UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingeniería Informática

Universitat Politècnica de València

[Título]

Proyecto Final de Carrera

[Titulación]

Autor: [Nombre del autor/es]

Director: [Nombre del director/es]

[Fecha]

Resumen

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed nisi turpis, iaculis a pulvinar quis, luctus et lorem. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nullam vitae purus eros, id auctor dolor. Sed et nisl quis nibh fermentum cursus ut at elit. Etiam condimentum porta leo quis tempor. Quisque commodo lobortis aliquet. Etiam tincidunt, libero ut vehicula euismod, justo augue lobortis sem, et facilisis velit lacus tristique dolor.

Palabras clave: integer, blandit, pharetra, urna, id.

Tabla de contenidos

1

2 Introducción

The study of the Instruction Set Architecture is a very important subject in studies of computer design and programing.

First year students cannot address on real processor because of its complexity and therefore more simple computers should be used. This is the case of the Easy8 computer although it is an educational and very simple computer, it includes the same components like real computers: CPU with registers, main memory, input/output system and a short but representative set of instructions.

3 Objective

Como el Easy8 no existe en el mundo real, para que los alumnos puedan hacer prácticas y experimentar, el The goal of this project is to make a computer simulator of the Easy8 computer as defined in FCO subject Grade Engineering Technology and Telecommunication Services ETSIT.

It must be easy to use by the student and if possible platform to work with.

Comment [1]: No entiendo que quieres decir c
esta frase

Also, we built it with java, so it could be used in window/linux/unix, and must include a graphical interface.

4 Requirements

4.1 List of requirements

i. **Execution of assembler on basic assembler language with minimal set of instruction list.**

ii. **Work with ASM files, compile them and reload them from the memory.**

iii. **Ability to stop/continue execution of the system.**
Ability to add Breakpoints.

iv. **Ability to have input/output to display/external system.**
User interface to change memory.

v. **Load/save memory for working again on the same system.**

vi. **Show to the user the impact of the memory.**

vii. **Working with hexdecimal base**

Comment [2]: Aunque ya lo has dicho antes, también son requirimientos y debes ponerlos aquí que tenga un interfaz gráfico y que sea multiplataforma.

Comment [3]: En lenguaje ensamblador, el compilador se llama ensamblador, igual que el lenguaje.

5 Application design

The application was ***done with java, using some design patterns. The project was done in object oriented methodology. Following framework was in used: Gui – Swing. JPanel, data model of JList were extended . XML - JAXB.***
The implementation of the handling of the instructions in the logic model – was done with TDD – Test driven development.

Utils class was created as a Helper class which will include the methods that not part of the Panel handling itself. It will shortly th code to be more viewable.

5.1 Used design patterns

5.1.1 Singleton pattern

Will be used for handling just one occurrence of every member of the CPU. LogicalCpu is the class of the Singleton. Sigleton implementation done with double checking to prevent entering double time on the same time.

5.1.2 Factory Pattern

to create the implementation of ActivityPiece in run time, for every step while execution the assembled code1. Factory class name is ActivityPieceFactory.

5.2 Pseudo_codes

5.2.1 Run/Step execution

#nextAddress = getNextAddress()

#step()

#updatePc()

#Show impact memory fields

#Show next step on the Instruction CPU window

5.2.2 Instruction implementation

The implementation of one instruction is built from the following parts:

#updateModel()

[1] The decision which implementation to create is done according to the input which contains the command, In the run time assumption is taken that current instruction is exist [after assemble has been pass already before]

#create new event for gui
#Create jump event

5.3 Seven digid display

 # FOR INT numOfDigits <=2

 # FOR int pieceNumber <=7

 # Draw the piece ON/OFF [using the graphics.fillPolygon method]

5.3.1.1 Explanations

1. The Gui would later go on list of received events and handling them to show impact memory, next command, new registers values.
2. Create Jump event – will update the CP to another value instead of using the address of the the next command.

