UIC – ECE 366: Computer Organization – Spring 2020

Project 2: MIPS simulator in Python

For this project, you will work in a group to build a Python program, which takes a text file containing MIPS machine code (in hex) as input, simulates its running, and output various intermediate and final results. The ultimate goal is to run your perceptron program in the previous project, with some nice output showing the results in memory arrays.

Python simulator specs:

Your simulator should support:

- a subset of non-pseudo instructions in MIPS.
- registers:
  - `$0 (always  = 0) to $31, lo, hi`
  - `PC: starting at 0, increment by 4 for normal instructions`
- Instruction memory:
  - read from a text file (containing a program in valid MIPS machine code in hex)
- Data memory address range:
  - `0x2000 - 0x3000`
- You can assume all the registers / data memory content are initialized to be 0

Input: a text file containing a valid MIPS assembly program (in hex)

Output: on screen and a text file, nicely formatted:

- Step-by-step information (instruction info, updated register content, PC)
- Final results:
  a) all the registers' content (including lo, hi, and PC)
  b) Memory content
  c) Instruction Count and Statistics (see MARS Tool -> Instruction Statistics)

Groupwork notes:

- Work with your assigned group - members in the same group can submit the same content, but across group is considered plagiarizing.
- Bottom-line: if your group partners do not contribute, you are responsible for your own submission. Members in the same group are allowed to submit their own work different from others in the same group.
- Project 3 will assign you to a group of different members. Exception: your entire group can stay together if all the members make request via email to the instructor, by the submission deadline.

Project Report (PDF file) components:

(10 pts) Part I: Group Effort Log and Reflection

1) Summary of group activity – for example:

Meeting Time / Location	**Achieved / To-Do**	**Attended**
Week 5, Thu 5:30pm – 7pm @ Library 2nd floor	Done: - python installation and testing for all members - GitHub setup - homework 3 discussion - P1 MIPS code discussion and comparison To-do by end of Sunday: - individually finish -	Alice Bob Charlie
…		
Week 6, Sat 4-8pm, CS lounge	Done: - python code runs correctly for addi, add, sub, slt with 3 test cases - An updated Perceptron program (eliminating pseudo-instructions). To-do: - Fix bugs in beq and bne part (2 out of 4 testcases did not pass) - Test the correctness of the updated Perceptron program	Alice Charlie
…		

2) List of activities done by yourself

3) Reflections

a) List 3 significant things you learned (to do, or not to do) from your group members, or through the process of building this project.

b) List one thing that you will do differently for future projects like this.

(10 pts) Part II: Individual draft

Provide: 1) your own Python pilot program (cut off time: by the end of week 6) and 2) indicate how the final submission differs from your draft.

(80 pts) Part III: Project Results

1) (40pts) Testcase design and results

- Provide 4 testcases that you have designed to test the functionality of your simulator:
    A. ALU instructions
    B. branch instructions
    C. memory access instructions
    D. for everything

  For each testcases, provide:

    - .asm and machine code (in hex) side-by-side
    - screenshots or output of your Python simulator:
        - Step-by-step info per instruction
        - Final results
    - MARS screenshots including register content, memory content, and Instruction Statistics.

2) (40pts) Perceptron Program results

Prepare the following two versions of the perceptron program:

```
V1:     n = 100,  A = 5,    B = -6
V2:     n = 200,  A = -3,   B = -66
```

Perceptron protocol:

`Should begin with setting (n, A, B) in ($3, $1, $2):`

```
addi $3, $0, 100     # n = 100
addi $1, $0, 5       # A = 5
addi $2, $0, -6      # B = -6
```

`Should end with all the results written into data memory:`

```
a, b at M[0x2000], M[0x2004]
array x, y, C, starting from M[0x2020], M[0x2420], M[0x2820]
```

For each version, provide:

- .asm and machine code (in hex) side-by-side
- screenshots or output of your Python simulator – final result only
- MARS screenshots including register content, memory content, and Instruction Statistics.

up to 10 pts of extra credit will be provided for nice formatting f the Python program output.

Submission instructions

Submission Instructions:

Deadline: Mar 1st (week 7 Sunday) 11:59pm (end of day)

Late submission policy and penalty:

By the end of	late penalty on total score
Mon	5%
Tue	10%
Wed	15%
Thu	20%

What to submit: on Blackboard, individually (-5 if any code file is missing):

1. **`Report.pdf`:** **a self-contain PDF report writeup**

2. **`MIPS-sim.py:`**
   Python simulator code

3. **`testcaseA.asm, testcaseB.asm, testcaseC.asm, testcaseD.asm:`**
   MIPS asm code for the 4 testcases

4. **`PerceptronV1.asm, PerceptronV2.asm:`**
   MIPS asm code of the perceptron program, V1 and V2.

Appendix: Task breakdown

You can Use “Issues” on GitHub for task identification / assignment.

Begin with the following suggestions:

1. **File reading**: a text file of hex characters (example, memory dump from MARS) into a list of binary strings (MIPS program in machine code).
2. **Data structure design:** PC accessing instruction, update, and register file
3. **addi instruction:** simulation implementation
    a) Testcase building: MIPS programs consisting solely of addi instructions.
    b) positive imm number support
    c) negative imm number support
    d) thorough testing
4. **Output design:** what to show after executing an instruction
5. **General ALU instructions**
    a) Python coding
    b) Testcases building and testing for ALU instructions thoroughly
6. **Branch instructions**
    a) Python coding for beq / bne / j
    b) Testcases building and testing for branches thoroughly
    c) Integration (with other instructions), testing, and debugging
7. **Memory instructions**
    a) Data structure design for memory array [0x2000 – 0x3000]
    b) Python coding for lw / sw
    c) Testcases building and integration with other instructions
    d) Integration, testing, and debugging
8. **I/O enhancement** and other features
9. **Perceptron** on Simulator
    a) MIPS asm / machine code preparation
    b) Integration, testing, and debugging
10. **Final wrap-up**: testcase run and report preparation

Group 1st meeting notes

366 Project 2 Group ___________________

1. **Group member contact info & availability:**

2. **Timeline and meeting plan**

3. **Other notes**