A Simple Manual for the Simulator

By Group 2

Meng Lin

Siyuan Feng

Rong Wang

Tingting XU

Contents

Layout explanation 1

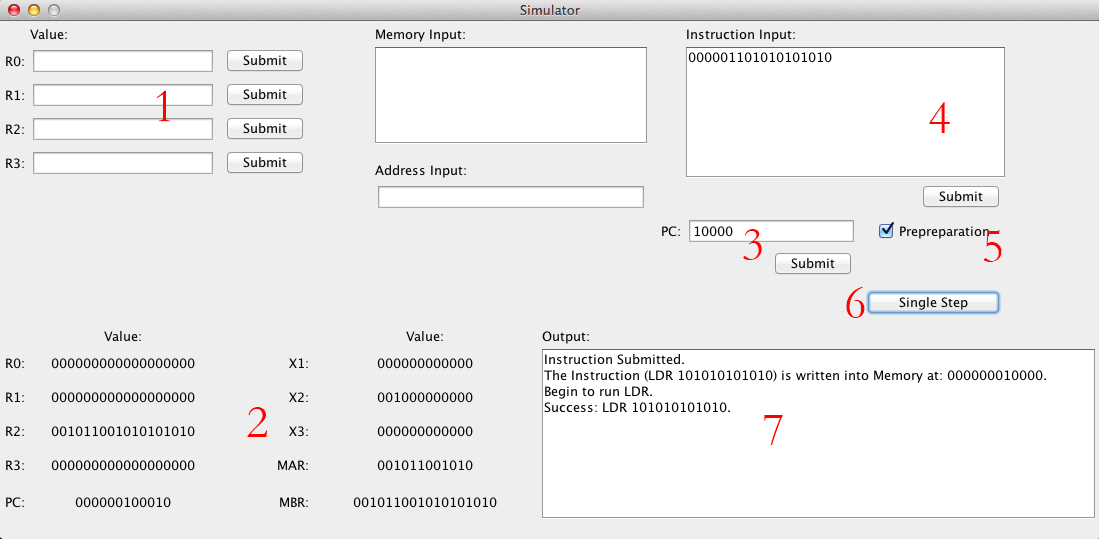
Appendix I Code format 3

Appendix Two Preparation for every instruction 4

Appendix Three An Example 6

Layout explanation

This is a simple document describing How To Use the simulator.



1: This is the place where you can set the value of R0-4. Remember to Submit after you operation.

*Since R0-R4 are 18-bit register, your input will be expanded to 18-bit. For example, if you input for R0 is 11101, the R0 will be 000000011101. You are not allowed to input more than 18 words or any character other than ‘1’ and ‘0’.*

2. This is the panel show the states of registers in the Simulator. Up to now, we only show these 10 states, which, we think, are most important.

3. This is the place where you can change the value of PC. When you input an instruction into the memory, the PC in CPU will not point to the address of the instruction. Therefore, to execute your instruction, you need to set the PC manually or use instructions instead, which, however, are not realized yet, that means you can only change the PC yourself. You can check the address of your instruction in the output window.

*The value of PC is examined strictly as the R’s.*

4. This is the place where you can input an instruction. The instruction will be place into the memory properly by our program, and the address will be given in the Output window.

*The instruction input will be examined. Undefined instructions will not be accepted and an error will be given in the output window.*

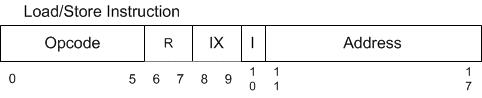
5. This is an important option. When you execute an instruction, you may need some data in the memory. However, the memory is empty, which means 0s, when you start up the program. Thus, your output might be 0s!!! To show you our simulator works correctly, we prepare some data to write into the memory before the execution of every instruction. You can check the content of the prepreparation in our appendix.

6. This is the button to run your instruction. The address of instruction is stored in PC. So make sure the PC is correct every time you execute the instruction. If the instruction pointed by PC is undefined, it will not be executed, and an error will be given in the output window.

7. The result of every operation will be shown here. Check the information after your operation.

Appendix I Code format

Every instruction must obey the code format.



|  |  |  |  |
| --- | --- | --- | --- |
| **OpCode8** | **Instruction** | **Code** | **Description** |
| 01 | LDR r, x, address[,I] | 000001 rr xx I addr | Load Register From Memory, r = 0..3  r  c(EA)  r <- c(c(EA)), if I bit set |
| 02 | STR r, x, address[,I] | 000010 rr xx I addr | Store Register To Memory, r = 0..3  EA  c(r)  c(c(EA)) <- c(r), if I-bit set |
| 03 | LDA r, x, address[,I] | 000011 rr xx I addr | Load Register with Address, r = 0..3  r  EA  r  c(EA), if I bit set |
| 41 | LDX x, address[,I] | 100001 rr xx I addr | Load Index Register from Memory, x = 1..3  Xx <- c(EA) |
| 42 | STX x, address[,I] | 100010 rr xx I addr | Store Index Register to Memory. X = 1..3  EA <- c(X0)  C(EA) <- c(Xx), if I-bit set |