5.4 Customization of classes

1. JPanel – customized for display the Seven Digit

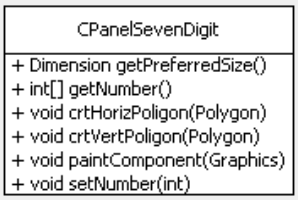
5.5 Implementations

5.5.1 Seven digit

5.5.1.1 High level design*:*

The implementation of the seven digit display was done by extends the JPanel class of Swing. In the constructor of the panel we created list of polygons, every piece of the display is one polygon. Every number has list of On/OFF for the polygons. Select one value – get the required pieces that should be on and display them.

5.5.1.2 UML class diagram of class



5.5.1.3 Methods

getPreferredSize – used by the parent panel to create the customized panel with the required Size.

paintComponent - does the creation of the display.

5.5.1.4 Interface*:*

Creation – as in the following line, in the generated code:[2]

```
sdPanel = new meirdev.simulator.gui.customized.panels.CPanelSevenDigit();
```

Update value – using the method setNumber to update the value.

[2] NetBeans add the customized creation code. The component has ***Custom creation code*** in the Code properties tab.

6 Known Limitations-improvements that could be done

6.1 Error handling of asm file

The assemble process with good error viewing is viewable only in the console. Please see in the manual the steps which should be done for compile asm file.

6.2 Seven digit update

The Seven-digit display regular definition exists only for regular decimal number. No display for the case that output value has one of the additional characters. Previous value will remain in case that RA contains value which has a characte which is part of hexdecimal base, instead of a number.

7 Manual

7.1 Execution of Gui

7.1.1 Windows	1. Extract the files to working directory. 2. Execute the Gui.bat
7.1.2 Linux	1. Extract the file to local folder. 2. Open the shell to this folder 3. Execute the following java command: 4. java -cp SimEasy8-jar-with-dependencies.jar meirdev.simulator.gui.frmae.GuiSimulator

7.2 Execution of compiler

7.2.1 Windows	1. Extract the files to working directory. 2. Execute the Show.bat
7.2.2 Linux	1. Extract the file to local folder. 2. Open the shell to this folder 3. Execute the following java command: 4. java -cp SimEasy8-jar-with-dependencies.jar meirdev.simulator.simeasy8.AssemblerReader test.asm

The last parameter for the java is the assembler file to be assembled.

8 Gui

8.1 Parts view

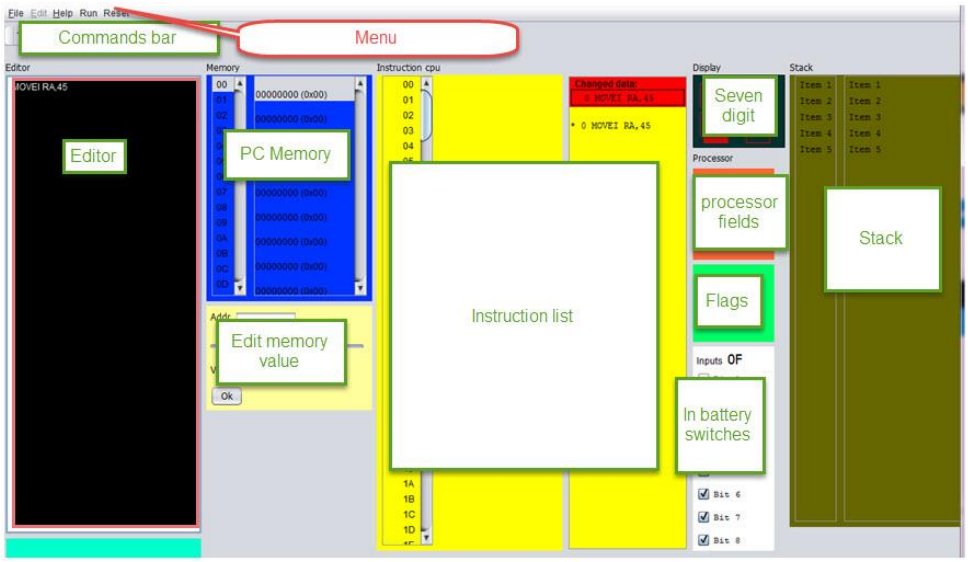


Figure 1- Simulator parts

8.2 File types used in the simulator

There are two file types which are used in the simulator:

8.3 Optional files

8.3.1 ASM

8.3.1.1 This type of file contains the assembler code.

1. Could be loaded from param when execution is from console or by Menu item when execution is done by the Gui.

8.3.2 MEM

8.3.2.1 This type of file contains the memory code.

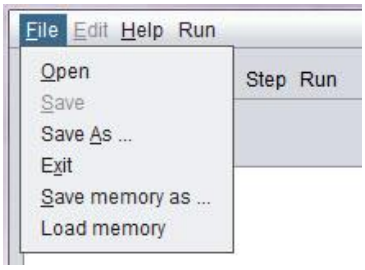
The graphic interface must include the following elements:

8.4 Editing/saving assembler

- Window (or frame) editing to introduce programs in assembly language. Buttons to save disk drive or loaded from an assembly program.

8.5 Open assembler file

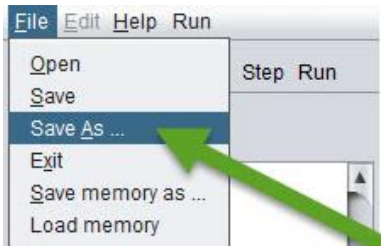
The file will be opened as in the following image, using the ***File/Open*** menu item /Sub menu item. Only ***asm*** files will be viewed from this dialog window.



2Open file

8.6 Save assembler file

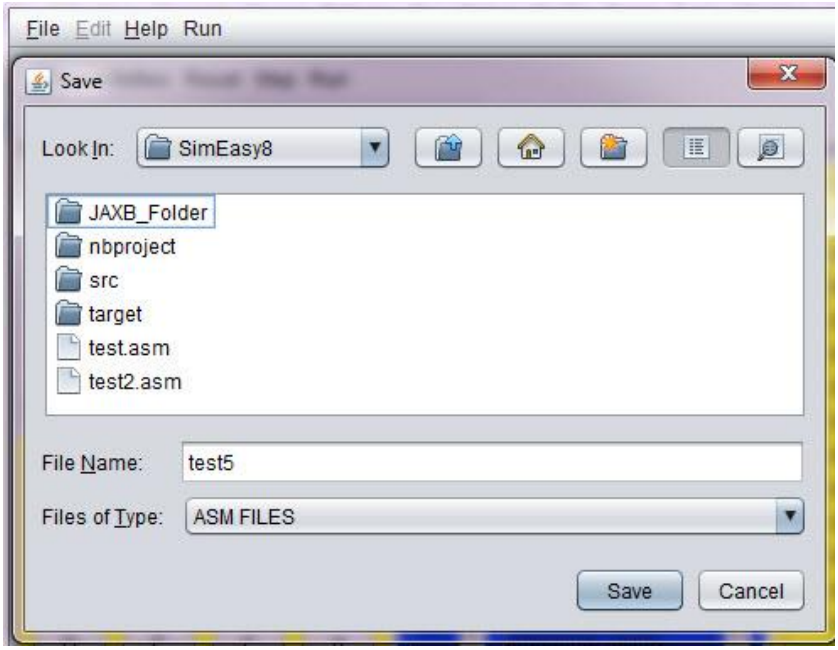
2. Save file – Use the ***Save*** / ***Save as*** sub menu. Extention of the file name could be ignored in this activity.



