Alisha Patel
I pledge my honor that I have abided by the Stevens Honor System.
CPU Name: Duff

How to use the assembler program:

The assembler converts the user's instructions into binary (through binary encoding, which is shown below), based on the operation specified, and then creates the machine code. After that, it converts the machine code to hex code to be interpreted in the ram. After filing the 4 instructions, it should fill up the rest of the space with 0s. You can edit the text file as to input the instructions, and then run the assembler on that. It should edit the image file upon that.

Architecture description of the CPU:

The first stage of the datapath is adapted from the memory lab done prior, in this case, the ram memory serves as the instruction memory which will generate instruction code, and perform instructions on that basis. This CPU works around 8-bit instructions. Then, comes the register file (which is also adapted from the lab), which contains 4 general purpose registers to store the temporary data for calculation purposes. All of these stages are synchronized by the same clock, for the instructions do not overwrite each other. However, the data memory's clock will be different from the instruction memory. Because RAM takes more time to access data, thus we need data memory controlled by its own clock and speed which is greater than that of the CPU. For the execution of operations, like in this, ADD, ORR, and LDR, the RegData1 and RegData2 values need to be determined, where it is going to input the immediate values or register data.

Reg Write is always going to be on since we will be writing values for all instructions accepted; load, add, and or boolean. Reg2Loc control signal will be on when the opcode will be 01 or 10, using the loading and adding with immediate. Which will be fed into the mux, which will determine whether to use the RegData2 or the immediate. ALUsrc will also choose, depending, on which value should be chosen to execute ADD or ORR. Based on the ALUop will determine which operation should be taken into consideration (if 00, then the mux will choose to add, or else ORR). And when the opcode indicates to load, the MemToReg signal will choose to load the element from memory to register.

General Format for the instructions:

The binary encoding for the instruction is based on the first 2 bits being the opcode for the desired operations. Since the CPU takes in 8-bit instructions, the rest 6 bits will be used for the registers and immediate values that need to be computed.

Opcodes (the first two bits for the instructions):

- ADD Rd Rn Rm = 00
- ADD Rd Rn imm = 01
- LDR Rd Rn imm = 10
- ORR Rd Rn Rm = 11

Examples:

ADD X1 X2 X3 → 00 01 10 11

ADR X2 X3 1 → 01 10 11 01

LDR X1 X3 X2 → 10 01 11 10

ORR X3 X1 X2 → 11 11 01 10