Temple University

College of Engineering

Department of Electrical and Computer Engineering (ECE)

Student Lab Report Cover Page

Course Number : 3613

Course Section : 001 / 002

Experiment # : Lab #5

Student Name (print) : Von Kaukeano

TUid# : 915596703

Date : 10/3/19

Grade : _____ **/100**

TA Name : Sung Choi

ACTIVITIES:

Activity 1

Write an assembly code using the stack pointer (SP). Store values in the memory location using SP and retrieve values from the memory locations to output values to the ports (total 60 points).

Show the resulted outputs of the activities using screenshot (use snipping tool) and your code (copy and paste).

Requirement
Draw a flowchart for Activity 1 to show the flow of the code. The flowchart will include the whole procedure of activity 1 (10 points). NOTE: Flowchart and code section will be found at the end of Activity 1.

Activity 1.1

Store the given values in the memory locations using Stack Pointer (SP). (Total 20 points)

Required procedure:

(1) Initialize SP at 0x0210.

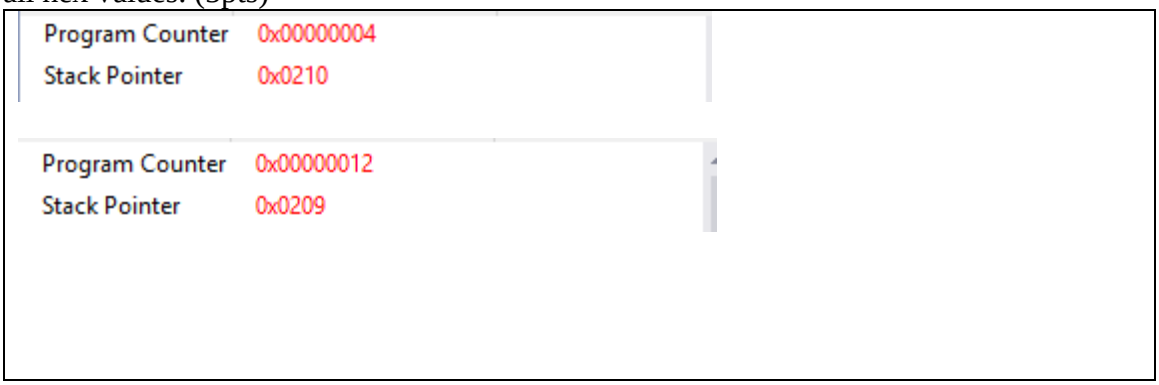
(2) Use PUSH instruction to store values, $34, $11, $92, $0F, $10, $ C5, and $67 in the locations from 0x0210, 0x020F, 0x020E, 0x020D, 0x020C, 0x020B, and 0x020A respectively (see Table 1).

Table 1. SP values and the contents of the memory locations

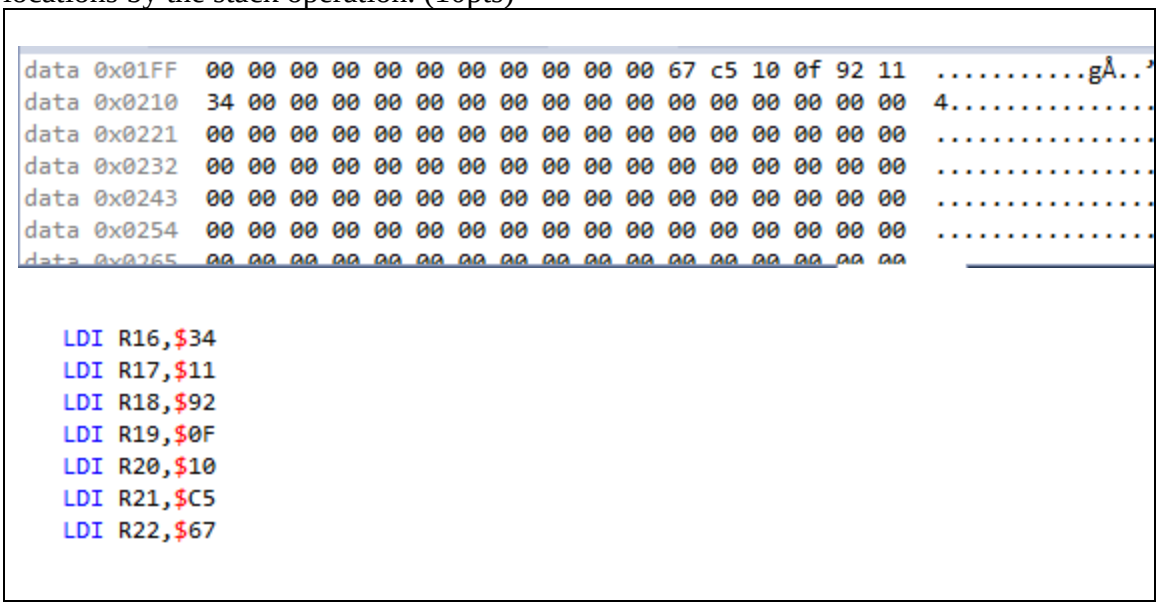
Value	Stack Operation	SP Value
$34	Push	0x0210
$11	Push	0x020F
$92	Push	0x020E
$0F	Push	0x020D
$10	Push	0x020C
$C5	Push	0x020B
$67	Push	0x020A

