Ain Shams University / Faculty of Engineering

CSE Department

Computer Architecture CSE116

MIPS and Control Unit Simulation

Submitted By

Mostafa Mahmoud El-Rosasy 16P8113

Omar Magdy Shaaban 16P6034

Peter Nabil Zaghloul 16P8100

Seif El Din Abdel Hakim 15P8111

Seif El Din Mohamed 16P8109

Group 2 Section 1

Submitted To

Dr. Cherif Ramzi Salama

Cairo, 2018

Abstract

This project delivers a MIPS simulation implemented in Java and runs with a GUI (Educational). All executions of the program are illustrated during runtime. The project shall handle exceptions to avoid possible failures of the application. The project will assume 4x1 MUXs for the “RegDst”, and “MemtoReg” MUXs (to choose “$ra” as a destination register, and “PC+4” will be stored in $ra so “jal” can be called properly). Also, we will add another 2x1 MUX which will choose between the “j” address and “jr” address according to the control signal “jr” which is produced from the “ALU Control”. This report will illustrate the datapath used briefly, provide truth table and logic diagram of the “Control Unit”, provide a user-guide with screenshots, a few programs tested to make sure of the proper functionality of the application, a small project charter illustrating whom did what, and a small section in which every participant of the project will share his gained experience from such project.

Contents

Figures

1. Datapath

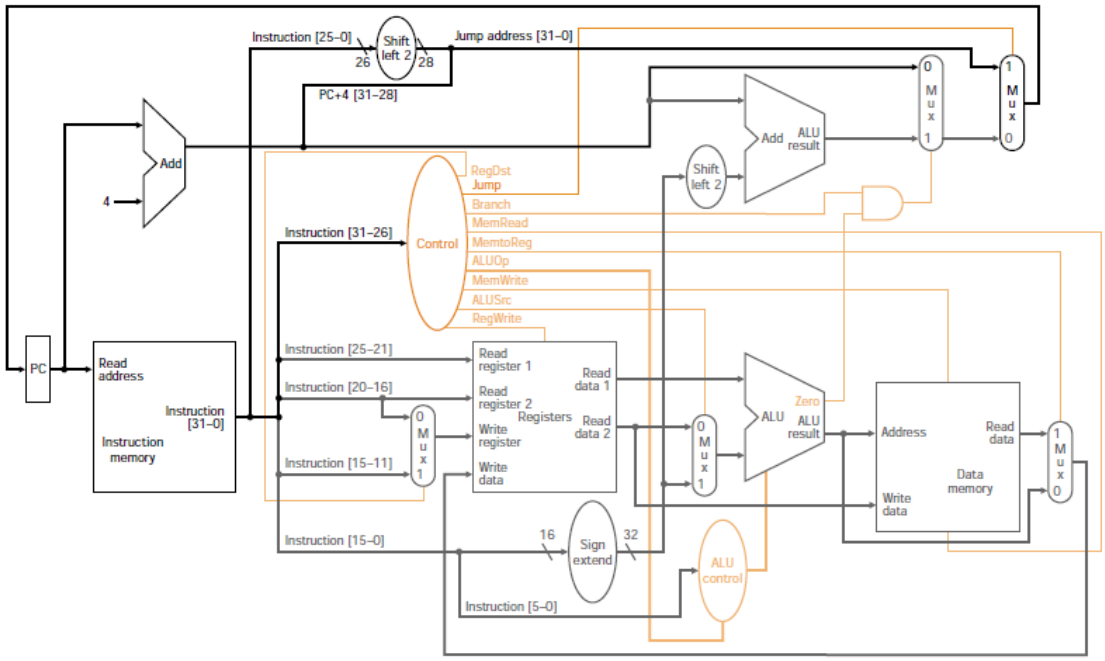

Figure 1: MIPS Datapath (1)

The datapath used is the single-cycle datapath, in which “PC” points at a certain instruction in the “Instruction Memory” where it will be fetched and divided according to its format (R-type, I-type, J-type). The “Register File” will read the values of the registers which will enter the “ALU”, the “ALU” then performs the operation according to the input from the “Control Unit” and “ALU Control”. The “Memory” is used if the control signal has read/write signals (Load/Store instructions), if not, then the result from the “ALU” can be either stored in the destination register in the “Register File”, or this result will be used in the branch instruction. However, there is an exception like the J-type instructions for example which will only be fetched, undergoes division of its bits according to its format then concatenate with the 4 last bits of the “PC”+4 and in the end update the “PC” with the new value.

