**Computer Simulator Simple Documentation**

# Introduction:

This document describes how to use our simulator in simple steps. The document is split into five sections:

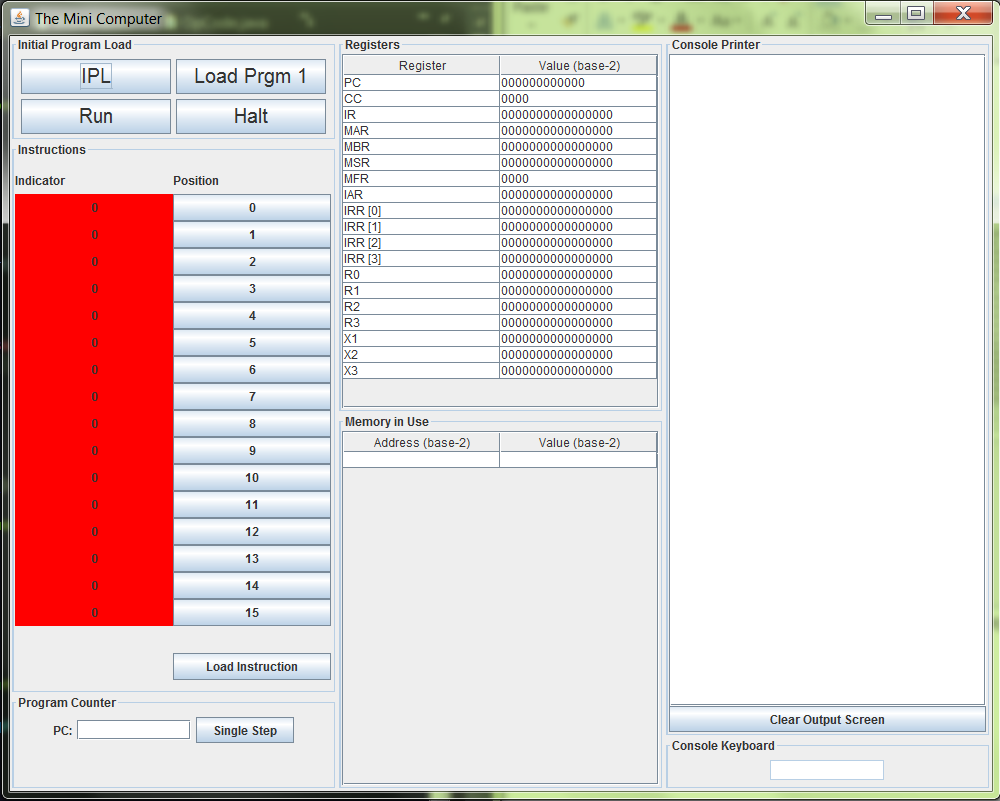
1. Introduction
2. How to launch the Simulator
3. What is the console layout
4. How to operate the simulator from the console.
5. User’s Bit-Instruction Manual

This document will evolve to include additional steps as the simulator project progresses through the four planned phases of agile development.

# How to launch the Simulator?

1. Please make sure JAVA 1.8 or later is installed.
2. Download the JAR file, boot program file, and program 1 files
3. The program files and the JAR file need to be in the same directory
4. Double-click the JAR file to launch the console