Outputs: (Total 15pts)

(1) The initialized SP value and the last SP value after PUSH instruction is completed to store all hex values. (5pts)



(2) The loaded values to the general-purpose registers and the values stored in the memory locations by the stack operation. (10pts)



Code for the activity 1.1 only: (5pts)

```
start:
        LDI R16, 0x10
        OUT SPL, R16
        LDI R17, 0x02
        OUT SPH, R17



        LDI R16,$34
        LDI R17,$11
        LDI R18,$92
        LDI R19,$0F
        LDI R20,$10
        LDI R21,$C5
        LDI R22,$67

        PUSH R16
        PUSH R17
        PUSH R18
        PUSH R19
        PUSH R20
        PUSH R21
        PUSH R22

HERE:RJMP HERE
```

Activity 1.2

Retrieve the stored values in the memory locations using Stack Pointer (SP) and do the specific arithmetic operations. The results of the arithmetic operations and the carry flag values must be shown as the outputs of the specified ports. (30 points)

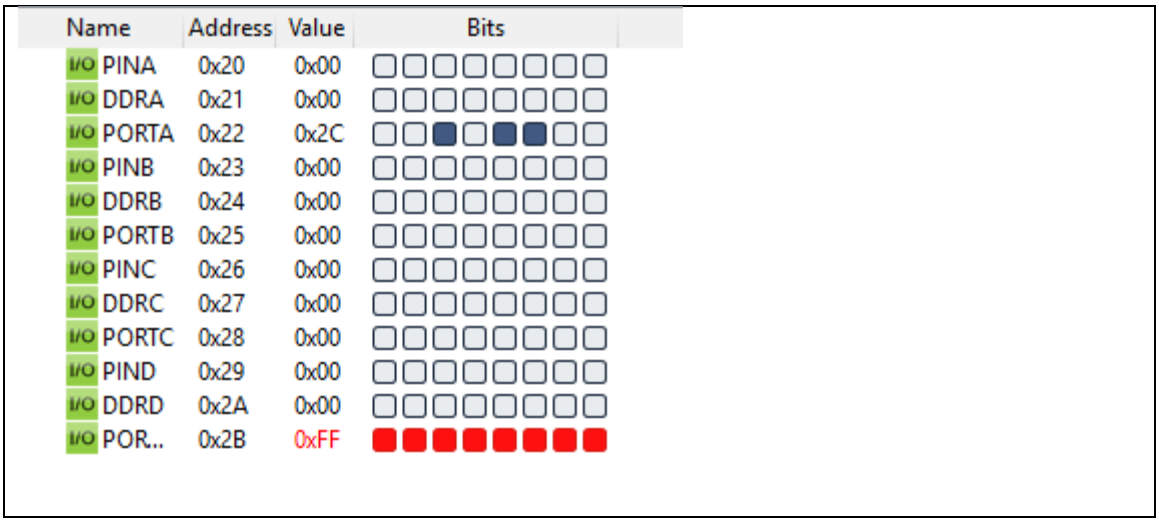
Required procedure:

(1) Use POP instruction to retrieve the values for the arithmetic operations. The values from the memory locations from 0x0210, 0x020F, 0x020E, 0x020D, 0x020C, 0x020B, and 0x020A must be used to do the arithmetic operations (see Table 2).
(2) Retrieve the values from the memory locations and load them in the registers first before the arithmetic operation is started. Set the SP location to the end of RAM after all the values are retrieved. Then, do the arithmetic operations.

(3) Use the single step execution to show each output.

Outputs: (10pts each)

(1) PORTA = $67 + $C5
PORTD = 00000000 while the carry bit is clear (C=0), but PORTD = 11111111 if the carry bit is set (C=1)
Result screenshot:

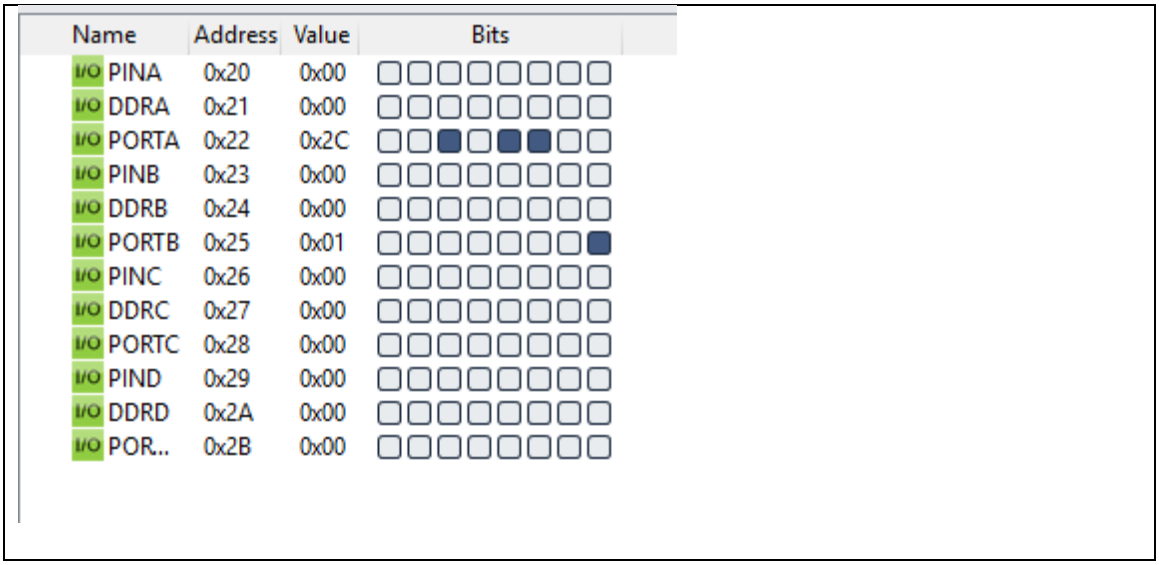


Name	Address	Value	Bits
PINA	0x20	0x00	
DDRA	0x21	0x00	
PORTA	0x22	0x2C	
PINB	0x23	0x00	
DDRB	0x24	0x00	
PORTB	0x25	0x00	
PINC	0x26	0x00	
DDRC	0x27	0x00	
PORTC	0x28	0x00	
PIND	0x29	0x00	
DDRD	0x2A	0x00	
POR...	0x2B	0xFF	

(2) PORTB = $10 - $0F

PORTD = 00000000 while the carry bit is clear (C=0), but PORTD = 11111111 if the carry bit is set (C=1)

Result screenshot:



Name	Address	Value	Bits
PINA	0x20	0x00	
DDRA	0x21	0x00	
PORTA	0x22	0x2C	
PINB	0x23	0x00	
DDRB	0x24	0x00	
PORTB	0x25	0x01	
PINC	0x26	0x00	
DDRC	0x27	0x00	
PORTC	0x28	0x00	
PIND	0x29	0x00	
DDRD	0x2A	0x00	
POR...	0x2B	0x00	

(3) PORTC = $92 + $11 - $34

PORTD = 00000000 while the carry bit is clear (C=0), but PORTD = 11111111 if the carry bit is set (C=1)

Result screenshot:

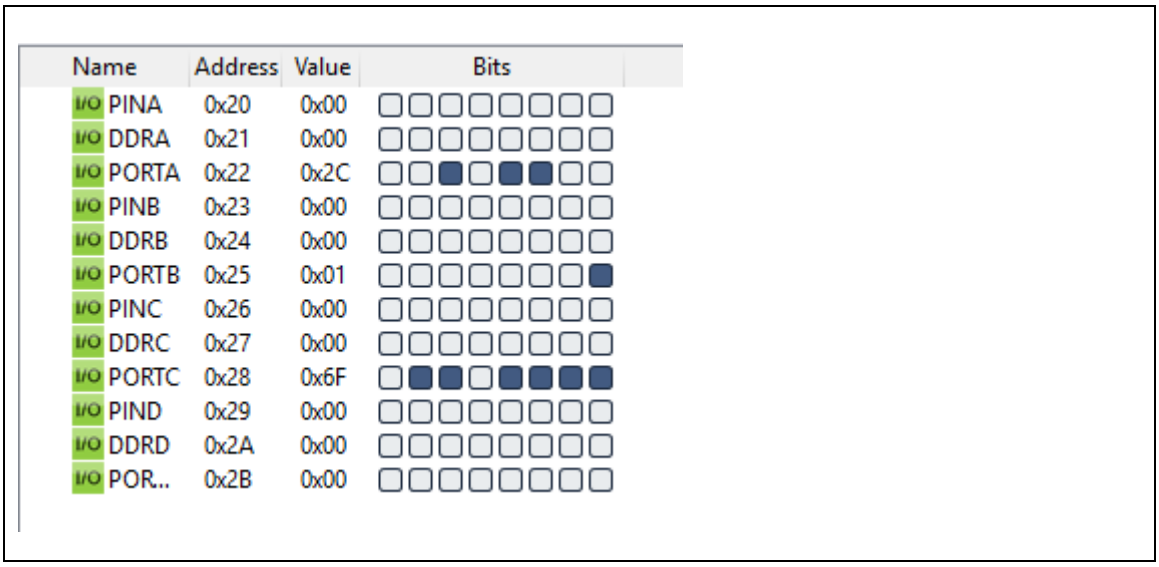
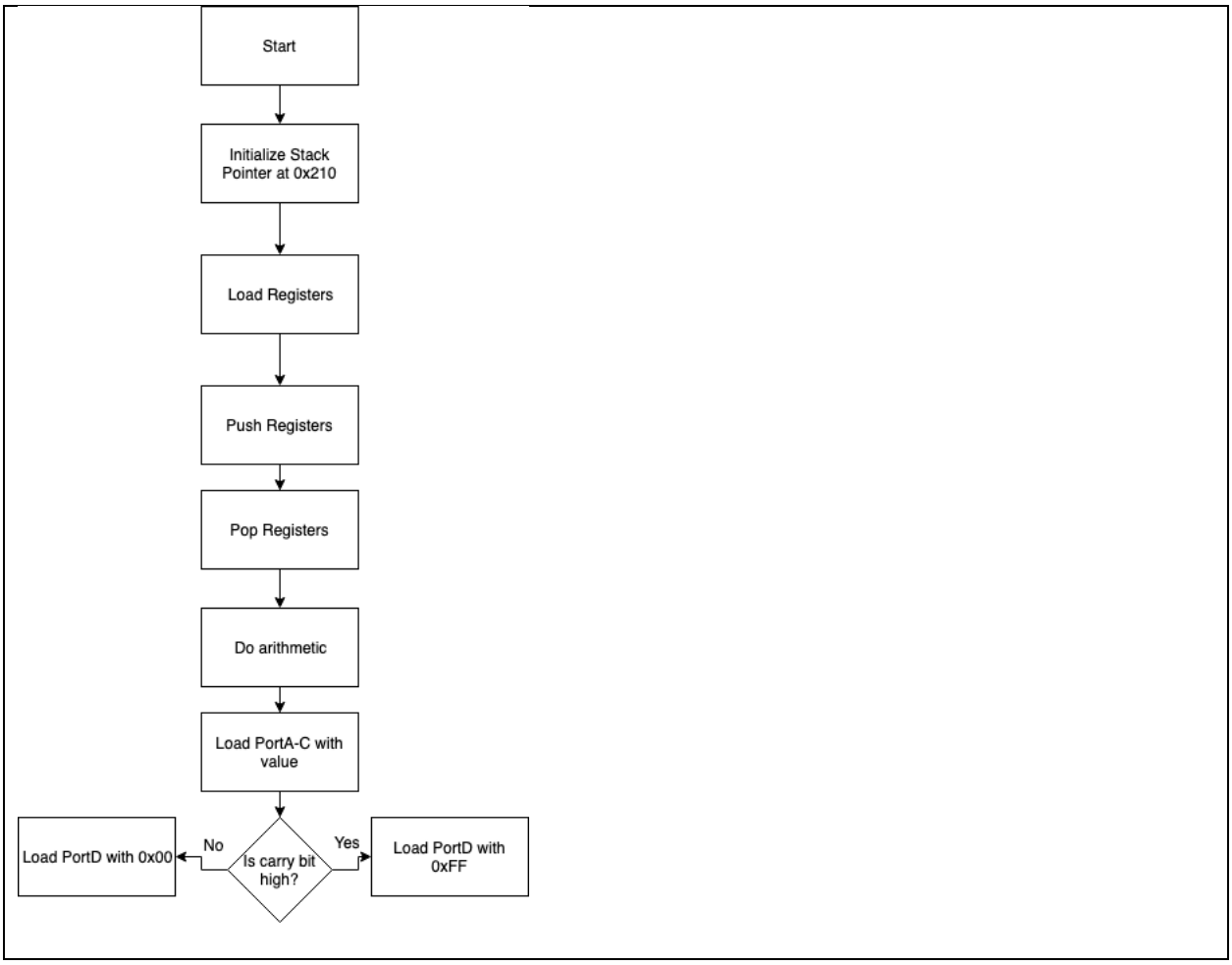


