Assignment on **8-bit MIPS Pipelined Execution**

Labgroup B1
Group No: 6
Group Members:
1705077
1705085
1705087
1705088
1705089

July 4, 2021

Contents

1 Introduction

In this assignment, we have designed an 8-bit processor that supports pipelined datapath. It will take MIPS instructions converted to machine code(by our program) and execute those instructions with better throughput than a single cycle implementation.

In this design, each instruction is divided into five stages: instruction fetch (IF), instruction decode (ID), execution and address calculation (EX), data memory access (MEM), and write back (WB).

Therefore, it will take five clock cycles to complete each instruction, where the length of each clock equals to the maximum time to execute a single stage.

The main components of the processor are as follows: instruction memory, data memory, register file, ALU, control unit, five pipeline registers, and a forwarding unit.

2 Instruction Set

In this assignment we have considered only 4 **R type** instructions:-

OpCode	Instruction ID	Category	Type	Instruction
0010	B	Arithmetic	R	sub
1001	A	Arithmetic	R	add
1010	D	Logic	R	or
1111	C	Logic	R	and

So, we implemented each instruction with 20-bits long data having the given formula:

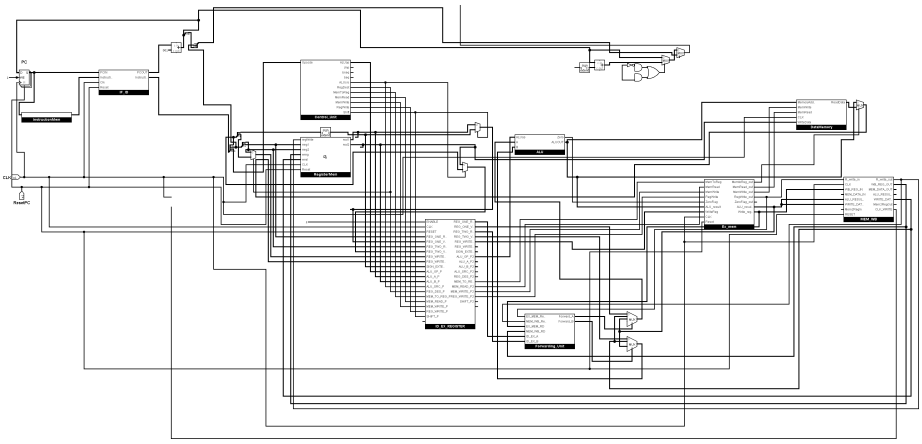
Opcode	Src Reg 1	Src Reg 2	Dst Reg	Shft Amnt
4 bits	4 bits	4 bits	4 bits	4 bits

We have the following numbering for addressing registers on the Register File:

Register	Corresponding Register File Address
$t0	0000
$t1	0001
$t2	0010
$t3	0011
$t4	0100
$Zero	0101
$sp	0110

3 High-Level Block Diagram

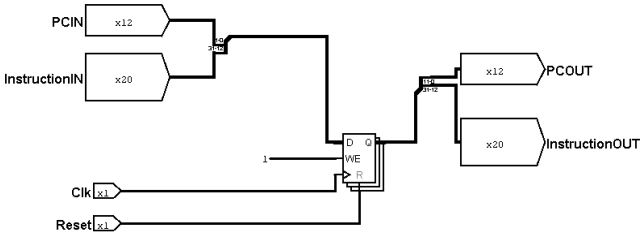
4 Complete Block diagram of an 8-bit MIPS processor with pipelined registers