Appendix Two Preparation for every instruction

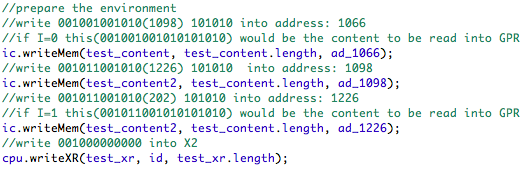
1. LDR

I=1

Instruction Input: 000001101010101010

I=0

Instruction Input: 000001101000101010



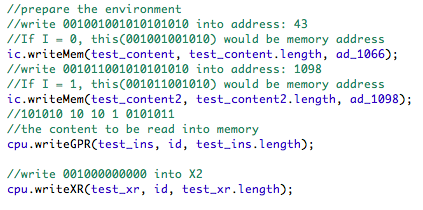
2. STR

I=1

Instruction Input: 000010101010101010

I=0

Instruction Input: 000010101000101010

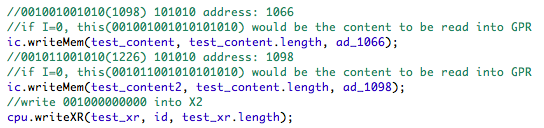


LDA I=0

Instruction Input: 000011101000101010

I=1

Instruction Input: 000011101010101010

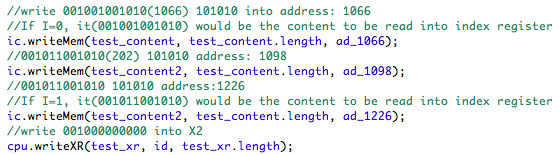


LDX I=0

Instruction Input: 100001101000101010

I=1

Instruction Input: 100001101010101010

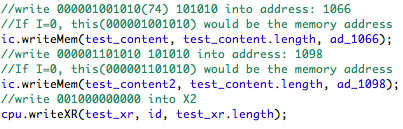


STX I=0

Instruction Input: 100010101000101010

I=1

Instruction Input: 100010101010101010

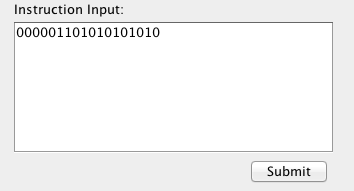


Appendix Three An Example

This is an example to run an instruction.

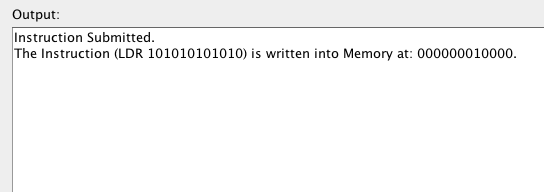
Input: LDR I=1

000001101010101010



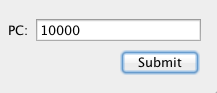
Press “Sumbit”

Look at the output window

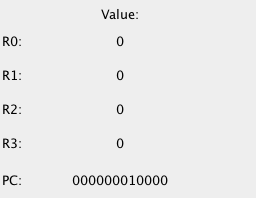


We can see that, the Instruction is stored into the Memory at 10000 in binary, which is 16 in decimal.

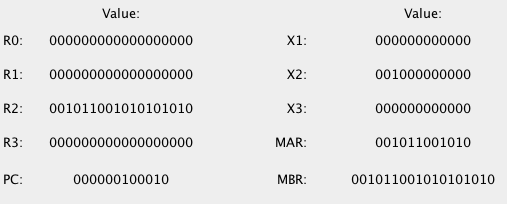
Thus to execute the Instruction, we need to set the PC to 16.



Press the button, and we can see that the PC is 000000010000 now.

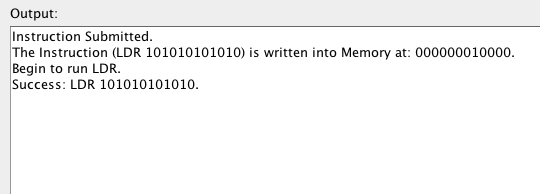


Now press the Single Step button, we can see the result. Remember to check the Prepreparation option, or your result will be all 0s, which is caused by the almost empty (except the instruction input) memory.



The R2 is 001011001010101010, and you can check the Appendix Two.

The PC is 100010, which is 32 in decimal. It equals to (16+18), which should be the address of the next Instruction.



From the output window, we can see that, the LDR instruction works successfully.