# CDA 3103 Computer Organization Homework

The problems in this assignment will be divided into two parts: Part A and Part B. You can select to complete this assignment with a partner (pair programming) or independent.

- If you select to complete this assignment independently, you only need to complete the problems in Part A and you can use problems in Part B for extra exercise.
- If you select pair programming, one student only needs to submit answers to problems in Part A and the other student only needs to submit the answers to problems in Part B.

How the pair programming works:

- Let us call two students in one team as student A and student B.
- Student A is the primary developer for problems in Part A and student B is the primary developer for problems in Part B.
- Student A should submit the final answers for Part A while student B should submit the final answers for Part B.
- For each problem, students A and B should use one computer for assembly code developing. The primary student develops the code and explains the code to the other student. If two students have different opinions, please continue further discussion, and learn from each other or other resources until come to the final conclusion.

**For the whole assignment, pseudo-instructions are not allowed except "j target_label".** One suggestion for assembly programming problems is that you can include comments to each instruction.

## Section I: Problems

### Part A:

1. (3 points): Please circle your choice: Pair programming or Independent.
   If your choice is pair programming, please specify the name of your partner: and which part of problems for your submission:_____.
2. (5 points) The **`NOR`** instruction is not part of the RISC-V instruction set because the same functionality can be implemented using existing instructions. Write a short assembly code snippet that has the following functionality: **`s3 = s4 NOR s5`**. Use as few instructions as possible.
3. (12 Points) Write RISC-V assembly code for placing the following immediate constants in register **`s7`**. Use a minimum number of instructions.

   a) 59

   b) -199

   c) 0xDDCBE289

   d) 0x11236BDF

4. (16 Points) Convert the following high-level code into RISC-V assembly language. Assume that the signed integer variables **g** and **h** are in registers **t0** and **t1**, respectively. You can use other temporary registers like **t2** and **t3** if needed.

**a)**

```
if (g > h)
   g = (g + 7) * 2.5;
else
   h = (h – 6)/16;
```

**b)**

```
if (g <= h)
   g = (g - h) % 32;
else
   g = (g + 3 * h) * 14;
```

5. (24 Points) Convert the following high-level code into RISC-V assembly language. Assume that the signed integer variables **g** and **h** are in registers **t0** and **t1**, respectively. You can use other temporary registers like **t2** and **t3** if needed.

**a)**

```
if (g % 8 == 3 || g % 8 == 5)
   h = h * 5.5;
else
   h = h % 16;
```

**b)**

```
if (g % 16 > 4 && g % 16 < 12)
   h = (g + 5 * h) % 8;
else
   h = (g + 8.5 * h) * 9;
```

6. (15 points) Comment on each snippet with what the snippet does. Assume that there is an array, **int arr [6] = {3, 1, 4, 1, 5, 9}**, which starts at memory address **0xBFFFFF00**. You may assume that each integer is stored in 4 bytes. Register **a0** contains **arr**'s address **0xBFFFFF00.**

**a) (5 points)**

```
lw    t0, 0(a0)
lw    t1, 8(a0)
add   t2, t0, t1
sw    t2, 4(a0)
```

**b) (10 points)**

```
          add    t0, x0, x0
loop:     slti   t1, t0, 6
          beq    t1, x0, end
          slli   t2, t0, 2
          add    t3, a0, t2
          lw     t4, 0(t3)
```

```
            sub    t4, x0, t4
            sw     t4, 0(t3)
            addi   t0, t0, 1
            j      loop
    end:
```

7. (25 points) Write a RISC-V assembly snippet code to find the maximum and minimum elements in an array. Assume that the base address of array **arr** and the **size** of the array are held in register **a0** and **a1**, respectively. You can use temporary registers if needed.

## Part B:

1. (3 points): Please circle your choice: Pair programming or Independent.
   If your choice is pair programming, please specify the name of your partner: and which part of problems for your submission:_____.
2. (5 points) The **NAND** instruction is not part of the RISC-V instruction set because the same functionality can be implemented using existing instructions. Write a short assembly code snippet that has the following functionality: **s3 = s4 NAND s5**. Use as few instructions as possible.
3. (12 Points) Write RISC-V assembly code for placing the following immediate constants in register **s7**. Use a minimum number of instructions.
   a) 45
   b) -119
   c) 0xFEDCB8AB
   d) 0xAABCD325
4. (16 Points) Convert the following high-level code into RISC-V assembly language. Assume that the signed integer variables **g** and **h** are in registers **t0** and **t1**, respectively.

```
a)
    if (g > h)
       g = (g + 9) * 4.5;
    else
       h = (h – 5) / 8;
b)
    if (g <= h)
       g = (g + 3 * h) % 16;
    else
       g = (g – 2 * h) * 12;
```

5. (24 Points) Convert the following high-level code into RISC-V assembly language. Assume that the signed integer variables **g** and **h** are in registers **t0** and **t1**, respectively. You can use other temporary registers like **t2** and **t3** if needed.

```
a)
    if (g % 16 == 9 || g % 8 == 12)
       h = (g - 0.5 * h) * 3.5;
    else
       h = (g + 1.5 * h) % 8;
b)
    if (g % 32 > 6 && g % 32 < 17)
       h = (g + 6 * h) % 4;
    else
       h = (g + 7.5 * h) * 7;
```

6. (15 points) Comment on each snippet with what the snippet does. Assume that there is an array, **`int arr [6] = {3, 1, 4, 1, 5, 9}`**, which starts at memory address **`0xBFFFFF00`**. You may assume each integer is stored in 4 bytes. Register **`a0`** contains **`arr`**'s address **`0xBFFFFF00.`**

```
a)  (5 points)
    lw    t0, 4(a0)
    lw    t1, 8(a0)
    add   t2, t0, t1
    sw    t2, 0(a0)
b)  (10 points)
              addi   t0, a0, 0
              addi   t1, a0, 24
    loop:     beq    t0, t1, end
              lw     t2, 0(t0)
              sub    t2, x0, t2
              sw     t4, 0(t0)
              addi   t0, t0, 4
              j      loop
    end:
```

7. (25 points) Write a RISC-V assembly snippet code to find the length of a string. Assume that the base address of string **`str`** is held in register **`a0`**. You may assume that each character is stored in 1 byte. You can use temporary registers if needed.

## Section II: Submission Requirements

The following requirements are for electronic submission via Canvas.

- Your solutions must be in a single file with a file name yourname-module4-assignment-1.
- Upload the file by following the link where you download the homework description on Canvas.
- If scanned from hand-written copies, then the writing must be legible, or loss of credits may occur.
- Only submissions via the link on Canvas where this description is downloaded are

graded. Submissions to any other locations on Canvas will be ignored.