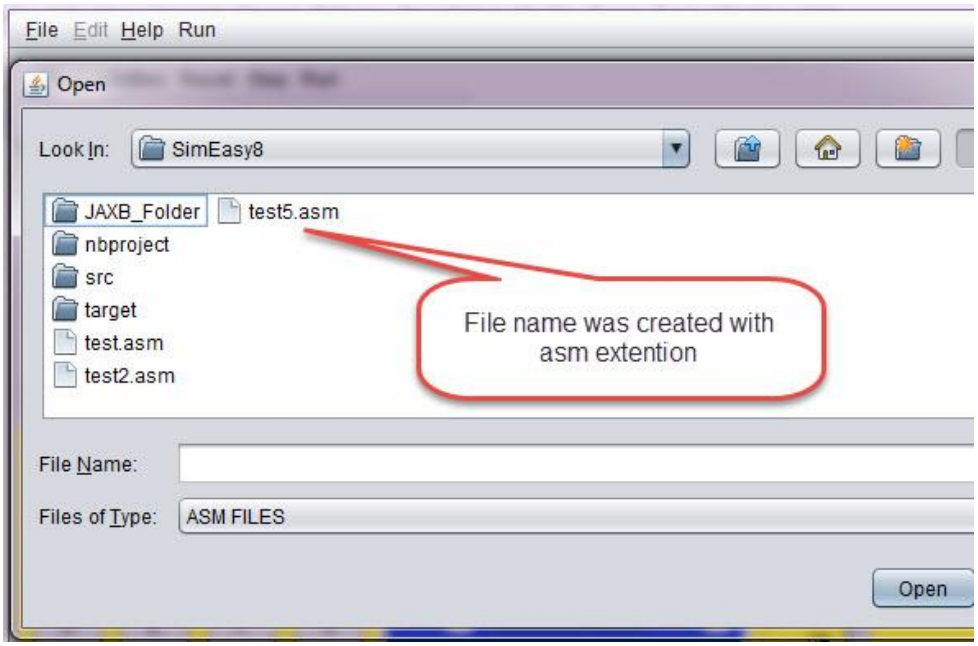
3Save as - menu item q Sub menu item

8.6.1 Save file flow:

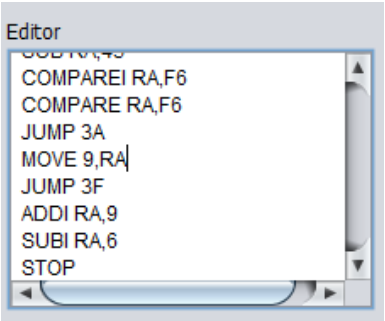
Saving the file name with no extention, will save the file with the corect extention, as in the folowing pictures. The result is – that open file will view the file and file will be opened with no issues:



4save asm file - test5 - with no extension



5The file was saved corectly - as test5.asm[Viewdlated by the open file]

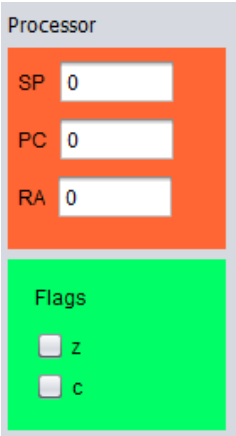


6Editor window

The editor window – contains window with editor for the ASM code.

Window (or frame) to display processor registers (PC, SP and RA) and outcome indicators (C, N, Z and V). These records will be updated either by the execution of the instructions or because the user modifies the content. You should view / edit both binary and hexadecimal.

In the current version – the edit of the values is done currently in hexadecimal.



7Processor registers and outcome indicators.

