October 16, 2015 (Late Submission)
Ke (kq32@cornell.edu) | Joseph (jrl336@cornell.edu)

Pipelined Mini-MIPS for CS 3410

Design Documentation

---

1 Introduction

In this project, we will be building a 32-bit pipelined MIPS processor to perform a subset of the MIPS ISA.

The processor supports the instructions in the table below:

Immediate arithmetic	ADDIU, ANDI, ORI, XORI, SLTI, SLTIU
Register arithmetic	ADDU, SUBU, AND, OR, XOR, NOR, SLT, SLTU
Move	MOVN, MOVZ
Shifts (constant and variable)	SLL, SRL, SRA, SLLV, SRLV, SRAV
Immediate load	LUI
Jumps (with one delay slot)	J, JR, JAL, JALR
Branches (with one delay slot)	BEQ, BNE, BLEZ, BGTZ, BLTZ, BGEZ
Memory Load/Store (little endian, using stall)	LW, LB, LBU, SW, SB

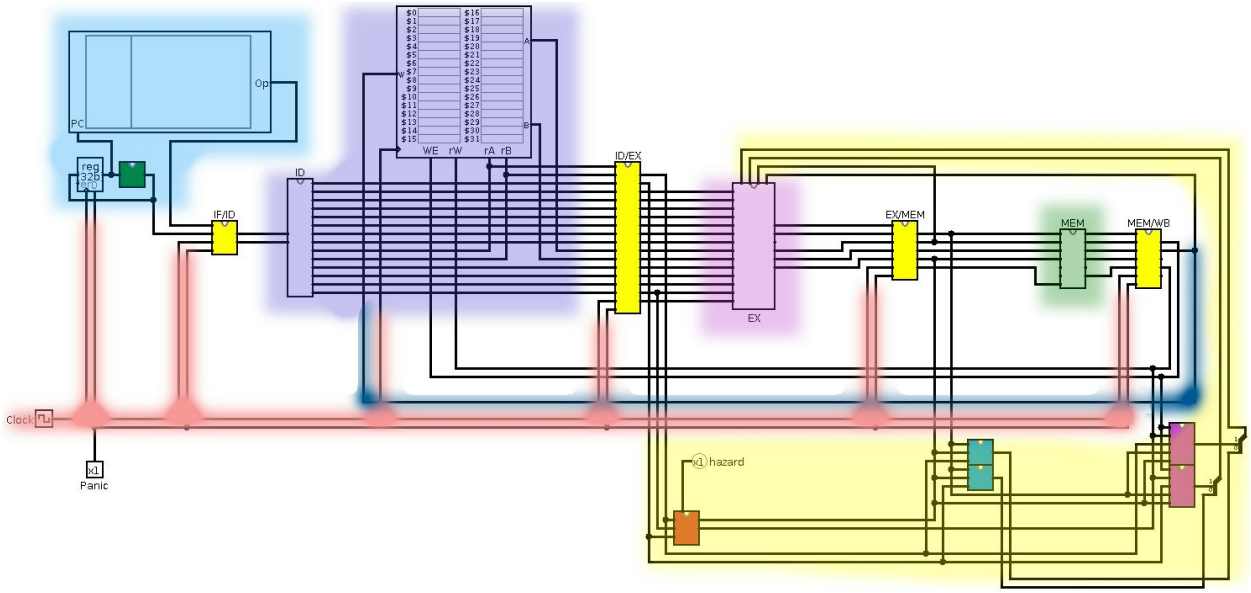
We created our pipeline by dividing the full cycle into 5 stages, as seen in lecture. We placed pipeline registers between each stage to store the information necessary to complete the next stage.

This documentation starts with an overview of the processor.

Ke (kq32@cornell.edu) | Joseph (jrl336@cornell.edu)