Table 2. Values in the Stack and the output ports

SP value	Stack Operation	Value	Output Number
0x0210	Pop	$34	(3)
0x020F	Pop	$11	
0x020E	Pop	$92	
0x020D	Pop	$0F	(2)
0x020C	Pop	$10	
0x020B	Pop	$C5	(1)
0x020A	Pop	$67	

Flowchart for Activity 1 (10 pts)



Code for activity 1 (copy and paste the entire code showing activity 1.1 and activity 1.2)

NOTE: If you do not include your code, you will get '0' points for this activity.

```
start:
        LDI R16, 0x10
        OUT SPL, R16
        LDI R17, 0x02
        OUT SPH, R17

        LDI R16,$34
        LDI R17,$11
        LDI R18,$92
        LDI R19,$0F
        LDI R20,$10
        LDI R21,$C5
        LDI R22,$67
```

```
        PUSH R16
        PUSH R17
        PUSH R18
        PUSH R19
        PUSH R20
        PUSH R21
        PUSH R22

        POP R16
        POP R17
        ADD R17,R16
        OUT PORTA, R17
        CALL PORT

        BCLR 0
        POP R17
        POP R16
        SUB R17,R16
        OUT PORTB,R17
        CALL PORT

        BCLR 0
        POP R16
        POP R17
        POP R18
        ADD R17,R16
        SUB R17,R18
        OUT PORTC,R17
        CALL PORT

HERE: RJMP HERE

PORT:
        LDI R20,0X00
        OUT PORTD,R20
        SBRS R17,0
        LDI R20,0XFF
        OUT PORTD,R20
        RET
```

Activity 2

Write an assembly code to use SP, subroutine, and directives for the arithmetic operation. Store two sets of 32-bits number using directives and store the resulted output in the designated ports after the required logic operation (total 50 points).

Show the resulted outputs of the activities using screenshot (use snipping tool) and your code (copy and paste).

Requirement: Draw a flowchart for Activity 2 to show the flow of the code. The flowchart will include the whole procedure of activity 2 (10 points). NOTE: Flowchart and code section will be found at the end of Activity 2.

Required procedure:

(1) Use directives to store the value of the number sets in the memory locations (see Table 3). Use the SP or the regular pointers (X or Y) for this operation.