5 Block diagrams of the pipelined registers

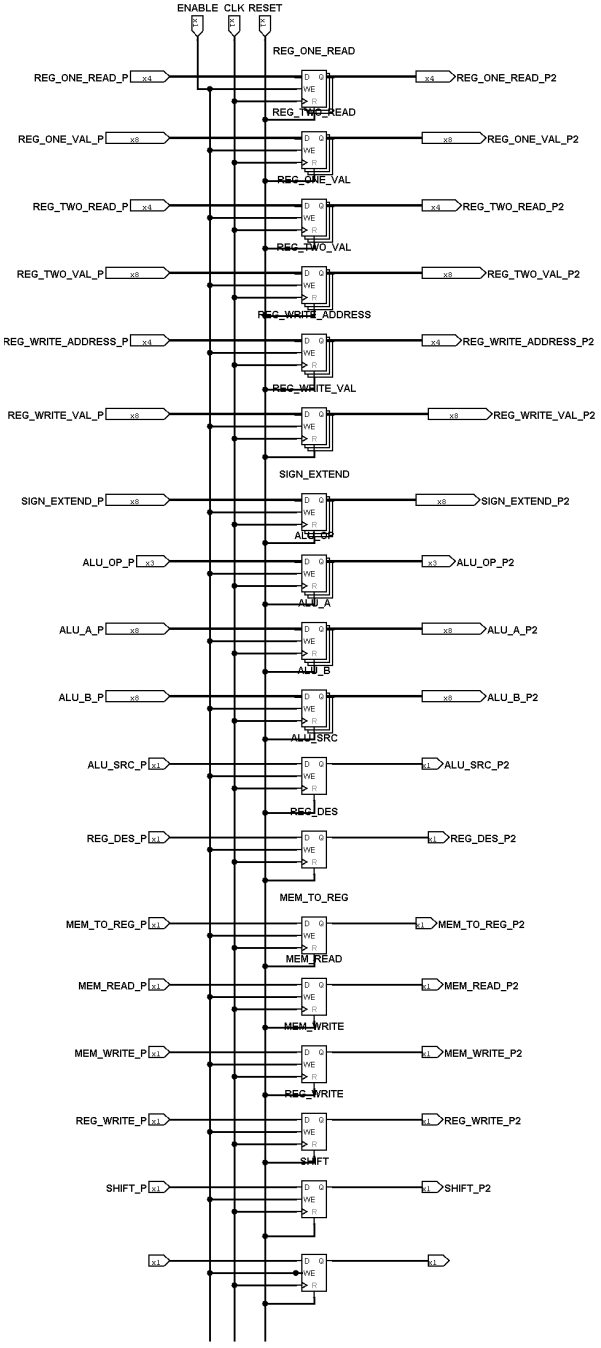
5.1 IF/ID

Size of this register is 32 bit.



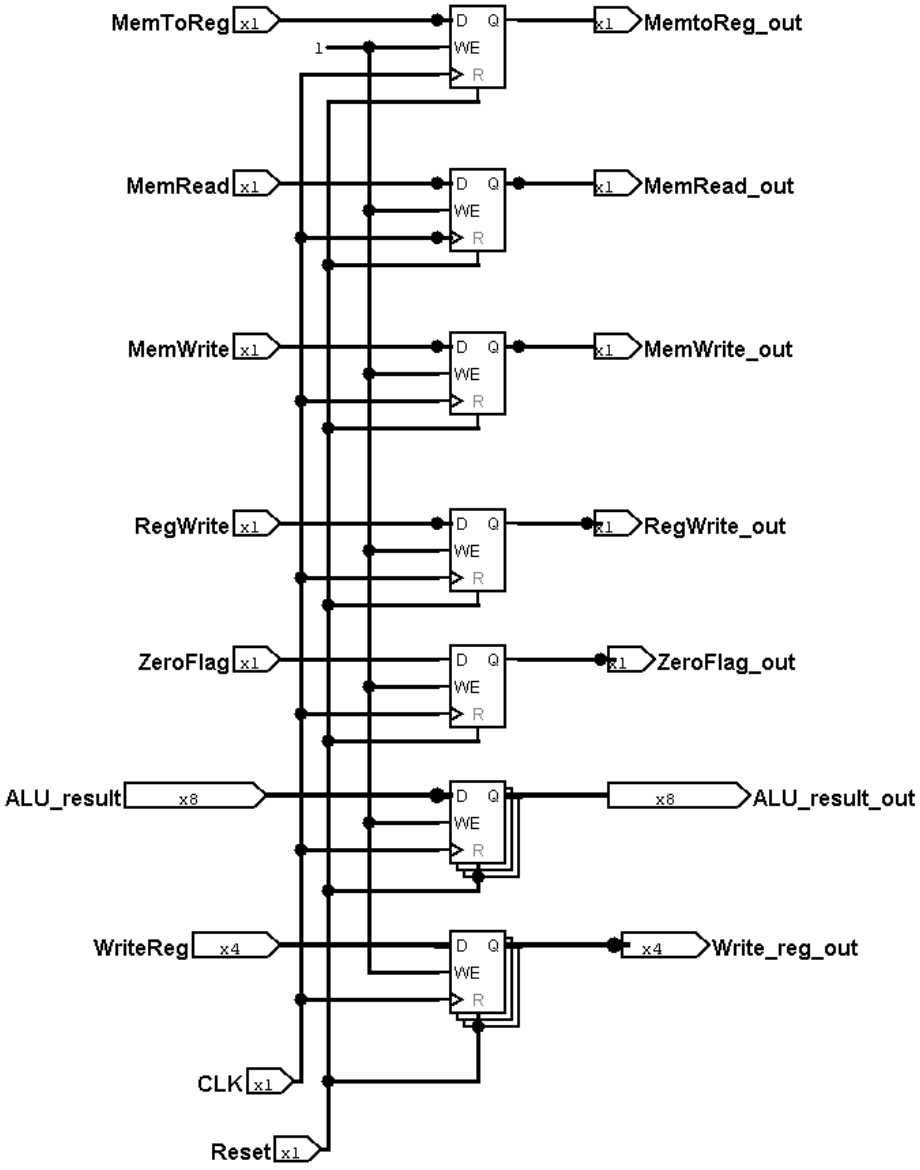
5.2 ID/EX

Size of this register is 71 bit.