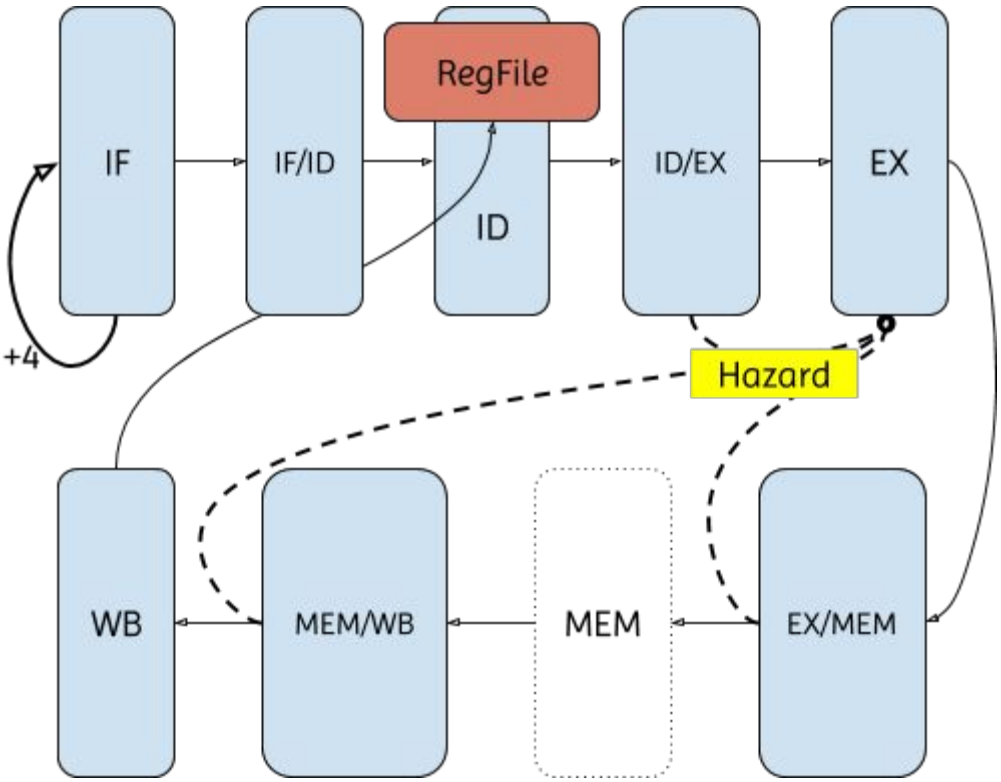
2 Overview

2.1 High Level Circuit Diagram (Color Overlay)



Instruction Fetch	Fetch instruction from instruction memory. Increment PC + 4.
Instruction Decode	Parse ALU OpCode, select bits, and fetch operands from register file. Passes addresses.
Instruction Execute	Numerical, logical and address computation.
Memory	Used by LW / SW instructions, not implemented.
Write Back	Write back to register if write enabled.
Clock	Clock signal shared by pipelines, PC, and register file.
Hazard Logic	Detect hazard. Select between register read, EX forward and MEM forward.
Hazard Detect	Detect hazard.
EX Forward Unit	Generate select bit 0 for forward muxes.
MEM Forward Unit	Coupled with EX Forward Unit, generate select bit 1 for forward muxes.
Pipelines	IF/ID; ID/EX; EX/MEM; MEM/WB
PC+4	

2.2 Pipeline and Stages



2.3 Pipeline Stages (Walkthrough)

- IF Stage:
  - Fetch new instruction to run through the cycle
  - Write instruction bits and PC+4 to IF/ID Register
  - Increment the PC
- ID Stage:
  - Read IF/ID for instruction bits
  - Decode instruction and generate control signals
  - Read from register file and write to ID/EX Register
  - Write control signals, AddrRs, Rs, AddrRt, Rt, Rd, Imm, SA, PC+4 to ID/EX Register
- Ex Stage:
  - Implement hazard logic, using AddrRs, AddrRt, and Rd to determine hazard
  - Perform ALU operations, comparisons, etc., based on OpCode and control signals

2.4 Decoding Logic

DecodingLogic subcircuit

OpCode / Func [6]	**Rtype**	RegDst	RegWrite	ALUSrc	ALUOp [4]	Sign	Comp	Move	VS	Lui

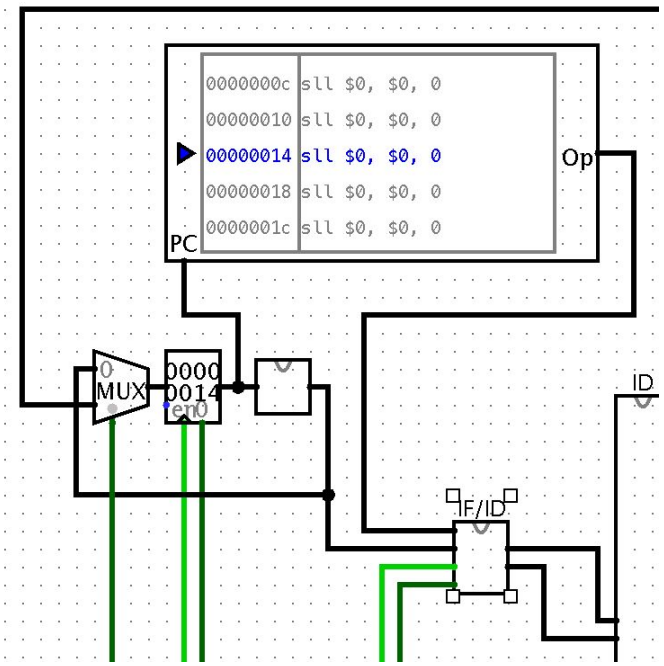
In ID Stage

Instruction [32]	RegWrite	ALUSrc	ALUOp [4]	Sign	Comp	Move	VS	Lui	Rs	Rt	SA	Imm	Rd

See Section 4.

