System Hardware – COMP 228 Assignment 4

Team Members: Johanson Felix & George Mavroeidis

REPORT

Step 1: Implementation of IF-THEN-ELSE

Step 1: Implementation of IF-THEN-ELSE.
Write the following code segment in MARIE's assembly language:

```
Set initial values of X, Y as shown below (a, b, c)
    X:=20
    IF  X < 5  THEN
        Y:= X - 2; X:= 20;
    Else
        Y:= -1*(X ^3); X:= 8;
    ENDIF;
    Output X
    Output Y
```

Run your program in MarieSim environment, test it with different initial values of X and Y as follows:

a) Initial values: X = 15 , Y = 0; (all values are in decimal
b) Initial values: X = -25 , Y = 12; (all values are in decimal)
c) Initial values: X = 14 , Y = -14; (all values are in decimal)

Make sure your programs work correctly in each case.
Print your MARIE assembly codes and the results of your program in each of the above cases.
Show the content of memory and all registers at each important step.

Program Explanation:

In short (2-3 lines),

This program initializes a value in an address 'X' to 20, then changes the value of X to a new value. If the new value of X is less than 5 then the value in an address 'Y' is changed to the value of X subtracted by 2 and the value in address X is changed to 20. Else if X is not less than 5 then then the value in an address 'Y' is changed to negated X cubed (done by a loop of addition) and the value of in address X is changed to 8. Both Values of X and Y are then loaded and printed.

In long (detailed),

This program loads the value twenty (20) into the accumulator and stores this value in an address 'X'. It then loads a new value (newX) to the accumulator and this is stored in address X. This 'newX' value (case a, b, c) is additionally stored in an address 'count' that will be used as a counter in a subsequent loop.

Conditional branching is then used to determine whether X is greater than 5 by subtracting 5 from the accumulator. If the accumulator is left negative (SKIPCOND 000), meaning X is less than 5, then the line that proceeds is skipped and the address Y is stored as the value of X minus 2. Finally, X is assigned 20 from the accumulator. The program jumps to the end block which loads and outputs X and Y.

If the accumulator is not left negative, meaning X is greater or equal to 5, then Y is stored as the value of X cubed (by simulation of an addition loop), then this value is negated. Finally, X is assigned the value eight from the accumulator. The program jumps to the end block which loads and outputs X and Y.

Note: The program is tested with 3 case values –

a) X=20, X=15, Y=0
b) X=20, X=-25, Y=12
c) X=20, X=14, Y=-14

Source Code:

```
ORG         000   //Store program using memory location x000

//Initial value of X is loaded as 20
            Load        X
//New value for X is loaded into accumulator and stored in X
            Load newX
            Store X
//Count for subsequent loop takes value of X
            Store       count
// Subtract five from X, result will be in AC, this will be used to check if X<5
            Subt        five
//Check to see if AC is negative (implies X<5). If so, skip line Jump Else, and continue from If block
            Skipcond    000
//If AC is not negative (X>=5), then execute Jump Else to skip If block
            Jump        Else
```

```
//if AC is negative (implies X<5)  then load X back into AC, Subtract two from AC, Store value
in AC as Y, Load twentty to AC and Store as X, Jump to end block

If,     Load    X
        Subt    two
        Store   Y
        Load    twenty
        Store   X
        Jump    End

//If  AC is not negative (X>=5), then Jump to Loop1

Else,   Jump    Loop1

//Simulation of cubing a number by an addition Loop( Loop1 with Count2, Loop2):

//Load X into accumulator, Add value of sum1 (initially 0), Store value of AC in sum1, load
count to AC (count initially set to value of X), subtract one from AC, Store AC to count, if
AC>0 then skip next line and jump to Loop1 block again, else Jump to Count2 block

Loop1,  Load    X
        Add     sum1
        Store   sum1
        Load    count
        Subt    one
        Store   count
        Skipcond 800
        Jump    Count2
        Jump    Loop1

//load X into AC, store value of AC (X) in count and follow code to Loop2

Count2, Load    X
        Store   count

//Load sum1 into AC, Add value of sum to AC, Store AC to sum, Load count to AC, subtract
one from AC and store AC to count, if  if AC>0 then skip next line and jump to Loop2 block
again, else Jump to End block

Loop2,  Load    sum1
        Add     sum
        Store   sum
        Load    count
        Subt    one
        Store   count
        SkipCond 800
        Jump    Negate
        Jump    Loop2
```

