Temple University

College of Engineering

Department of Electrical and Computer Engineering (ECE)

Student Lab Report Cover Page

Course Number	**:**	3613
Course Section	**:**	002
Experiment #	**:**	Lab # 4

Student Name (print)	**:**	Robert Bara
TUid#	**:**	915614617
Date	**:**	9/23/2020

Grade	**:**	______ **/100**
TA Name	**:**	Sung Choi

ACTIVITIES:

Activity 1

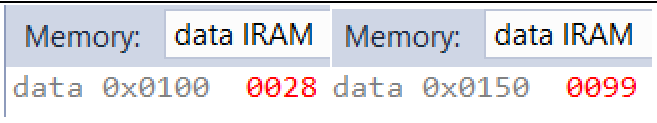
Write assembly codes to perform the given instructions (30 pts total).

1.1 Load the number of $28 to the data memory location 0x0100 and load the number of $99 to the data memory location 0x0150. Then, swap the values of the memory locations **(10 pts).**

Code (with full comment)

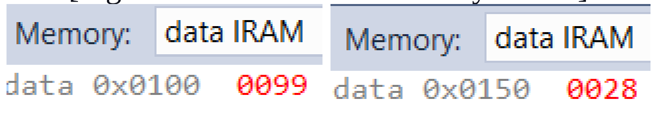
```
Act1:
        Ldi r18, $28 ;loading r18 with hex value $28
        Sts 0x0100,r18 ;storing the contents of r18 into memory location 0x0100
        Ldi r19,$99 ;loading r19 with the hex value $99
        Sts 0x0150,r19 ;storing the contents of r18 into memory location 0x0150
        //Swapping each values of memory locations
        Lds r5,0x0100 ;loading r5 with the contents of 0x0100 for swapping
        Lds r6,0x0150 ;loading r6 with the contents of 0x0150 for swapping
        Sts 0x0100,r5 ;swapping the contents by storing r5's contents into 0x0100
        Sts 0x0150,r6 ;swapping the contents by storing r6's contents into 0x0150
jmp Act1 ;jump back to label Act1
NOP ;No operation
```

Result (screenshot)



```
Memory: data IRAM            Memory: data IRAM
data 0x0100   0028           data 0x0150   0099
```

[Figure 1A &1B: Initial memory values]



```
Memory: data IRAM            Memory: data IRAM
data 0x0100   0099           data 0x0150   0028
```

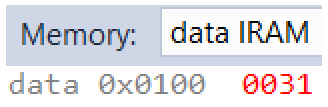
[Figure 1C & D: Swapped final memory values]

1.2 Use a directive, <.EQU>, to set three labels, data_BINARY, data_ASCII, and data_HEX and assign the values, 0b00110001, 'P,' and $28 to the labels respectively. Then, set three labels for the data memory locations, DATA1 for 0x100, DATA2 for 0x101, and DATA3 for 0x102. Store the value of data_BINARY, data_ASCII, and data_HEX to DATA1, DATA2, and DATA3, respectively. For storing operation, you must use the labels. **(20 pts)**

Code (with full comment)

```
Act1_2:
        .EQU data_BINARY=0b00110001 ;Creating label data_BINARY and setting it's value
        .EQU data_ASCII= 'P' ;Creating label data_ASCII and setting it's value
        .EQU data_HEX=$28 ;Creating label data_HEX and setting it's value
        .SET DATA1=0x100 ;Setting DATA1 equal to memory location 0x100
        .SET DATA2=0x101 ;Setting DATA2 equal to memory location 0x101
        .SET DATA3=0x102 ;Setting DATA3 equal to memory location 0x102
        Ldi R17,data_BINARY ;loading data_BINARY's value to R17 for storing
        Ldi R18,data_ASCII ;loading data_ASCII's value to R18 for storing
        Ldi R19,data_HEX ;loading data_HEX's value to R19 for storing
        sts DATA1,r17 ;Storing data_BINARY's value from R17 to DATA1
        sts DATA2,r18 ; Storing data_ASCII's value from R18 to DATA2
        sts DATA3,r19 ; Storing data_HEX's value from R19 to DATA3
        jmp Act1_2 ;jump back to label Act1_2
NOP ;No operation
```

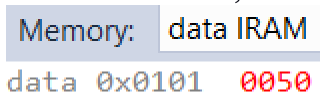
Result (screenshot)



Memory: data IRAM

```
data 0x0100  0031
```

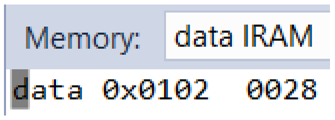
[Figure1E: Final values stored in DATA1 label, which is memory location 0x0100]



Memory: data IRAM

```
data 0x0101  0050
```

[Figure1F: Final values stored in DATA2 label, which is memory location 0x0101]



Memory: data IRAM

```
data 0x0102  0028
```

[Figure1G: Final values stored in DATA3 label, which is memory location 0x0102]

Activity 2

Write assembly codes to perform the basic arithmetic operations and observe the flags related the arithmetic operations (20 pts).

