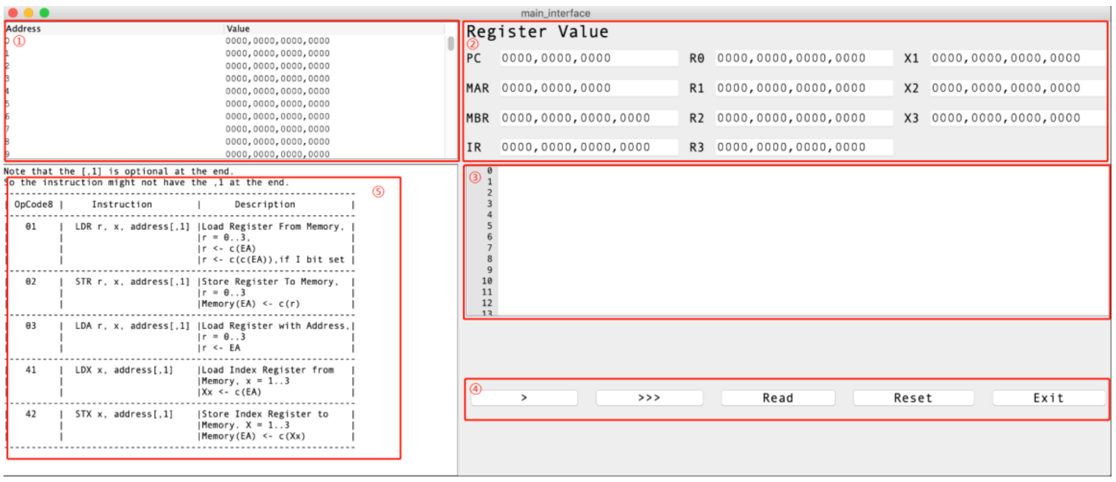
This design is for CSCI6461 project1.

By: Kaibo Zhang, Xinlong Han, Ashvi Soni

Introduction

This a Java application that allows the user to run machine language and display on user interface. It can realize five functions, which are” LDR”, “STR”, “LDA”, “LDX” and “STX”, by receiving code from user input panel. The explanation of each instruction code and how to use this program will explain in following part.

Simulator Layout



Address	Value
0	0000,0000,0000,0000
1	0000,0000,0000,0000
2	0000,0000,0000,0000
3	0000,0000,0000,0000
4	0000,0000,0000,0000
5	0000,0000,0000,0000
6	0000,0000,0000,0000
7	0000,0000,0000,0000
8	0000,0000,0000,0000
9	0000,0000,0000,0000

Register Value

PC	0000,0000,0000	R0	0000,0000,0000,0000	X1	0000,0000,0000,0000
MAR	0000,0000,0000	R1	0000,0000,0000,0000	X2	0000,0000,0000,0000
MBR	0000,0000,0000,0000	R2	0000,0000,0000,0000	X3	0000,0000,0000,0000
IR	0000,0000,0000,0000	R3	0000,0000,0000,0000		

```
Note that the [,1] is optional at the end.
So the instruction might not have the ,1 at the end.
------------------------------------------------------------
| OpCode8 |     Instruction         |       Description        |
------------------------------------------------------------
|   01    |  LDR r, x, address[,1]  |Load Register From Memory, |
|         |                         |r = 0..3,                  |
|         |                         |r <- c(EA)                 |
|         |                         |r <- c(c(EA)),if I bit set |
------------------------------------------------------------
|   02    |  STR r, x, address[,1]  |Store Register To Memory,  |
|         |                         |r = 0..3                   |
|         |                         |Memory(EA) <- c(r)         |
------------------------------------------------------------
|   03    |  LDA r, x, address[,1]  |Load Register with Address,|
|         |                         |r = 0..3                   |
|         |                         |r <- EA                    |
------------------------------------------------------------
|   41    |  LDX x, address[,1]     |Load Index Register from   |
|         |                         |Memory, x = 1..3           |
|         |                         |Xx <- c(EA)                |
------------------------------------------------------------
|   42    |  STX x, address[,1]     |Store Index Register to    |
|         |                         |Memory. X = 1..3           |
|         |                         |Memory(EA) <- c(Xx)        |
------------------------------------------------------------
```

Part 1: This part shows every memory address and corresponding value. Total memory addresses are 2048 and user can drag on the right side or use mouse wheel to shift.

Part 2: This part shows register status after user input instruction. It includes PC, MAR, MBR, IR, CC, MFR, four general registers and three index registers.

Part 3: This part is user input panel. User can input instruction code here and click button to observe different status of each register and memory.

Part 4: This part has five buttons listed from left to right and their function is 1. step run instruction code in input panel row by row, 2. run all instruction code in input panel, 3. read code from file, 4. reset program and 5. exit program.

Part 5: This part shows a quick instruction reference card.

Instruction Code

The input panel can **ONLY** receive those five instructions code listed in the following. If user input other than those five instructions code or not following syntax, this program will throw exception or have no response.

1. LDR r, x, address [, I] (ldr r, x, address [, I])
   Read content from memory address and load it to register r. x means which index register you want to use. You can write a capitalized "I" at the end of this code and it means index indirect addressing. E.g. Without "I", r1🡐c(EA). if code with "I", r 🡐c(c(EA)).
   Example: LDR 1,1,10
   It can read as: load register 1 with contents of memory location 10 and add with index register x.
2. STR r, x, address [, I] (str r, x, address [, I])
   Store register r content to the memory address. x means which index register you want use. "I" has same function as LDR function explained.
   Example: STR 1,1,10
   It can read as store content in register 1 to memory address 10.
3. LDA r, x, address [, I] (lda r, x, address [, I])
   Store address number that user input to register r and add with index register x. User can use "I" to realize index indirect addressing. If write "I" at the end of this instruction, it will have the same function as LDR does.
   Example: LDA 3,1,30
   It can read as store address number 30 to register 3.
4. LDX x, address [, I] (ldx r, x, address [, I])
   Store content in memory address to index register x. User can use "I" to realize index indirect addressing.
   Example: LDX 3,10
   It can read as store content in memory address 10 to index register 3.
5. STX x, address [, I] (stx r, x, address [, I])
   Store content in index register x to the memory address. User can use "I" to realize index indirect addressing.
   Example: STX 2,15
   It can read as store content in index register 2 to memory address 15.

Input Formats

For instruction for project part one, the input parameter will have r for the register, x for the index register and address for memory address. For parameter r, it can only take from 0 to 3. For parameter x, it can only take from 1 to 3. For parameter address, it can only take from 0 to 31. If user input other than those ranges, the program will occur an error window and marked it as red.

By input panel: Directly input instruction code listed above and following the syntax tips listed below.

By input file: User can write code in a .TXT document and use read button to import it to program.

Syntax tips:

1. Leave one space between the function name and parameter.
2. Using a comma and no space to separate each input parameter, such as register 1,register 2,address 3.
3. To run multiple instruction code, directly change the line for each complete instruction code.

   For example:

   ```
   STR 1,0,10
   LDR 2,3,10
   LDA 3,2,30
   ```
4. Input by input panel and the file should start from first line and if user leaves any line empty. It will occur an error.