Faculty of Engineering, Alexandria University

# Pipeline Project Report

Computer Architecture

Ahmed Magdy 3003

Dina Nashaat 3124

Nader El Sharkawy 3001

Mohamed Amgad 3009

# Contents


Introduction: ..... 2

Controller ..... 2

Controller Component: ..... 2

Controller Input/Output: ..... 3

Control Unit Component: ..... 3

Main Decoder: ..... 3

Input/ Output Signals: ..... 3

ALU Decoder: ..... 3

Input/ Output Signals: ..... 3

Latches: ..... 3

Hazzard Unit ..... 4

Input / Output Signals: ..... 4

Hazzard Unit Behavior: ..... 4

DATAPATH Entity ..... 5

MIPS Entity ..... 6

Input /Output: ..... 6

Instruction Memory: ..... 6

Data Memory: ..... 6

Top Entity ..... 6

Test Bench ..... **Error! Bookmark not defined.**

# Introduction:

In this report we will demonstrate the components and architecture of the Pipeline MIPS processor, showing all components of the entity, its encapsulating entity and the testbench and simulation.

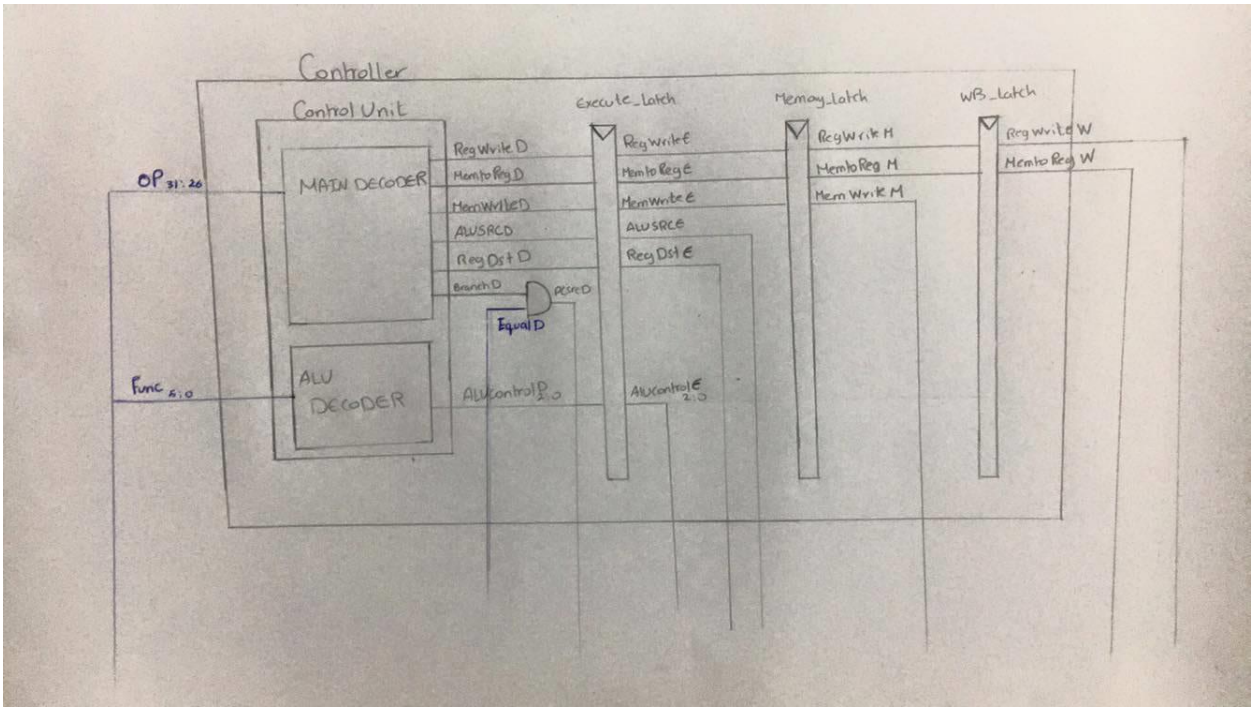
This report is divided into Controller, Data Path, Instruction and Data Memory Entities, following that, describe the Top and Test bench.

# Controller

The Controller of a MIPS Pipeline is similar to that of the Single Cycle, with one difference, it has register clocks over each stage.

In our design, the Controller is divided into Main Decoder and ALU Decoder. The ALU Decoder is the same as the Single Cycle one, however, the Main Decoder constitutes of several Clock Registers.

Below is the Diagram for the Controller:




Blue Lines are Input Signals, while the others are output signals.

## Controller Component:

The Main decoder consists of:

- Control Unit Component
- Decode Latch Component
- Execute Latch Component
- Memory Latch Component

## Controller Input/Output:

| INPUT SIGNALS | OUTPUT SIGNALS | INTERNAL SIGNALS |
|---|---|---|
| CLK | MEMTOREGE | BRANCHD |
| RESET | REGDSTE | REGWRITED |
| OP | MEMTOREGW | MEMTOREGD |
| FUNC | PCSRCD | MEMWRITED |
| EQUALD | ALUSRCE | ALUSRCD |
| | ALUSRCB (1 DOWNTO 0) | REGDSTD |
| | IRWRITE | MEMWRITEE |
| | MEMWRITEM | REGWRITEM |
| | PCENABLE | MEMTOREGM |
| | REGWRITEW | REGWRITEE |
| | ALUCONTROLE | |

## Control Unit Component:

The control Unit consists of:

- Main Decoder
- ALU Decoder

## Main Decoder:

The main decoder takes the Opcode and transfers the input signals to the data path.

