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 # 3

Student Name (print) : Robert Bara

TUid# : 915614617

Date : 9/16/2020

Grade : _____ **/100**

TA Name : Sung Choi

ACTIVITIES:

Activity 1

Activity 1.1

Write and assemble a program to add the following data and then use the simulator to examine the C, H and Z flags after the execution of each addition. Copy and paste your code in the Code box. Write your solution in the result table. (10pts)

Values:

$92, $23, $66, $87, $F5

Code (copy and paste your code here):

```
ldi r18, $92
ldi r19, $23
ldi r20, $66
ldi r21, $87
ldi r22, $F5
ADD R18,R19
ADD R18,R20
ADD R18,R21
ADD R18,R22
```

Result:

Addition	Values of Flags C, H, and Z after each addition		
Flag	**C**	**H**	**Z**
$92+$23	0	0	0
$92+ $23+ $66,	1	0	0
$92+ $23+ $66+$87	0	1	0
$92+$23+ $66+$87+$F5	1	0	0

Activity 1.2

Write and assemble the following programs. Use the simulator to single-step and examine the flags and register content after the execution of each instruction. Check the SUB operation by hand and

see what flags are set and check it with the Status Register in Atmel Studio. Write your solutions in the result table. (10pts)

```
        LDI     R20,$27
        LDI     R21,$15
        SUB     R20, R21 ;Look at the Status Register and check by hand

        LDI     R20,$20
        LDI     R21,$15
        SUB     R20, R21 ;Look at the Status Register and check by hand


        LDI     R24,95
        LDI     R25,95
        SUB     R24, R25 ;Look at the Status Register and check by hand

        LDI     R22,50
        LDI     R23,70
        SUB     R22, R23 ;Look at the Status Register and check by hand

L1:     RJMP    L1
```

Result:

Code line	Registers' Values	Status Register (SR) Value (Show all 8bits)	Mathematic Expression and Result
EX. LDI R16,$f	**R16=$f**	SR = 0000 0000	**R16=$f**
LDI R17,$4	**R17=$4**	SR = 0000 0000	**R17=$4**
ADD R16,R17	**R16=$13** **R17=$4**	SR = 0010 0000	**R16=R16+R17** **=$f+$4 =$13**
LDI R20, $27	R20=$27	SR= 0000 0000	R20=$27
LDI R21, $15	R21=$15	SR=0000 0000	R21=$15
SUB R20, R21	R20=$12 R21=$15	SR = 0000 0000	R20=R20-R21 =$27-$15=$12
LDI R20, $20	R20=$20	SR=0000 0000	R20=$20
LDI R21, $15	R21=$15	SR=0000 0000	R21=$15
SUB R20, R21	R20=$B R21=$15	SR = 0010 0000	R20=R20-R21 =$20-$15=$B
LDI R24, 95	R24=95	SR=0000 0000	R24=95
LDI R25, 95	R25=95	SR=0000 0000	R24=95
SUB R24, R25	R24=0 R25=95	SR =0000 0010	R24=R24-R25 =95-95=0
LDI R22, 50	R22=50	SR=0000 0000	R22=50
LDI R23, 70	R23=70	SR=0000 0000	R23=70
SUB R22, R23	R22=-20	SR = 0011 0101	R22=R22-R23

	R23=70		=50-70=-20

Activity 1.3

Find the value of the C, Z and H flags after the execution of the following codes, check by hand. Write your answer in the result table. (10pts)

(a)
```
LDI   R20, $85
LDI   R21, $92
ADD   R20, R21
```

(b)
```
LDI   R16, $15
LDI   R17, $72
ADD   R16, R17
```

(c)
```
LDI   R25, $F5
LDI   R26, $52
ADD   R25, R26
```

(d)
```
LDI   R25, $FE
INC   R25
INC   R25
```

Result:

Section	C flag	Z flag	H flag	Mathematic Expression and Result
(a)	1	0	0	R20=R20+R21 =$85+$92=$117
(b)	0	0	0	R16=R16+R17 =$15+$72=$87
(c)	1	0	0	R25=R25+R26 =$F5+$52=47
(d)	0	1	0	R25=R25+1 =$FE+1=$FF R25=R25+1 =$FF+1=$0

Activity 2

Read each code and write its own mathematical or logical expression (30 pts, 10pts/ea).

