Luis Fernando Escobar-Driver

CS3A MW 6 – 9:20pm

11/23/2015

Abstract

This packet will provide a user of the simple computer emulator with a comprehensive & through instruction manual. This instruction manual includes opcodes, what the emulator does, how to clearly use the emulator,, what type of computers the emulator will execute on, how to start the execution of the emulator, printed screens of the main menu & sub-menus, help screens, relevant source code & printed demonstrations of four execution cycles. The four execution cycles are; division, shifting digits, absolute value & a loader program.

Computer Emulator

Term Project

**TABLE OF CONTENTS**

Op Codes 2

User Manual 3 - 13

* What the emulator does 3
* What type of computer to execute on 3
* How to use the emulator 4 - 11
* Printed copies of menus & sub-menus 4 - 11
* How to start execution 11
* Printed copy of help screen 11 - 13

Relevant Source Code 14 - 55

Division Program 56 - 61

Shifting Digits Program 62 - 67

Absolute Value Program 68 - 73

Loader Program 74 - 79

Emulator Progress Worksheets 80

**OP CODES**

0\_ \_ INP : Copy the input card into cell number \_ \_ , and advance the input device to the next card. If the input card is blank, then advance the input device, set the contents of the program counter to 00, and halt the SC processor.

1 \_ \_ OUT : Copy the contents of cell number \_ \_ onto the output card and advance the output device one card.

2 \_ \_ ADD : Add the contents of cell number \_ \_ to the value of the accumulator.

3 \_ \_ SUB : Subtract the contents of cell number \_ \_ from the value of the accumulator.

4 \_ \_ LDA : Load accumulator by clearing the accumulator and copying the contents of cell number \_ \_ into the accumulator.

5 \_ \_ STA : Store accumulator. Copy the least significant three digits of the accumulator into cell number \_ \_ .

6 \_ \_ JMP : Place the present value of the program counter into cell 99. Then, change the value of the program counter to correspond to cell number \_ \_ .

7 \_ \_ TAC : Test accumulator. If the value of the accumulator is negative, change the value of the program counter to correspond to that of cell number \_ \_ .

8 X Y SHF : Shift the accumulator left x digits, and then shift the result right y digits. With all left shifts, zeros enter the accumulator on the right. Similarly, with right shifts, zeros will enter the accumulator on the left.

9 \_ \_ HALT : Replace the value of the program counter with cell number \_ \_ and then halt the SC processor.

**User Manual**

**What the emulator does**

This simple computer emulator mimics precisely the function and nature of the simple computer which can be found at <http://www.cs.xu.edu/csci170/07f/simpleComputer/sc.html> . The simple computer emulator uses Operation Codes to run a series of instructions and implement a program which has been designed by the user to function in a very specific manner. The users’ tools are the Op Codes. The Op Codes tell the simple computer emulator (SCE) what the program should do and how to do it precisely, accurately & efficiently. The user is able to input into the SCE any of the acceptable Op Codes into any acceptable memory cell; input card or main memory. A user can program just about any design with these Op Codes, meaning the SCE can run just about any program. In essence, the SCE purpose and goal is to take user input, Op Codes, and expel output, the users intended program design.

**What type of computer to execute on**

The SCE can execute on any PC or Mac which has Eclipse installed on it and which can code a C++ source file. One simply needs to copy and paste the source/header files into eclipse. If your computer can code a C++ source file on eclipse then you can run this SCE. Simple copy and paste the SCE folder into eclipse, build the code and then run it.

**How to use the emulator** **/ Printed copies of menus & sub-menus**

Using the SCE is very simple and straight forward. The SCE has been set up as a menu / sub-menu interface which allows a user to intuitively run through the SCE and input their desired Op Codes. The main menu is the command and control center of the SCE. It looks like this

Command List

---------------

#0 - List all memory components to the screen

#1 - Set Input Card

#2 - Set Program Counter

#3 - Set Main Memory

#4 - Clear input card

#5 - Clear Main Memory

#6 - Clear Output Cards

#7 - Clear All

#8 - Run

#9 - Step

#10- Show the CPU

#11- Save the contents of memory in their current form

#12- Load program from Option.txt

#-1 TO EXIT PROGRAM

Enter your selection:

Lets’ walk through this. This main menu will allow any user to control the SCE and manipulate data in any way. The user can select any option right from the beginning, 0 through -1.

Option 0 allows the code to print all the contents of the SCE into the console for the viewership of the user. Option 0 simple prints the contents of all memory; input cards, output cards & main memory cards, into the console in a very user friendly manner.