3 The Fetch Stage [IF]

3.1 Circuit Diagram



Description: The PC is incremented by 4 unless a jump is occurring, which will cause the next PC value to be the calculated JumpAddress, (Jump control bit)

3.1.1 ROM (Instruction Memory)

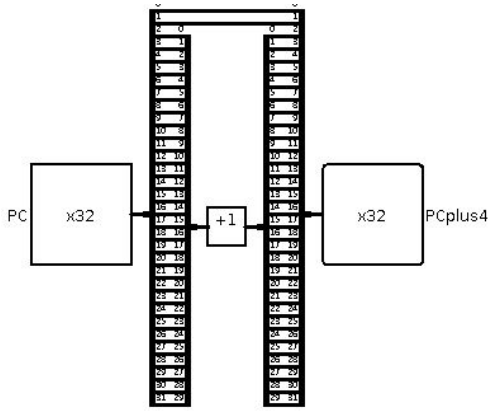
This is the ROM provided in JAR. Click on the ROM to load test programs.

3.1.2 PC

A small register that holds the current instruction address.

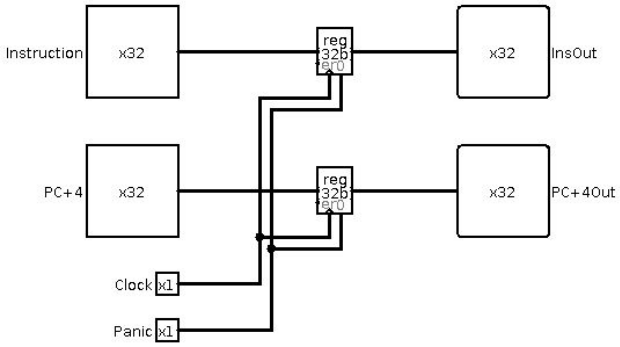
3.1.3 PC+4

An incrementer that increments [MSB ... Bit 2] of an address, equivalent to PC + 4.



3.1.4 IF/ID Register

A pipeline register that stores 1. PC+4, and 2. instruction.



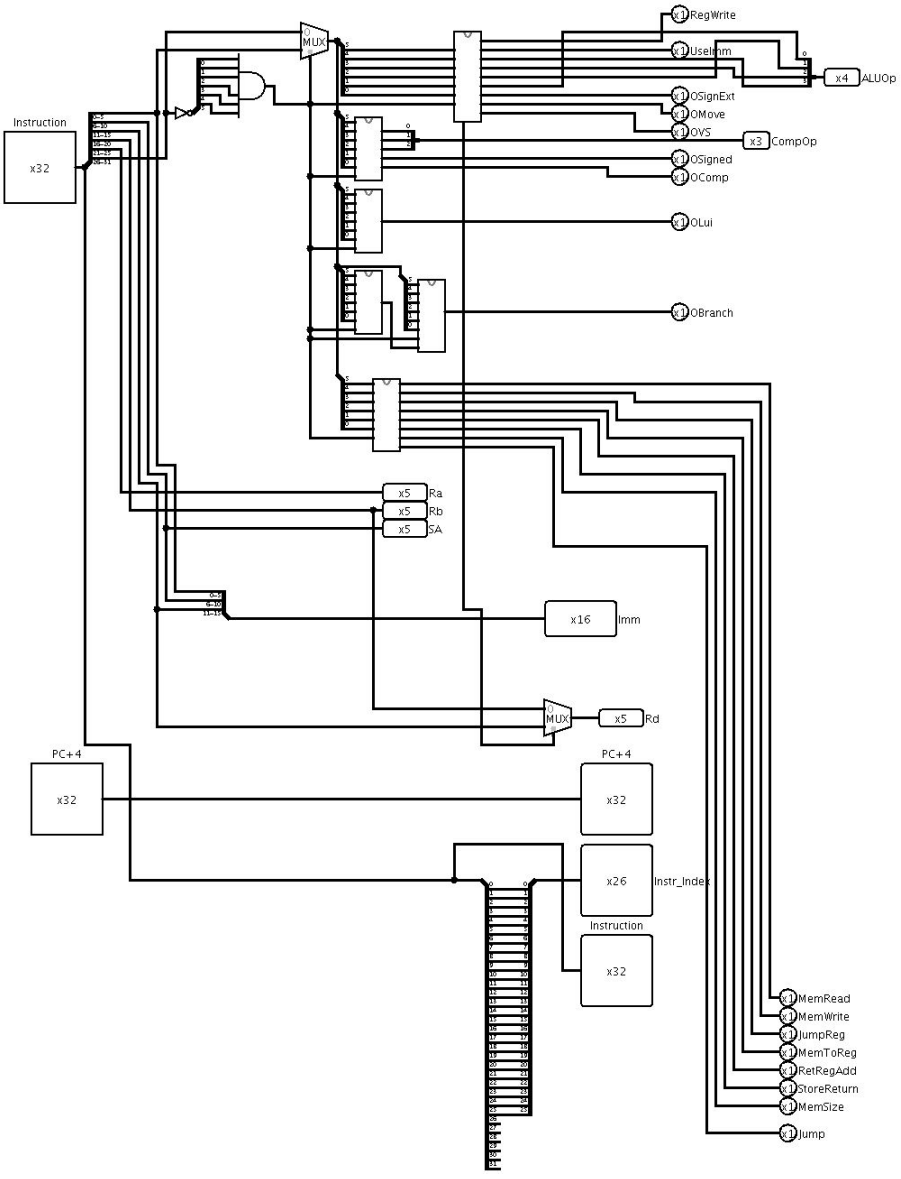
3.2 Test Phase

We will create a bunch of test instructions for each type. For example, we will perform an addition on two known values from the register file and compare the result. We will work on this project in stages. First, we will decide which OpCodes will represent which operation. Then we will also decide how to turn the code into varying control signals. We already know the fundamentals of a lot of this stuff, we just need to put everything together.

