Project 3: Designing a 32-bit CPU

Adam Sumner - A20283081, **Contribution** - 25%
Bobby Unverzagt - A2028923, **Contribution** - 25%
Emilie Woog - A20265269, **Contribution** - 25%
Nash Kaminski - A20283999, **Contribution** - 25%

ECE 485

December 5th, 2015

1 Introduction

This goal of this project is to design a stripped down version of the MIPS processor. The processor will be a 32-bit version of the processor discussed in class and the text book, however, its instruction set will be a small subset of the MIPS processor's full capability.

1.1 Background Information

1.1.1 MIPS

MIPS is a reduced instruction set computer (RISC) instruction set architecture (ISA). It defines three types of instruction types: R (register), I (Immediate), and J (Jump). For the implementation that this project is focused on, only R and I instructions will be executed. R type instructions are the most common form of instructions. The format for an r-type instruction is:

`Bits[31:26]`	`Bits[25:21]`	`Bits[20:16]`	`Bits[15:11]`	`Bits[10:6]`	`Bits[5:0]`
opcode	Rs	Rt	Rd	shamt	funct

For this instruction, the opcode field is always 000000_2, while the function code `funct` is used to determine which instruction is to be carried out. Rs and Rt are the two registers in which the operation reads and Rd is the destination of the result. Some instructions require a shift amount (`shamt`), so it is specified explicitly.

The I type instruction involves an immediate value, so the instruction format must accommodate this. The format of this type of instruction is:

`Bits[31:26]`	`Bits[25:21]`	`Bits[20:16]`	`Bits[15:0]`
opcode	Rs	Rt	immediate

For this instruction, the op code field is used to define the specific instruction, Rs is the register in which the operation acts on along with the immediate value as the other operand. Rt is the destination register in which the result is stored.

1.1.2 Datapath and Control

A datapath is a collection of functional units that perform data processing operations. It includes units such as a program counter, a register file, instruction memory, an ALU, data memory, and a control unit. Figure 1 shows a high level overview of a simple datapath with control.

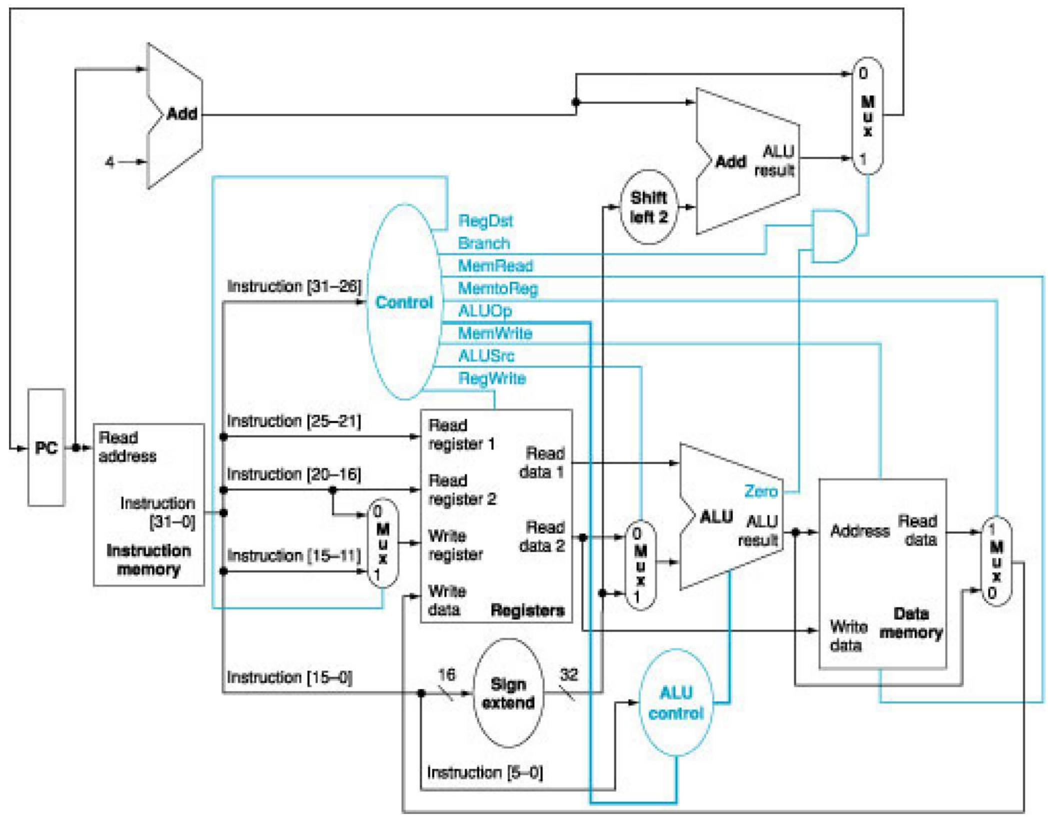

Figure 1: Datapath Overview

2 Design

2.1 Instruction Set

Table 1 shows the instructions that were chosen to be implemented in the CPU with the respective OpCode and Function Field for each instruction.