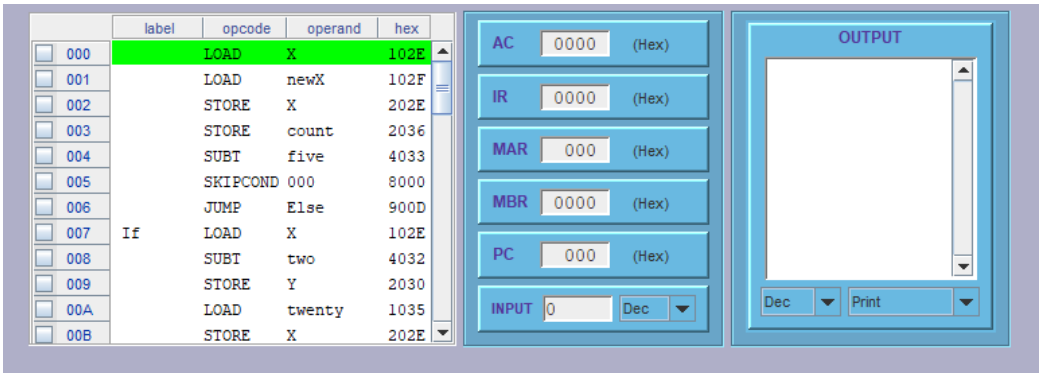
```
//Simulation of negating a number by subtracting the number from itself which results in 0 then
subtracting the number from 0 which results in the number negated

Negate,     Load    sum
            Subt    sum
            Subt    sum
            Store   Y
            Load    eight
            Store   X
            Jump    End


//Load X and Y to AC, output X and Y, halt program

End,        Load    X
            Output
            Load    Y
            Output
            Halt

//Identifier Declarations:

X,          DEC     20      /'initial value of X as 20'
newX,       DEC     15      /'value of X in each case(a,b,c) 15, -25, 14'
Y,          DEC     0       /'value of Y in each case(a,b,c) 0, 12, -14'
one,        DEC     1       /'value of 1'
two,        DEC     2       /'value of 2'
five,       DEC     5       /'value of 5'
eight,      DEC     8       /'value of 8'
twenty,     DEC     20      /'value of 20'
count,      DEC     0       /'count for loops'
sum1,       DEC     0       /'sum of Loop1'
sum,        DEC     0       /'Overall sum after looping'
```

Results, Contents/Outputs of important variables in Program:

