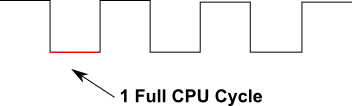
CS385 – Final Report

Robert Rotaru, Bryan Bigelow, Anthony Cerritelli

# Introduction

# Architecture

**Description –** The CPU is built using a 16-bit architecture that includes pipelining – meant to improve the cycle efficiency and protect against certain kinds of hazards. It follows the standard single cycle datapath. What this means is that areas of the CPU ready to execute run on a single clock cycle. This happens during a negative edge of the clock.

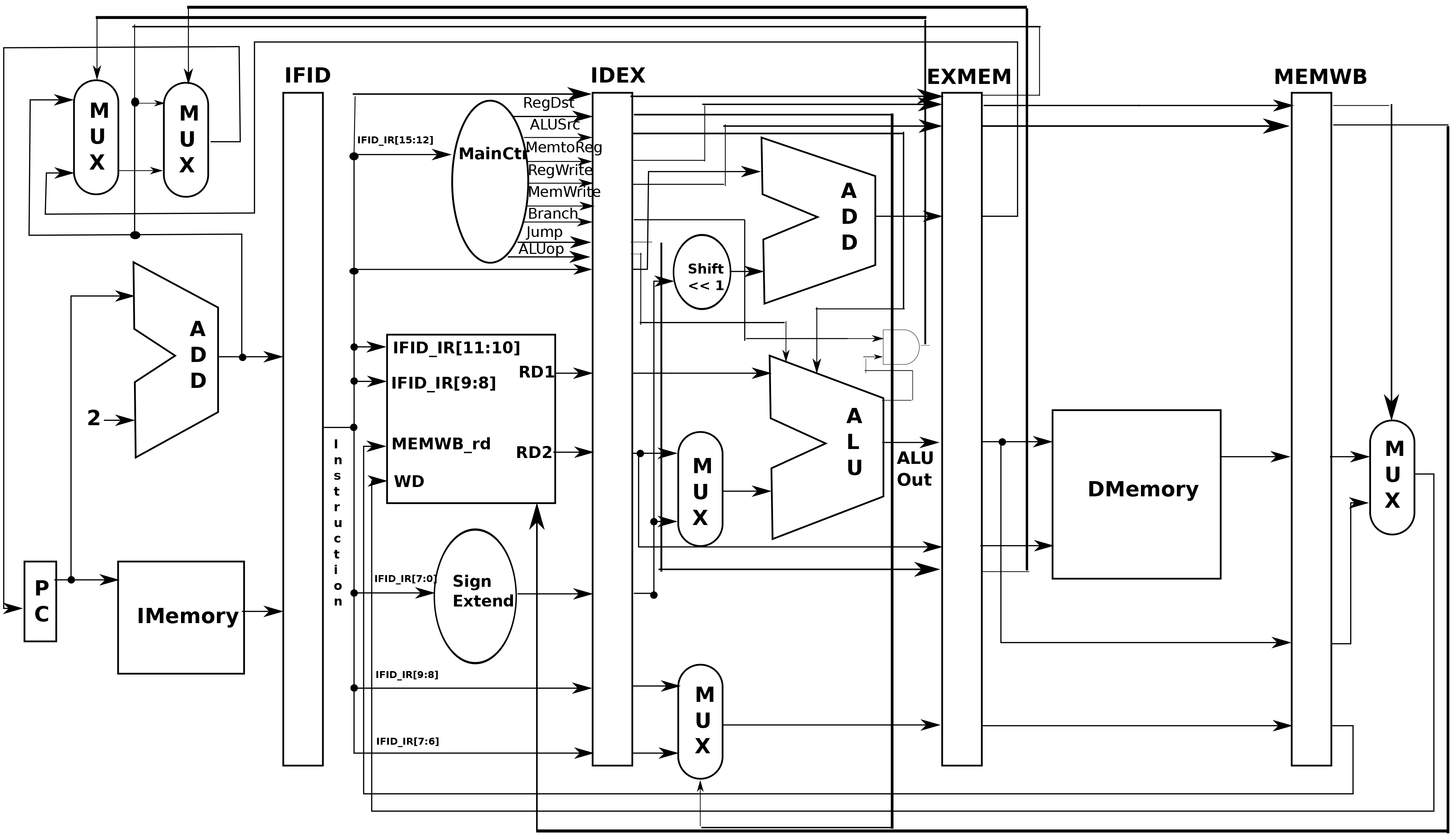


The execution will occur within five distinct areas:

* IF – Includes instruction memory and program counter
* ID – Includes registers, instruction controllers, and branching logic
* EX – Includes the ALU
* MEM – Includes the data memory of the program
* WB – Includes the write back logic

Registers are defined based on a 2-bit number with a max of 4 registers. Only 3 of those registers are available for writing. The last one, the 0 register, is read only for the constant 0.