Run the following codes in the Atmel Studio environment. Examine the values of each register. Fill each section (it's operation) with mathematical or logical expression.

Section	Assembly Code	Mathematical and Logical Expression

Example	LDI R17, 0xA3
	LDI R18, 4
	ADD R17, R18
	label: DEC R18
	BRNE label
	NOP
		R17 = 0xA3
		R18 = 4
		R17 = R17 + R18
		R18 = R18 – 1
		If Z=0, then branch to **label**, else go to the next line NOP

Section	Assembly Code	Mathematical and Logical Expression

2-1.	LDI R22, $B5
	LDI R21, $4
	here: SUB R22, R21
	BRNE here
	NOP
		R22=$B5
		R21=$4
		R22=R22-R21
		If Z=0, then branch to **here**, else go to next line NOP

2-2.	LDI R16, $05
	LDI R17, $E6
	L1: ADD R16, R17
	BRCC L1
	NOP
		R16=$05
		R17=$E6
		R16=R16+R17
		If C=0, then branch to **L1**, else go to NOP

2-3.	LDI R25, $04
	LDI R26, $F4
	JAZZ: CP R25, R26
	DEC R25
	BRNE JAZZ
	NOP
		R25=$04
		R26=$F4
		Compare R25 and R26, C Flag
		R25=R25-1
		If Z=0, then branch to **Jazz**, else go to next line NOP

Activity 3

Read the error message and correct the error parts in the code (40 pts, 20pts/ea).

[Example] This assembly code loads the hex values to the general purposed registers, R16 and R17. Then, it swaps the contents of the registers. Answer the questions.

```
LDI R16, 560          ; load a value 560 in decimal to R16
LD R17, 0x81          ; load a value 0x81 in hex to R17

; The following section is for swapping the values of the register 16 and 17
```

```
MOV R22, R16        ; move the value of R16 to R22
MOV R16, R17        ; move the value of R17 to R16
MOV R17, R22        ; move the value of R22 to R17
```

Questions:

Q. What are the syntax errors in this code and how to fix?

➔Error 1. LDI R16, 560 ; load a value 560 in decimal to R16

Reason: General Purpose Register has only 8-bit space. So, the maximum number that we can load is 255 in decimal ($FF in hex).

Fix: Reduce the number we want to load to R16 or break the number into a higher byte and a lower byte part. For instance, the decimal number 560 is 0x0230 in hex. So, we can load the higher byte 0x02 into R17 and the lower byte 0x30 to R16.

➔Error 2. LD R17, 0x81 ; load a value 0x81 in hex to R17

Reason: The assembly instruction LD (load indirect from data space to Register) only load the value to Register using indirectly using pointers X, Y, and Z. For instance, the value in a space where X pointer is pointing is loaded to R10 using the code line - LDI R10, X.

Fix: To fix the line, change the instruction LD to LDI (load immediate).

Screenshots: Attach the screenshots of the modified code and register values.

1) Show your modified code.

```
LDI R16, 255        ; load a value 255 in decimal to R16
LDI R17, 0x81       ; load a value 0x81 in hex to R17
; The following section is for swapping the values of the register 16 and 17
MOV R22, R16        ; move the value of R16 to R22
MOV R16, R17        ; move the value of R17 to R16
MOV R17, R22        ; move the value of R22 to R17
```

2) Show the final value of the register 16 and 17.

R16	0x81
R17	0xFF

3-1. This assembly code loads four hex values ($10, $42, $a8, and $11) to the address 0x0100, 0x101, 0x0102 and 0x0103 in the data memory.

```
LDI $10, R0         ; load a hex value $10 to a GPR
STS 0x100,R0        ; store the value to the memory location 0x0100
LDI R17, $42        ; load a hex value $42 to a GPR
```

```
STS 0x101,R17        ; store the value to the memory location 0x0101
LDI R1, $A8          ; load a hex value $A8 to a GPR
STS 0x102,R1         ; store the value to the memory location 0x0102
LDI R18, $11         ; load a hex value $11 to a GPR
STS 0x103,R18        ; store the value to the memory location 0x0103
```

Questions:

Q. What are the syntax errors in this code and how to fix?

Solution:

Answer:

Error - Reason: Cannot ldi into R0-R15. Also, there is a syntax error for ldi into R0, register must come first before what you are loading in.

Fix: Change the instruction for ldi $10, r0 (also a syntax error) to ldi into a register r16-r31.