8.7 Memory view

8.7.1 Memory

8.7.2 Instruction cpu

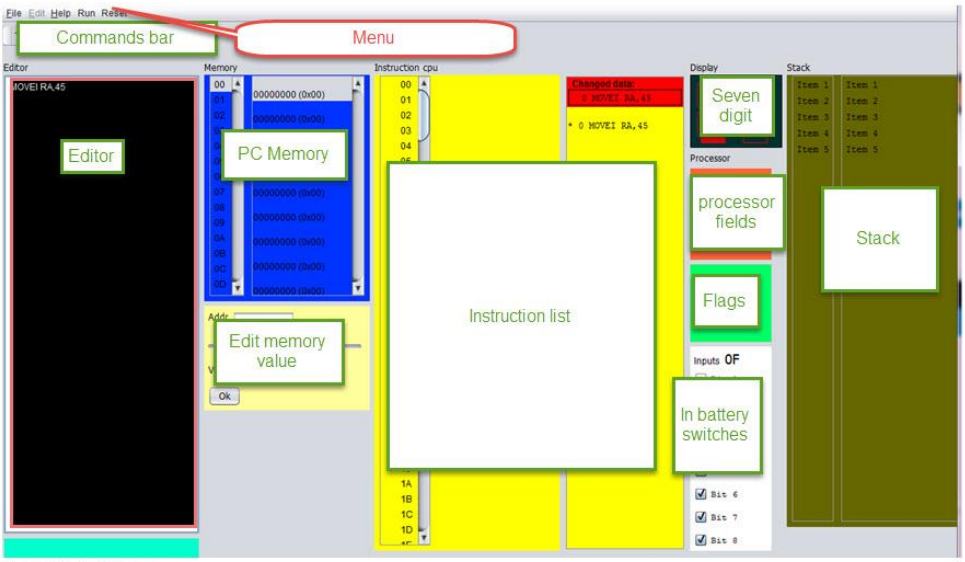
8.7.3 Stack

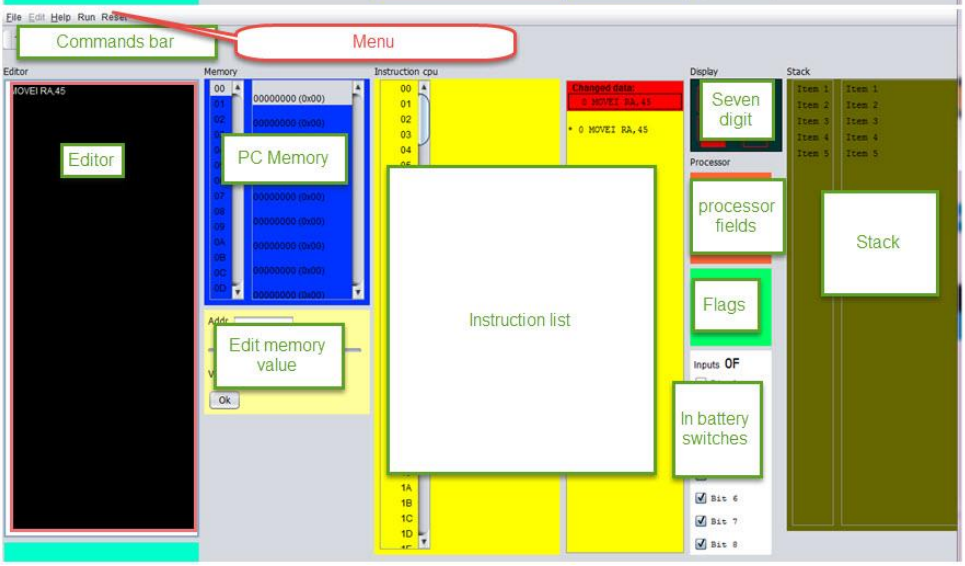
- Two windows (or frames) to display the computer's memory (256 bytes). This allows the student displayed at the same time the bottom of memory where your program into machine code, and in the other window another address range, which may have data or be the stack. Both memory addresses and their contents are displayed in hexadecimal. It must be possible to modify the contents of a memory address. The way it is now does not quite like, but not if it is possible to make a memory location by clicking on its value and changing it is changed.

8.8 Seven segment display

- A seven-segment display two digits to display the output of 0x01 OUT instruction (is as it is in the current simulator)

- A peripheral output to your liking. Something chart, which will send a byte or two and change color, shape or position. For example, a needle that moves right or left depending on the value that you send it ... – Only thing wasn't implemented





8.9 Input battery of 8 switches

- A battery of 8 switches, to generate the input to the instruction IN 0x00. The user must be able to change the position of each of the switches (on / off) with the mouse.

From the high bit at the top to the lower bit .