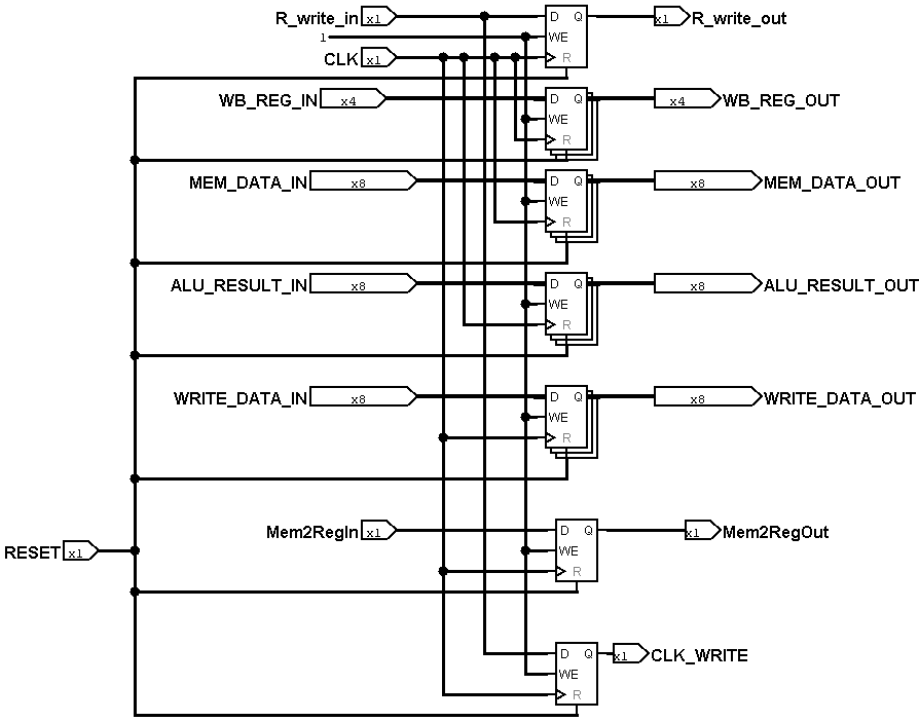
5.3 EX/MEM

Size of this register is 17 bit.

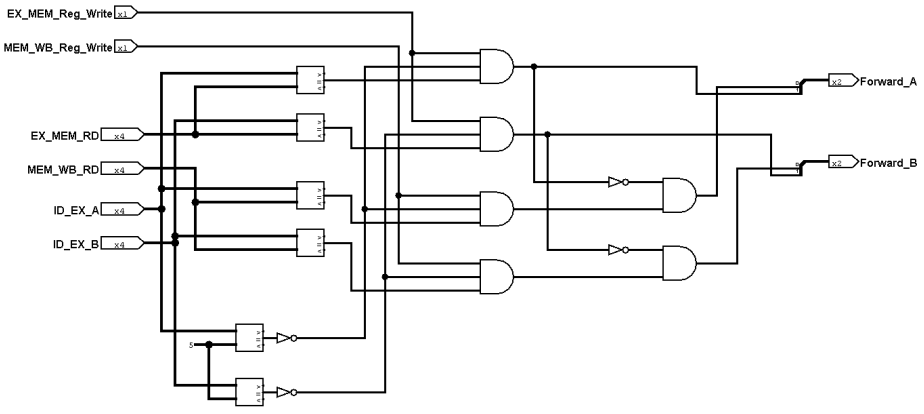


5.4 MEM/WB

Size of this register is 31 bit.



5.5 Forwarding Unit



6 Mechanism of forwarding unit

The forwarding unit takes the registers from EX/MEM and MEM/WB and compare it with the registers of from ID/EX stages. And decides from which level the register value should be passed to ALU. The logic is:

```
if(EX/MEM.RegRD == ID/EX.Reg) MUX selects EX/MEM.RegRD.val
elif(MEM/WB.RegRD == ID/EX.Reg) MUX selects MEM/WB.RegRD.val
```

This help us to overcome the Data Hazard.

7 About The Simulator

The simulator used for making this design is Logisim EVO 3.5. It is a free software for simulating digital logic circuits. It was downloaded from: https: //sourceforge.net/projects/circuit/

8 Discussion

We have used the pipeline registers to improve the throughput instruction. But since the pipeline registers can introduce data hazards, we have used a Forwarding unit that determines the from where the register value will be passed.