**Single Cycle Datapath–**



# Instruction Set

|  |  |  |
| --- | --- | --- |
| **Instruction Set** | | **Instruction Format** |
| add | 0000 | |  |  |  |  |  | | --- | --- | --- | --- | --- | | op | rs | rt | rd | Unused | | 4 | 2 | 2 | 2 | 6 | |
| sub | 0001 |
| and | 0010 |
| or | 0011 |
| slt | 0111 |
| addi | 0100 | |  |  |  |  | | --- | --- | --- | --- | | op | rs | rt | value / address | | 4 | 2 | 2 | 8 | |
| sw | 0110 |
| lw | 0101 |
| beq | 1000 | |  |  | | --- | --- | | op | address | | 4 | 12 | |
| bne | 1001 |
| j | 1010 |

# Components

## Program Counter

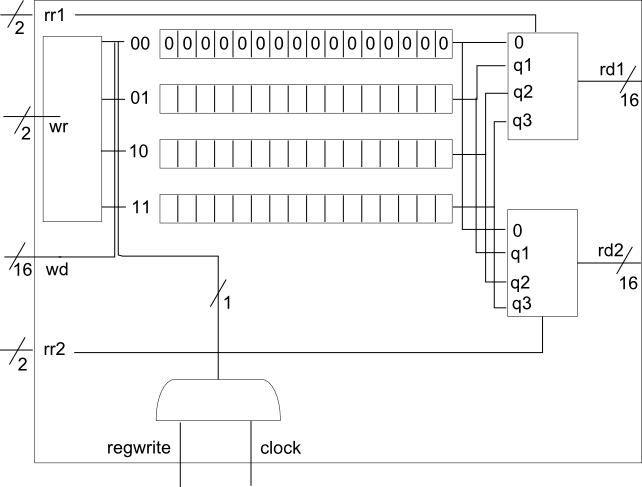
**Description** – The program counter is a requirement for running programs greater than a single instruction. It determines the current location of the program by adding the current size of an instruction to the current value of the program counter. Additionally, to support breaks in the cycle using loops or logic statements it must be capable of being set.

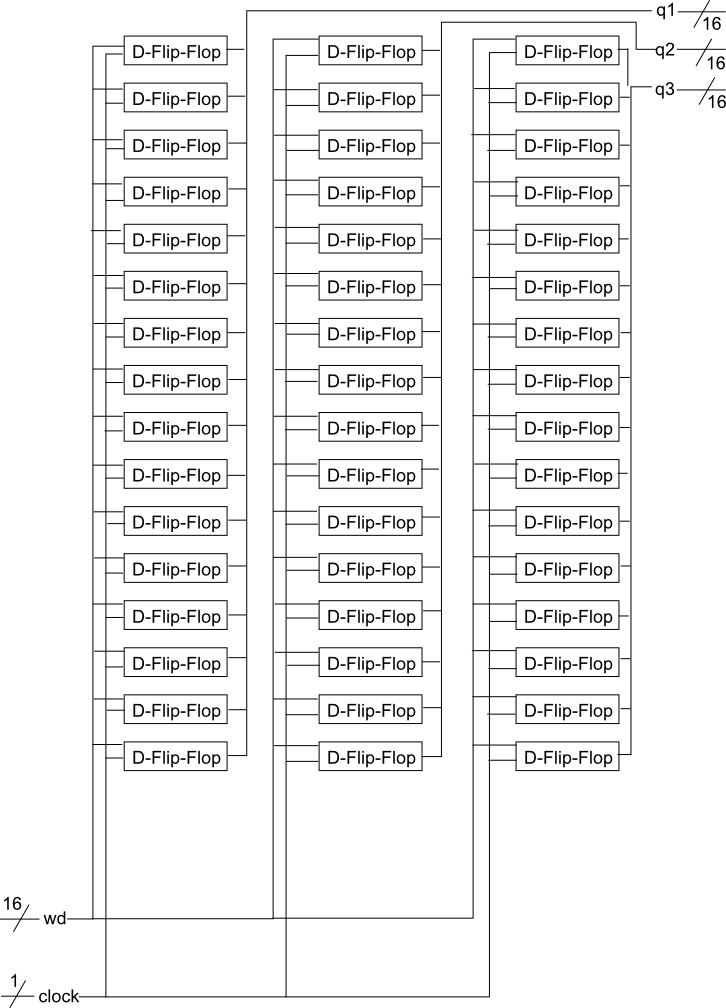
## Registers

**Description –** Registers store and maintain data for use between each clock pulse cycle. They are used both in reading data and for writing back into registers for future use. Because of this, they are one of the few pieces of the CPU utilized in multiple cycles and is more prone to hazards in a pipelined data path.

This CPU design utilizes a 16-bit D-flip-flop architecture for register storage and retrieval. This is achieved by chaining together 1-bit D-flip-flops on a negative edge change. Additionally, each D-flip-flop is actually pieced together from two D-latchs.