Show the code section to set the pointer and directives: (5pts)

```
.equ NUM_SET1 = 0x0250
.equ NUM_SET2 = 0x0254
.equ OUT_LOG  = 0x0270

LDI R16,LOW(RAMEND)
OUT SPL, R16
LDI R17,HIGH(RAMEND)
OUT SPH, R17

LDI XL, LOW(NUM_SET1)
LDI XH, HIGH(NUM_SET1)

LDI YL, LOW(NUM_SET2)
LDI YH, HIGH(NUM_SET2)
```

(2) Set the SP value to the end of the Stack. Then, call the subroutines for each logic operation for each byte: AND, OR, NAND, and EX-OR, as shown in Table 3. Use the specific name of the label for each subroutine call:

Subroutine call labels:

- Num_and for AND operation
- Num_or for OR operation
- Num_nand for NAND operation
- Num_exor for EX-OR operation

Table 3. Directives, their memory location value, and the contents

Directives	Initial Memory Location (Highest Byte location)	Highest Byte			Lowest Byte
NUM_SET1	0x0250	$19	$02	$C5	$66
NUM_SET2	0x0254	$4A	$18	$23	$F4
OUT_LOG	0x0270	AND	OR	NAND	EX-OR

(3) Store the results from (2) to the location from 0x026D to 0x0270 using SP (PUSH instruction). See Table 4.
Note: Initialize the SP value to 0x0270 for this operation.

Show the code section to do the logic operations and push operation: (5pts)

```
LDI ZL, LOW(OUT_LOG)
LDI ZH, HIGH(OUT_LOG)
```

(4) Retrieve the resulted output of OUT_LOG using SP and load them to the registers.

Show the code section for this pointer setup and pop operation: (5pts)

```
Num_and:
LD R16,X
LD R17,Y
AND R16,R17
ST Z,R16
OUT PORTD,R16
RET

Num_or:
```

```
LD R16,-X
LD R17,-Y
OR R16,R17
ST -Z,R16
OUT PORTC,R16
RET

Num_nand:
LD R16,-X
LD R17,-Y
AND R16,R17
COM R16
ST -Z,R16
OUT PORTB,R16
RET

Num_exor:
LD R16,-X
LD R17,-Y
EOR R16,R17
ST -Z,R16
OUT PORTA,R16
RET
```

(5) Set the SP value to the end of the Stack (RAM).
(6) Show the OUT_LOG values in the ports, as shown below:

Table 4. Memory location and their contents that will be out through PORTA

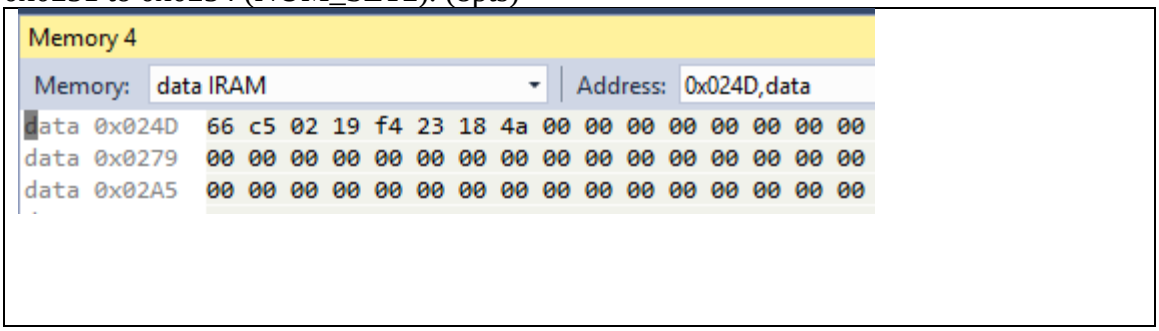
Memory Location	Contents	PORT
0x026D	EX-OR ($66, $F4)	A
0x026E	NAND($C5,$23)	B
0x026F	OR($02,$18)	C
0x0270	AND ($19, $4A)	D

Outputs: (25pts total)