2.1 Store the number $a7 and $62 to the location 0x0200 and 0x201 respectively. Store the 2's complement of $a7 and $62 to the location 0x0205 and 0x206 . List the flags that turn on for each 2's complement operation and explain why the flags turn on **(10 pts).**

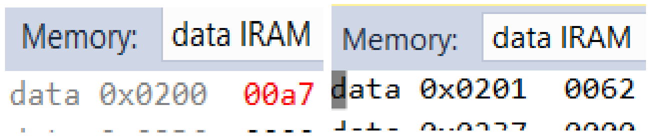
Code (with full comment)

```
Act2:
        Ldi r17,$a7 ;load $a7 into r17
        Ldi r18,$62 ;load $62 into r18
        sts 0x0200,r17 ;store r17's content into memory location 0x0200
        sts 0x201,r18 ;store r18's content into memory location 0x201

//Ben said using neg is fine rather than using com and adding 1, it does not say we can't use it

        neg r17 ;Takes the 2's complement of r17 and updates r17
        neg r18 ;Takes the 2's complement of r18 and updates r18
        sts 0x205,r17 ;Storing the 2’s complemented value into memory address 0x205
        sts 0x206,r18 ;Storing the 2’s complemented value into memory address 0x206
jmp act2 ;jump back when done to Act2 label
NOP ;no operation
```

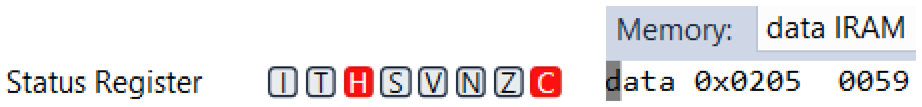
Result (screenshot) and Explanation (flags and reasons)



```
data 0x0200   00a7
data 0x0201   0062
```

[Figure 2.1 A &B: Storing the initial values into their memory locations]

Taking the 2’s complement of R17, the flags are as follows and the value $59 gets stored into memory location 0x0205:

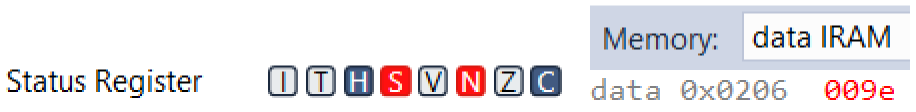


```
data 0x0205   0059
```

[Figure 2.1 C & D: Flags and stored memory location of 2’s Complement for R17]

This is because R17 holds $a7 which in binary is 8b1010 0111, negating this becomes 8b0101 1000 and adding 1 to store the two’s complement yeilds 8b0101 1001, which is also $59. By performing a bit by bit binary operation it is clear that the Cary and Half flags go high.

Taking the 2’s complement of R18, the flags are as follows and the value $9E gets stored into memory location 0x0206:

