Microprocessors Project

IITB-RISC-23

Anoushka Dey (210010010)

Aparna Agrawal (21B090004)

Apoorva Rajadhyaksha (210070066)

Shravani Kode (210070082)

Under the guidance of

Prof. Virendra Singh

Department of Electrical Engineering

Indian Institute of Technology Bombay

May 2023

Noteworthy features

- IITB-RISC-23 consists of 8 registers and is a 16-bit computer system.
- R0 stores the Program Counter
- The datapath components are Instruction Decoder, Register File, Instruction Memory, Data Memory, three 16-bit ALUs, two sign extenders SE6 and SE9 that give 16-bit outputs for 6 and 9-bit inputs respectively and a 1-bit shifter 1S.
- One 16-bit temporary registers T1 has been created to facilitate execution of the instructions.

Instructions:

ADA

$PC \rightarrow MEM1\_ADD, ALU2\_a$

$+2 \rightarrow ALU2\_b$

$ALU2\_c \rightarrow PC$

$Mem1\_D_{11-9} \rightarrow RF\_A1$

$RF\_D1 \rightarrow ALU1\_a$

$Mem1\_D_{8-6} \rightarrow RF\_A2$

$RF\_D2 \rightarrow ALU1\_b$

$ALU1\_c \rightarrow RF\_D3$

$Mem1\_D_{5-3} \rightarrow RF\_A3$

ADC

$PC \rightarrow MEM1\_ADD, ALU2\_a$

$+2 \rightarrow ALU2\_b$

$ALU2\_c \rightarrow PC$

if (C==1)

$Mem1\_D_{11-9} \rightarrow RF\_A1$

$RF\_D1 \rightarrow ALU1\_a$

$Mem1\_D_{8-6} \rightarrow RF\_A2$

$RF\_D2 \rightarrow ALU1\_b$

$ALU1\_c \rightarrow RF\_D3$

$Mem1\_D_{5-3} \rightarrow RF\_A3$

ADZ

```
$PC \rightarrow MEM1\_ADD, ALU2\_a$
$+2 \rightarrow ALU2\_b$
$ALU2\_c \rightarrow PC$
if (Z==1)
$Mem1\_D_{11-9} \rightarrow RF\_A1$
$RF\_D1 \rightarrow ALU1\_a$
$Mem1\_D_{8-6} \rightarrow RF\_A2$
$RF\_D2 \rightarrow ALU1\_b$
$ALU1\_c \rightarrow RF\_D3$
$Mem1\_D_{5-3} \rightarrow RF\_A3$
```

AWC

```
$PC \rightarrow MEM1\_ADD, ALU2\_a$
$+2 \rightarrow ALU2\_b$
$ALU2\_c \rightarrow PC$
$Mem1\_D_{11-9} \rightarrow RF\_A1$
$RF\_D1 \rightarrow ALU1\_a$
$Mem1\_D_{8-6} \rightarrow RF\_A2$
$RF\_D2 \rightarrow ALU1\_b$
if (C==1)
$\{ALU1\_c \rightarrow ALU3_a$
$+1 \rightarrow ALU3\_b$
$ALU3\_c \rightarrow RF\_D3\}$
else
$ALU1\_c \rightarrow RF\_D3$
$Mem1\_D_{5-3} \rightarrow RF\_A3$
```

ACA

$$\boxed{\begin{array}{c} PC \rightarrow MEM1\_ADD, ALU2\_a \\ +2 \rightarrow ALU2\_b \\ ALU2\_c \rightarrow PC \\ Mem1\_D_{11-9} \rightarrow RF\_A1 \\ RF\_D1 \rightarrow ALU1\_a \\ Mem1\_D_{8-6} \rightarrow RF\_A2 \\ RF\_D2 \rightarrow ALU1\_b \\ ALU1\_c \rightarrow RF\_D3 \\ Mem1\_D_{5-3} \rightarrow RF\_A3 \end{array}}$$

ACC

$$\boxed{\begin{array}{c} PC \rightarrow MEM1\_ADD, ALU2\_a \\ +2 \rightarrow ALU2\_b \\ ALU2\_c \rightarrow PC \\ \text{if (C==1)} \\ Mem1\_D_{11-9} \rightarrow RF\_A1 \\ RF\_D1 \rightarrow ALU1\_a \\ Mem1\_D_{8-6} \rightarrow RF\_A2 \\ RF\_D2 \rightarrow ALU1\_b \\ ALU1\_c \rightarrow RF\_D3 \\ Mem1\_D_{5-3} \rightarrow RF\_A3 \end{array}}$$

