Names: Sandya Devanahally and Vinayak Ranjan
Honor Pledge: I pledge my honor that I have abided by the Stevens Honor System.

Name of the CPU: Byte Knight ⚔️🛡️

This project is eligible for extra credit. We have implemented a fun little LED diagram for a sword and shield, inspired by our CPU name. In addition, we implemented a little LED matrix that displays the current instruction hex code.

Job Description:
Both partners contributed equally to the creation of this CPU project. The creation of the .circ file was done on Vinayak's computer, with both team members working collaboratively to figure out the circuits. The coding portion was completed on Sandya's computer, with both partners working on the logic of the code together. Both partners contributed to the user manual at the same time. Finally, the project was put together to test various parts of the assignment.

How to use assembler program:
Step 1: Ensure that **instructions.txt, assembler.py, and byteknight.circ** are all located in the same folder.

Step 2: Open the **instructions.txt,** and input any instructions desired in the following format:

<INSTRUCTION> <REG1> <REG2/IMMEDIATE>

Where instruction is either ADD, SUB, LDR, STR, or MUL
Reg1 and Reg2 is a register, e.g. X0
Or Immediate- an integer (simm 8), e.g. 2

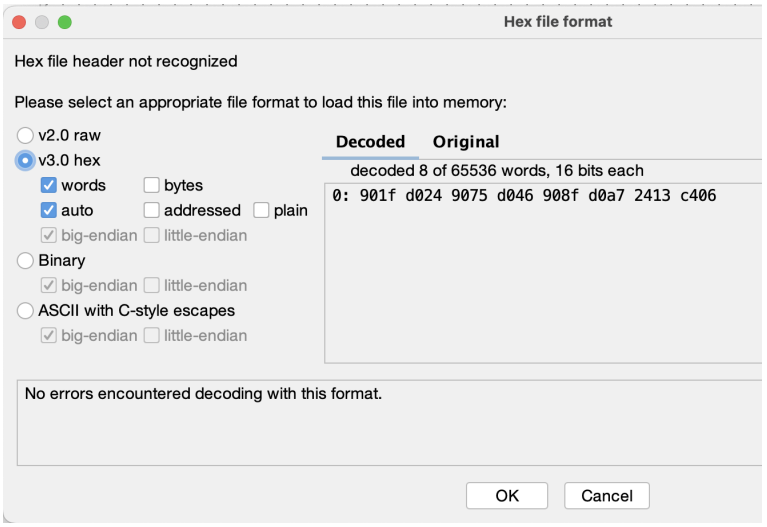
Put each instruction in a new line if you would like to have multiple instructions

Step 3: Save the file.

Step 4: Open and run **assembler.py**

Step 5: in the folder, there will be an **image.txt** file generated that contains the necessary hexadecimal that you need to put into the **.circ** file

Step 6: Open the .**circ** file and then simulate the CPU. Click the simulation tab and press reset simulation and then make sure auto-tick is enabled from the same tab. Right click the ROM labeled **Instruction Memory** and load the **image.txt** file. Make sure you load it as **v3.0 hex**. Next, press simulate to run the clock.



The CPU has **4 general purpose registers: X0, X1, X2, X3**. These registers are identified using **2 bit binary representations**:
X0: 00
X1: 01
X2: 10
X3: 11

The CPU supports the following instructions:
ADD: Adds two registers or a register and an immediate value
SUB: Subtracts one register or immediate value from another register
MUL: Multiplies two registers or a register and immediate value
LDR: Loads data from memory into a register with offset
STR: Stores data from a register into memory with offset

ADD
With Immediate:
ADD <DST_REG> <REG1> <IMM (simm8)>
ADD Rd Rn Imm

Without Immediate:
ADD <DST_REG> <REG1> <REG2>
ADD Rd Rn Rm

SUB
With Immediate:
SUB <DST_REG> <REG1> <IMM (simm8)>
SUB Rd Rn Imm

Without Immediate:
SUB <DST_REG> <REG1> <REG2>
SUB Rd Rn Rm

MUL
With Immediate:
MUL <DST_REG> <REG1> <IMM (simm8)>
MUL Rd Rn Imm

Without Immediate:
MUL <DST_REG> <REG1> <REG2>
MUL Rd Rn Rm

LDR
LDR <TARGET_REG> <REG1> <OFFSET (simm8)>
LDR Rt Rn offset

STR
STR <TARGET_REG> <REG1> <OFFSET (simm8)>

STR Rt Rn offset

If there is no offset, a value of 0 is needed.

Each instruction is 16 bits wide. Here is the breakdown of the bitstring:
Bit Organization:
6-Bit Opcode: 15-10
    * Bit 11 is a space holder
ALU Opcode: 9-6
Reg2 or Immediate: 5-4
Reg1: 3-2
Destination/Target Register: 1-0

Instruction	Opcode	ALU Opcode
ADD	110100	0000
SUB	110001	0001
LDR	11000	0000
STR	001001	0000
MUL	110100	0010

16 bits are used in this CPU. Two bits are necessary per each register and immediate input, Four are used for the ALU output.