Eric Last

CS203 Project 3

1) Problem Statement and Solution

With this project, our task was to create and implement a CPU simulator that will take and process instructions from an assembly file. Given the foundations of a computer (RAM, Registers, etc.), we were tasked to expand upon and build a full simulation. Using an ALU, the simulator will be able to process basic arithmetic and logical calculations. The CPU will also be able to do basic memory manipulation and branching, much like how our computers work.

The solution to this problem is not simple, but very methodical. An ISA must be created to define the instructions that the computer can recognize, and each instruction needs to have an RTN representation coded for the computer to process through each clock step. The ALU must be programmed to understand how to compute additions, subtractions, and logical operations. All of these components combined with the existing framework will create a fully functional basic CPU simulation.

2) ISA Documentation

I decided to keep the wordcount at 16, which only allowed me 14 operations, keeping the two provided in the skeleton. Each operation has a 4-bit opcode, and 12 available bits to control the operation (source(s), destinations, immediate, etc.) Luckily, this was enough to fully implement the functionality of my design. The table below explains the ISA:

Instruction Name	**Hex opcode**
No Operation (NOP)	0x0
Load Immediate (LOADI)	0x1
Add (ADD)	0x2
Subtract (SUB)	0x3
Add Immediate (ADDI)	0x4
Subtract Immediate (SUBI)	0x5
Add and Set Flags (ADDS)	0x6
Subtract and Set Flags (SUBS)	0x7
AND (AND)	0x8
OR (OR)	0x9
XOR (XOR)	0xA
Logical Shift Left (LSL)	0xB
Logical Shift Right (LSR)	0xC
Branch (B)	0xD
Load Register (LDUR)	0xE
Store Register (STUR)	0xF

Table 1: ISA Implementation

3) ISA Discussion

I made a few design decisions on my ISA to allow the simulation to operate the way I want it to. First of all, I decided to implement ADD, ADDI, and ADDS, as well as the subtraction counterparts, because I felt that those are the most useful instructions a computer can have; the

ability to *compute* is important for a *computer*. These each take a destination register and two sources to carry out addition or subtraction. The ALU handles all 6 of these operations, sourcing from a register and either the master bus or another register depending on the instruction. The logical operations are very standard, and use the ALU for the calculations. Branching is important for being able to control the location of the PC, and memory operations are fundamental to the design of these computers. Unfortunately, I was unable to implement the memory operations, as I ran out of time on the assignment. I ordered each operation by classification, which is why they are given the opcodes shown in Table 1.

To implement the ISA, I had to create RTN instructions for each operation. Some could not be done in a single clock cycle, and thus three or four steps were needed to carry out one instruction. Once completed, the Controller class had over 1200 lines of code, mostly composed of RTN classes for each step of every instruction. Each step needed to be placed in a specific spot in the control memory so that the computer can understand and carry out the instructions. All of this control is implemented in the Controller class of the project.

4) Example Programs

I've submitted, along with the code, a few example programs that demonstrate how the simulation works. Below are explanations of each program:

prog1.as: This program demonstrates the functionality of adding and subtracting, with a few lines performing different operations

prog2.as: This program demonstrates the functionality of logical operations, with a few lines performing different operations

prog3.as: This program combines arithmetic and logical operations, creating a random sampling of these operations

prog4.as: This program demonstrates branching. The program jumps back and forth in the memory, executing several instructions.

These four programs are able to demonstrate the ability of my CPU to handle the instructions laid out in my ISA. The first is a simple example of the six arithmetic operations, and the second shows the five logical operations. The third puts the eleven aforementioned instructions together, and the fourth demonstrates how branching can be used to move around the memory file. I constructed these four files so each can adequately demonstrate how the computer can work, There is no significance of the values initially loaded into the first few register at the beginning of each file, just randomly chosen numbers. Each file can be run by changing the file name in line 69 of the RAM class.

5) Conclusion

This project provided a challenge to really understand the organization of computers and the truth behind what happens when processes are carried out. I learned a great deal about the CPU from this project, and how so many components work together to produce the outcome we know to be computations. I was able to fully understand the PC, IR, and other important information about registers and the fundamental parts that make a computer.