ACZ

```
$PC \rightarrow MEM1\_ADD, ALU2\_a$
$+2 \rightarrow ALU2\_b$
$ALU2\_c \rightarrow PC$
if (Z==1)
$Mem1\_D_{11-9} \rightarrow RF\_A1$
$RF\_D1 \rightarrow ALU1\_a$
$Mem1\_D_{5-0} \rightarrow SE-16 \rightarrow ALU1\_b$
$ALU1\_c \rightarrow RF\_D3$
$Mem1\_D_{8-6} \rightarrow RF\_A3$
```

ACW

```
$PC \rightarrow MEM1\_ADD, ALU2\_a$
$+2 \rightarrow ALU2\_b$
$ALU2\_c \rightarrow PC$
$Mem1\_D_{11-9} \rightarrow RF\_A1$
$RF\_D1 \rightarrow ALU1\_a$
$Mem1\_D_{8-6} \rightarrow RF\_A2$
$RF\_D2 \rightarrow ALU1\_b$
if (C==1)
$\{ALU1\_c \rightarrow ALU3_a$
$+1 \rightarrow ALU3\_b$
$ALU3\_c \rightarrow RF\_D3\}$
else
$ALU1\_c \rightarrow RF\_D3$
$Mem1\_D_{5-3} \rightarrow RF\_A3$
```

ADI

$$\boxed{\begin{array}{c}
PC \rightarrow MEM1\_ADD, ALU2\_a \\
+2 \rightarrow ALU2\_b \\
ALU2\_c \rightarrow PC \\
Mem1\_D_{11-9} \rightarrow RF\_A1 \\
RF\_D1 \rightarrow ALU1\_a \\
Mem1\_D_{5-0} \rightarrow SE6 \rightarrow ALU1\_b \\
RF\_D2 \rightarrow ALU1\_b \\
ALU1\_c \rightarrow RF\_D3 \\
Mem1\_D_{5-3} \rightarrow RF\_A3
\end{array}}$$

NDU

$$\boxed{\begin{array}{c}
PC \rightarrow MEM1\_ADD, ALU2\_a \\
+2 \rightarrow ALU2\_b \\
ALU2\_c \rightarrow PC \\
Mem1\_D_{11-9} \rightarrow RF\_A1 \\
RF\_D1 \rightarrow ALU1\_a \\
Mem1\_D_{8-6} \rightarrow RF\_A2 \\
RF\_D2 \rightarrow ALU1\_b \\
ALU1\_c \rightarrow RF\_D3 \\
Mem1\_D_{5-3} \rightarrow RF\_A3
\end{array}}$$

NDC

$$\boxed{\begin{array}{c} PC \rightarrow MEM1\_ADD, ALU2\_a \\ +2 \rightarrow ALU2\_b \\ ALU2\_c \rightarrow PC \\ \text{if (C==1)} \\ Mem1\_D_{11-9} \rightarrow RF\_A1 \\ RF\_D1 \rightarrow ALU1\_a \\ Mem1\_D_{8-6} \rightarrow RF\_A2 \\ RF\_D2 \rightarrow ALU1\_b \\ ALU1\_c \rightarrow RF\_D3 \\ Mem1\_D_{5-3} \rightarrow RF\_A3 \end{array}}$$

NDZ

$$\boxed{\begin{array}{c} PC \rightarrow MEM1\_ADD, ALU2\_a \\ +2 \rightarrow ALU2\_b \\ ALU2\_c \rightarrow PC \\ \text{if (Z==1)} \\ Mem1\_D_{11-9} \rightarrow RF\_A1 \\ RF\_D1 \rightarrow ALU1\_a \\ Mem1\_D_{8-6} \rightarrow RF\_A2 \\ RF\_D2 \rightarrow ALU1\_b \\ ALU1\_c \rightarrow RF\_D3 \\ Mem1\_D_{5-3} \rightarrow RF\_A3 \end{array}}$$

