Gasser Ahmed
gasser18@vt.edu
ECE 5484, Project 2

Section 1 – Objectives:

Design and write an assembly language program for the MARIE processor that inputs, transforms, stores, and then outputs a sequence of characters from the set A-Z.

By following the steps below:

1. Understand, design, and write the Rotate-13 subroutine then test with constants i.e. ‘A’.
2. Design/write and test the input phase.
3. Design/write and test the output phase.
4. Initialize/define symbols, constants, and data that will be used in both input and output phases.
5. Design/write code that combines all the steps above (Rotate-13 subroutine, input, output, and initialization phases) including the processing phase that takes every input, processes Rotate-13 subroutine on it, then stores the result at the current pointer memory location so it can be used later in the output phase.
6. Debug and test the full code by following Test 1 (inputting “VIRGINIA”) and Test 2 (inputting “GRPU”) scenarios and make sure the output is correct i.e. “IVETVAVN” and “TECH” respectively after running both scenarios without reassembling or reloading. Also, make sure the memory locations contain the right values.

Hint: To make the output cleaner, I added a space character to be displayed after every run to separate between the output of multiple runs. Also, after each character output, I clear value stored for that character in the memory location to avoid outputting previous run values in the next new runs.

Section 2 – Design Description:

The program inputs a sequence of characters from the set A-Z (capital letters only), stores each character in memory after it is transformed by the trivial ROT13 cipher, and then, after character input completes, it outputs the transformed characters.

To achieve that, we start the program at location 100 hexadecimal and initialize character pointer (Ptr) to start of block (200 hexadecimal). We also initialize InVal and Hold variables with 0 and other constants like ChA, ChZ, ChPe, Val13, etc. so they can be used in later phases. After we finish initialization, we start the input phase by taking an input value from the user and store it into InVal to be used in the transformation process. Then, we subtract ChPe from that value, if the result is 0, that means the input was period '.' so we need to end that phase and reset the Ptr location to point to the start of the string to be ready for the output phase, otherwise, we will proceed with the transformation operation by calling "Process" which will start "ROT13" subroutine and start the transformation, store the result in memory (location stored in Ptr), increment Ptr location for the next input character and repeat the loop until user inputs '.' to end that phase and move to the output phase.