a) X= 8 Y= -3375
b) X=20 Y= -27
c) X=8 Y= -2744

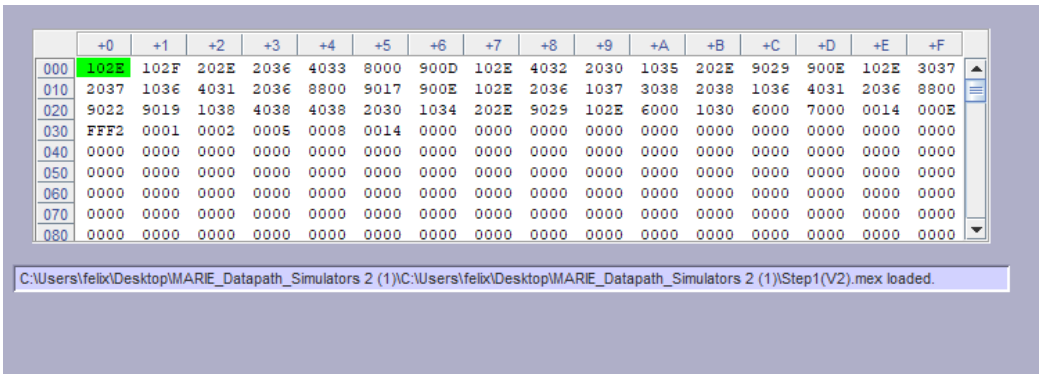
Snapshots

Registers and Contents of Memory before program runs



	label	opcode	operand	hex
000		LOAD	X	102E
001		LOAD	newX	102F
002		STORE	X	202E
003		STORE	count	2036
004		SUBT	five	4033
005		SKIPCOND	000	8000
006		JUMP	Else	900D
007	If	LOAD	X	102E
008		SUBT	two	4032
009		STORE	Y	2030
00A		LOAD	twenty	1035
00B		STORE	X	202E

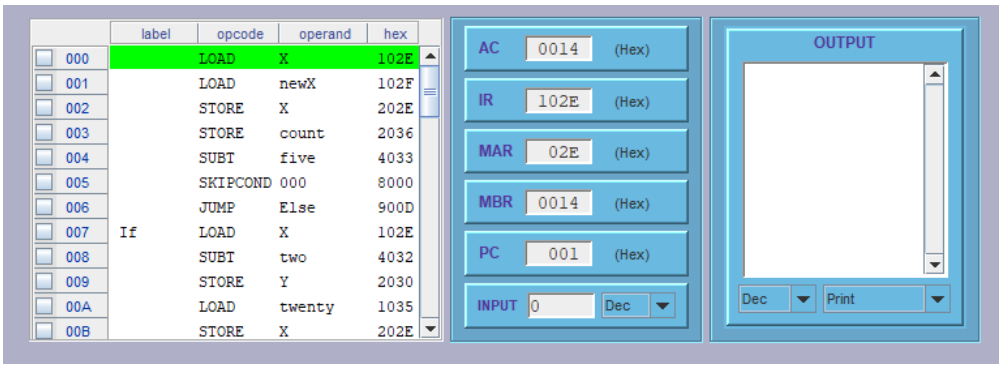
AC 0000 (Hex) | IR 0000 (Hex) | MAR 000 (Hex) | MBR 0000 (Hex) | PC 000 (Hex) | INPUT 0 Dec | OUTPUT | Dec | Print



	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	102E	102F	202E	2036	4033	8000	900D	102E	4032	2030	1035	202E	9029	900E	102E	3037
010	2037	1036	4031	2036	8800	9017	900E	102E	2036	1037	3038	2038	1036	4031	2036	8800
020	9022	9019	1038	4038	4038	2030	1034	202E	9029	102E	6000	1030	6000	7000	0014	000E
030	FFF2	0001	0002	0005	0008	0014	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

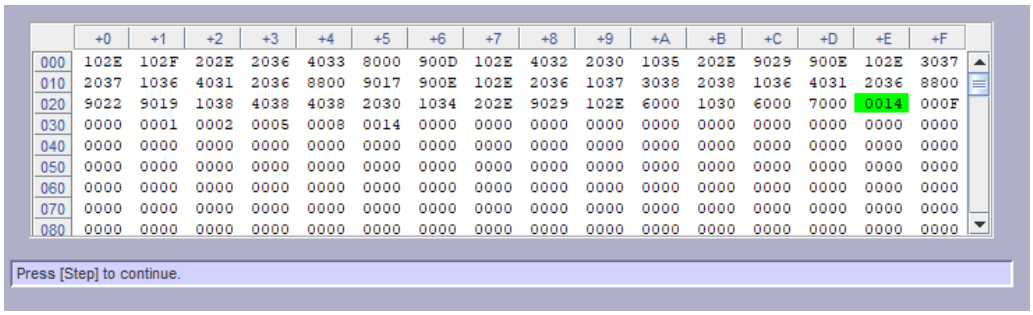
C:\Users\felix\Desktop\MARIE_Datapath_Simulators 2 (1)\C:\Users\felix\Desktop\MARIE_Datapath_Simulators 2 (1)\Step1(V2).mex loaded.

