Project Report

MIPS-32 PIPELINED PROCESSOR

Submitted by

Name	**Roll Number**
Souvik Malakar	213079026
Shivajee Tiwari	21307R013

Contents

1 Introduction

A MIPS32 processor is a type of microprocessor architecture that emphasizes simplicity and efficiency in its design. The key idea behind RISC processors is to use a small set of simple and highly optimized instructions that can be executed in a single clock cycle. This approach aims to maximize the performance of common operations while minimizing the complexity of the processor's internal logic.

RISC processors prioritize simplicity, regularity, and efficiency in their design. By focusing on a reduced set of instructions optimized for fast execution, RISC architectures achieve high performance and are well-suited for a wide range of applications.

2 ISA

The ISA as per the problem statement is shown below. It has 3 instruction formats (R, I and J) and a total of 15 instructions taken from MIPS 32.

R Type Instruction format

Opcode (6 bit)	Register A (RA) (5-bit)	Register B (RB) (5-bit)	Register C (RC) (5-bit)	(11 - bits) (reduntant)

I Type Instruction format

Opcode (6 bit)	Register A (RA) (5 bit)	Register C (RC) (5-bit)	Immediate (16 bits signed)

J Type Instruction format

Opcode (6 bit)	Immediate (26 bitssigned)

Figure 1: Instruction Set

The datapath is conceived using Hardware flowchart and evolves according to the instruction requirement. The implementation for the instructions are shown below.

2.1 Instruction Encoding :

ADD:	001110	RA	RB	RC	(11 - bits)
SUB:	000001	RA	RB	RC	(11 - bits)
MUL:	000010	RA	RB	RC	(11 - bits)
OR:	000011	RA	RB	RC	(11 - bits)
AND:	000100	RA	RB	RC	(11 - bits)
SLT:	000101	RA	RB	RC	(11 - bits)
					I-Type Instruction
ADDI:	001000	RA	RB	16 bit Immediate	
SUBI:	001001	RA	RB	16 bit Immediate	
SLTI:	001010	RA	RB	16 bit Immediate	
BEQ:	001011	RA	RB	16 bit Immediate	
BNEQ:	001100	RA	RB	16 bit Immediate	
			J- TYPE Instruction		
JMP:	001101	26 – Bits Immediate			
HLT:	111111	26 – Bits (Reduntant)			

3 Implementation

3.1 Overview

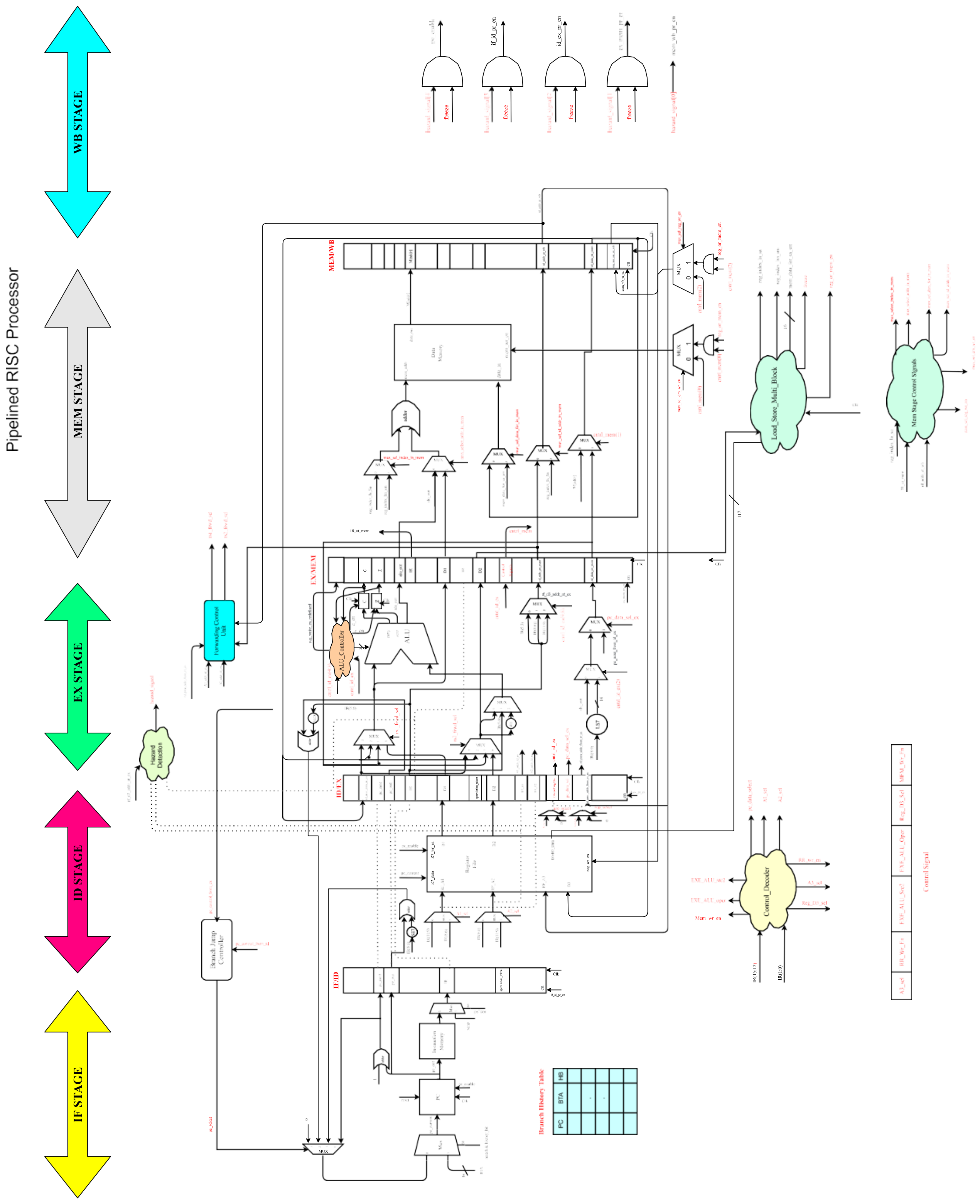
This project implements a 5 stage Pipelined Processor, MIPS32. It follows the standard 5 stage pipelines (Instruction fetch, instruction decode register read, execute, memory access, and write back).
It has 32 general purpose registers (R0 to R31). It has 15 instructions .