4 The Decode Stage [ID]

Here we decode the 32 bit MIPS instruction. All the necessary information for R and I type instructions are taken and put into the ID/EX register. We choose whether to look at the OpCode or Function Field to generate the control signals by checking for a 000-000 OpCode. We then put the OpCode or Function Field into a decoder (called DecodingLogic) with a 0 or 1 appended to the 6 bit value (append 0 for I-type and 1 for R-type) to produce all the control signals.

4.1 Circuit Diagram



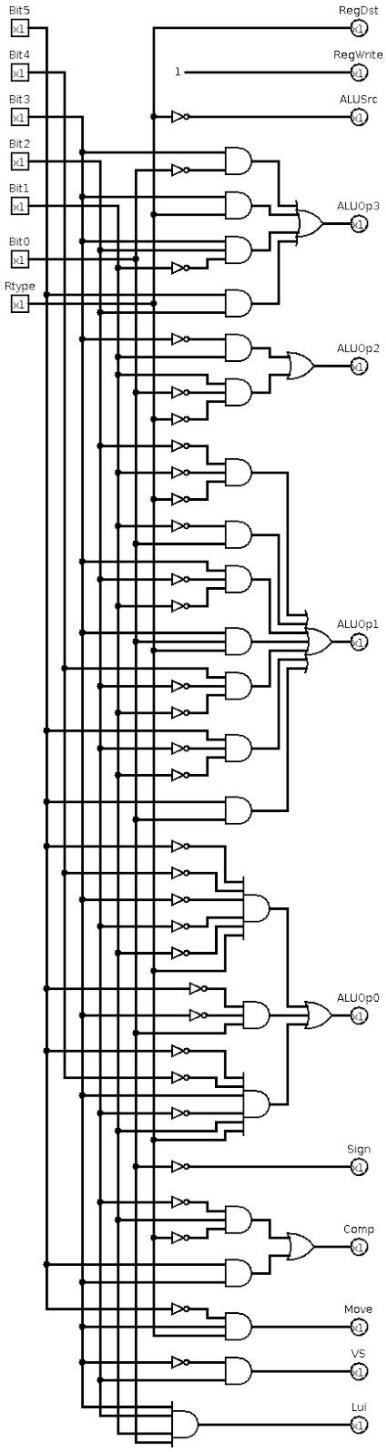
Description: This stage uses several decoding modules (DecodingLogic, DecodingLogicComp, DecodingLogicLui, DecodingREGIMM, DecodingBranch, DecodingJumps/Mem) to produce

the control bits and also takes the important information from the instruction to pass to Execute

