# Programming Project – Basic Machine Simulator

This product consists of 4 parts, numbered with Roman Numerals I, II, III, and IV. Information regarding the project and its requirements are provided in the course web page. The following key elements of the project information set are:

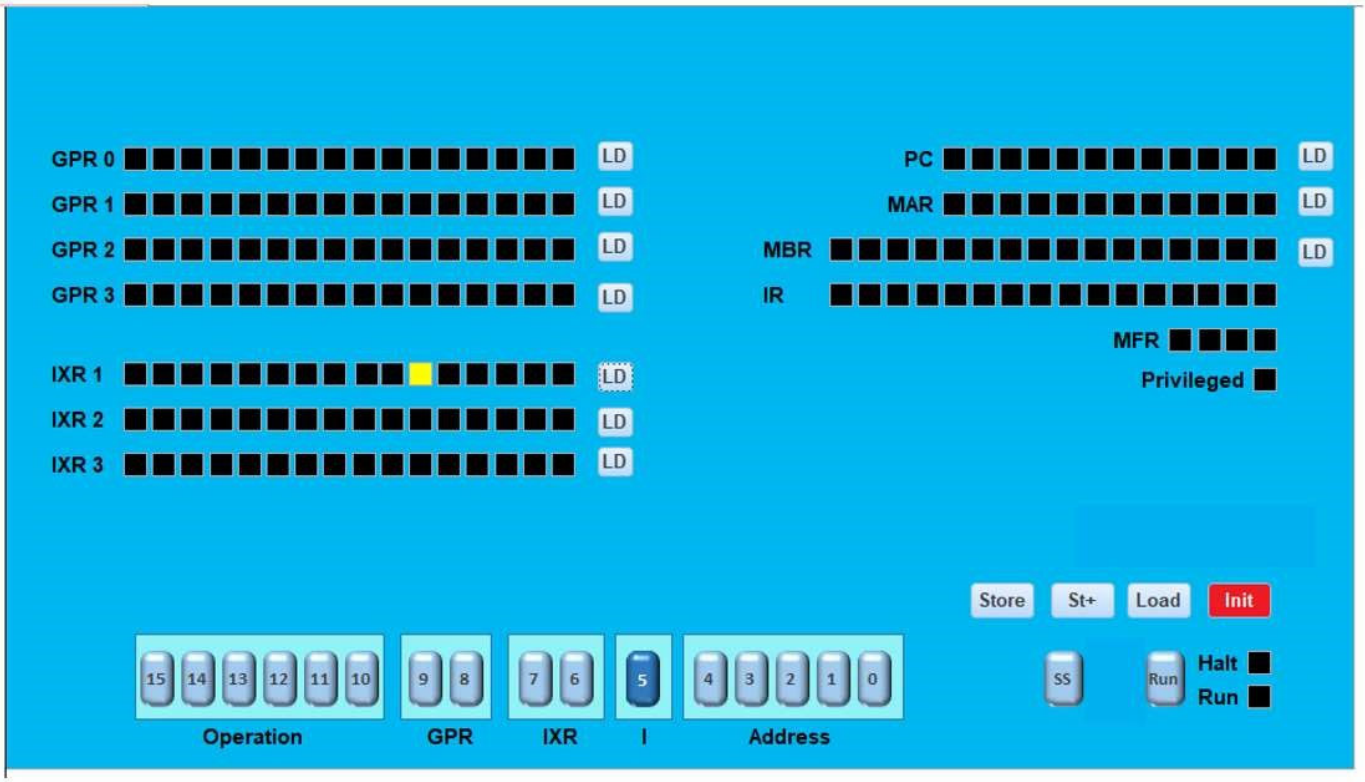
- Project Description Document
- Project Rubric
- Project Team List

The project will be developed by teams of 3 to 4 students. Teams will be assigned by me. Teams will be frozen by the end of the second week. I will make an initial assignment of the teams. The reason for the delay in team assignments is to confirm that students will be taking the course. My assignments of teams will be final. The project will require that you work well with other members of the team. All members should be able to participate in the development of the project.

## Project 1 Front Panel Requirement

| Machine Register | Function |
|---|---|
| PC | Display and Change |
| MAR | Display and Change |
| MBR | Display and Change |
| GPR Registers 0-3 | Display and Change |
| Index Registers 1-3 | Display and Change |
| CC | Display |
| MFR (memory fault register) | Display |
| IR | Display and Change |

Below is a sample front panel display. Yours does not have to match exactly except that the required items above must be on the front panel.




Note that the switches at the bottom were set to 0000 0000 0010 0000. The panel shows that the load button (LD) was pressed next to IXR 1 resulting in IXR 1 being loaded with this value and the display values set so that the appropriate light (yellow) is set.