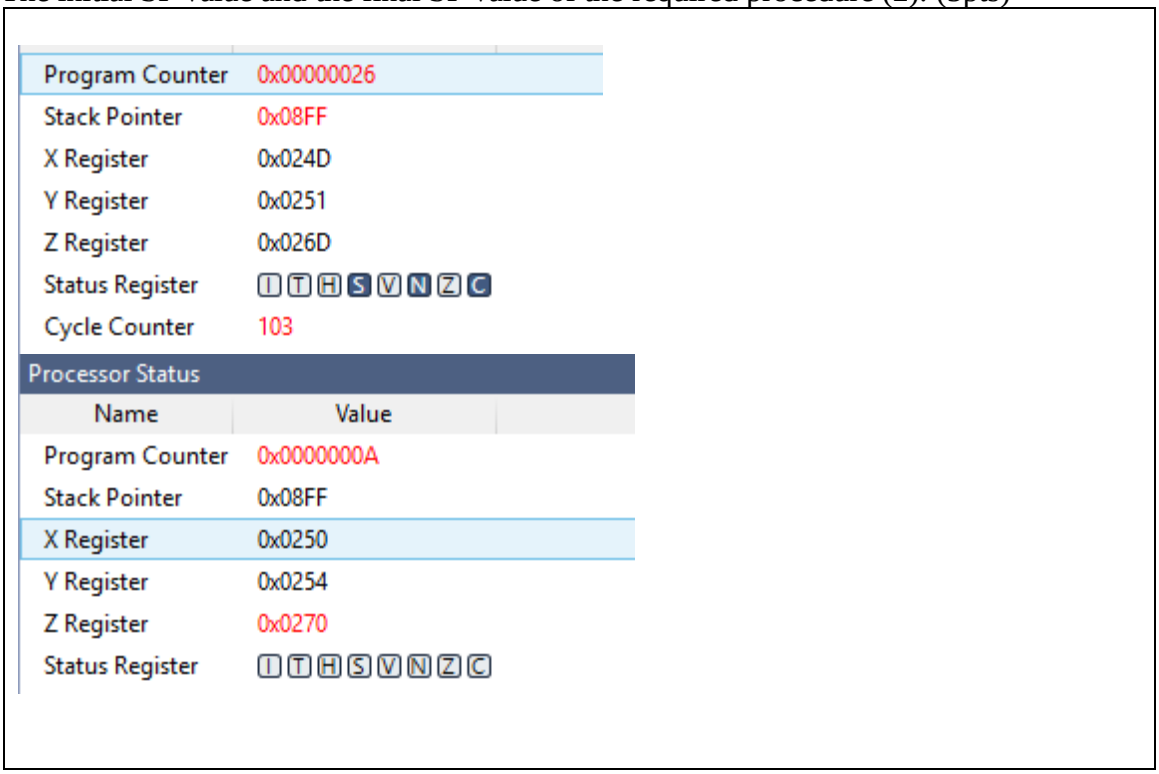
(1) Directives and their values in the watch view. (2pts)

Name	Value
Num_Set1	592
Num_Set2	596
OUT_LOG	624

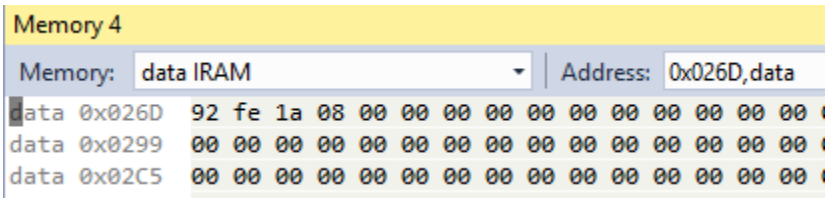
(2) The contents of the memory locations: from 0x024D to 0x0250 (NUM_SET1) and from 0x0251 to 0x0254 (NUM_SET2). (8pts)



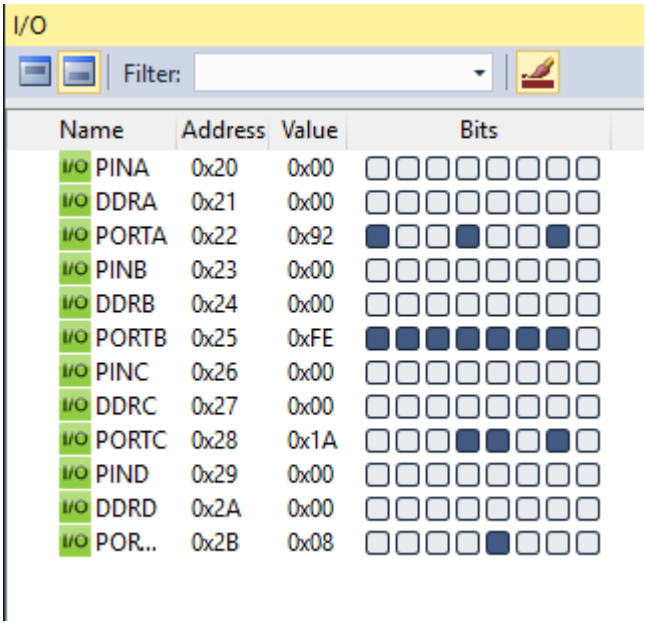
(3) The initial SP value and the final SP value of the required procedure (2). (5pts)