Screenshots: Attach the screenshots of the modified code and SRAM locations 0x0100 through 0x0103

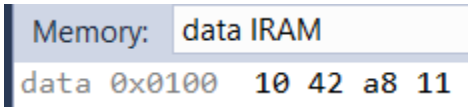
1) Show your modified code (copy and paste code)

Code:

```
;Activity 3
LDI r20,$10          ; load a hex value $10 to a GPR
STS 0x100,R20        ; store the value to the memory location 0x0100
LDI R17, $42               ; load a hex value $42 to a GPR
STS 0x101,R17        ; store the value to the memory location 0x0101
LDI R21, $A8               ; load a hex value $A8 to a GPR
STS 0x102,R21                ; store the value to the memory location 0x0102
LDI R18, $11         ; load a hex value $11 to a GPR
STS 0x103,R18        ; store the value to the memory location 0x0103
```

2) Show the final values of the memory location 0x100, 0x101, 0x102, and 0x103

Screenshots:



<Figure. Final values in memory location 0x100>

<Figure. Final values in memory location 0x101>

<Figure. Final values in memory location 0x102>

<Figure. Final values in memory location 0x103>

3-2. This assembly code loads value $7 to the memory address 0x0200. Then, load the contents of 0x0200 to the memory location 0x220. The content of the location 0x220 is decremented continuously until the value hits zero by using the register R1.

```
            LD R17, $7          ; load $7 to R17
            STS 0x200, R17      ; store R17 value to 0x200
LABEL:      LDI R1, 0x200       ; load the content of 0x200 to R1
            STS 0x220, R1       ; store the value of R1 to the location 0x220
            DEC R1              ; decrement R1 value by 1
            BRNE LABEL          ; If Z=0, then branch to label, else go to the next line

HERE:       RJMP HERE           ; jump to the label here
```

Questions:

Q. What are the syntax errors in this code and how to fix?

Solution:

Error – Reason: Cannot ld into R17. Cannot ldi into R1.

Fix: Change ld to ldi R17, change the register for R1 to a register between R16-R31, also use lds for that register.

Screenshots: Attach the screenshots of the modified code and the register/memory values.

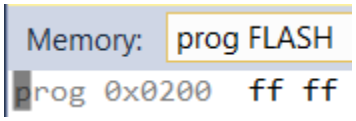
1. Show your modified code (Copy and paste code here)

Code:

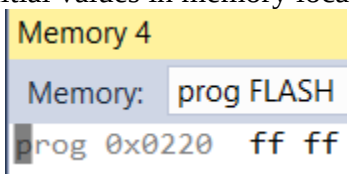
```
            LDI R17, $7         ; load $7 to R17
            STS 0x200, R17      ; store R17 value to 0x200
LABEL:      LDS R19, 0x200      ; load the content of 0x200 to R1
            STS 0x220, R19      ; store the value of R19 to the location 0x220
            DEC R19             ; decrement R19 value by 1
            BRNE LABEL    ; If Z=0, then branch to label, else go to the next line

HERE:  RJMP HERE          ; jump to the label here
```

2. Show the initial and final values in the memory location 0x200 and 0x0220

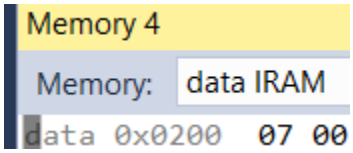
Screenshot:



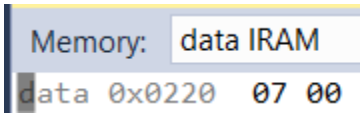
< Figure. Initial values in memory location 0x200>



< Figure. Initial values in memory location 0x220>



< Figure. Final values in memory location 0x200>



< Figure. Final values in memory location 0x220>

ECE3613 Processor System Laboratory Rubric
Lab #: 3
Section: 001 / 002
Name: _____

Activity	Section	Task	Full Points	Earned Points	Comment
1	1.1	Code & Result	10		
	1.2	Result Table	10		
	1.3	Result Table	10		
Subtotal			30		
2	2.1	Math/logic Exp	10		
	2.2	Math /logic Exp	10		
	2.3	Math/logic Exp	10		
Subtotal			30		
3	3.1	Question	10		
		Code & Result	10		
	3.2	Question	10		
		Code & Result	10		
Subtotal			40		
Total			100		