NCU

$$\boxed{\begin{array}{c} PC \rightarrow MEM1\_ADD, ALU2\_a \\ +2 \rightarrow ALU2\_b \\ ALU2\_c \rightarrow PC \\ Mem1\_D_{8-6} \rightarrow RF\_A1 \\ RF\_D1 \rightarrow ALU1\_a \\ Mem1\_D_{11-9} \rightarrow RF\_A2 \\ RF\_D2 \rightarrow ALU1\_b \\ ALU1\_c \rightarrow RF\_D3 \\ Mem1\_D_{5-3} \rightarrow RF\_A3 \end{array}}$$

NCC

$$\boxed{\begin{array}{c} PC \rightarrow MEM1\_ADD, ALU2\_a \\ +2 \rightarrow ALU2\_b \\ ALU2\_c \rightarrow PC \\ \text{if (C==1)} \\ Mem1\_D_{8-6} \rightarrow RF\_A1 \\ RF\_D1 \rightarrow ALU1\_a \\ Mem1\_D_{11-9} \rightarrow RF\_A2 \\ RF\_D2 \rightarrow ALU1\_b \\ ALU1\_c \rightarrow RF\_D3 \\ Mem1\_D_{5-3} \rightarrow RF\_A3 \end{array}}$$

NCZ

$$\boxed{\begin{array}{c}
PC \rightarrow MEM1\_ADD, ALU2\_a \\
+2 \rightarrow ALU2\_b \\
ALU2\_c \rightarrow PC \\
\text{if (Z==1)} \\
Mem1\_D_{8-6} \rightarrow RF\_A1 \\
RF\_D1 \rightarrow ALU1\_a \\
Mem1\_D_{11-9} \rightarrow RF\_A2 \\
RF\_D2 \rightarrow ALU1\_b \\
ALU1\_c \rightarrow RF\_D3 \\
Mem1\_D_{5-3} \rightarrow RF\_A3
\end{array}}$$

LLI

$$\boxed{\begin{array}{c}
PC \rightarrow MEM1\_ADD, ALU2\_a \\
+2 \rightarrow ALU2\_b \\
ALU2\_c \rightarrow PC \\
Mem1\_D_{8-0} \rightarrow SE9 \rightarrow RF\_D3 \\
Mem1\_D_{11-9} \rightarrow RF\_A3
\end{array}}$$

LW

$$\boxed{\begin{array}{c}
PC \rightarrow MEM1\_ADD, ALU2\_a \\
+2 \rightarrow ALU2\_b \\
ALU2\_c \rightarrow PC \\
Mem1\_D_{5-0} \rightarrow SE6 \rightarrow ALU1\_B \\
Mem1\_D_{8-6} \rightarrow RF\_A1 \\
RF_D1 \rightarrow ALU1\_a \\
ALU\_c \rightarrow MEM2\_ADD \\
MEM2\_D \rightarrow RF\_D3 \\
MEM1\_D_{11-9} \rightarrow RF\_A3
\end{array}}$$

SW

$$\boxed{\begin{array}{c} PC \rightarrow MEM1\_ADD, ALU2\_a \\ +2 \rightarrow ALU2\_b \\ ALU2\_c \rightarrow PC \\ Mem1\_D_{8-6} \rightarrow RF\_A1 \\ RF\_D1 \rightarrow ALU1\_a \\ MEM1\_D_{11-9} \rightarrow RF\_A2 \\ RF\_D2 \rightarrow MEM2\_DATA \\ Mem1\_D_{5-0} \rightarrow SE6 \rightarrow ALU1\_B \\ ALU\_c \rightarrow MEM2\_ADD \end{array}}$$

LM and SM

- The number of registers to be modified (LM) or read (SM) can be determined from the number of 1s in the last 8 bits of the instruction. This value is evaluated and stored as **count**.
- Thereafter in the ID stage, using priority encoder we generated dummy load/store instructions with the immediate given by the number of registers to be read/modified minus 1, that is **count-1**.
- We recursively generated LM/SM instructions using priority encoder and priority encoder modifier, which sets the first bit set as 1 to 0 and sent these instructions to the IF stage.