The architecture is optimized for performance, it has Forwarding Mechanism to limit the stalls due to RAW Hazards and to improve performance we have also implemented hazard techniques for Data , jumps , branches and Loads.

Implementing a MIPS32 processor involves designing and building the hardware components that make up the processor, as well as creating the control logic and instruction set architecture. Here is a high-level overview of the steps involved in implementing a MIPS32 pipeline processor:

- **Datapath**
- **Stages in Pipeline**
- **Interface Registers in Pipeline**
- **Hazard Techniques**

3.2 Complete Datapath



3.3 Stages in the Pipeline

- **Instruction Fetch Stage**

The first stage in the pipeline, that fetches the four byte instruction from the memory. Also consists of a PC Selector MUX, which selects between PC+1 or various other Target address based on the address resolved from various hazards mitigation logic (unconditional jump , conditional branch ,raw data dependency and also immediate load data dependency).

- **Instruction Decode + Operand Read Stage**

The second stage in the pipeline, decodes the operand address, the target destination address and type of instructions. This stage also read the operand from the Register File.

- **Execution Stage**

The third stage in the pipeline, based on data forwarding logic ,ALU got operand either of register file that was read in ID stage or from the instruction those are still in the pipeline.

- **Memory Stage**

Load and store instructions only utilizes this stage. Load data comes into pipeline(interface register) and the store data got stored into memory.

- **Write Back Stage**

In this final stage all the output data of instruction got written into their corresponding destination.

3.4 Signals present in the Pipeline Registers

- **IF_ID Register**
  - **PC, IF_ID_NPC, IF_ID_IR**
- **ID_EX Register**
  - **ID_EX_IR, ID_EX_NPC, ID_EX_A,ID_EX_B, ID_EX_Imm, ID_EX_type**
- **EX_MEM Register**
  - **EX_MEM_IR, EX_MEM_ALUout , EX_MEM_B, EX_MEM_type**
- **MEM_WB Register**
  - **MEM_WB_IR , MEM_WB_ALUout, MEM_WB_LMD, MEM_WB_type**

3.5 Hazards Mitigation

Hazard mitigation is the most crucial part in pipeline processor design. We may encounter following Hazards,

- **<u>Conditional Branch</u>:** Branch instructions control the flow the programme based on some certain condition. So we may get a whole set wrong instruction in the pipeline as new PC value will be updated only after WB stage causing 5 cycle of penalty. But this condition got resoved in EX stage only, so we forward the new PC value from that stage only hence we get only 2 cycle of penalty.