For example:

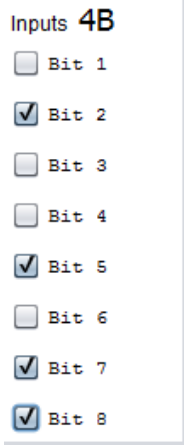


Figure 8 battery of 8 switches - 4B value

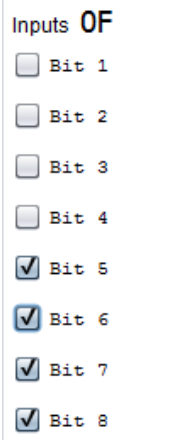


Figure 9 0F value

Figure 10 battery of 8 switches - 4B value

- A hexadecimal as having the current simulator keyboard, but instead of an OK button, you will have something like a push button (https://electrosome.com/wp-content/uploads/2012/12/Push-Button -Switch.jpg).

8.10 Toolbar

- A panel with buttons (or bar, although I prefer panel with large buttons so you can read the function of each button) with the following buttons:

Assemble, Step (Step by Step Run) Run (Uninterruptible Execution), Reset PC (Set to 0 the PC), Reset RA (Reset the RA), Reset SP (Reset the SP), Reset memory (Reset all memory) and RESET (Resets all of the above).

The RESET button in the tool bar will have confirmation, to prevent from reset by mistake.[i]

STOP button

8.11 Gui Menu

Reset menu – all the reset options.

9 Gui behaviour

Behavior of the different elements:

-buttons Reset PC, RA, SP and memory.

These buttons should reset the corresponding record (or all memory). The outcome indicators are reset in conjunction with RA or when reset everything.

Joining -button

Once introduced into an assembly program editor, the simulator must analyze it to translate into machine code. If an error is detected, for example, the instruction does not exist or is missing an operand, display an error message and indicate where the error occurred. It takes no great sophistication, as the language is simple and regular.

If no errors are loaded into memory the result of code the program into machine code.

10 Debugger

10.1 - Step Button

Execute the instruction pointed to by the PC and stop updating all the graphics and state of the computer elements.

10.2 - Breakpoint [Wasn't part of the requirement]

The user could add break point in the required address.

NOTE: Breakpoint could be added only after finished of assemble activity.

10.3 - Run Button

Execute the instruction pointed to by the PC, update the machine status and graphic elements, and continue with the next instruction. The execution will stop when it reaches the Stop button or press the STOP instruction.

10.4 - Stop Button

Stops program execution.

- Records and memory must change according to the execution of instructions (and if the user forces a value, of course)

- It would be very interesting that when a memory location is changed in the second window that shows the memory will be displayed, if it is not already, the memory area around the modified position.

- When a step execution is done, it should light or at least an arrow pointing instruction to be executed. If you can do both in the assembly code and source code would be fine.

Instructions to run:

In addition to the instructions in the documentation, including byte (which is not an instruction, is a thing of assembler) are two more who would want will implement instructions. And CLEAR are COMPAREP PORT PORT, but these'll talk later, when you start scheduling the execution of instructions.

The code to update indicators C, N, Z and V you'll pass, because although it is not complex is not trivial.

11 TABLE 1

11.1 EASY8 INSTRUCTION SET.

12 Instruction	13 Operation code	14 Description
15 MOVEIRA, VALUE	16	17
18 MOVERA, [ADDRESS]	19	20
21 MOVE [ADDRESS], RA	22	23
24 ADDIRA,	25	26

VALUE		
ADD RA, [ADDRESS] 27	28	29
SUBI RA,VALUE 30	31	32
SUB RA, [ADDRESS] 33	34	35
INC RA 36	37	38
39 DEC RA	40	41
COMPAREI RA, VALUE	42	43
44 COMPARE RA, [ADDRESS]	45	46
JUMP ADDRESS 47	48	49
JLESS ADDRESS 50	51	52

JGREATER ADDRESS 53	54	55
56 JEQUAL ADDRESS	57	58
PUSH RA 59	60	61
62 POP RA	63	64
CALL ADDRES	65	66
67 RET	68	69
70	71	72

73

74 Appndix

74.1 Assembler file with IO

74.1.1 Basic flow – output to the Seven digit

Basic flow – execution of a program that will display hex value of 77.