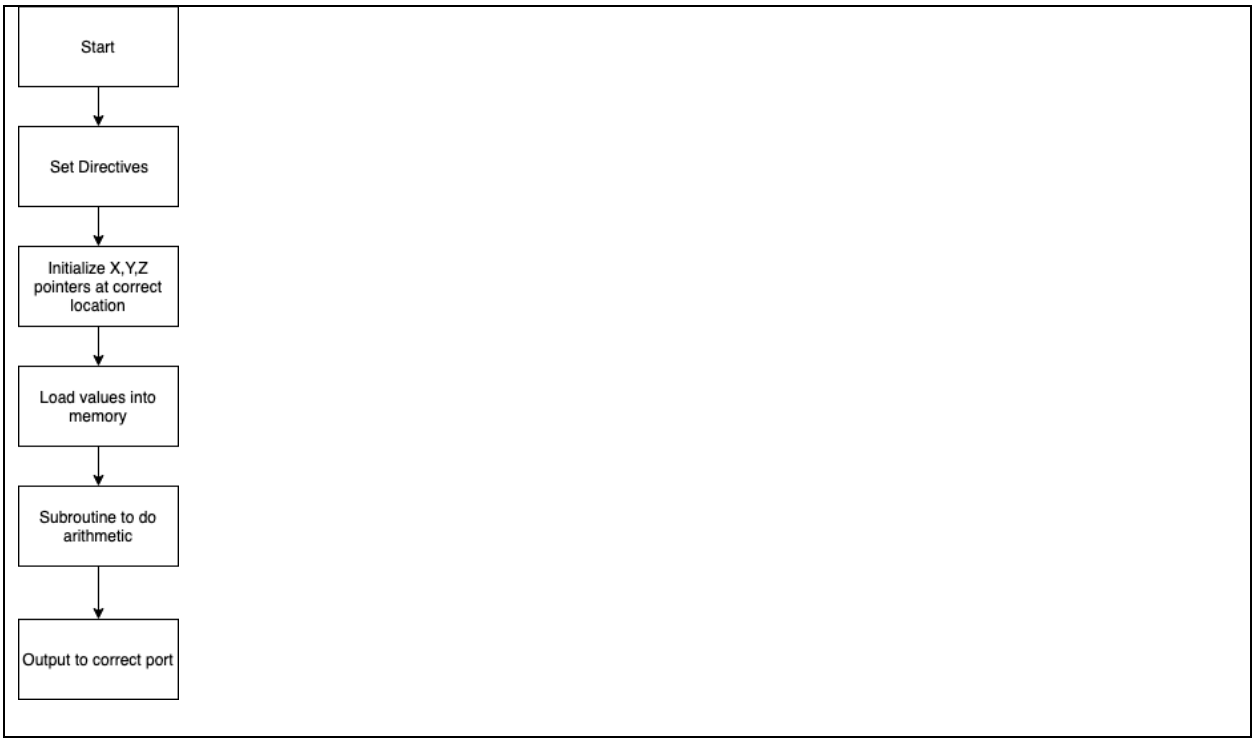
(4) The contents of the memory locations from 0x026D to 0x0270 (OUT_LOG). (5pts)



(5) The ports outputs: PORTA, PORTB, PORTC, and PORTD. (5pts)



Flowchart for activity 2 (10pts)



Code for activity 2 (copy and paste the entire code showing the activity)

NOTE: If you do not include your code, you will get '0' points for this activity.

```
start:
        .equ NUM_SET1 = 0x0250
        .equ NUM_SET2 = 0x0254
        .equ OUT_LOG  = 0x0270

        LDI R16,LOW(RAMEND)
        OUT SPL, R16
        LDI R17,HIGH(RAMEND)
        OUT SPH, R17

        LDI XL, LOW(NUM_SET1)
        LDI XH, HIGH(NUM_SET1)

        LDI YL, LOW(NUM_SET2)
        LDI YH, HIGH(NUM_SET2)

        LDI ZL, LOW(OUT_LOG)
        LDI ZH, HIGH(OUT_LOG)

        ; NUM_SET1
        LDI R16,0x19
```

```
        ST X,R16

        LDI R16,0x02
        ST -X,R16

        LDI R16,0xC5
        ST -X,R16

        LDI R16,0x66
        ST -X,R16

        ; NUM_SET2
        LDI R16,0x4A
        ST Y,R16

        LDI R16,0x18
        ST -Y,R16

        LDI R16,0x23
        ST -Y,R16

        LDI R16,0xF4
        ST -Y,R16

        LDI XL, LOW(NUM_SET1)
        LDI XH, HIGH(NUM_SET1)

        LDI YL, LOW(NUM_SET2)
        LDI YH, HIGH(NUM_SET2)

        CALL Num_and
        CALL Num_or
        CALL Num_nand
        CALL Num_exor

HERE: RJMP HERE

        Num_and:
        LD R16,X
        LD R17,Y
        AND R16,R17
        ST Z,R16
        OUT PORTD,R16
        RET
```

```
Num_or:
LD R16,-X
LD R17,-Y
OR R16,R17
ST -Z,R16
OUT PORTC,R16
RET

Num_nand:
LD R16,-X
LD R17,-Y
AND R16,R17
COM R16
ST -Z,R16
OUT PORTB,R16
RET

Num_exor:
LD R16,-X
LD R17,-Y
EOR R16,R17
ST -Z,R16
OUT PORTA,R16
RET
```