Registers and Contents of Memory at first Step



	label	opcode	operand	hex
000		LOAD	X	102E
001		LOAD	newX	102F
002		STORE	X	202E
003		STORE	count	2036
004		SUBT	five	4033
005		SKIPCOND	000	8000
006		JUMP	Else	900D
007	If	LOAD	X	102E
008		SUBT	two	4032
009		STORE	Y	2030
00A		LOAD	twenty	1035
00B		STORE	X	202E

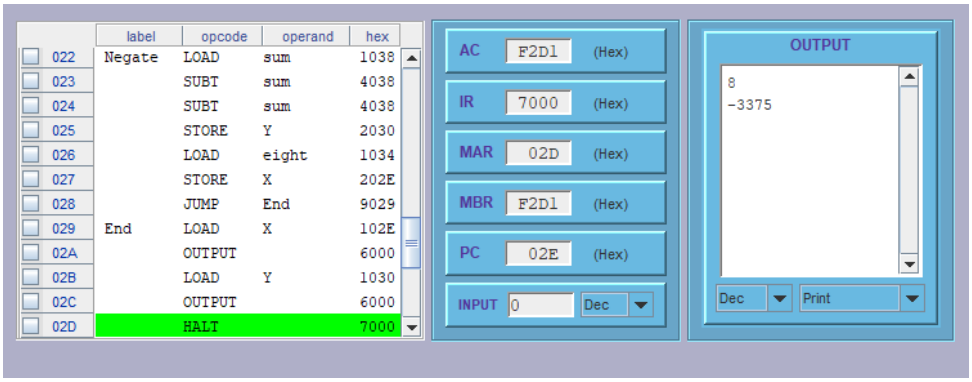
AC 0014 (Hex) | IR 102E (Hex) | MAR 02E (Hex) | MBR 0014 (Hex) | PC 001 (Hex) | INPUT 0 Dec | OUTPUT | Dec | Print



	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	102E	102F	202E	2036	4033	8000	900D	102E	4032	2030	1035	202E	9029	900E	102E	3037
010	2037	1036	4031	2036	8800	9017	900E	102E	2036	1037	3038	2038	1036	4031	2036	8800
020	9022	9019	1038	4038	4038	2030	1034	202E	9029	102E	6000	1030	6000	7000	0014	000F
030	0000	0001	0002	0005	0008	0014	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

Press [Step] to continue.

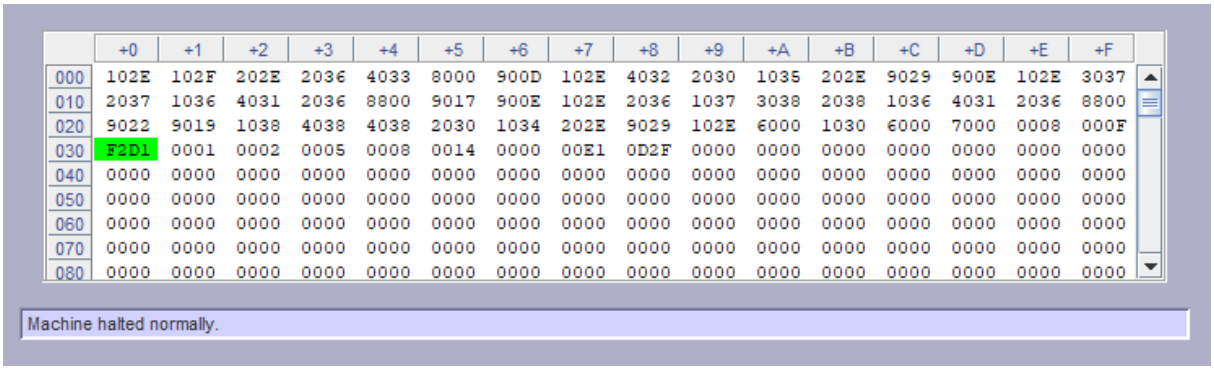
CASE A



	label	opcode	operand	hex
022	Negate	LOAD	sum	1038
023		SUBT	sum	4038
024		SUBT	sum	4038
025		STORE	Y	2030
026		LOAD	eight	1034
027		STORE	X	202E
028		JUMP	End	9029
029	End	LOAD	X	102E
02A		OUTPUT		6000
02B		LOAD	Y	1030
02C		OUTPUT		6000
02D		HALT		7000