`OpCode[31:26]`	`Function Field [5:0]`	Instruction	Operation
100011_2	`--`	`lw`	`lw $t3, 200($s2)`
101011_2	`--`	`sw`	`sw $t4, 100($t3)`
000000_2	100000_2	`add`	`add $s3, $t2, $s2`
000000_2	110000_2	`sub`	`sub $s3, $t2, $s2`
000100_2	`--`	`beq`	`beq $s5, $s2, 500`
000000_2	000001_2	`nand`	`tbd`
000010_2	`--`	`andi`	`tbd`
000000_2	000010_2	`or`	`tbd`
000011_2	`--`	`ori`	`tbd`

Table 1: CPU Instruction Set

Because it was only required to implement 9 instructions, and the MIPS instruction set format requires 6 bits for op code and function field, it was an easy decision to choose these values for the implemented instructions. For all R-type instructions, the functions fields were chosen to be vastly different from one another to make debugging easier for the team. Likewise, the same approach was taken for the op code decisions for the I-type instructions.

2.2 Memory

For this project, it seemed unnecessary to implement memory of 4GB (2^{32}). It was chosen to use an array of 256 words instead. If need be, this memory size could be upgraded easily, so this choice does not hinder performance on the actual design of the CPU.

2.3 DataPath

Figure 2 shows the overview block diagram of the implemented datapath for the CPU.

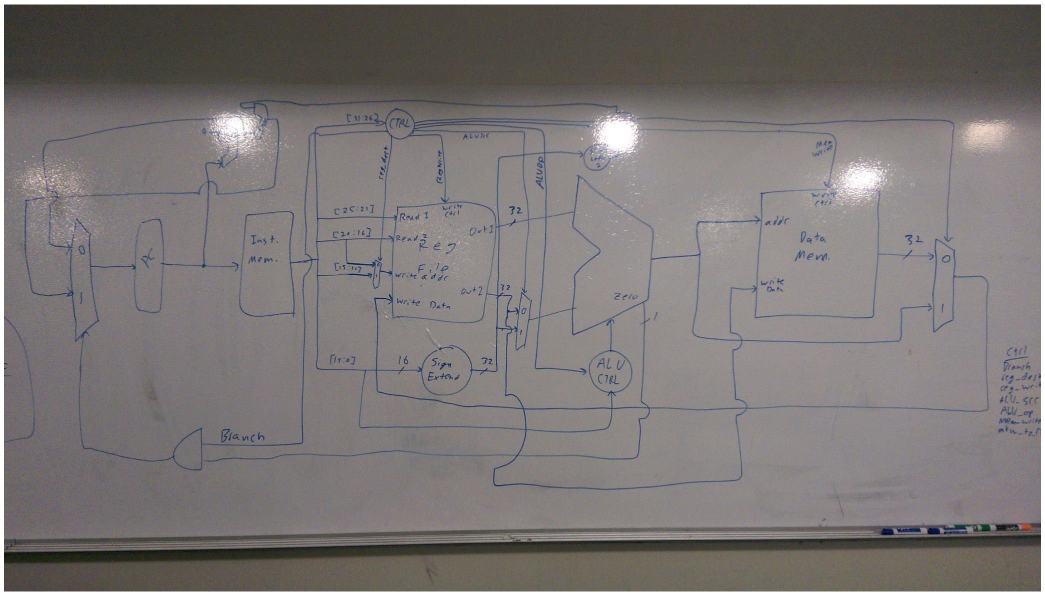

Figure 2: Implemented Data Path Overview

2.4 Control

The control can also be seen in Figure 2. It is a simple design of several signals acting as the `sel` line of a series of multiplexers. Based on the op codes and function field read from the instruction memory, the signals are asserted accordingly to relay the correct signals into the Register File, ALU, and Data memory.

3 Analysis

While this processor is designed to be able to fully accomplish the tasks specified in the business requirements document, it could still be improved. In its current stage, it can be considered a bare bones prototype. To transform the current design into a processor on par with the current industry standard, a complete instruction set would have to be implemented. Furthermore, pipelining is a necessity to add. Any processor that doesn't implement pipelining is not making efficient use of its own components.

4 Simulation Results

5 Conclusion

The design and implementation of a 32-bit CPU was a success. a set of 9 instructions were successfully implemented and verified with test bench code. All requested functionality was achieved. This 32-bit CPU can now be used in further projects.