BEQ

$$\boxed{\begin{array}{c} PC \rightarrow MEM1\_ADD, ALU2\_a, T1 \\ +2 \rightarrow ALU2\_b \\ ALU2\_c \rightarrow PC \\ Mem1\_D_{11-9} \rightarrow RF\_A1 \\ RF\_D1 \rightarrow ALU1\_a \\ Mem1\_D_{8-6} \rightarrow RF\_A2 \\ RF\_D2 \rightarrow ALU1\_b \\ T1 \rightarrow ALU3\_b \\ Mem1\_D_{5-0} \rightarrow SE6 \rightarrow 1S \rightarrow ALU3\_a \\ \text{if (Z==1)} \\ ALU3\_c \rightarrow PC \end{array}}$$

BLT

$PC \rightarrow MEM1\_ADD, ALU2\_a, T1$

$+2 \rightarrow ALU2\_b$

$ALU2\_c \rightarrow PC$

$Mem1\_D_{11-9} \rightarrow RF\_A1$

$RF\_D1 \rightarrow ALU1\_a$

$Mem1\_D_{8-6} \rightarrow RF\_A2$

$RF\_D2 \rightarrow ALU1\_b$

$T1 \rightarrow ALU3\_b$

$Mem1\_D_{5-0} \rightarrow SE6 \rightarrow 1S \rightarrow ALU3\_a$

if (C==1)

$ALU3\_c \rightarrow PC$

BLE

$PC \rightarrow MEM1\_ADD, ALU2\_a, T1$

$+2 \rightarrow ALU2\_b$

$ALU2\_c \rightarrow PC$

$Mem1\_D_{11-9} \rightarrow RF\_A1$

$RF\_D1 \rightarrow ALU1\_a$

$Mem1\_D_{8-6} \rightarrow RF\_A2$

$RF\_D2 \rightarrow ALU1\_b$

$T1 \rightarrow ALU3\_b$

$Mem1\_D_{5-0} \rightarrow SE6 \rightarrow 1S \rightarrow ALU3\_a$

if (C==1 or Z==1)

$ALU3\_c \rightarrow PC$

JAL

$$\boxed{\begin{array}{c}
PC \rightarrow MEM1\_ADD, ALU2\_a, T1 \\
+2 \rightarrow ALU2\_b \\
Mem1\_D_{8-0} \rightarrow SE9 \rightarrow 1S \rightarrow ALU3\_a \\
T1 \rightarrow ALU3\_b \\
ALU3\_c \rightarrow PC \\
ALU2\_c \rightarrow RF\_D3 \\
MEM1\_D_{11-9} \rightarrow RF\_A3
\end{array}}$$

JLR

$$\boxed{\begin{array}{c}
PC \rightarrow MEM1\_ADD, ALU2\_a, T1 \\
+2 \rightarrow ALU2\_b \\
Mem1\_D_{8-0} \rightarrow SE9 \rightarrow 1S \rightarrow ALU3\_a \\
ALU2\_c \rightarrow PC \\
Mem1\_D_{8-6} \rightarrow RF\_A2 \\
RF\_D2 \rightarrow PC \\
T1 \rightarrow ALU1\_a \\
+2 \rightarrow ALU1\_b \\
MEM1\_D_{11-9} \rightarrow RF\_A3 \\
ALU1\_c \rightarrow RF\_D3
\end{array}}$$

JRI

$$\boxed{\begin{array}{c}
PC \rightarrow MEM1\_ADD, ALU2\_a \\
+2 \rightarrow ALU2\_b \\
ALU2\_c \rightarrow PC \\
Mem1\_D_{11-9} \rightarrow RF\_A1 \\
RF\_D1 \rightarrow ALU1\_a \\
Mem1\_D_{8-0} \rightarrow SE9 \rightarrow 1S \rightarrow ALU1\_b \\
ALU1\_c \rightarrow PC
\end{array}}$$

Hazard Management

Branch Instructions

- Branches are resolved at the end of the execute (**EX**) stage of the pipeline.
- If the branch is taken, we will have to eliminate the instructions in instruction fetch, instruction decode, register read stages. This is achieved through '**throwing**'.
- There are 5 pipelines each between two consecutive stages, namely Pipeline 1 between Instruction Fetch and Instruction Decode, Pipeline 2 between Instruction Decode and Register read, and so on. We have provided control signals that enable/disable each pipeline register and a write signal to enable writing into it.
- Suppose when a branch instruction is in EX stage, I1 is in Instruction Fetch, I2 is in Instruction Decode and I3 is in Register Read stage. If the branch is determined to be taken at the end of that clock cycle, we disable the write signal of Pipeline 2 and 3, and since Pipeline register 1 doesn't have a write signal, we introduce a dummy instruction I1' with unused Opcode 1110, for which all control signals are disabled.
- This ensures that I1, I2, I3 do not carry forward to the next stages of the pipeline. Simultaneously, if the next instruction in the taken branch is I4, we load I4 into IF.
- This is how we are **throwing** instructions I1, I2 and I3 and resolving the branch hazard.