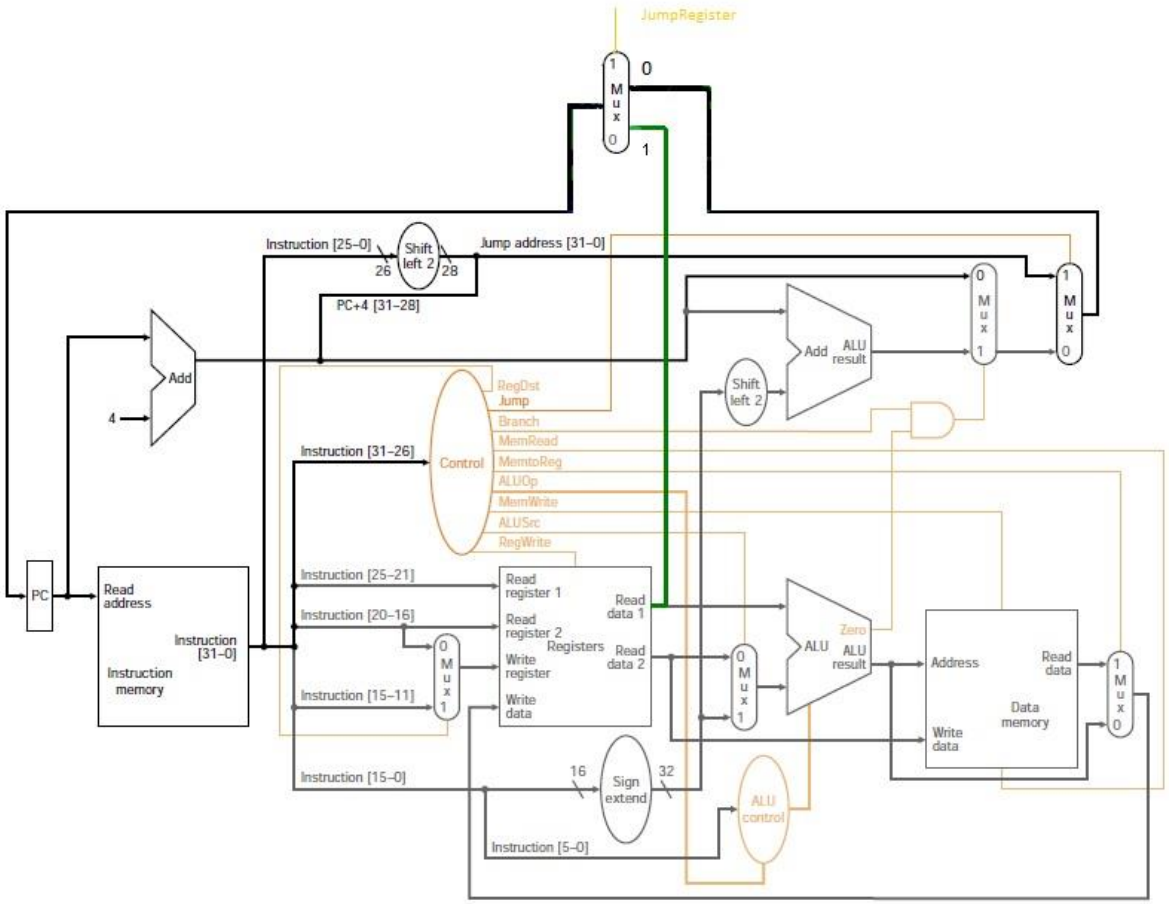

Figure 2: MIPS Datapath (2)

This datapath has added another control signal “jr” (but not in the “Control” unit. It is produced from the “ALU Control” unit). As “jr” instruction is an R-type instruction, then by the first datapath, it won’t jump and will have to write in a register, so there is a necessity to add another control signal to detect “jr” and will make it jump to another register without writing any data in the “Registers File”. Also, the “RegWrite” signal will be put in an “AND” gate with the inverse of the “jr” signal, so that if it was a true “jr” signal, then no writing of data in the “Registers File” will occur, otherwise, R-type instructions will write in their destination registers as usual.

2. Control Unit

2.1 Truth Table

Name	Opcode	RegDst	Jump	Branch	MemRead	MemtoReg	ALUOp	MemWrite	ALUSrc	RegWrite
R-type (except "jr")	000000	01	0	0	0	00	10	0	0	1
Lw	100011	00	0	0	1	01	00	0	1	1
Lb	100000	00	0	0	1	01	00	0	1	1
Lbu	100100	00	0	0	1	01	00	0	1	1
Sw	101011	XX	0	0	0	XX	00	1	1	0
Sb	101000	XX	0	0	0	XX	00	1	1	0
Beq	000100	XX	0	1	0	XX	01	0	0	0
Addi	001000	00	0	0	0	00	00	0	1	1
Slti	001010	00	0	0	0	00	10	0	1	1
J	000010	XX	1	X	0	XX	XX	0	X	0
Jal	000011	10	1	X	0	10	XX	0	X	1
Jr	000000	XX	0	X	0	XX	XX	0	X	0

The "jr" instruction has the same control signals as R-type instructions, but we wanted to show what "jr" actually needs on its own. The "jr" instruction will have "jr" signal which will make a jump to a certain register and disable writing in a register. The "opcode" input signals are ordered A → F from left to right, to be used in the logic implementation.