### Input/ Output Signals:

| INPUT SIGNALS | OUTPUT SIGNALS | INTERNAL SIGNALS |
|---|---|---|
| OPCODE (5 DOWNTO 0) | REGWRITE, | Controls $_{8:0}$ |
| ZERO | MEMTOREG, | |
| | MEMWRITE, | |
| | ALUSRC, | |
| | REGDST, | |
| | BRANCH | |

## ALU Decoder:

The ALU Decoder takes the FUNC and gives the corresponding ALUCONTROL Signal

### Input/ Output Signals:

| INPUT SIGNALS | OUTPUT SIGNALS |
|---|---|
| FUNC | ALLUCONTROL |
| ALUOP | |

## Latches:

Latches are used to transfer signals from one state to another on the clock events. There are three latches inside the Controller, EXECUTELATCH, MEMORYLATCH, and WRITEBACKLATCH.

Input and Output Signals can be concluded from the above Controller diagram, and the Controller Output Table.

# Hazzard Unit

The Hazzard Unit, one of the very important and characteristic part of a Pipeline processor. The Hazzard Unit, manages three main hazards that is; LW Stall, Decode Stage Forwarding, Stall Detection.

## Input / Output Signals:

| INPUT SIGNAL | OUTPUT SIGNAL | INTERNAL SIGNAL |
|---|---|---|
| FORWARDING | | |
| REGWRITEW | FORWARDBE | LWSTALL |
| REGWRITEM | FORWARDAE | |
| RSE, RTE | | |
| WRITEREGM, WRITEREGW | | |
| STALLING | | |
| MEMTOREGE | FLUSHE | |
| RSD, RTD | STALLF, STALLD | |
| DECODE STAGE FORWARDING | | |
| | FORWARDAD | |
| | FORWARDBD | |
| STALL DETECTION LOGIC | | |
| REGWRITEE, MEMTOREGM | BRANCHSTALL | |
| WRITEREGE | | |
| BRANCHD | | |

## Hazzard Unit Behavior:

### *Forwarding Behavior::*

Forwarding solves RAW data Hazzard, when an instruction in the Execute Stage has a source register matching the destination register of an instruction in Memory or WriteBack stage.

Its logic is stated as follows:

IF ((RSE != 0) AND (RSE == WRITEREGM) AND REGWRITEM) THEN FORWARDAE = 10;

ELSE IF ((RSE != 0) AND (RSE == WRITEREGW) AND REWRITEW) THEN FORWARDAE = 01;

ELSE FORWARDAE = 00;

FORWARDAB is the same logic, except that it checks RT rather than RS.

### *Stalling:*

Stalls are needed to handle LW instruction, where Forwarding would not work in that case because the Load Work doesn't finish until its Memory stage, that is why stalling, which is holding up the operations until the data is available.

LWSTALL = ((RSD == RTE) OR (RTD == RTE)) AND MEMTOREGE

STALLF = STALLD = FLUSHE = LWSTALL

### *Decode Stage Forwarding:*

This logic solves Control Hazards by BEQ, where the branch decision happens in a later stage and is not known by the time the next instruction is fetched. To solve the issue, the processor must predict whether the branch will be taken in an earlier stage, and then proceed accordingly, reducing branch misprediction penalty.

FORWARDAD = (RSD != 0) AND (RSD == WRITEREGM) AND REGWRITEM

FORWARDBD = (RTD != 0) AND (RTD == WRITEREGM) AND REGWRITEM

### *Stall Detection Logic*

BRANCHSTALL = (BRANCHD AND REGWRITEE AND (WRITEREGE == RSD OR WRITEREGE == RTD))

OR (BRANCHD AND MEMTOREGM AND (WRITEREGM == RSD OR WRITEREGM == RTD))

## DATAPATH Entity

The DATAPATH constitutes of all the components that make up the main logic of the processor, the DATAPATH is identical to the single cycle, however, it is cut into stages; fetch, decode, execute, Memory, and WRITEBACK.

Each stage is modeled in the entity, using muxes, latches and components if needed.

The Fetch stage fetches the instruction from Data Memory, and handles PC Calculations

The decode stage fetches addresses of instruction form the Instruction Memory

The Execute stage is where the Instruction takes place using the ALU

The Memory Stage is where data is read from or written to the Memory

The Write Back stage is where data is written back to the register file

Input/ Output:

| INPUT | OUTPUT |
|---|---|
| CLK | OP |
| RESET | FUNCT |
| PCSRCD | |
| REGDSTE | |
| ALUSRCE | |
| MEMWRITEM | |
| MEMTOREGW | |
| REGWRITEW | |
| STALLF, STALLD | |
| FORWARDAD, FORWARDBD | |
| FLUSHE | |
| RSE, RTE | |
| ALUOUTM | |

| | |
|---|---|
| WRITDATAM | |
| RSD, RTD | |
| INSTRD | |
| PCF | |

## MIPS Entity

To encapsulate the above Component, MIPS Entity was created to provide a more structural view and coherence between its components and the memory.

### Input /Output:

| INPUT SIGNAL | OUTPUT SIGNAL |
|---|---|
| CLK | PCF |
| RESET | INSTRD |
| DATARD | DATAA |
| | WD |
| | WE |

### Instruction Memory:

Just like the single cycle, the Instruction memory takes as an input the Address, and return the respective Data residing at that Address. The instruction file reads the memfile.dat and saves all the instructions in Memory.

### Data Memory:

The Data Memory hasn't changed much from Single cycle, it accepts the same input and outputs signals. When writing to the memory WE and CLK must be true, and when reading the address must be provided to fetch the data.

## Top Entity

Top Entity contains all the above entites, the TOP mainly encapsulates and connectes The IMEM, DMEM and MIPS.

Input/ Output Signals

| Input Signal | Buffer Signals |
|---|---|
| Clk | MemWrite |
| Reset | Adr |
| | WD |