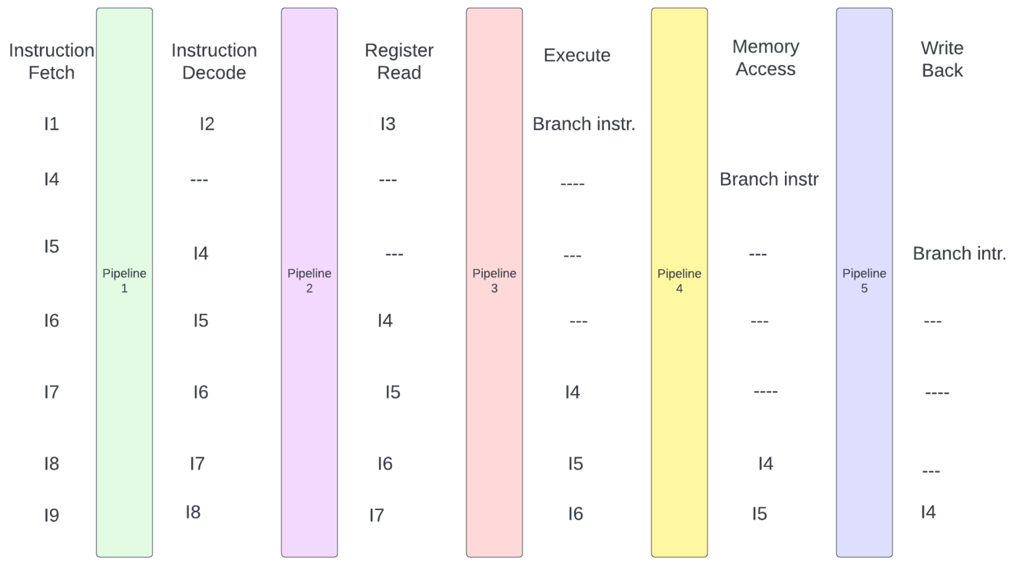

Figure 1: Throwing

Load instructions

- When a load instruction is encountered, the pipeline stages are **stalled**, and a copy of the instruction is thrown away. This is accomplished by disabling the enable signals of the stalled stages.
- To handle the load instruction hazard, a variable count is assigned, which remains zero for the first cycle. This variable is used to set the PC mux signal, which determines the starting value of the PC.
- After the initial cycle, the PC mux signal is set based on the instruction in stage 4 or stage 1.
- PC intermediate 2 refers to the output of ALU2, which is used to set the PC mux signal. The PC mux signal 011 signifies PC+2.
- To avoid any errors in the execution, it is essential to check if the PC mux signal is not equal to 011 for the PC enable signal.