- **<u>Data Hazard</u>**: If any previous instruction that are still in pipeline is updating any operand of current instruction then we may face data hazard. To resolve it we do data forwarding to ALU from the past instructions which are in later stages.

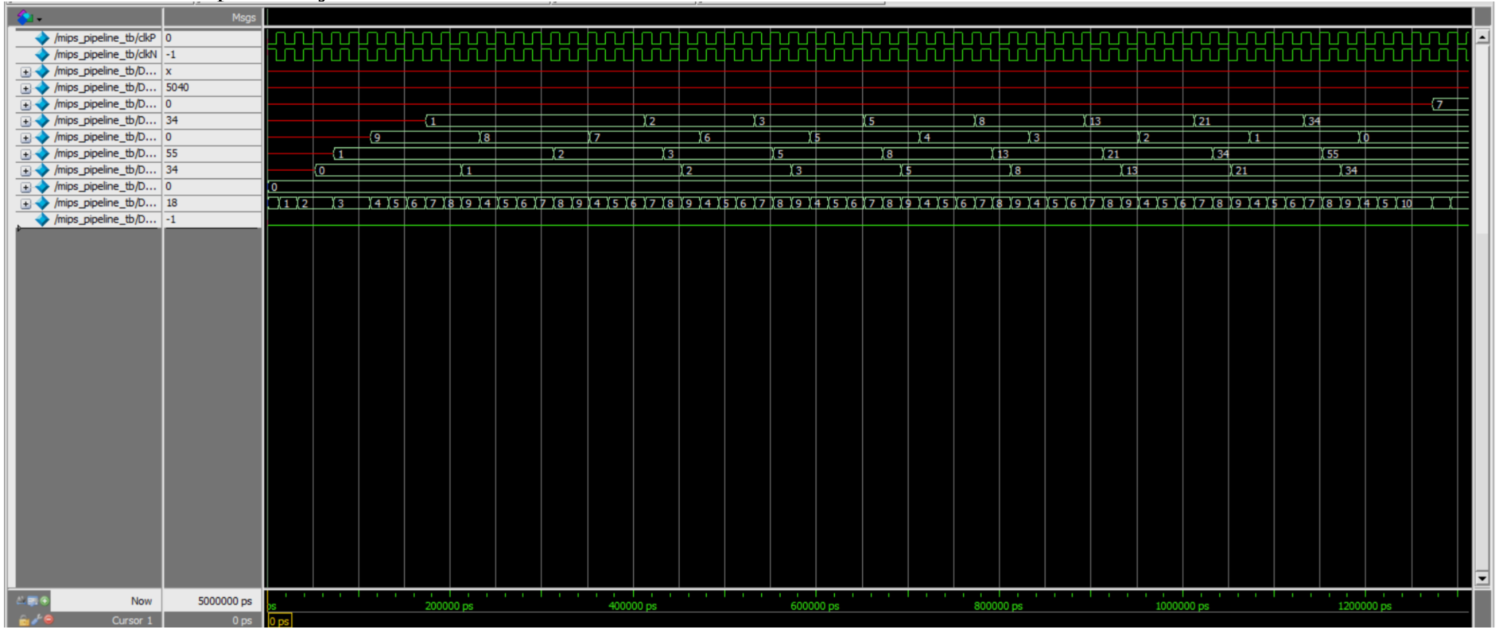
- **<u>Immediate Load Data Dependency Hazard</u> :** If any operand register is being loaded from memory in its immediate previous LW instruction then we can not just simply do data forwarding there. We must have a single cycle penalty in this case.

- **<u>Unconditional Jump</u>** : Unconditional jump destination got reserved in decode stage it self. So in the next cycle PC points to that location only. So Unconditional jump does not cause any single penalty.

• <u>Experimental</u> <u>Results</u> :

We are demonstrating below two (Fibonacci, Factorial) of the programs those were being executed in our implemented processor,

Program is to calculate the Fibonacci series upto 11th term and Factorial of 7. The expected output should be 55& 5040 respectively.