AC F2D1 (Hex)
IR 7000 (Hex)
MAR 02D (Hex)
MBR F2D1 (Hex)
PC 02E (Hex)
INPUT 0 Dec

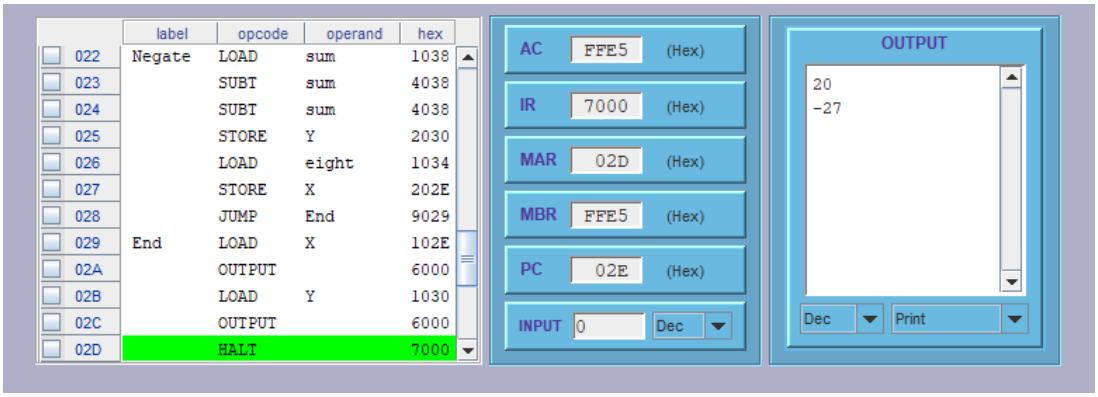
OUTPUT
8
-3375
Dec Print



	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	102E	102F	202E	2036	4033	8000	900D	102E	4032	2030	1035	202E	9029	900E	102E	3037
010	2037	1036	4031	2036	8800	9017	900E	102E	2036	1037	3038	2038	1036	4031	2036	8800
020	9022	9019	1038	4038	4038	2030	1034	202E	9029	102E	6000	1030	6000	7000	0008	000F
030	F2D1	0001	0002	0005	0008	0014	0000	00E1	0D2F	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.

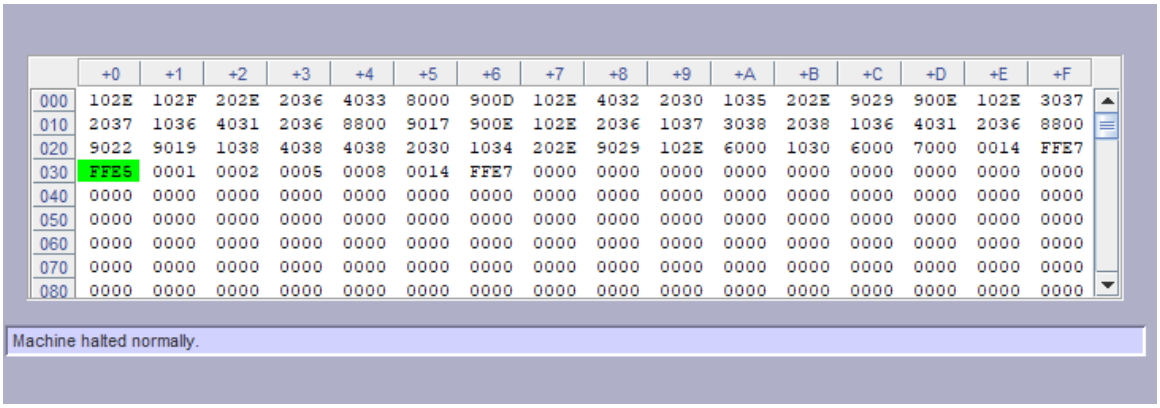
CASE B



	label	opcode	operand	hex
022	Negate	LOAD	sum	1038
023		SUBT	sum	4038
024		SUBT	sum	4038
025		STORE	Y	2030
026		LOAD	eight	1034
027		STORE	X	202E
028		JUMP	End	9029
029	End	LOAD	X	102E
02A		OUTPUT		6000
02B		LOAD	Y	1030
02C		OUTPUT		6000
02D		HALT		7000