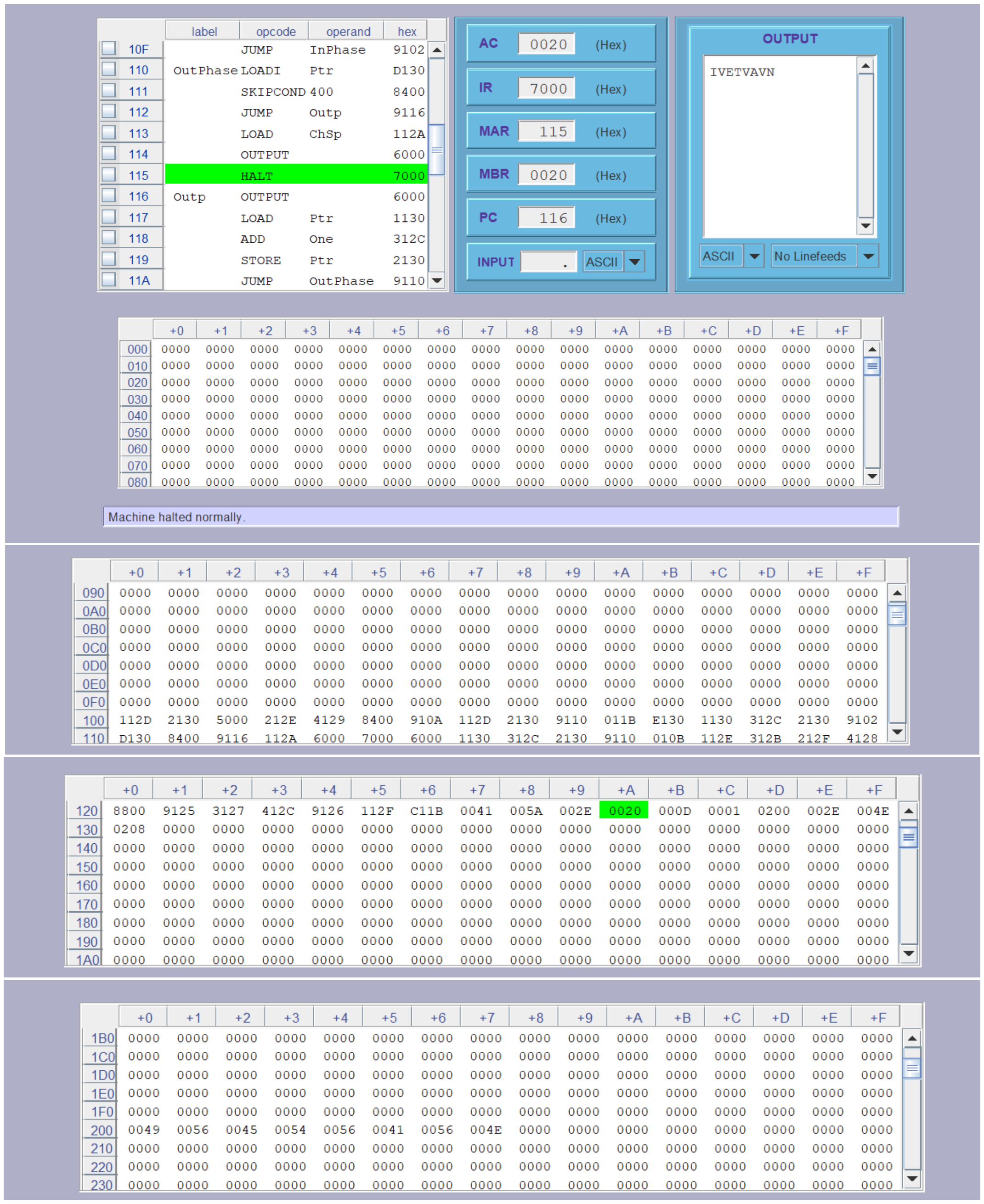
Once we move to the output phase, we load the character found at address Ptr and check if the character is null, that means we reached the end of the string so we exit/halt the program, otherwise, we output that character, clear it from the memory (to avoid printing old previous run values in the next new runs) then we increment Ptr to move to the next character and we repeat the output loop until the end of the string (null character found).

Once the output phase is done, we add a space character to the output to separate current output from next run's output then we quit/halt the program.

Section 3 – Testing and Results:

After running Test 1 with input “VIRGINIA”, I was able to get the right “IVETVAVN” output, then after making another run without reassembling or reloading and with input “GRPU”, I got “TECH” as the output for that run.

Test 1 Output & Memory Screenshots:



```
     label     opcode    operand    hex
10F            JUMP      InPhase    9102
110  OutPhase  LOADI     Ptr        D130
111            SKIPCOND  400        8400
112            JUMP      Outp       9116
113            LOAD      ChSp       112A
114            OUTPUT               6000
115            HALT                 7000
116  Outp      OUTPUT               6000
117            LOAD      Ptr        1130
118            ADD       One        312C
119            STORE     Ptr        2130
11A            JUMP      OutPhase   9110
```

```
AC   0020 (Hex)
IR   7000 (Hex)
MAR  115  (Hex)
MBR  0020 (Hex)
PC   116  (Hex)
INPUT  .   ASCII
```

```
OUTPUT
IVETVAVN
ASCII   No Linefeeds
```

```
     +0   +1   +2   +3   +4   +5   +6   +7   +8   +9   +A   +B   +C   +D   +E   +F
000  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
010  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
020  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
030  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
040  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
050  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
060  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
070  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
080  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

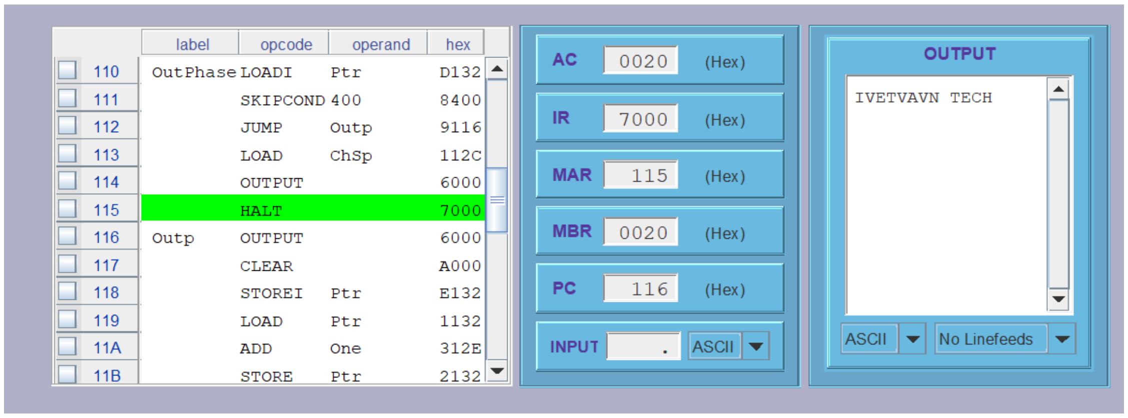
Machine halted normally.