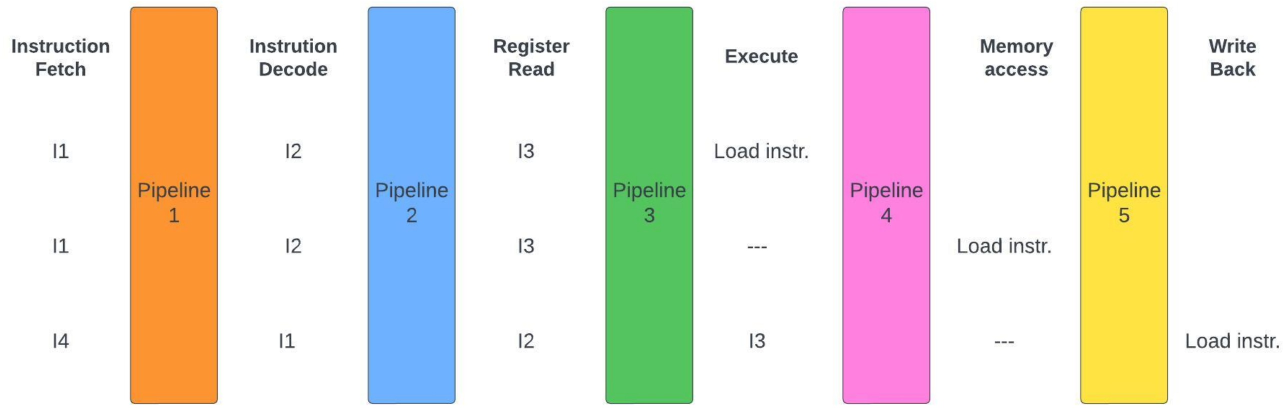

Figure 2: Stalling

Pipeline registers

Pipeline Register	Number of bits
PP1	32
PP2	46
PP3	78
PP4	90
PP5	56

	PP1	PP2	PP3	PP4	PP5
15-0	Instruction	Instruction	Instruction	Instruction	Instruction
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					
28					
29					
30					
31	PC out	PC out	PC out	PC out	PC out
32		PC_en	PC_en	Z-en	Z-en
33		Z-en	Z-en	C_en	C_en

34		C_en	C_en	RF_wr	current_zero
35		Az9	Az9	mem_data_rd	current_carry
36		OneS_en	OneS_en	mem_data_wr	RF_wr
37		SE6_en	SE6_en		
38		SE9_en	SE9_en		
39		RF_wr	RF_wr		A3
40		mem_data_rd	mem_data_rd		
41		mem_data_wr	mem_data_wr		
42					
43		ALU1_INST	ALU1_INST		
44		Mux2	prev_zero		
45		Mux1	prev_carry		
46					
47					
48					
49					
50					
51					
52				D1	
53					
54					
55					D3
56					
57					
58					
59					
60					
61			D1		
62					
63					
64					
65					
66			D2	temp_D3	

67					
68					
69					
70					
71					
72					
73					
74					
75					
76					
77					
78					
79					
80					
81					
82					
83					
84				ALU1 c	
85				current zero	
86				current_carry	
87					
88					
89				A3	

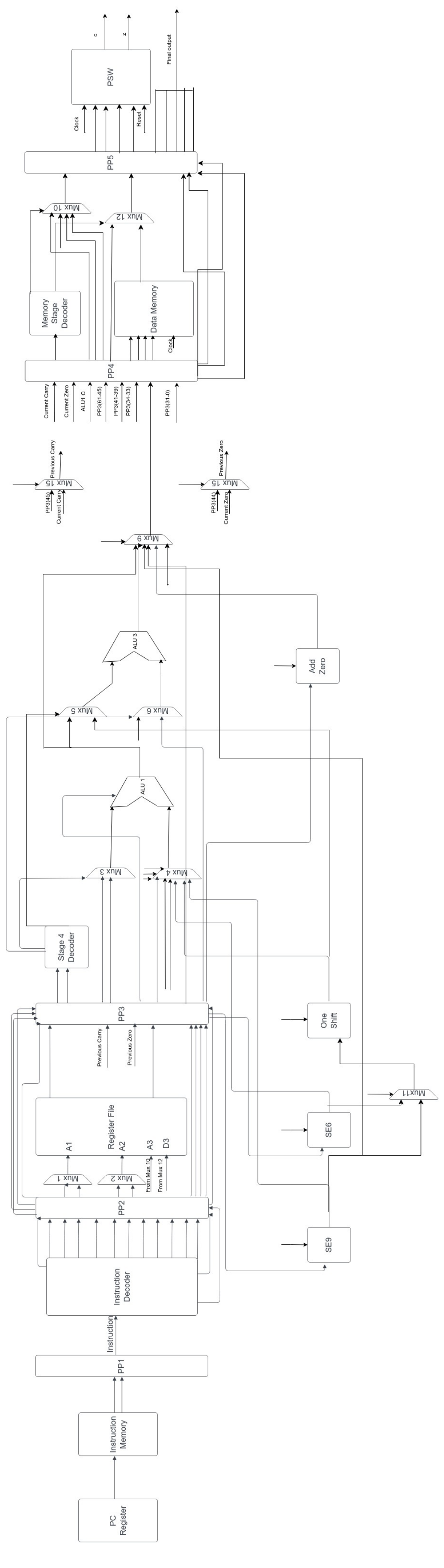

Figure 3: Datapath