AC FFE5 (Hex)
IR 7000 (Hex)
MAR 02D (Hex)
MBR FFE5 (Hex)
PC 02E (Hex)
INPUT 0 Dec

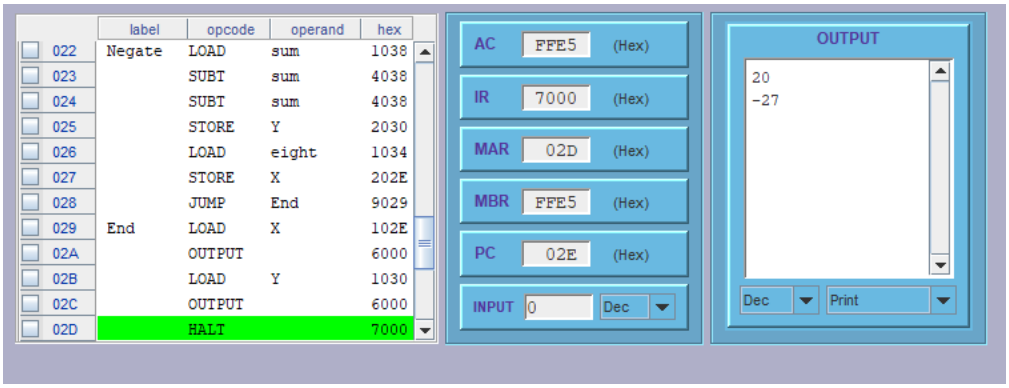
OUTPUT
20
-27
Dec Print

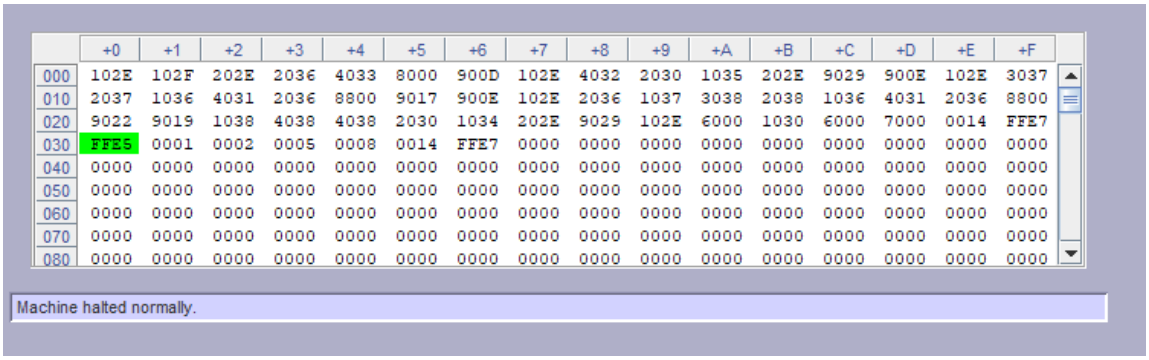


	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	102E	102F	202E	2036	4033	8000	900D	102E	4032	2030	1035	202E	9029	900E	102E	3037
010	2037	1036	4031	2036	8800	9017	900E	102E	2036	1037	3038	2038	1036	4031	2036	8800
020	9022	9019	1038	4038	4038	2030	1034	202E	9029	102E	6000	1030	6000	7000	0014	FFE7
030	FFE5	0001	0002	0005	0008	0014	FFE7	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.

CASE C





References

[1] L. Null and J. Lobur, The Essentials of Computer Organization and Architecture, -: Jones and Bartlett Publishers, 2014.

[2] L. Null and J. Lobur, " MarieSim: The MARIE computer," pp. 2, 3, 2003.