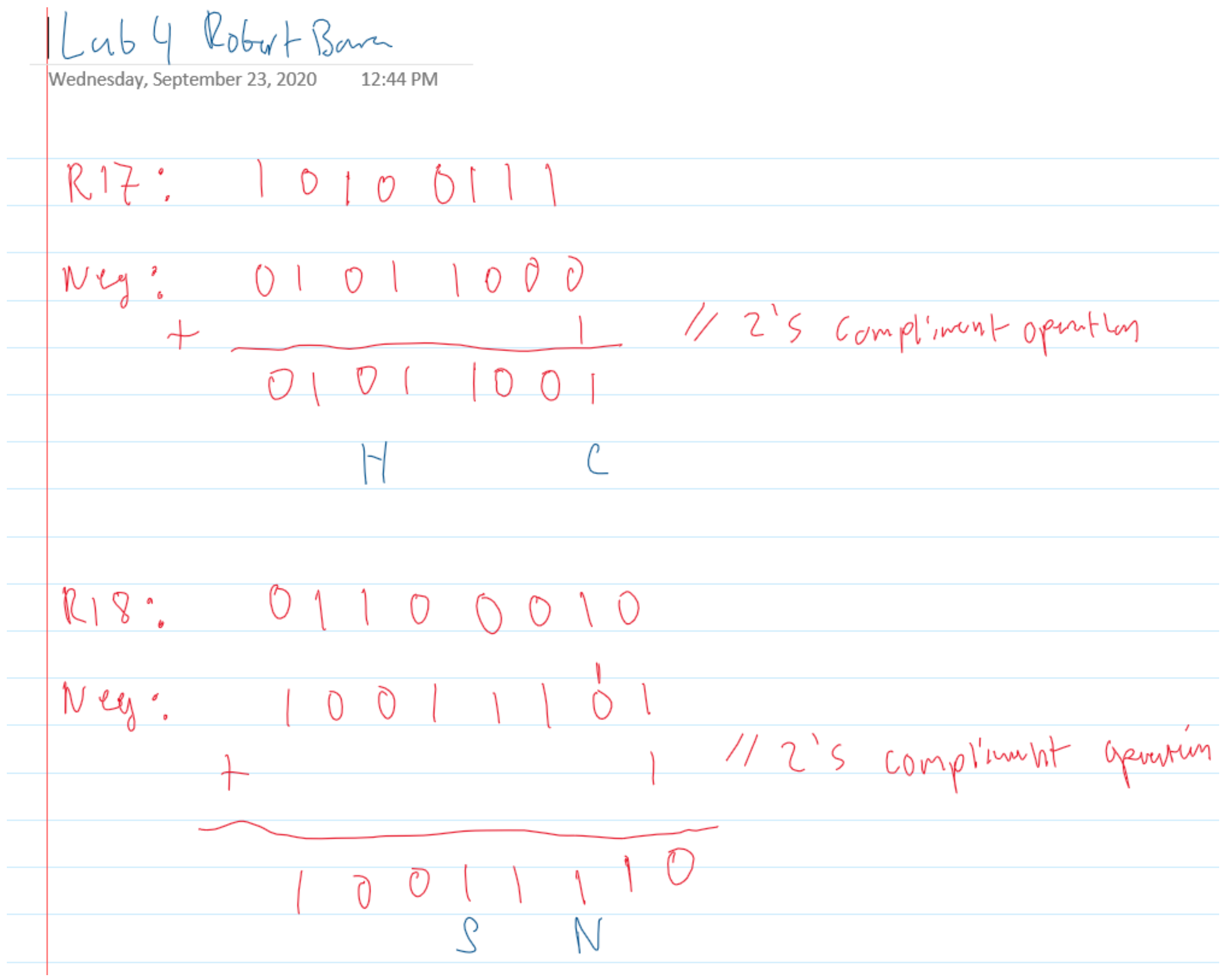


```
data 0x0206   009e
```

[Figure 2.1 E & F: Flags and stored memory location of 2’s Complement for R18]

Similary, this is because R18 holds $62 which in binary is 8b0110 0010, negating this becomes 8b1001 1101 and adding 1 to store the two’s complement yeilds 8b1001 1110, which is also $9E. By performing a bit by bit binary operation it is clear that the Sign and Negative flags go high and C and H go low from the previous operation.

Bit by Bit operation for both registers can be found below:


Lab 4 Robert Barn

Wednesday, September 23, 2020 12:44 PM

```
R17:  1010 0111
Neg:  0101 1000
   +          1   // 2's compliment operation
      ---------
      0101 1001
        H     C

R18:  0110 0010
Neg:  1001 1101
   +          1   // 2's compliment operation
      ---------
      1001 1110
          S  N
```

[Figure 2.1G: Bit by Bit operations for 2's Compliment of R17 and R18]

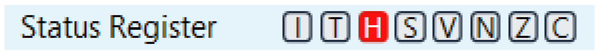
2.2 Subtract $7 from $12 four time. Find the flags that turn on when you perform each subtraction and list them **(10 pts).**

Code (with full comment)

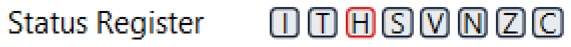
```
Act2_2:
        Ldi r17, 4 ;loading our counter for the loop
        Ldi r20,$7 ;loading r20 with $7 for subtraction
        Ldi r21,$12 ;loading r21 with $12 for subtraction

Loop: Sub r21,r20 ;subtracting r21=r21-r20
        Dec r17 ;decrementing the counter until it reaches 0, so the loop executes 4 times
        Brne Loop ;If branch is not equal to Z=0, loop again, if it is 0, exit loop
Jmp Act2_2 ;jump back to label Act2_2
NOP ;no operation
```

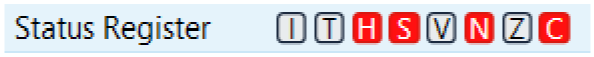
Result (screenshot)