The following steps should be done to work with ASM file.

The display of the seven digit for the default value will be as in the following image:

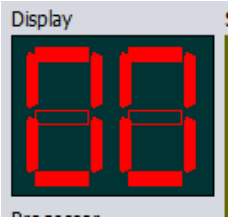


Figure 11Seven digit contain default value

1. Load ASM file with Open file menu/Sub menu
2. Edit the file in the Editor section
3. Clear the contents of the editor
4. Add the following lines:
    a. OUT 77
    b. STOP

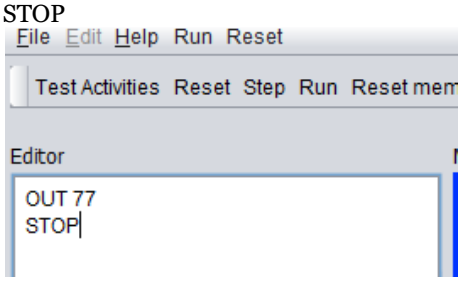


Figure 12Editie -ASM lines to display 77 in the seven digit

5. Assemble the file.
6. Run the assembled code.

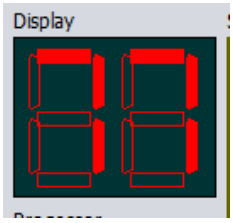


Figure 13Seven digit in the ending of the execution - hex 77

74.1.2 Basic flow – Input from the seven switches battery

The following steps should be done to work with ASM file.

The display of the seven digit for the default value will be as in the following image:

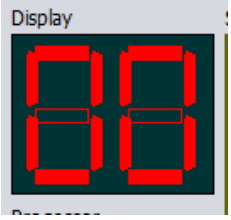


Figure 14Seven digit contain default value - 00

7. Load ASM file with Open file menu/Sub menu
8. Edit the file in the Editor section
9. Clear the contents of the editor
10. Add the following lines:
    a. IN
    b. OUT RA
    c. STOP

11. Assemble the file.
12. Run the assembled code.

Figure 15Seven digit in the ending of the execution - hex 77

75 CONCLUSIONS AND FUTURE WORK

This work is basic simulator with very simple input/output.

Additional improvements - already exist in another section.

New features could be storing the asm file in the internet, grouped by classes.

Programing improvements – could be do the next step and split the code of the logical model and the GUI itself, so another model could be used.

Index

List of pictures

/*

1. Introduction 2. Objective 3. Requirements 4. Design of the application 5. Implementation 6. User Manual 7. Conclusions 8. Bibliography I mention what I would each of these sections. 1. Introduction: where you say that the study of the Instruction Set Architecture is very important in studies .... but first course can not address a real processor because of its complexity and therefore ... use simplifications, educational systems that do not really exist ... ... and to do simulators. 2. Objective: To say that the goal is to make a computer simulator Easy8 as defined in FCO subject Grade Engineering Technology and Telecommunication Services ETSIT. It is also easy to use by the student and if possible platform. And for that you propose the development of a Java application with graphical interface. 3. Requirement: You put a list of what you want to do the application, in all its facets: edit, assemble, execute step by step graphical interface, input and output ... 4. Application design: more or less what has commanded me, but divided into sections. An introduction with an image of the different elements of the interface, and then a section for each of them: editor, view memory, I / O, etc .. 5. Implementation: explains the language and tools used for implementation. If you have made modular programming you put the different files that

have organized the source code. And usually put a list of the functions that you created. Not put the code, only the prototype of the functions and a short description of the function. This list of functions you leave for last. 6. Manual: for that, a short manual with steps to use the simulator. 7. Conclusions 8. Bibliography

*/

[i] The reset menu – do not have this option. It'