Option 1 allows the user to place values into the input card. When a user selects option 1 from the main menu, a prompt screen in the console will display this.

Enter your selection: 1

Set input card cell #0 to:

The user can select any valid input to place into the input card and may only place 15 inputs into the input cards. After the user has placed a valid input into the input card the SCE will print this into the console view.

Enter your selection: 1

Set input card cell #0 to: 10

Input has been registered!

Setting main memory cell #0 to: 010

Would you like to add another input(Y/N)?

At this point the user can continue to input values until the maximum of 15 has been reached or select ‘N’ to go back to the main menu.

Option 2, from the main menu, allows the user to select which cell the program counter should be pointing to. It looks like this.

Enter your selection: 2

Which cell should the program counter be set to:

Once the user makes a valid selection (00 – 99) the program will prompt this.

Enter your selection: 2

Which cell should the program counter be set to: 5

The Program Counter has been set to: 05

The program will then go back to the main menu.

Option 3 allows the user to place values into main memory. First the user will be prompted to select a main memory cell into which they would like to start inputting values into.

Enter your selection: 3

Which cell would you like to start in:

After the user makes a valid selection for the starting cell (01-99) the code will prompt this.

Enter your selection: 3

Which cell would you like to start in: 5

Enter value into main memory cell #5:

At this point, the user will be able to make a valid input selection (-999 – 999) into main memory cell #5. Once a user makes a valid selection the SCE will prompt this.

Enter your selection: 3

Which cell would you like to start in: 5

Enter value into main memory cell #5: 020

Input has been registered!

Setting main memory cell #5 to: 020

Continue onto main memory cell #6 (Y/N)?:

If the user wants to continue onto a sequential input pattern the user will simply continue to entire ‘Y/y’. It will look like this.

Enter your selection: 3

Which cell would you like to start in: 5

Enter value into main memory cell #5: 020

Input has been registered!

Setting main memory cell #5 to: 020

Continue onto main memory cell #6 (Y/N)?: y

Enter value into main memory cell #6:

The user will again make a valid selection. If the user wishes to make a new entry into the main memory but in a cell which is not sequentially attached to the previous cell the user will simply enter ‘N/n’ when prompted if they want to continue onto main memory cell #\_. It will look like this.

Enter your selection: 3

Which cell would you like to start in: 5

Enter value into main memory cell #5: 020

Input has been registered!

Setting main memory cell #5 to: 020

Continue onto main memory cell #6 (Y/N)?: n

Would you like to begin in a new main memory cell (Y/N)?:

The user will simply input which main memory cell they wish to input into. If the user wants to exit to the main menu, they will simply enter ‘N/n’ into Would you like to being in a new main memory cell.

That will look like this.

Enter your selection: 3

Which cell would you like to start in: 5

Enter value into main memory cell #5: 020

Input has been registered!

Setting main memory cell #5 to: 020

Continue onto main memory cell #6 (Y/N)?: n

Would you like to begin in a new main memory cell (Y/N)?: n

This will take the user right back to the main menu.

Option 4 allows the user to clear the input card. If the input card is empty the SCE efficiently informs the user that the input card is already empty. If the input card is not empty the SCE will simply clear over the cards which have input in them. That will look like this.

Enter your selection: 4

Input card is already empty

Or

Enter your selection: 4

Input card has been cleared!

Option 5 allows the user to clear main memory. This option simply allows the code to clear out every single main memory cell except cell #00. Cell #00 is protected in every single part of the SCE.

Enter your selection: 5

Main Memory has been cleared!

Option 6 allows the user to clear the output cards. Like option 4, if the output cards are empty the SCE efficiently informs the user that the output cards are already empty. If the output cards are not empty the SCE will simply clear over the cards which have input in them. That will look like this.

Enter your selection: 6

Output cards are empty already!

Option 7 allows the user to clear out all memory slots; input cards, output cards and main memory. That will look like this.

Enter your selection: 7

The entire program has been cleared!

Option 8 allows the SCE to run through the entire program at once. The SCE automatically aggregates all the inputs in all memory cards and runs the SCE to whatever end the user had in mind. Once the run option has been selected and the SCE executed, the code will save a final copy of all the values into an out file names ‘Save.txt’; that includes the input cards, output cards, main memory cells and the CPU. This will look something like this.

