Homework 2: CPUlator
CPE221

Instructor: Rahul Bhadani

Due: January 24, 2025, 11:59 PM
100 points

You are allowed to use a generative model-based AI tool for your assignment. However, you must submit an accompanying reflection report detailing how you used the AI tool, the specific query you made, and how it improved your understanding of the subject. You are also required to submit screenshots of your conversation with any large language model (LLM) or equivalent conversational AI, clearly showing the prompts and your login avatar. Some conversational AIs provide a way to share a conversation link, and such a link is desirable for authenticity. Failure to do so may result in actions taken in compliance with the plagiarism policy.

Additionally, you must include your thoughts on how you would approach the assignment if such a tool were not available. Failure to provide a reflection report for every assignment where an AI tool is used may result in a penalty, and subsequent actions will be taken in line with the plagiarism policy.

Submission instruction:

Upload a .pdf on Canvas with the format {firstname_lastname}_CPE221_hw02.pdf. For example, if your name is Sam Wells, your file name should be sam_wells_CPE221_hw02.pdf. If there is a programming assignment, then you should include your source code along with your PDF files in a zip file {firstname_lastname}_CPE221_hw02.zip. Your submission must contain your name, and UAH Charger ID or the UAH email address. Please number your pages as well.

---

1 Learning CPUlator

This assignment requires you to take an ARM assembly language program and simulate it.

The program is available in the file `array_plus_scalar.asm` on
https://github.com/rahulbhadani/CPE221_Sp2025/blob/main/Code/array_plus_scalar.asm

Perform the task instructed in Part A, do the screen recording using Zoom or any other screen

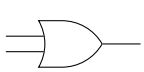
recording software, and upload your screen recording video to YouTube. Ensure your YouTube video's visibility is set to either unlisted or public. Provide the URL of your video as a part of this homework.

1.1 Part A: Understanding CPUlator (50 points)

1. Go to https://cpulator.01xz.net/ to access the CPUlator tool
2. From the "Choose a system to simulate" pop-up window select the Architecture **ARMv7** and the **System ARMv7 generic** then press **go**.
3. On the toolbar at the top of the window select **Help** → **Documentation** to pull up the documentation for CPUlator. Read the section entitled **Using the Simulator**.
4. Upload the file `array_plus_scalar.asm` to CPUlator by going to **File** → **Open** and browsing to the file location on your computer. Select the file and press **OK** (Note: You may have to change from searching for **S** Files to **All Files** in your file explorer to locate the `.asm` file).
5. Before compiling select the **Disassembly** and **Memory** windows to see their initial states.
6. The source code for the file should now be visible in the **Editor Window** of CPUlator. Compile the code by pressing the **Compile and Load** button at the top of the **Editor Window**. Once successful you should see the message **Compile succeeded** in the **Messages Window**.
7. Select the **Disassembly** and **Memory** windows again and note what has changed from the initial state. How have the assembly instructions changed between the Editor and Disassembly views? (**Note:** The actual lines of code that will be executed by the simulator have an address and opcode associated with them in the lefthand columns of the Disassembly window. Their equivalent instructions from the editor window are shown in grey above them and will not have an address/opcode in the Disassembly view).
8. In the **Disassembly Window** set a break point on the **done: B done** instruction (line 34).
9. Once you have set a breakpoint, press the **Continue** button on the toolbar to run the program until the breakpoint. Alternatively, you can step through the code instruction by instruction using the **Step Into** button.
10. Once you feel comfortable using the CPUlator tool answer the questions in Part B.

1.2 Part B: Assessment (50 points)

1. What is the value of the PC (decimal and hexadecimal) when the program ends execution?



2. What is the range of memory addresses associated with x?
3. What is the range of memory addresses associated with y?
4. What values are stored in y? List the value at each index of y in decimal and 32-bit hexadecimal.
5. Are the values for y correct for the computation performed?