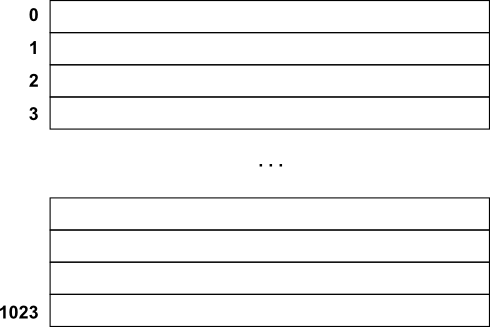
**Diagram –**



****

## Instruction / Data Memory

**Description –** Data memory is implemented as an array of 16-bit storage locations for data. The data is stored within these array indexes, referenced, and read or written to in the MEM section of the CPU cycle. Indices are calculated based on the program counter shifted by the size of the program counter increase.



Instruction memory is handled similarly – though it requires less functionality as it is meant only to be retrieved and not written to.

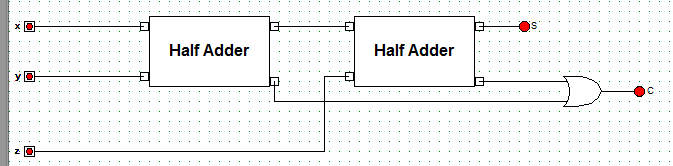
## ALU

**Description –** The Arithmetic Logic Unit, or ALU, is the workhorse of the CPU producing the results of different operations – with the exception of branch and memory instructions. Depending upon the Opcode it may produce several different instructions with the supplied register data.

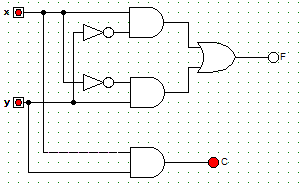
This CPU uses a 16-bit ALU built upon a hierarchical style. The 16-bit ALU is built upon 16, 1-bit ALU’s (including a special most significant bit ALU). Those 1 bit ALU’s are made using a full-adder that in turn is made up of half-adders.

**Diagram –**

Full Adder



Half Adder



**Truth Table(s) –**

Half Adder

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | C | S |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

## Pipelines

**Description –** The four pipelines of the single cycle data path CPU provide monitoring, hazard protection, and performance improvements to the running of programs:

* Monitoring – Storing certain values within the pipeline and transferring it forward to the next level pipelines, the CPU may monitor the instructions as they pass through.
* Hazard Protection – By storing certain pieces of information between cycles it can be used in future cycles before it completely makes its way through the CPU. By doing this there is a large decrease in hazards for the program.
* Performance Improvements – By utilizing the hazard protection explained above there is less of a need for artificial no operations (dummy instructions) to introduce delays for the CPU to catch up.

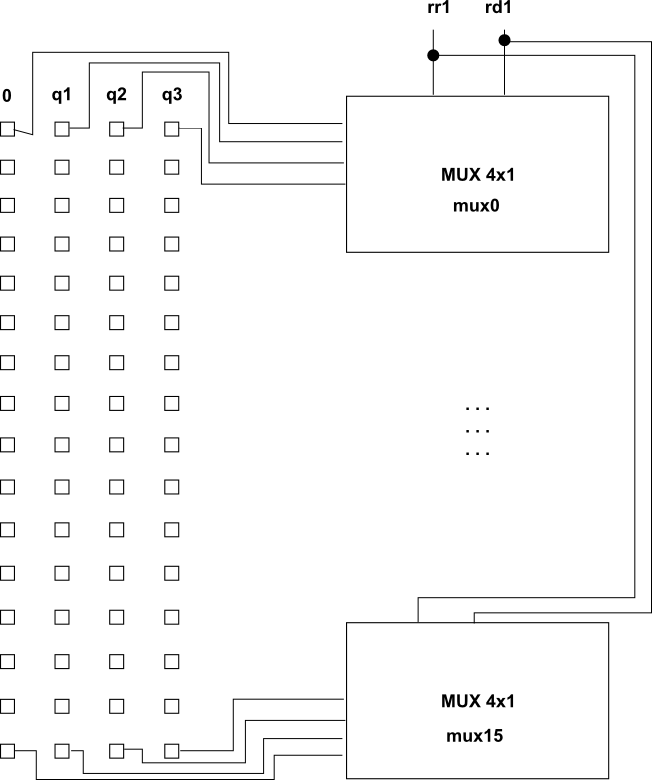
## Multiplexers

**Description –** Several different kinds of multiplexers are contained within the CPU architecture. They provide support for selecting appropriate values from several potential values based on things like instruction operation codes, write back enabled, etc.

A 16-bit 4x1 multiplexer, made from chained 1-bit 4x1 multiplexers, is utilized within the register file to determine register data. Additionally, a 16-bit 2x1 multiplexer is used for data in the WB phase of the CPU.

**Diagram –**

16 bit 4x1 Mux (Register File)



**Truth Table(s) –**

4x1 Multiplexer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| s1 | s2 | x0 | x1 | x2 | x3 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

# Program

# Simulation Results