4.1.1 IF/ID Register

See 3.1.4

4.1.2 DecodingLogic



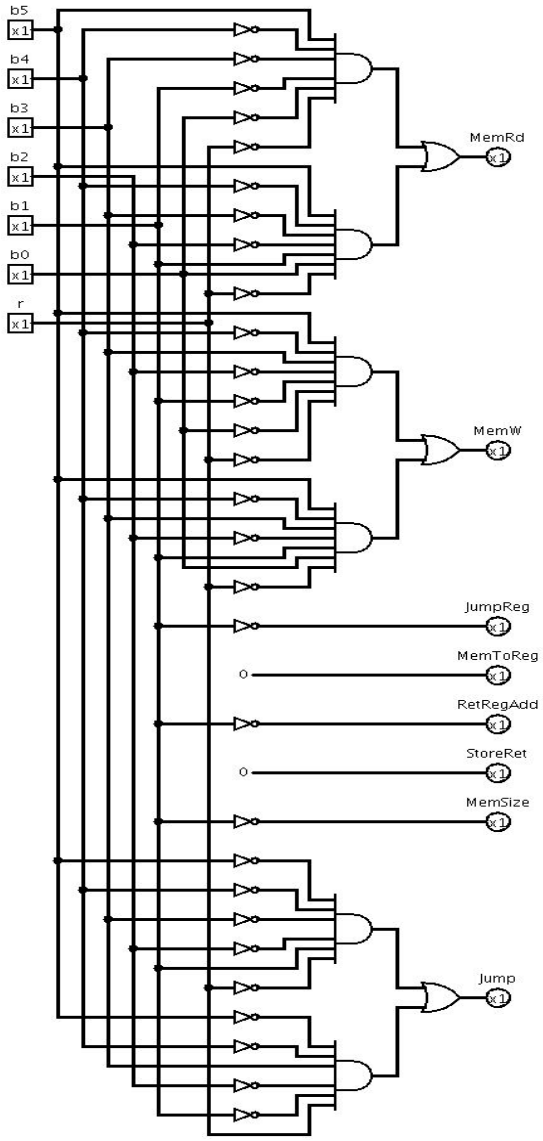
Inputs

Pin	Aka
Bit5	OpCode / Func MSB
Bit4	OpCode / Func Bit 4
Bit3	OpCode / Func Bit 3
Bit2	OpCode / Func Bit 2
Bit1	OpCode / Func Bit 1
Bit0	OpCode / Func LSB
Rtype	Indicates R-type Instruction

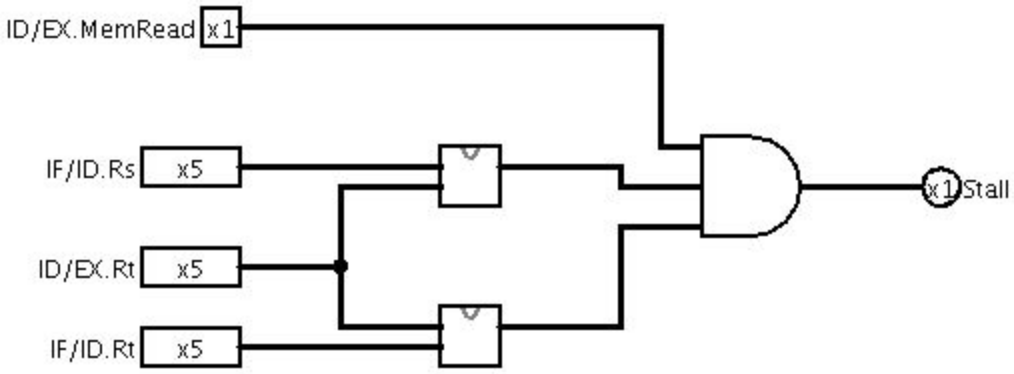
Outputs

Pin	Aka
RegOut	Sel Bit for Rd
RegWrite	WriteEnabled
ALUSrc	Sel Bit for IMM v. Rb
ALUOp3	ALU OpCode MSB
ALUOp2	ALU OpCode Bit 2
ALUOp1	ALU OpCode Bit 1
ALUOp0	ALU OpCode LSB
Sign	Indicates Signed Operations
Comp	Indicates Comparison Ops
Move	Indicates Move Operations
VS	Indicates Var Shift Ops
Lui	Indicates LUI Operation

Decoding Mem/Jumps



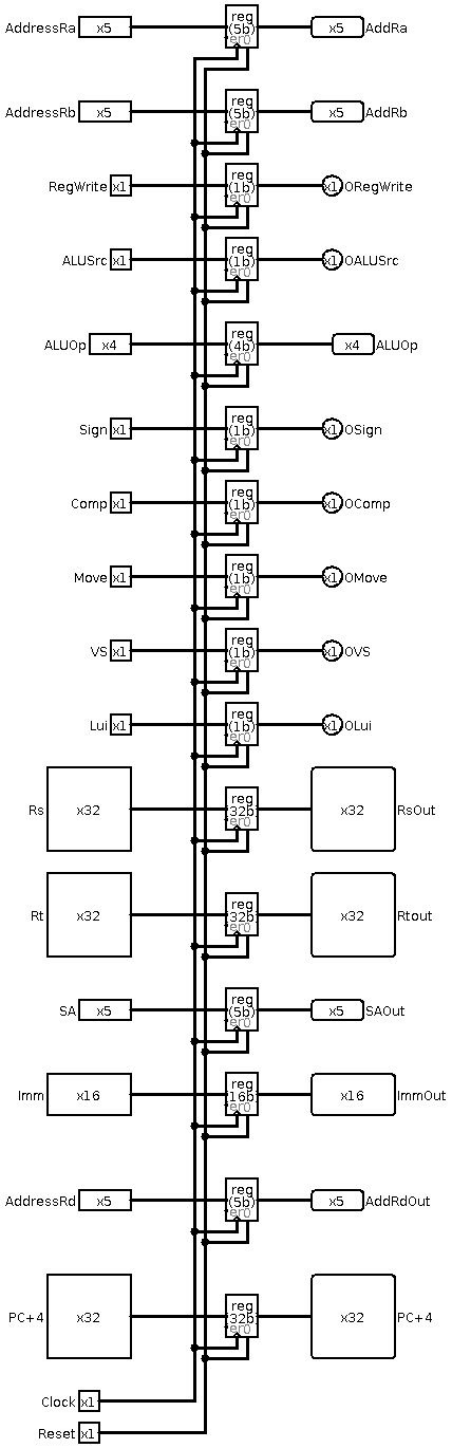
StallChecker



This checks whether there is a hazard with a memory operation and will set stall control bit to 1 if there is.

