# Computer Organization

**University at Albany**
**Department of Computer Science**
**ICSI 504 – Fall 2023**

**Project-2**

**Assigned: Thursday, October 12th, 2023.**

**Due: Tuesday, November 7th, by 11:59 PM. No submission will be accepted after the due date.**
**<u>Unlimited number of submissions is allowed.</u>**

## Purpose of the Project

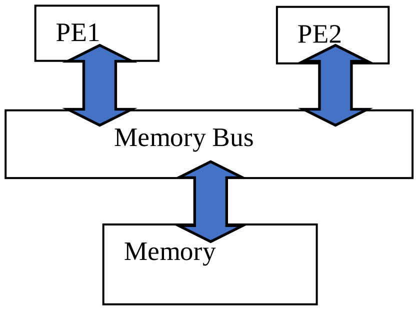
The goal of this exercise is to extend your stack-based microprocessor simulator by developing a *duo core processor*. The additional core will share the same properties of the original processor you have developed in project-1.

## The Details

You are to develop a simulator for a duo-core version of the **abm** abstract machine. Your simulator must support all instructions included in the **abm-instruction-set.doc** document. The following new instructions are to be included in your new version of the simulator.

| **Syntax** | **Semantics** |
|---|---|
| **.data** | Indicates that the following data items are stored in the data segment. All variables declared in this segment are global. This means they can be manipulated by all cores. |
| **.text** | Indicates that the items (instructions) that follows it are stored in the user text segment. |
| **.int $d_1$, …, $d_n$** | Stores ***n*** integers in successive data memory locations. |
| **:&** | Stack top is replaced by the **lvalue** below it, and both are popped. Operands are both memory addresses.<br>Ex.: Assign a memory address to another memory address.<br>Usage: **lvalue x**<br>**lvalue y**<br>**:&** |
| **+, -** | Both addition and subtraction original operators are overloaded to represent the addition and subtraction of memory address respectively.<br>Ex.: Increment a memory address and assign it to another memory address.<br>Usage: **lvalue x**<br>**lvalue y**<br>**push 1**<br>**+**<br>**:&** |

The new version of your simulator will have access to a shared memory by means of a shared memory bus.



All processor communications with memory will be under the control of the memory bus component. An adequate processor and memory synchronization mechanism is to be developed to ensure data integrity of the memory component.

## What to report

Each student must provide an itemized list of his or her own contributions to the following:

1. Software design (whole system or list specific modules),
2. Coding (whole system or list specific modules),
3. Debugging (whole system or list specific modules),
4. Report preparation (whole report or list specific sections/diagrams),
5. Other (any other relevant contribution).

## How to report

The report must contain the following sections and sub-sections. Each section/subsection should be clearly delineated with a proper heading.

1. External Documentation including [5-10 pages]:
    a. Title page (Names and UAlbany IDs of group members must be included here).
    b. A table of contents.
    c. [12%] System documentation.
        i. A high-level data flow diagram for the complete system.
        ii. A list of routines and their brief descriptions.
        iii. Implementation details.
    d. [10%] Test documentation.
        i. How you tested your program.
            - Data used for all tests done.
            - Testing outputs produced.
        ii. List of program bugs and your plans to address them.
        iii. Executable version of your solution.
        iv. Difficulties and solutions involved in creating the program.
    e. [4%] Algorithms and data structures.
        i. Algorithms. Does your system use any complex algorithm?

If NO, skip to the next item,
If YES, describe your algorithm(s).

ii. Data structures. Does your system use any complex data structures such as arrays, linked lists, stacks, queues, graphs, hash tables, or trees?
If NO, skip to the next item,
If YES, what criteria you used in deciding each data structure used, e.g., performance vs. flexibility?

f. [5%] User documentation.
    i. What operating system was used.
    ii. How to compile your program.
    iii. What other applications (tools) does your system require.
    iv. How to run your program.
    v. Describe parameters (if any).
    vi. Other requirements needed to run your executable.

2. Source Code
    a. [67%] Correctness.
    b. [2%] Programming style.
        i. Layering.
        ii. Readability.
        iii. Comments.
        iv. Efficiency.

## What to Submit

- Your solutions must be submitted through Brightspace.
- Your files for *Project-2* must include the source code for all modules used for the exercise as well as an executable file to allow your solution to be tested. You must place all your source files as well as your testing results in a Microsoft Compressed (zipped) folder (.zip). Your (.zip) folder must follow the format: *504 Project-2 Your Name*. Marks will be deducted if you do not follow this requirement.