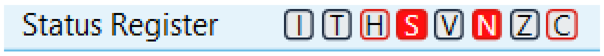
[Figure 2.2A: 1st Subtraction's Flags]



[Figure 2.2B: 2nd Subtraction's Flags]



[Figure 2.2C: 3rd Subtraction's Flags]



[Figure 2.2D: 4th and Final Subtraction's Flags]

Activity 3

Write two codes to complete the given task with two different branch instructions and labels. Each branch instructions must use different flags. Explain the branch instructions you select and draw the flowchart of each code. (25 pts / each code)

Task: Make the output of PORTA using the R16 value. The R16 initial value is $F. Decrease R16 by 3 and update the PORTA value until the value becomes zero. Store each value to the data memory locations listed below (see the expected result as followings):

Expected Result:

1st output of PORTA = $F
2nd output of PORTA = $C
3rd output of PORTA = $9
4th output of PORTA = $6
5th output of PORTA = $3
6th output of PORTA = $0

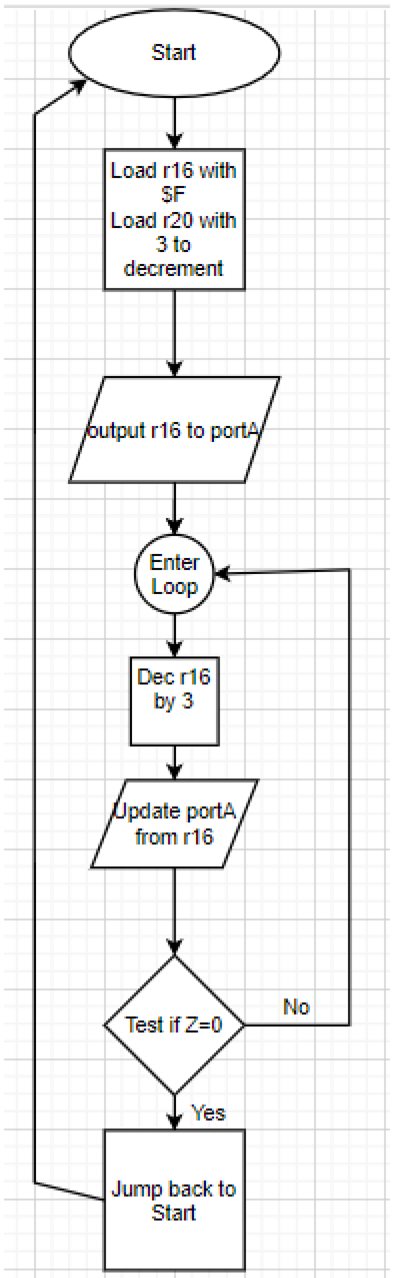
Code 1 with full comments and Branch Instruction Explanation

```
Act3: ;label Act3
        Ldi r16,$F ;Load r16 with $F
        Ldi r20,3   ;load r20 with 3 to decrement r16, loop counter if you will
        OUT PORTA, r16 ;Outputting the contents of r16 to PORTA
Loop1: Sub r16, r20 ;subtracting r16=r16-3, this is done by subtracting r20 from r16
        OUT PORTA, r16 ;updating the output of PortA
```

Brne Loop1 ; If branch is not equal to Z=0, loop again, if it is 0, exit loop
Jmp Act3 ;After the value of r16 is =0 so port A is 0, jump back to Act3 label
NOP ;No Operation

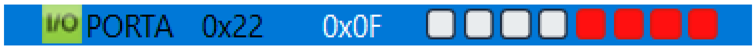
Using Brne, this will continue to loop until the z flag is =0, meaning that the value of PortA has become 0.

Flowchart

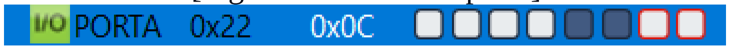


[Figure 3: Flowchart for 1st method]

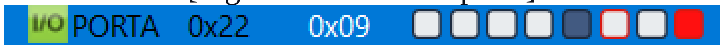
Result 1 (screenshot, 6 PORTA outputs)


PORTA 0x22 0x0F

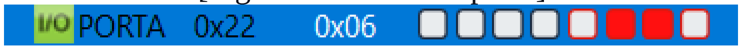
[Figure3A:PortA Output 1]


PORTA 0x22 0x0C

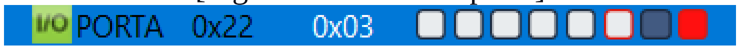
[Figure3B:PortA Output 2]


PORTA 0x22 0x09

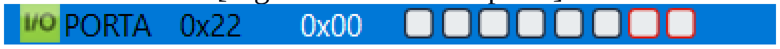
[Figure3C:PortA Output 3]