DelaySlot: Holds the instruction for the next stage in the pipeline

4.1.3 ID/EX Register (split into 2 separate modules to make wiring easier)



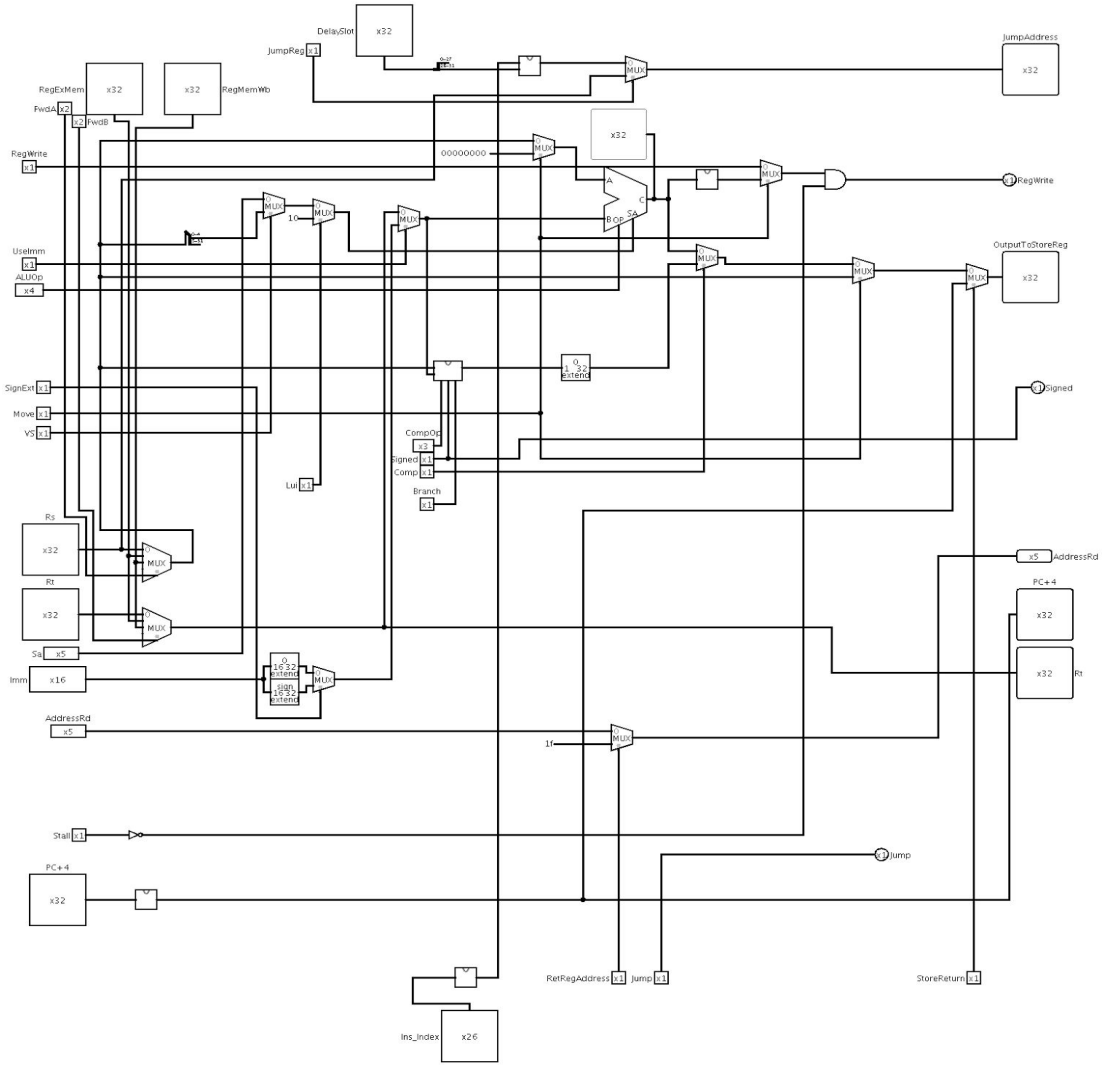
4.3 Test Phase

The test phase is a part of the full system test using MIPS programs.

5 The Execute Stage [EX]

The interesting control signals are Move, Comp, Sign, SV, and LUI. If Move is 1 then we are executing a Move type command and the output of the ALU will create the updated RegWrite value (MOVZ: ALU Op is EQ, MOVN: ALU Op is NE). Move also determines whether to store the register value to be moved or ALU result in EX/Mem register. The Comp (for comparison operations) control bit determines whether to take the result of the comparators rather than the ALU. The Sign bit determines whether you are dealing with unsigned or signed comparison. SV determines where you get your Shift Amount from (SV=1 for variable shifts). LUI determines where you get your shift amount also (shifts 16 when LUI=1).

5.1 Circuit Diagram



Description:

Jumps) The jump address is determined by several control bits. JumpReg determines whether the address is calculated using a value from a register or immediate. RetRegAddress determines whether to pass $31 or rd address to the return address. StoreReturn is 1 for linked jumps and 0 for every other operation (makes AddressRD set to result).