- You may use text boxes that display either the binary values in 0s and 1s or hexadecimal numbers. You DO NOT have to implement the lights independently.
- I have specified , such as SS which would go through a single cycle, that is, would go through fetch, then decode steps, etc. The St+ button is a neat, not required button that when you set the MAR and MBR and press St+, the machine stores the value in the MBR into the memory location indicated by the MAR and increments the MAR to the next machine location so that one only has to load the MBR for the next location. Useful in loading programs.
- The Privileged indicator is not required. It was put there to indicate that the user program tried to execute within the first 6 locations.
- You should have a condition code display, (after Project 1).
- The Init button is the IPL (Initial Program Load) button, you may label it as you like.

## Additional Comments on Project I

- You may use text boxes that display either the binary values in 0s and 1s or hexadecimal numbers. You DO NOT have to implement the lights independently.

- I have specified , such as SS which would go through a single cycle, that is, would go through fetch, then decode steps, etc. The St+ button is a neat, not required button that when you set

  the MAR and MBR and press St+, the machine stores the value in the MBR into the memory location indicated by the MAR and increments the MAR to the next machine location so that one only has to load the MBR for the next location. Useful in loading programs.
- The Privileged indicator is not required. It was put there to indicate that the user program tried to execute within the first 6 locations.
- You should have a condition code display, (after Project 1).

## Loading a Program

The machine front panel should contain an Initial Program Load (IPL) button that restarts the machine and loads a file IPL.txt into memory. The file is in a format that contains two hexadecimal numbers per line, the first being a machine address, and the second being the content at that address. Example:

```
0000    004F
000A    0009
0020    000A
```

The results of loading the above would be (in

decimal) location 0 would get 79 (004F)
location 10 would get 9 (0009)
location 32 would get 10 (000A)

This will be the means in which we load programs from a file. I will use files in this format to test the simulator. You should provide the file name that the machine loads from.

Additional note. The simulator example above numbers the bits increasing from right to left to reflect the numerical significance of the bits. Note that the project description shows bit positions increasing left to right.

# Project 1 Delivery Requirement

Refer to the Project Rubric linked to the course syllabus web site. It shows how the projects will be graded. A sample of project 1 is provided below.

Note the deliverable column

**GRADING RUBRIC FOR TEAM PROJECTS**

**The grading rubric below itemizes the requirements and point value for each of the 4 deliverables for the computer simulator. The grader may give partial credit in increments of .1 . The grade will be posted for each project in the form of 0-100 percent.**

| REQUIREMENT | SCORING | PER CENT | ADDITIONAL ADVICE |
|---|---|---|---|
| | | | |
| **Deliverable I** | | | |
| Delivered as working JAR file | 2.5 | 50% | Ensure that your JAR File will execute on the grader's machine - Test the JAR thoroughly |
| Includes correct user guide which is easy to understand and use. | 1 | 20% | The grader should not have to contact the team for clarification on loading the file |
| Source code included and well documented | 1 | 20% | Code fully documented with header comments for each module/class file. Comments provided under each section of code describing purpose. |
| Design notes included | 0.5 | 10% | Design notes provided and match code structure. Clear with respect to description of the classes and code architecture, including additional functions. |
| | | | |

## Sample Programming Quick Start Guide for Users

To facilitate ease of grading and gain a sense of quick start documentation for users not familiar with the implementation language or environment, the following quick start are to be provided. This quick start guide is part of the Rubric evaluation of the "user" (actually the grader) being able to execute the simulator.

Usage Instructions

1. Download the following files from the blackboard

   <add the file names to be downloaded>

2. Preparation instructions
   a. How to unzip, start or prepare the environment for execution. Programs should be developed so as to execute with minimal preparation. Any special environment instructions are to be in detail.

   b. This item also contains the locations of:
      i. The file to run the simulator. This includes, if necessary, any ribbon item selections in the environment in which the program is load (for example, File->Run indicates clicking the File menu in the window and then clicking Run.

         Remember the user will not necessarily know how to use command line interfaces. All should be push button. No Unix or Windows command line interfaces required. If necessary, develop batch files.

ii. Input file names and required location (IPL.txt, etc.). Where are they located relative to the executable file? iii. All these should be placed within the zip file in proper location so as to eliminate necessity to move them.

3. Operation of the Simulator
    a. Starting location where the simulator executes code (7, 10, 100?)
    b. How to IPL
    c. How to Single Step
    d. How to Run the machine code
    e. How to read memory locations

## Floating Point

In the implementation of Floating Point instructions, you will use 16 bit floating point. A Wikipedia and other articles are available. See this link

https://en.wikipedia.org/wiki/Half-precision_floating-point_format