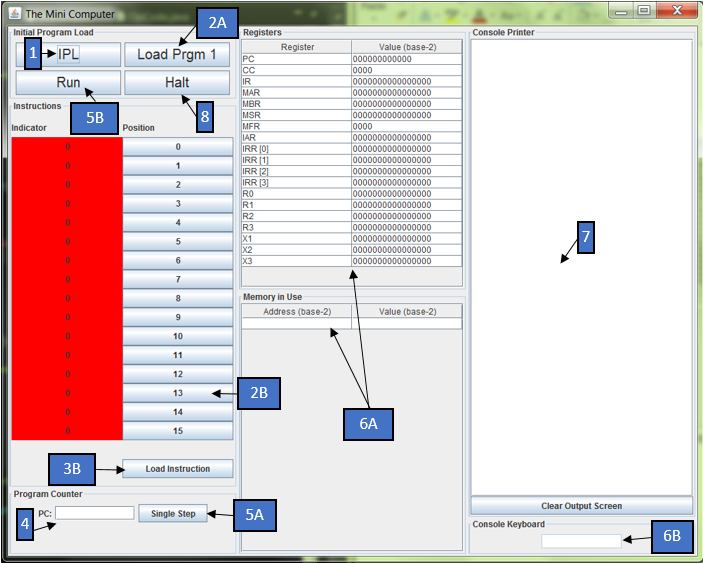
# What is the console layout?



Our mini-computer has a graphical user interface (GUI) console. It is split into seven panels:

1. Initial Program Load (IPL)
   1. Initial Program Load (IPL) – Contains the initial booting sequence options by activating the IPL button.
   2. Halt – The Halt button will stop the execution of any program that is running and return control to the user interface. The PC will remain at the address of the instruction when Halt was executed.
2. Instructions – the console allows 16 bits of information to be loaded into the computer one instruction at a time from the console.
3. Program Counter – Allows the user to specify the program counter to single step
4. Registers – displays the register and the value contained within the register
5. Memory in Use – displays the address and value stored in memory in base-2 notation.
6. Load Prgm 1 button – loads program 1 into memory (Starting at address 30), IPL should be pressed before pressing this button
7. Run button – execute the instruction pointed to by PC until either a HLT instruction is encountered or the user presses the Halt button on the GUI
8. Console Printer – displays output
9. Console Keyboard – is used for entering input.
   1. The console keyboard will only accept digits and the Enter key, and only when the current instruction is IN
   2. Enter key - used when the entire number has been entered by the user

# How to operate the simulator from the console.



Step 1: From the console the minicomputer simulator can be launched by activating (clicking) the IPL button. This IPL button loads and executes the boot program, which loads values into memory and registers (see label 6 for memory and register values).

Step 2A: Press the Load Prgm 1 button to load Program1\_bin.txt into memory, starting at address 30 (000000011110 in binary)

Step 2B: Alternatively, instructions to the minicomputer can be entered into the instruction panel by activating bit-toggles (i.e. clicking buttons enumerated from 0-15). A green light indicates the corresponding bit position’s inclusion into the instruction. The user manual explains the available commands and basic bit-instruction usage.

Step 3B: Activating the ‘Load Instruction’ button will load the operator’s instruction into the minicomputer. Note: The default location for console instructions is “000000000110”

Step 4: From the Program Counter Panel the operator can elect to execute step-by-step instructions that have been loaded into the computer. The PC should contain the address of the (first) instruction to be executed.

Step 5A: Activate the Single Step button to execute the instruction.

Step 5B: Alternatively, click the Run button to execute instructions until a Halt instruction is encountered or the Halt button is pressed.

Step 6A: Observe the effect of your bit instruction(s) in the minicomputer’s Registers and Memory.

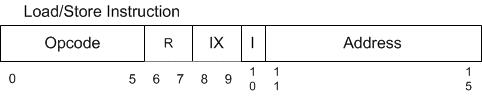
Step 6B: If the bit instruction is IN, input one digit, or the Enter key if the entire number has been entered.

Step 7: The user will be able to see output to the console in the Console Printer.

Step 8: To return control back to the console the Halt button can be used to interrupt a program running.

# User’s Bit-Instruction Manual

Instructions for the minicomputer and the accompanying options for that instruction are broken down by the bit-position ranges in the figures and tables below. The Opcodes are the basic instruction options for the minicomputer. The general purpose register (R) will be used if an Opcode indicates a Load from register, or Store to register instruction. The index register (IX) will be used if an Opcode specifies a Load from index, or Store to index instruction. The indirect addressing indicator (I) tells the Opcode how to interpret the instruction’s address field. If I=0 then the Opcode will read the address value directly. However, if I=1 then the Opcode field will be a pointer to another address location.