If this operation is a stalled op, then it will set RegWrite, MemWrite to 0.

5.1.1 ID/EX Register

See 4.1.3

5.1.2 Main ALU

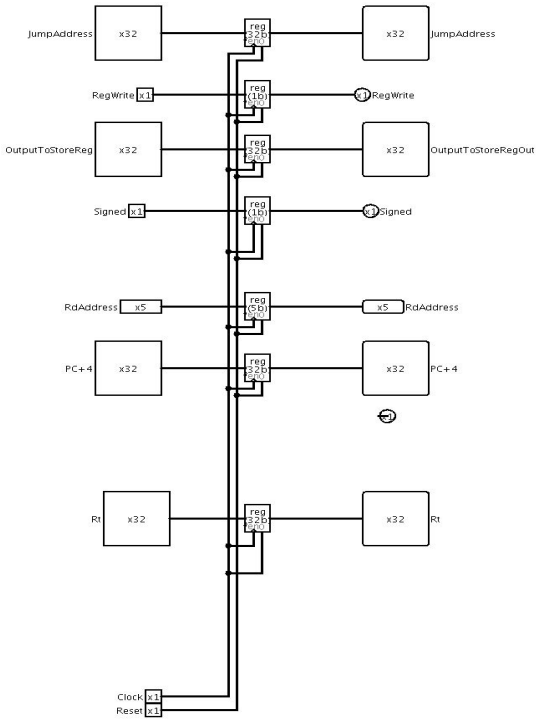
Use the ALU provided in JAR.

5.1.3 Hazard / Forward Unit

Additional logic for hazard control, forwarding result back to ID/EX.

5.1.4 EX/MEM Register

Use a register to store ALU result, jump address, SW operand. (several unnecessary storages but changing would require lots of logism wire moving/freezing)



5.2 Decoding Logic

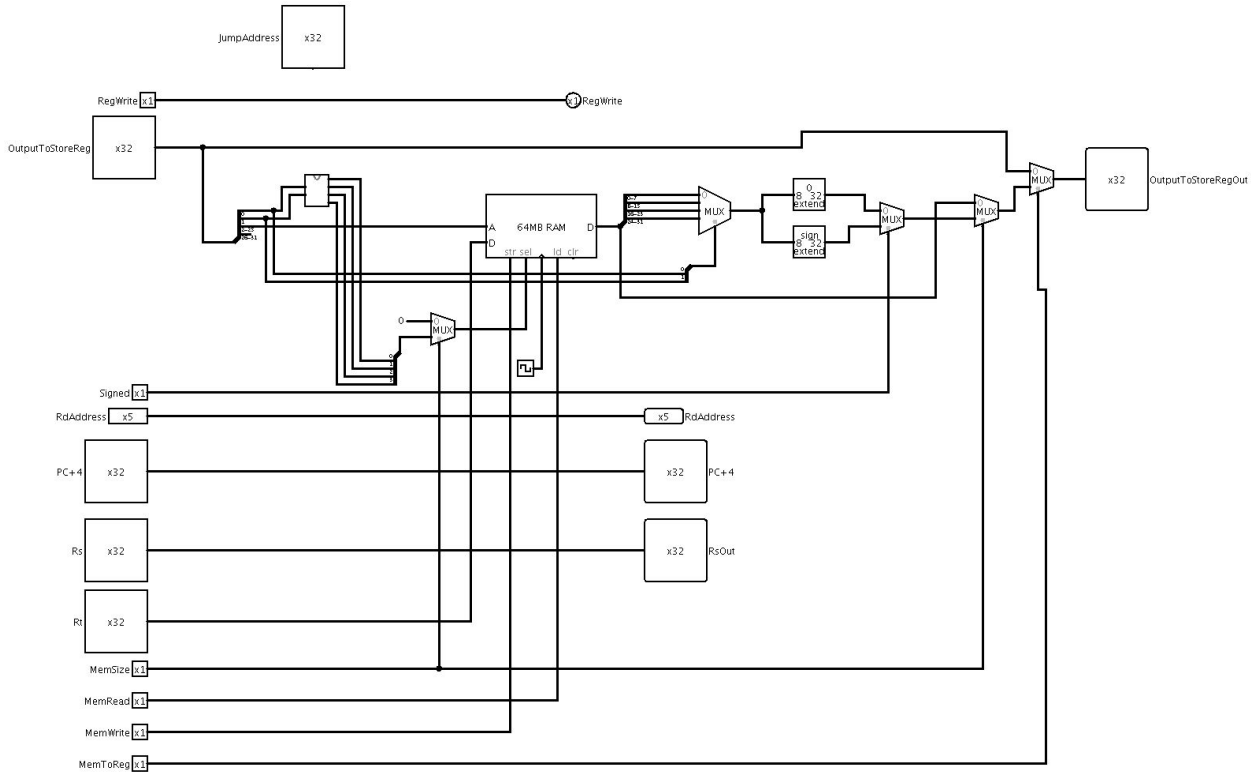
For the EX stage, the important signals are the read register values, and control bits (what type of arithmetic operation to perform, whether to compute a memory address, whether to extend an immediate, etc).