PORTA 0x22 0x06

[Figure3D:PortA Output 4]


PORTA 0x22 0x03

[Figure3E:PortA Output 5]


PORTA 0x22 0x00

[Figure3F:PortA Output 6]

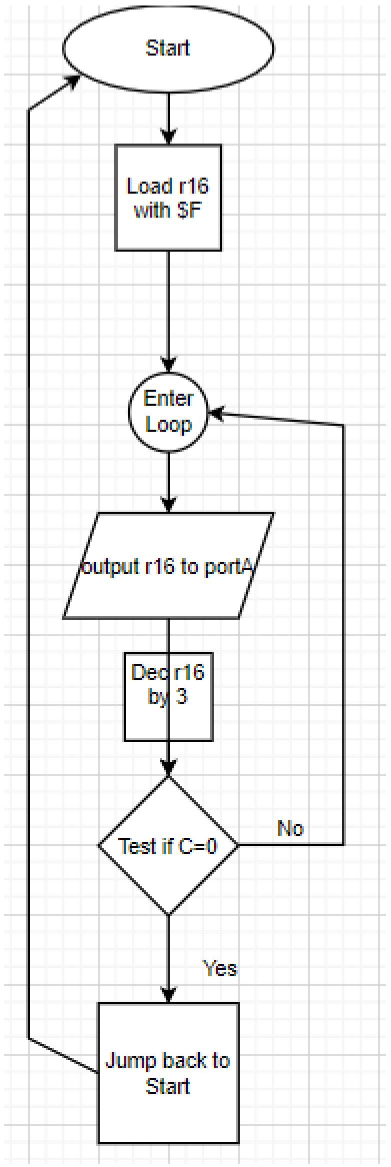
Code 2 with full comments and Branch Instruction Explanation

```
Act3ALT: ;label Act3 second method or Act3 Alternative
        Ldi r16,$F ;Load r16 with $F

Loop2: OUT PORTA, r16 ;Outputting the contents of r16 to PORTA
        Subi r16,3 ;Decrement r16 by 3
        Brcc Loop2 ; Loop until the carry is 0 (since C flag will be the last bit), exit loop
Jmp Act3ALT ;After the value of r16 is =0 so port A is 0, jump back to Act3 label
NOP ;No Operation
```

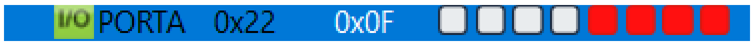
Using Brcc, you will keep looping back until the last bit is 0, when the C flag is cleared it will exit the loop. This will work since we want our last output to be 0.

Flowchart

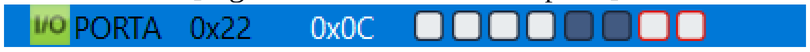


[Figure 3ALT: Flowchart for 2nd method]

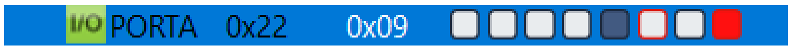
Result 2 (screenshot, 6 PORTA outputs)


PORTA 0x22 0x0F

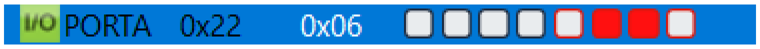
[Figure3ALT_A:PortA Output 1]


PORTA 0x22 0x0C

[Figure3ALT_B:PortA Output 2]

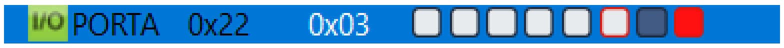

PORTA 0x22 0x09

[Figure3ALT_C:PortA Output 3]



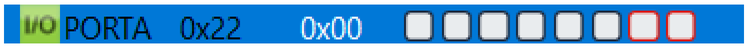
PORTA 0x22 0x06

[Figure3ALT_D:PortA Output 4]



PORTA 0x22 0x03

[Figure3ALT_E:PortA Output 5]



PORTA 0x22 0x00

[Figure3ALT_F:PortA Output 6]

ECE3613 Processor System Laboratory Rubric
Lab #: 4
Section: 001 / 002
Name: __________________________

Activity	**Section**	**Task**	**Full Points**	**Earned Points**	**Comment**
1	1.1	Code	10		
	1.2	Result Table	20		
Subtotal			30		
2	2.1	Code	10		
	2.2	Result	10		
Subtotal			20		
3	Code 1	Code	10		
		Flowchart	10		
	Code 2	Code	10		
		Flowchart	10		
	Result		10		Result for both Code 1 and Code 2 (5 pts each)
Subtotal			50		
Total			100		