UIC – ECE 366: Computer Organization - Fall 2020

Project 1: MIPS programming with MARS

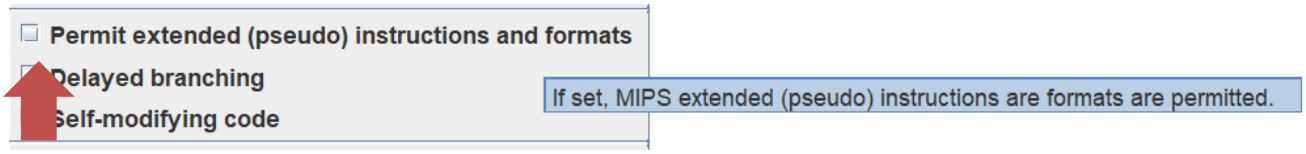
In this project, you will write a MIPS assembly program and verify that it runs correctly with the simulator MARS. This program consists of the following parts (I) generating an array of 100 numbers from 3 given values, (II) evaluating the width of each number to form another array of widths, and (III) forming the histogram array of all the widths.

This program will be used for subsequent projects of this class.

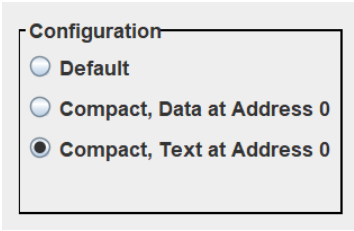
MARS Setup:

when programming your MIPS assembly code, MARS needs to be set with the following:

- "Permit extended (pseudo) instructions and formats" option UNCHECKED such that extended (pseudo) instructions are NOT permitted.



- MIPS Memory Configuration should be set to the 3rd option: Compact, Text at Address 0



Input / Output specs:

- Your program must begin with the following 4 lines of code to initialize A, B, and C:

```
addi $8, $0, 5          # A = 5
addi $9, $0, -6         # B = -6
lui $10, 0xCD           # C = 0x00CD0000
ori $10, $10, 0x1234    # C = 0x00CD1234
```

  Therefore, by changing these 4 instructions, we can initialize A, B, C to any values to test your program.

- Your program should produce the resulting arrays in the following memory locations:

```
A1 – A100:          at M[0x20A0, 0x20A4, … ]
W1 – W100:          at M[0x22A0, 0x22A4, … ]
H0 – H32:           at M[0x2000, 0x2004, … ]
```

Part (I) – generating an array of numbers A1 – A100:

With the given parameters A, B, and C, your MIPS program should produce an array with the following formula:

```
A1   = (A  * B) XOR C
A2   = (A1 * B) XOR C
A3   = (A2 * B) XOR C
...
A100 = (A99 * B) XOR C
```

Each of the number in Ai is a 32-bit word. When multiplication results in a larger than 32-bit number, use the lower 32-bit part.

Part (II) – generating width array W1 – W100:

For each number in A1 – A100, compute its "width" to form an array W1 – W100, such that Wi = width(Ai). Width is defined as the number of bits between the leftmost 1 and the rightmost 1 (including both) in the 32-bit binary representation.

For example:

```
width(0000 0110 0000 0000 0000 1100 0000 0000) = 21
width(0000 1100 0000 0000 0000 0000 0000 0000) = 2
width(1000 0000 0000 0000 0000 0000 0000 0011) = 32
width(0000 0000 0000 0000 0000 0000 0000 0000) = 0
width(1000 0000 0000 0000 0000 0000 0000 0000) = 1
```

Part (III) – generating the histogram array of the widths H0 – H32

All the 100 width numbers must be in the range of [0, 32]. Here, you will build a "Histogram" array H0 – H32, counting how many times a number occurs in the width array W1 – W100. Specifically, H0 stores the count of 0, H1 stores the count of 1, etc.

For example (with a smaller array to illustrate the point):

```
W1:W10 = [0, 3, 1, 5, 5, 5, 5, 0, 3, 0]

0 occurred 3 times, thus H0 = 3;
1 occurred 1 time, thus H1 = 1;
3 occurred 2 times, thus H3 = 2;
5 occurred 4 times, thus H5 = 4;

Thus H0:H5 = [3, 1, 0, 2, 0, 4]
```

Extra Credit:

Part (IV) Sort the array of A1 to A100 according to their width, and store this sorted array in `M[0x24A0, 0x24A4, … ]`, with their corresponding weight in `M[0x26A0, 0x26A4, …]`.

Levels of completion for the technical part:

1) (60%) Your code should achieve (I)
2) (80%) Your code should achieve (I) + (II)
3) (100%) Your code should achieve (I) + (II) + (III)
4) (5% extra credit) will be given to the top 3 individuals (or groups) with the lowest Dynamic Instruction Count to accomplish part I, II and III.
5) (5% extra credit) for part IV.

Technical Report components:

1. General intro and Q&A (-2 points each if answer is missing)
   a) Which parts (I, II, III, IV) does your program achieve?
   b) What was the most difficult part of this project?
   c) (if you worked individually) How many hours have you spend in total on this project?
   d) (if you worked in group) Provide a rough breakdown of each member's help / contribution to what you are able submitting here (for example: 30% / 30% / 40%).

2. Program functionality
   Run your program for each of the following configurations:
   a. `A = 5, B = 2, C = 0x00000000`
   b. `A = 5, B = 2, C = 0x0000FFFF`
   c. `A = -3, B = -7, C = 0xFFFFFFFF`
   d. `Your selection of A, B, C that best "spreads" out the histogram of H0 to H32.`

   Provide MARS screenshots of your program's results for each of the above configurations, including:
   - Final results of all the arrays
     - `show MARS' data memory content at regions such as:`
       - `0x2000, 0x2020, 0x2120, etc.`
   - Dynamic Instruction count of completing your program
     - `use MARS Tools -> Instruction Statistics`

3. Appendix
   Include your MIPS assembly code here (-5 if missing).

Overall Grade:

If you work individually, you will be graded 100% of the technical part.
If you work in a group, your grade will consist of 10% of group participation (based on Piazza activities) and 90% of the technical part.

What to submit: upload the following **`two files`** on Bb

- **`report.pdf`** the technical report specified above.
- **`p1.asm`** your MIPS assembly code (so that we can run your code on MARS – missing it will result in -5 points).

Deadline: **week 4 Sunday (end of the day) on Bb.**

Late submission policy and penalty:

submission by the end of	late penalty on your total score
Sun - due date	0%
Mon	5%
Tue	10%
Wed	15%
Thu	20%