5.3 Test Phase

The test phase is a part of the full system test using MIPS programs.

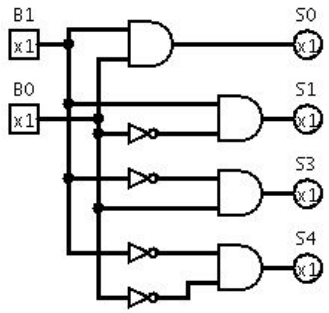
6 The Memory Stage [MEM]

6.1 Circuit Diagram



This stage is able to handle SW, SB, LW, LB, LBU operations. The RAM can select specific bits by supplying a 4 bit select signal. The MemSize control bit determines where the select signal is chosen from. The Signed bit determines whether to 0 extend or sign extend the loaded byte. The MemToReg control bit chooses whether to output the loaded memory item or the ALU computed value. (JumpAddress, PC+4 and RS are pointless, deleting them would require lots of dragging around wires in logism though so they are still pictured)

6.1.0 Last2BitsToSelects



This subcircuit converts the last 2 bits of a memory op and produces the select bits.

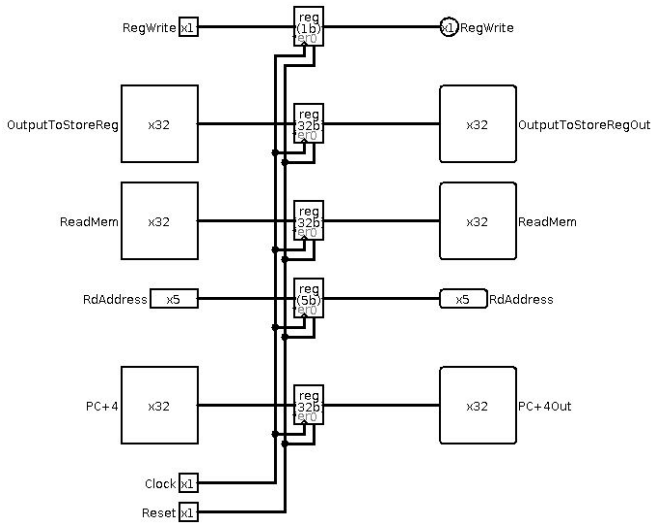
6.1.1 EX/MEM Register

See 5.1.4

6.1.2 Description

MEM stage passes through incoming inputs since we do not implement its features in PA1.

6.1.3 MEM/WB Register

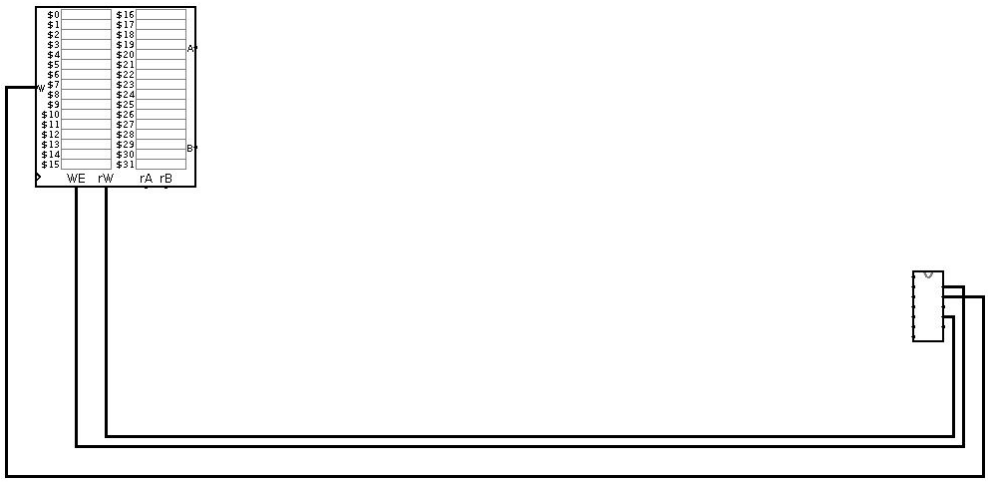


6.2 Test Phase

The test phase is a part of the full system test using MIPS programs.

7 The Writeback Stage [WB]

7.1 Circuit Diagram



7.1.1 MEM/WB Register

See 6.1.3

7.1.2 Descriptions

Writeback stage writes back to the register file based on three outputs of the MEM/WB Register, WriteEnabled, WriteOutput, and WriteAddress.

7.2 Test Phase

The test phase is a part of the full system test using MIPS programs.

8 Design Justification

Using a 5-stage pipeline that's divided into Fetch / Decode / Execute / Memory / Writeback, the processor sports better performance compared to a single cycle MIPS processor.