	Msgs
/mips_pipeline_tb/clkP	0
/mips_pipeline_tb/clkN	-1
/mips_pipeline_tb/D...	x
/mips_pipeline_tb/D...	5040
/mips_pipeline_tb/D...	0
/mips_pipeline_tb/D...	34
/mips_pipeline_tb/D...	0
/mips_pipeline_tb/D...	55
/mips_pipeline_tb/D...	34
/mips_pipeline_tb/D...	0
/mips_pipeline_tb/D...	18
/mips_pipeline_tb/D...	-1

Fig. : Simulation Result (Fibonacci - 10)

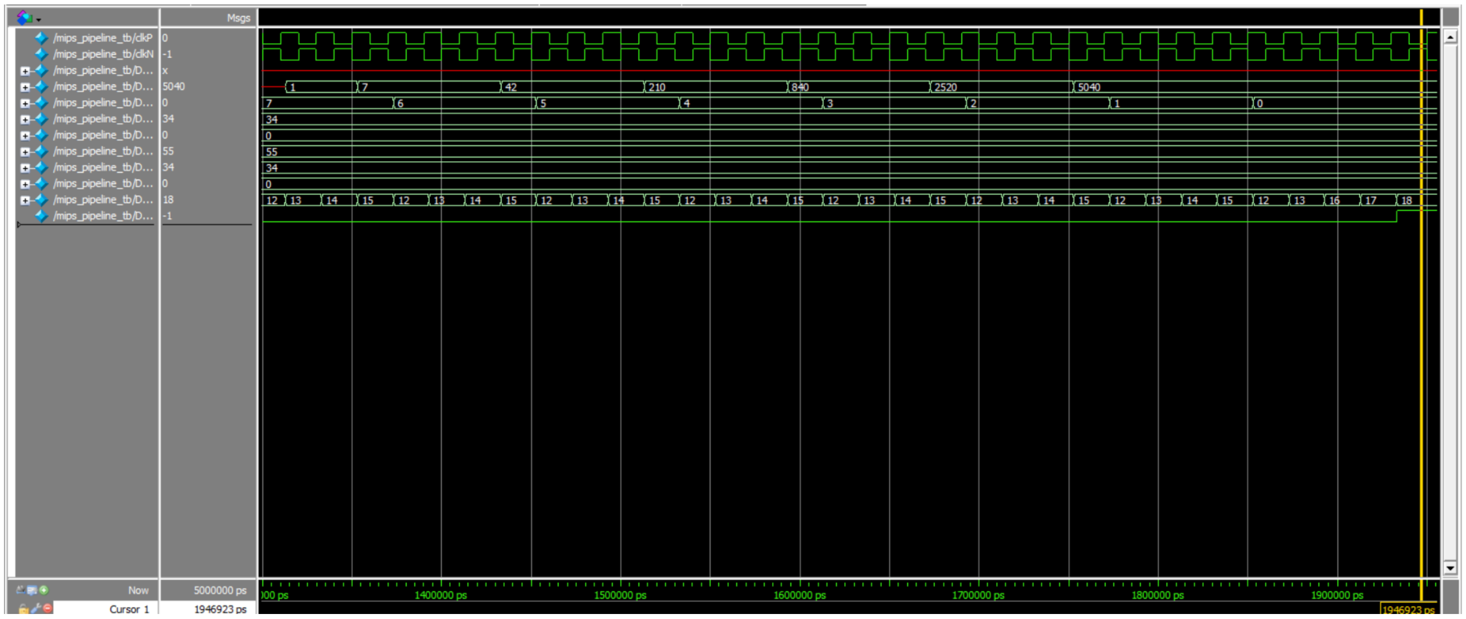


Figure : simulation result (Factorial)

- **<u>Observation</u>:**

No. of Instructions → Fibonacci – 3+ 6*9 +1 = 58
Factorial -- 2 + 4*7 + 1= 31
And 1 Halt Instruction

⇨ Total instructions = 90
Cycle time = 20ns
Latency of 1st Instruction = 50ns = 2.5 Cycle , as we are using 2 mutually complemented clock for each consecutive stages

Total execution time → 1930 ns = 96.5 cycles so rest of 89 Instructions are taking 94 cycles
This 5 cycles of penalty is due to, Branch instruction in Fibonacci and factorial program causes 2 cycles of penalty each during exit from loop and another 1 cycle of penalty we are getting due to immediate Load data dependency between instruction 3 and 4.

Average CPI = 96.5 / 90 = 1.072
Performance = 1/ (no. of instruction*CPI *cycle time) =1/(90 * 1.072 * 20 *10^-9)=518.242*10^3 s^(-1)
MIPS Rating : 1/(CPI *cycle time* 10^6) = 46.642 MIPS

4 Conclusion

A 5 stage, 32 bit pipelined processor that supports 15 instructions and optimized for performance having data forwarding and a HAZARD Mitigation Unit have been implemented. The design have been synthesized in Quartus 18.1 and RTL simulations have been carried out in Altera-ModelSim.