Yousaf Rajput
Project 2
I pledge my Honor that I have abided by the Stevens Honor System

Jet Z16

How to use the Assembler:

My assembler is a python file called Assembler.py. In order to use the assembler, you must have the python file (Assembler.py) in the same directory as your text file that contains your assembly code. For this assembler, you must make sure that the file containing your code is named 'Code.txt' otherwise the assembler will not be able to work and generate your image file. At that point, you can just run Assembler.py, and you'll see a text file called 'NewInstructions.txt' appear in the same directory containing your code and assembler. This is your image file containing the instruction memory. At this point you can go to the .circ file and right-click on the leftmost memory storage labeled, 'Instruction Mem.' You then click load image and locate 'NewInstructions.txt' in the directory of your project and click on it and click 'okay.' You should then see a window pop up called 'Hex file format.' Click on the button labeled 'v3.0 hex' and then click okay. Congratulations! You have just assembled your program and loaded the instructions into instruction memory!

Architecture Description:

Jet Z16 is a 16 bit CPU containing 4 general purpose registers, each storing 8 bits worth of data. You must reference a register in your code in the format 'Rx', where x can be any valued from 0-3 inclusive. For example, if I wanted to use Register 1 in my code, then you would type out 'R1.' My CPU can execute 4 different functions. It can add two values from any 2 registers and store the result into another one. For example, ADD R1 R1 R2 will perform R1+R1 and store the result into R2. My CPU can subtract the values from any two registers and store the result into any register. For example, SUB R0 R2 R3 will perform R0-R2 and store the result into R3. My CPU can also store any immediate value ranging from 0-15 into any register using MOV. For example, MOV R1 13 will store the value 13 converted to hex into R1. Lastly my CPU can load 1 byte from memory to a register using LD. For example, LD R1 R2 R3 will get the data at address R1+R2 and copy it from memory to R3. Those are all 4 functions currently implement into Jet Z16.

Binary Encodings:

Arithmetic and Memory Accessing Instructions

Instruction	15-6 Opcode	5-4 Rn First Register	3-2 Rm Second Register	1-0 Rd Destination Register
ADD Rn Rm Rd	0000000100			
SUB Rn Rm Rd	0000000101			
LD Rn Rm Rd	0000010110			

Imm to Reg Storing instructions

Instruction	15-6 Opcode	5-2 Imm4 4-bit immediate	1-0 Rd Destination Register
MOV Rd imm4	0000001100		

The opcode for the instructions are contained within bits 15-6. Although not all bits in the opcode are being used for instructions, this is purely because of a lack of time to implement other instructions that would take advantage of those bits. For all instructions, the destination register is encoded into bits 0-1 because there are only 4 registers ranging from R0-R3. For Arithmetic and Memory Accessing instructions bits 5-4 are used to encode the first register of the instruction and bits 3-2 are used to encode the second register of the instruction. They both only use 2 bits because there are only 4 registers to choose from and we can represent 4 values with 2 bits. For Imm to Reg Storing instructions, a 4-bit immediate number is encoded into bits 5-2 as we combine the bits we used for the first and second register Rn/Rm since we no longer use Rn and Rm. This allows us to use any positive value from 0-15 with those 4 bits.