```
     +0   +1   +2   +3   +4   +5   +6   +7   +8   +9   +A   +B   +C   +D   +E   +F
090  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0A0  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0B0  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0C0  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0D0  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0E0  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0F0  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
100  112D 2130 5000 212E 4129 8400 910A 112D 2130 9110 011B E130 1130 312C 2130 9102
110  D130 8400 9116 112A 6000 7000 6000 1130 312C 2130 9110 010B 112E 312B 212F 4128
```

```
     +0   +1   +2   +3   +4   +5   +6   +7   +8   +9   +A   +B   +C   +D   +E   +F
120  8800 9125 3127 412C 9126 112F C11B 0041 005A 002E 0020 000D 0001 0200 002E 004E
130  0208 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
140  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
150  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
160  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
170  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
180  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
190  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
1A0  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

```
     +0   +1   +2   +3   +4   +5   +6   +7   +8   +9   +A   +B   +C   +D   +E   +F
1B0  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
1C0  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
1D0  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
1E0  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
1F0  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
200  0049 0056 0045 0054 0056 0041 0056 004E 0000 0000 0000 0000 0000 0000 0000 0000
210  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
220  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
230  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

Test 2 Output Screenshot:



	label	opcode	operand	hex
110	OutPhase	LOADI	Ptr	D132
111		SKIPCOND	400	8400
112		JUMP	Outp	9116
113		LOAD	ChSp	112C
114		OUTPUT		6000
115		HALT		7000
116	Outp	OUTPUT		6000
117		CLEAR		A000
118		STOREI	Ptr	E132
119		LOAD	Ptr	1132
11A		ADD	One	312E
11B		STORE	Ptr	2132

AC	0020	(Hex)
IR	7000	(Hex)
MAR	115	(Hex)
MBR	0020	(Hex)
PC	116	(Hex)
INPUT	.	ASCII

OUTPUT

```
IVETVAVN TECH
```

ASCII — No Linefeeds

Section 4 – Conclusions:

After debugging and testing Test 1 scenario, I was able to get “IVETVAVN” as an output, then when I reran but with Test 2 scenario, I was able to get “IVETVAVNTECH” as the final output. Then after adding the space after each run, the final output became “IVETVAVN TECH”. However, in the beginning of the testing, I was confused by how the output letters were displaying vertically instead of horizontally, until I read the note on Ex4_3 mentioning that I need to change the output window control setting to "no linefeeds" so the text will print in a single line, rather than in a column of single characters. Also, knowing how inputting works in MarieSim was challenging in the beginning as I was missing the “Enter” step after typing the input. Besides, using Skipcond instruction was a little tricky in the beginning until I ran some trials and errors to understand how it works. In addition, after clearing the output characters from the memory during each run to avoid printing previous run values in the new runs, I wasn’t able to confirm the correctness of the output characters’ memory locations after the program halts, so I had to comment/uncomment the 2 lines that clear the output characters to confirm that my results are correctly located in the memory.

Accordingly, I learned that it’s important to read and test the tutorial examples firstly to get more familiar with the MarieSim environment which reduces the time spent on the project. I also learned how to divide the project into small pieces and then combine those little pieces together to be more efficient with my coding time.

Lastly, the approximate number of hours I devoted to the project was about 12-14 hours. In general, the project was very helpful in understanding assembly language for the Marie processor and it was very enjoyable to see the output of the ROT13 transform.