Enter your selection: 8

Program Halted: Input Card is empty

In this case, the loader program was loaded into the SCE, therefore the SCE ended with a program halt. If the program halted in any other way, the end message would reflect whichever ending the SCE had. The final contents are then saved into the specified out file in a user friendly easy to read manner.

Option 9 allows the user to step through the SCE operations one step at a time, with each step updating and displaying the contents of the CPU in the console view. All other memory cells are also updated but are not automatically displayed. This will look like this.

Enter your selection: 9

CPU

------

IR: 001

Program Counter: 01

Accumulator:

In this case, the loader program was loaded into the SCE. After a single step this is what the CPU would look like. After every step the user will see this in the console view. If the step option terminates the program, the final contents of the CPU and all memory will be saved into the previously mentioned out file ‘Save.txt’.

Option 10 simple allows the user to see the CPU at any time during the program. That will look like this.

Enter your selection: 10

CPU

----

Accumulator:

PC: 01

IR: 001

In this case, the CPU reflects a loader program with one step.

Option 11 allows the user to save all the contents of all memory and the CPU into an out file (Option.txt). The sole purpose of this is to re load the program from that same out file. This save option is not executed automatically and must be done by the user. The contents of the SCE are not saved in a user friendly manner, rather they are saved in a computer friendly manner. That means it allows the SCE to then read that out file and place the correct values back into their correct slots and execute properly. This will look like this.

Enter your selection: 11

Saved to file Option.txt

Option 12 allows the SCE to read from the previously saved out file ‘Option.txt’ and place all appropriate values back into the SCE. This will look like this.

Enter your selection: 12

Reading from file Option.txt!

After this has executed the SCE can continue to perform any action the user wishes.

**How to start execution**

The SCE is very easy and intuitive. To start the execution of the SCE simply copy the folder which contains the SCE into your eclipse. Build the code and then run it. Follow the instructions defined above.

**Printed copy of help screen**

//Main menu options

To list every single component of main memory, input cards &

output cards press option 0.

To put input into the input card press option 1.

To set the program counter press 2.

To set a main memory cell press 3. For a sequential series of

inputs into main memory continue to press 'y/Y' while still in option

3. If you want a non-sequential entry press 'n/N' after your

previous selection. If you want to exit option 3 press 'n/N' to the

sequential option and to the non-sequential option.

To clear the input card entirely, press 4.

To clear main memory press 5.

To clear the output cards press 6.

To clear the entire simple computer emulator press 7.

To run the entire program all at once press 8. This will save the

final vales of the CPU and all of memory into text file 'Save.txt'.

To step through the simple computer emulator one step at a time

press 9. This will provide, automatically, a console view update of

the CPU.

To show the CPU at any time during the run time of the simple

computer emulator press 10.

To save the contents of the simple computer emulator in their

current form press 11. This saves the SCE into text file 'Option.txt'.

To load a program from a text file press 12. This will load fom text

file 'Option.txt'.

To exit the main menu and the entire program press -1.

//This can be copied and pasted into the code right at the

//beginning for a quick execution cycles.

mainProgram[2] = "";

mainProgram[3] = "";

mainProgram[4] = "";

mainProgram[5] = "";

mainProgram[6] = "";

mainProgram[7] = "";

mainProgram[8] = "";

mainProgram[9] = "";

mainProgram[10] = "";

mainProgram[11] = "";

mainProgram[12] = "";

mainProgram[13] = "";

mainProgram[14] = "";

mainProgram[15] = "";

mainProgram[20] = "";

mainProgram[21] = "";

mainProgram[22] = "";

mainProgram[23] = "";

mainProgram[24] = "";

mainProgram[25] = "";

mainProgram[26] = "";

mainProgram[27] = "";

mainProgram[28] = "";

mainProgram[29] = "";

mainProgram[30] = "";

mainProgram[31] = "";

mainProgram[32] = "";

mainProgram[33] = "";

mainProgram[34] = "";

mainProgram[35] = "";

mainProgram[80] = "";

mainProgram[81] = "";

mainProgram[82] = "";

mainProgram[83] = "";

mainProgram[84] = "";

mainProgram[85] = "";

mainProgram[86] = "";

mainProgram[87] = "";

mainProgram[88] = "";

mainProgram[90] = "";

programCounter = ;

inputCard[0] = "";

inputCard[1] = "";

inputCard[2] = "";

inputCard[3] = "";

inputCard[4] = "";

inputCard[5] = "";

inputCard[6] = "";

inputCard[7] = "";

inputCard[8] = "";

inputCard[9] = "";

**Relevant Source Code**