C6461 Simulator - Quick Start Guide

Part 3 - Program Two (Card Reader)

To run program_two.txt, follow these steps:

1. Assemble the program using `data/program_two.txt`:

```
java -jar build/assembler/*.jar data/program_two.txt
```

2. Enter `002000` into the octal input field instead if it's not already set to that.

3. If you wish to update the contents of the card reader, you can do so by modifying the file `data/card.txt`.

4. Run the program with the following line:

```
java -jar build/simulator/*.jar
```

5. Select "IPL" to load the program into memory, and select the button under "PC" to set the PC to the correct location. Finally, select "Run" to execute the program.

6. In program two, the card reader contents will be printed to the OUT display field and you will receive a prompt to enter a word. Enter a word from the card reader output into the "Console Input" field and press enter. If the word you entered is in the card.txt file, the program will then print the word you typed, the sentence number it's found in, and the word number in that sentence. If the word you entered is not in the card.txt file, the program will print "Word not found".

7. Limitations to note:

• The card reader has a limitation of 767 characters. If the card reader exceeds this limit, overflow into the main program will occur. This could be improved some, but not significantly because the simulator memory is 2048 words in length, and the rest of the space is mostly taken by hard-coded messages, the main program, and subroutines.

• program_two.txt cannot currently find words if they are more than 9 words into the sentence. This is because the word number is stored in a single address. A future improvement would be to store the number, and re-use the "print_number" subroutine from program one.

Part 2 - Program One (Find Closest Number)

The UI displays the cache, an OUT display field, as well as a console input field to enter data through the IO stream. To test the functionality of Program 1, follow the steps below:

1. Assemble the program using `data/program.txt`:

```
java -jar build/assembler/*.jar data/program.txt
```

2. Run the simulator:

```
java -jar build/simulator/*.jar
```

3. The GUI should be preloaded with the correct data/load.txt file generated in step 1, and the octal input field should already be filled in with `000144`, which is the correct START location to run the program. Select "IPL" to load the program into memory, and select the button under "PC" to set the PC to the correct location. Finally, select "Run" to execute the program.

4. You will be prompted to enter the first 20 numbers into the console input field. You can do so by typing the number, from -32768 to 32767, into the console input field, and pressing enter.

5. You will be prompted to enter a guess number into the console input field. You can do so by typing the number, from -32768 to 32767, into the console input field, and pressing enter.

6. The closest number will be displayed in the OUT display field.

7. Optionally, you can modify the line `AIR 0, 20 ; R0 = 20` in the `data/program.txt` file to change the number of numbers to be entered.

Part 1 - Basic Simulator

Steps to operate the simulator:

1. Determine which file you want to run. The `data/assembly.txt` file from the C6461 documentation is automatically filled into the GUI to test. There is also a `data/load_store_test.txt` file that contains all LD/STR instructions.

2. Determine where the first instruction of the program is located and enter that into the PC register by typing the correct octal into the "OCTAL INPUT" field and selecting the button located below the PC register.

- For `data/load_store_test.txt`, the first instruction is located at octal `000015`.
- For `data/assembly.txt`, the first instruction is located at octal `000016`.

3. Initiate the program using the IPL button.

4. Step through the program using the Step button. Alternatively, you can run the program using the Run button.

5. You can also manually load and store values into the MAR and MBR using the "OCTAL INPUT" and their corresponding buttons.

Part 0 - Assembler

To run the assembler, execute the following command from the root directory:

```
java -jar build/assembler/*.jar data/assembly.txt data/listing.txt
data/load.txt
```

The output files will be placed as suggested above in the data/ folder.

Additionally, if you wish to run a full test of the encoder, run this command from the root directory:

```
java -jar build/encoder-test/*.jar
```