Lab 2: Arrays

Objective

Introduce the concepts and general practices for programming VerySimpleCPU using our Instruction Set Simulator (ISS).

Background

Bubble sort is a simple sorting algorithm. It means that the sort repeatedly compares pairs of elements and swaps them if needed. Bubble Sort is organized as a series of passes. In each pass, you compare the 1st and 2nd elements and swap if out of order; then the 2nd and 3rd elements; and so on. At the end of the first pass, the largest element is guaranteed to be at the end of the array.

What To Do

Make sure you have watched this video before you do the lab:
https://www.youtube.com/watch?v=wktP_xjbM7Q
The video is also here if you have trouble viewing it on Youtube:
https://ozu.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=e045f885-772c-444b-9a91-ace701009605

Part 1

Step 1

You have been given a sample assembly code. The changes in RAM while this code is running are shown. Examine the sample question and answer. Show the RAM changes for the given question. Note: If you have instructions you do not know, you can learn in the InstructionSet_README.txt file. Upload your solution to LMS.

Sample Question: Trace the contents of memory when the below VerySimpleCPU program is run. Express the progression of memory contents in a table.

```
0: CP 50 51
1: MUL 50 52
2: CP 100 50
3: SRLi 100 3
4: ADDi 100 1
5: CP 101 53
6: LT 101 100
7: BZJ 102 101
8: CPi 50 5
9: ADD 50 51
10: ADDi 50 1
11: BZJi 12 0
12: 11
50: 5
51: 4
52: 16
53: 10
102: 10
```

Answer:

PC:		0	1	2	3	4	5	6	7	10	11
50:	5	4	64							65	
100:				64	8	9					
101:							10	0			

Question: Trace the contents of memory when the below VerySimpleCPU program is run. Express the progression of memory contents in a table just like the one above. Note that the number after 0x is a hex number. Create a file called lab02_part1_RAM.txt and write your solution.

```
0: NAND 202 202
1: ADDi 202 1
2: ADD 201 202
3: CP 203 201
4: NANDi 203 0x40000000
5: BZJ 200 203
6: NAND 201 201
7: ADDi 201 1
8: BZJi 9 0
9: 8
200: 5
201: 8
202: 3
```

Step 2

You will write low level C code that sorts the numbers in **ascending order** using **bubble sort.**

1. Open and analyze the lab02_part1_hi.c file with a text editor. (I suggest you open with notepad++.)
2. Create a new file called lab2_part1_low.c
3. Convert lab2_part1_hi.c to lab2_part1_low.c (lab2_part1_low.c is low level c code of lab2_part1_hi.c)

Part 2

1. Make an assembly code incrementing by 10 (ten) each elements of an array A. The address of A[0] is 400, the address of A[1] is 401. Use CPI and CPIi when accessing A. Call this file lab02_part2.asm.
2. Test it for A[0] = 137 and A[1] = 224. Hence, at the end of your program we will see A[0] = 147 and A[1] = 234.

Part 3

1. Modify sortx.asm to make the assembly work correctly. You will be given comments on the appropriated task to be performed in low-level C, simply translate it into ASM.

Submission

- Submit the following files in LMS under the assignment LAB01. Do not zip your files, upload them directly on LMS!
    - lab02_part1_RAM.txt
    - lab02_part1_low.c
    - lab02_part2.asm
    - sortx.asm