Computer Architecture Fall 2025

Programming Assignment 1

- **Deadline**: 23:59, 2025-10-20 (no late submissions allowed)
- **Submission**: via NTU COOL, detailed instructions here
- **Contact**: If you have any questions, please email the TAs with the subject "[CA2025 HW1]".
  - Circle: r14943176@ntu.edu.tw
  - Alex: r14943141@ntu.edu.tw

Environmental Setup and General Description

In this assignment, you will write RISC-V assembly programs in a simulator named **Jupiter**.

- Download Jupiter here: https://github.com/andrescv/jupiter
- Documentation of Jupiter: https://jupitersim.gitbook.io/jupiter/

Download the corresponding app image for your OS, cf. its installation guide.

You will receive a compressed folder: `CA2025_PA1.tgz`, where you will find the following partially implemented assembly programs. Each program includes the predefined input format and console output, as well as sections marked with `### TODO ###`, where you should implement your solutions.

The folder structure is as follows:

```
CA2025_PA1/
│
├── p1_1.s   # Problem 1-1: Fibonacci (Recursive)
├── p1_2.s   # Problem 1-2: Fibonacci (Iterative)
├── p2.s     # Problem 2: Maximum Subarray Sum
├── p3.s     # Problem 3: 2-Sum
└── p3_bonus.s     # Problem 3 Bonus Hint
```

For more information on RISC-V assembly programs, please visit its official website for the programmer's manual. For example, you could find a list of *pseudo-instructions*, which may help with your assignment.

Problem 1: The n-th Fibonacci Number (30%)

Task:

Write a program that takes an integer `n` as input in register `a0` and outputs the n-th Fibonacci number in register `t0`.

This problem consists of two subtasks (15% each):

- **Problem 1-1 (Recursive)**: You must implement Fibonacci using a *recursive approach*. The function should call itself to compute fib(n) until reaching the base cases. Using loops is **not allowed** for this subtask.
- **Problem 1-2 (Iterative)**: You must implement Fibonacci using an *iterative loop*. Compute fib(n) by updating variables in a loop. Using recursion is **not allowed** for this subtask.

Definition:

- fib(0) = 0
- fib(1) = 1
- fib(n) = fib(n-1) + fib(n-2) (n $\geq$ 2)

Example:

- Input: `n` = 8 Output: 21
- Input: `n` = 12 Output: 144

Constraint:

- 0 $\leq$ `n` $\leq$ 30

Problem 2: Maximum Subarray Sum (30%)

Task:

Given an integer array `A` and its length `len`, write a program to compute the maximum value of the sum of any contiguous subarray in `A` and save it in register `t0`.

Example:

- Input: `A` = [$-$2, 1, $-$3, 4, $-$1, 2, 1, $-$5, 4], `len` = 9 Output: 6 (corresponding to the subarray [4, $-$1, 2, 1])
- Input: `A` = [5, -2, 3, -1, 2], `len` = 5 Output: 7 (corresponding to the subarray [5, -2, 3, -1, 2])

Constraint:

- 1 $\leq$ `len` $\leq$ 100
- -1000 $\leq$ `A[i]` $\leq$ 1000 for each element

Problem 3: Two Sum (30% & bonus 10%)

Task:

Write a program that takes an integer array `A` and its length `len` as input and outputs the indices of **two distinct elements** in `A` whose sum equals a given target value `target`.

Bonus: Compute the result in O(n) time complexity, and you will get a bonus 10%.

- If such a pair exists, store the indices in registers `t0` and `t1`. Else, store -1 in both `t0` and `t1`.
- You may assume that each input would have **exactly one solution**, and you may not use the same element twice.

Example:

- Input: `A` = [2, 7, 11, 15], `len` = 4, `target` = 9 Output: `t0` = 0, `t1` = 1
- Input: `A` = [1, 2, 3, 4], `len` = 4, `target` = 8 Output: `t0` = -1, `t1` = -1

Constraint:

- $1 \leq$ `len` ≤ 100
- $0 \leq$ `A[i]` ≤ 99
- -2000 $\leq$ `target` ≤ 2000
- Each element may only be used once (i.e., choose two distinct indices i and j, i $\neq$ j)

Submission Guidelines

Please follow the file structure below when submitting your solutions.

```
<studentID>_PA1/
│
├── p1_1.s   # Problem 1-1: Fibonacci (Recursive)
├── p1_2.s   # Problem 1-2: Fibonacci (Iterative)
├── p2.s     # Problem 2: Maximum Subarray Sum
└── p3.s     # Problem 3: 2-Sum
```

- **Do NOT modify the file names!**
- For Problem 3, submit only p3.s. The bonus points will be awarded automatically if your implementation meets the runtime requirements.
- The top-level folder must be named with your student ID (first letter **capitalized**).
- Compress the entire folder into a **`.tgz`** file and submit it via NTU Cool.

Deadline: 2025-10-20. No late submission.