c → T1	t1-write

S24

T2 → alu-a +2 → alu-b alu-a+alu-b → alu-c alu-c → T2	t2-write

S25

if IR [7:0] (7-int.(t2-d2))='1') T2 [2:0] → rf-a1 rf-d1 → mem-d1 T1 → mem-a1	mem-write

Figure 3: 3

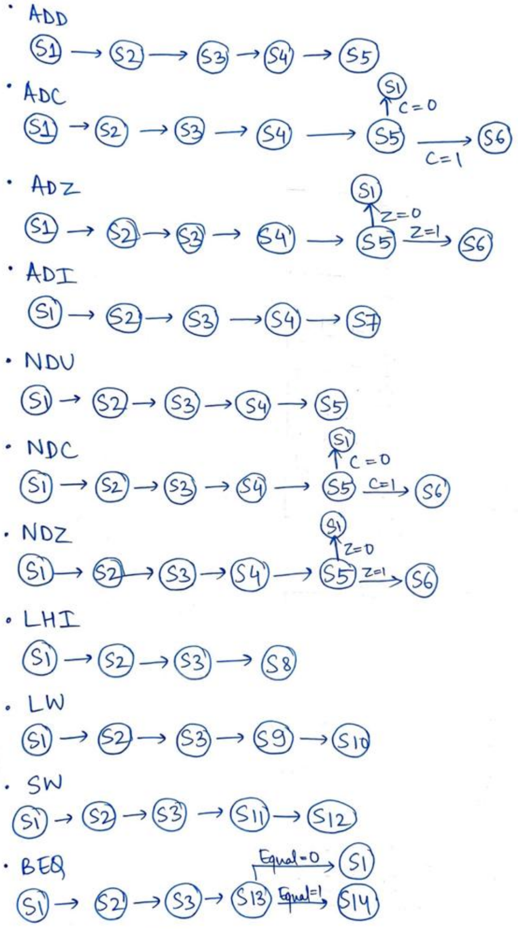


Figure 4: 4

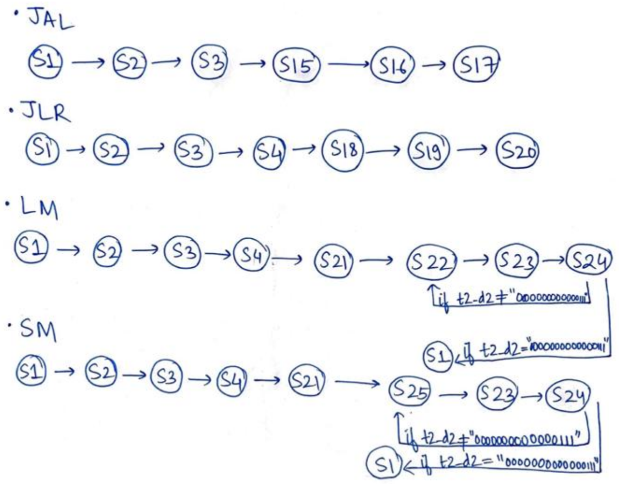

• JAL
S1 → S2 → S3 → S15 → S16 → S17
• JLR
S1 → S2 → S3 → S4 → S18 → S19 → S20
• LM
S1 → S2 → S3 → S4 → S21 → S22 → S23 → S24
if t2-d2 ≠ "0000000000000111"
S1 ← if t2-d2 = "0000000000000111"
• SM
S1 → S2 → S3 → S4 → S21 → S25 → S23 → S24
if t2-d2 ≠ "0000000000000111"
S1 ← if t2-d2 = "0000000000000111"

Figure 5: 5

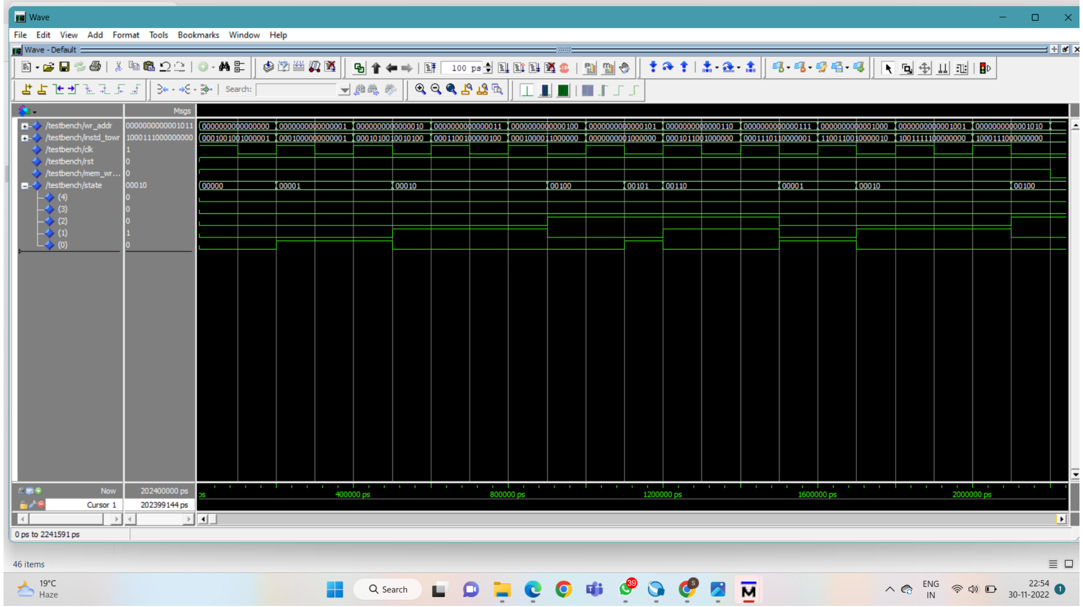


Figure 6: 6