The basic interpretation of the bit-wise instruction can be summarized in the table below.

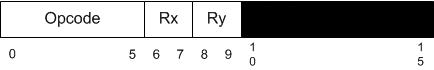
|  |  |  |
| --- | --- | --- |
| **Field** | **#Bits** | **Description** |
| Opcode | 6 | Specifies one of 64 possible **instructions**;  Not all may be defined in this project |
| R | 2 | Specifies one of four general purpose **registers**; may be referred to by R0 – R3 |
| IX | 2 | Specifies one of three **index registers**; may be referred to by X1 – X3. O value indicates no indexing. |
| I | 1 | If I =1, specifies **indirect addressing indicator**; otherwise, no indirect addressing. |
| Address | 5 | Specifies one of 32 address **locations in memory** |

The currently available instruction OpCodes are as defined below.

|  |  |  |
| --- | --- | --- |
| **OpCode2** | **Instruction\*** | **Description** |
| 000001 | LDR r, x, address[,I] | Load Register From Memory |
| 000010 | STR r, x, address[,I] | Store Register To Memory |
| 000011 | LDA r, x, address[,I] | Load Register with Address |
| 100001 | LDX x, address[,I] | Load Index Register from Memory |
| 100010 | STX x, address[,I] | Store Index Register to Memory |
| 001000 | JZ r, x, address[,I] | Jump If Zero |
| 001001 | JNE r, x, address[,I] | Jump If Not Equal |
| 001010 | JCC cc, x, address[,I] | Jump If Condition Code |
| 001011 | JMA x, address[,I] | Unconditional Jump To Address |
| 001100 | JSR x, address[,I] | Jump and Save Return Address |
| 001101 | RFS Immed | Return From Subroutine |
| 001110 | SOB r, x, address[,I] | Subtract One and Branch |
| 001111 | JGE r,x, address[,I] | Jump Greater Than or Equal |
| 000100 | AMR r, x, address[,I] | Add Memory To Register |
| 000101 | SMR r, x, address[,I] | Subtract Memory From Register |
| 000110 | AIR r, immed | Add Immediate to Register |
| 000111 | SIR r, immed | Subtract Immediate from Register |

\*[,I] = the Boolean indicator for using indirect addressing

OpCodes that are register to register use the following user entry pattern.



The currently available instruction OpCodes are as defined below.

|  |  |  |
| --- | --- | --- |
| **OpCode8** | **Instruction** | **Description** |
| 010000 | MLT rx,ry | Multiply Register by Register |
| 010001 | DVD rx,ry | Divide Register by Register |
| 010010 | TRR rx, ry | Test the Equality of Register and Register |
| 010011 | AND rx, ry | Logical And of Register and Register |
| 010100 | ORR rx, ry | Logical Or of Register and Register |
| 010101 | NOT rx | Logical Not of Register To Register |

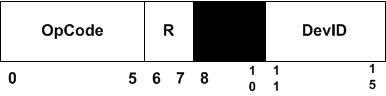
OpCodes that are logical instruction use the following user entry format.



Where Register (R), Arithmetic/Logic (A/L), Left/Right (L/R), and Count are the options to perform the shift or rotate operation.

|  |  |  |
| --- | --- | --- |
| **OpCode** | **Instruction** | **Description** |
| 011001 | SRC r, count, L/R, A/L | Shift Register by Count |
| 011010 | RRC r, count, L/R, A/L | Rotate Register by Count |

I/O Operations:



|  |  |  |
| --- | --- | --- |
| **OpCode** | **Instruction** | **Description** |
| 110001 | IN r, devid | Input Character To Register from Device |
| 110010 | OUT r, devid | Output Character to Device from Register |