2.2 Logic Diagram

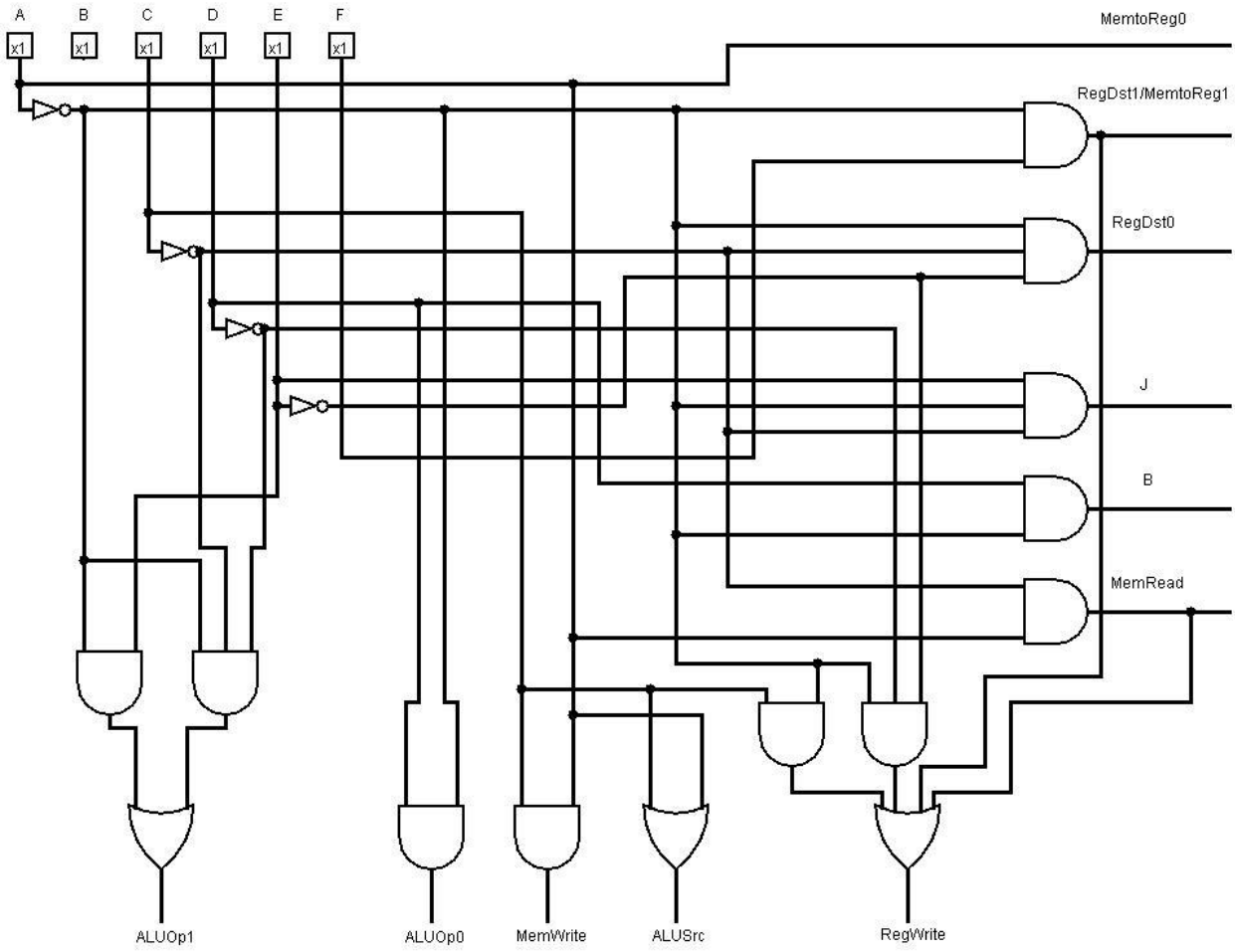

Figure 3: Control Unit Logic Diagram

3. User Guide

4. Tested Programs

5. Simple Project Charter

Name	Responsibility
Mostafa Mahmoud El-Rosasy	**Control Unit**
Omar Magdy Shaaban	**Memory, Control Unit, Report**
Peter Nabil Zaghloul	**Assembler, Instruction Memory, PC, Register File**
Seif El Din Abdel Hakim	**ALU, ALU Control**
Seif El Din Mohamed	**Components Integration, GUI (+Educational)**

6. Comments

- Mostafa Mahmoud: “’
- Omar Magdy: “This project has made me learn better about the MIPS single-cycle datapath, the simulation makes it easier to remember and also visualize what you’re thinking and what actually happens. Also, I’ve learned how to work in a team through discussing ideas, dividing work and using Github.”
- Peter Nabil: “”
- Seif El Din Abdel Hakim: “”
- Seif El Din Mohamed: “”