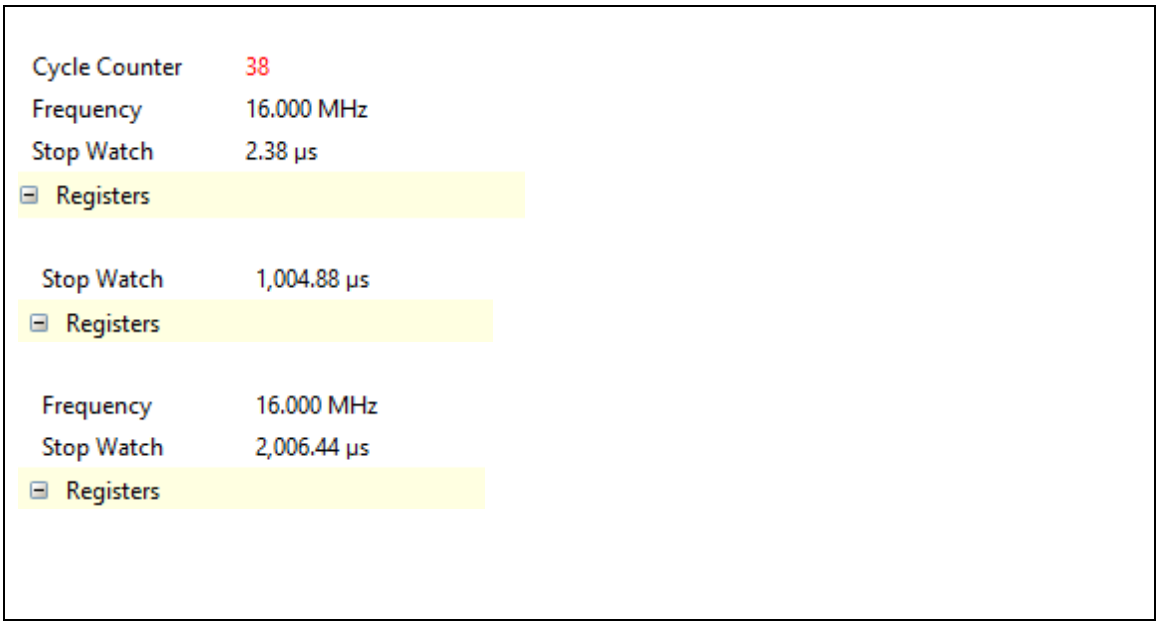
EXTRA Credit (20 points):

Use time delay to show the result of (6) using only PORTC with a time delay.

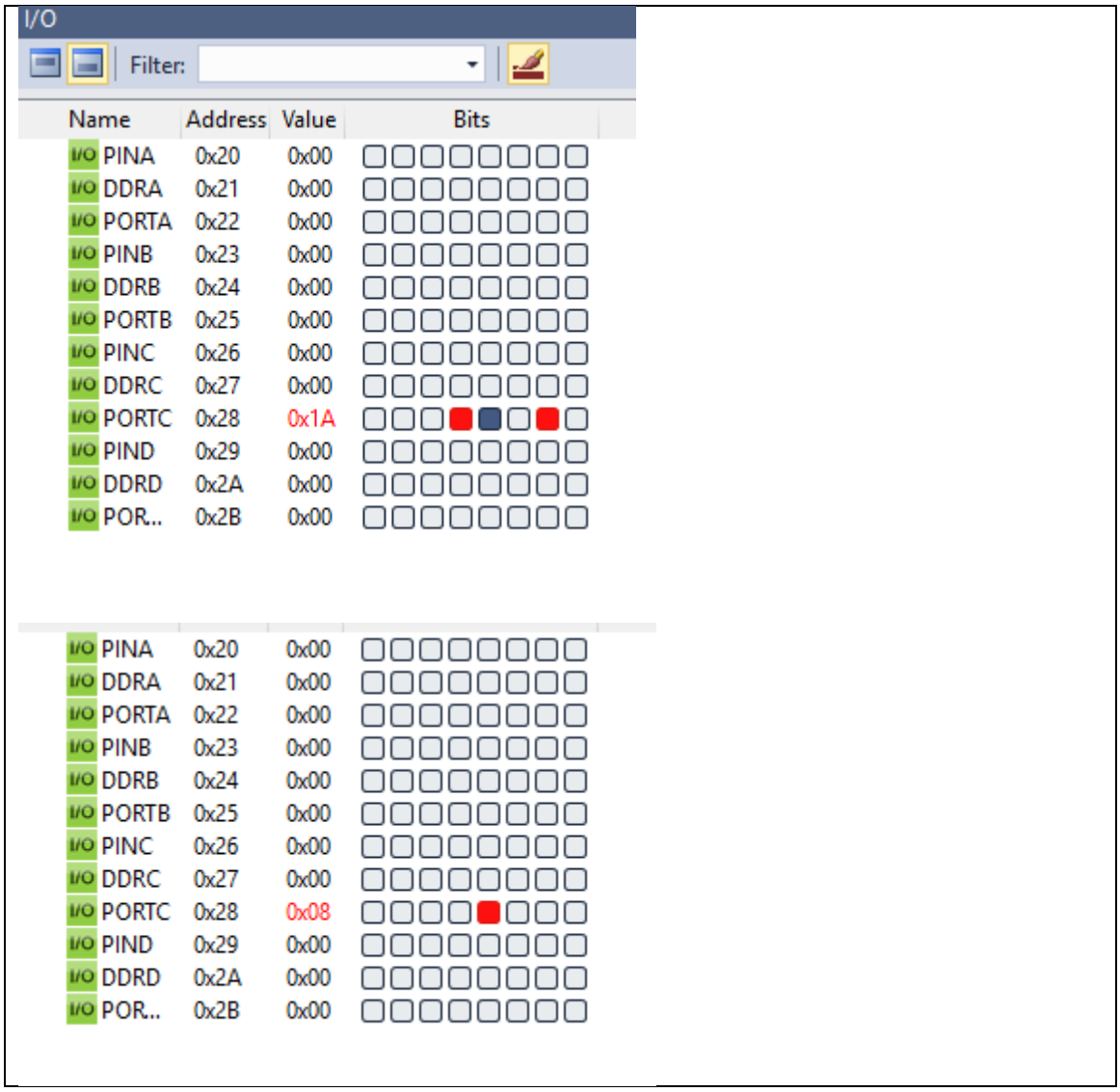
1. Show the time delay subroutine

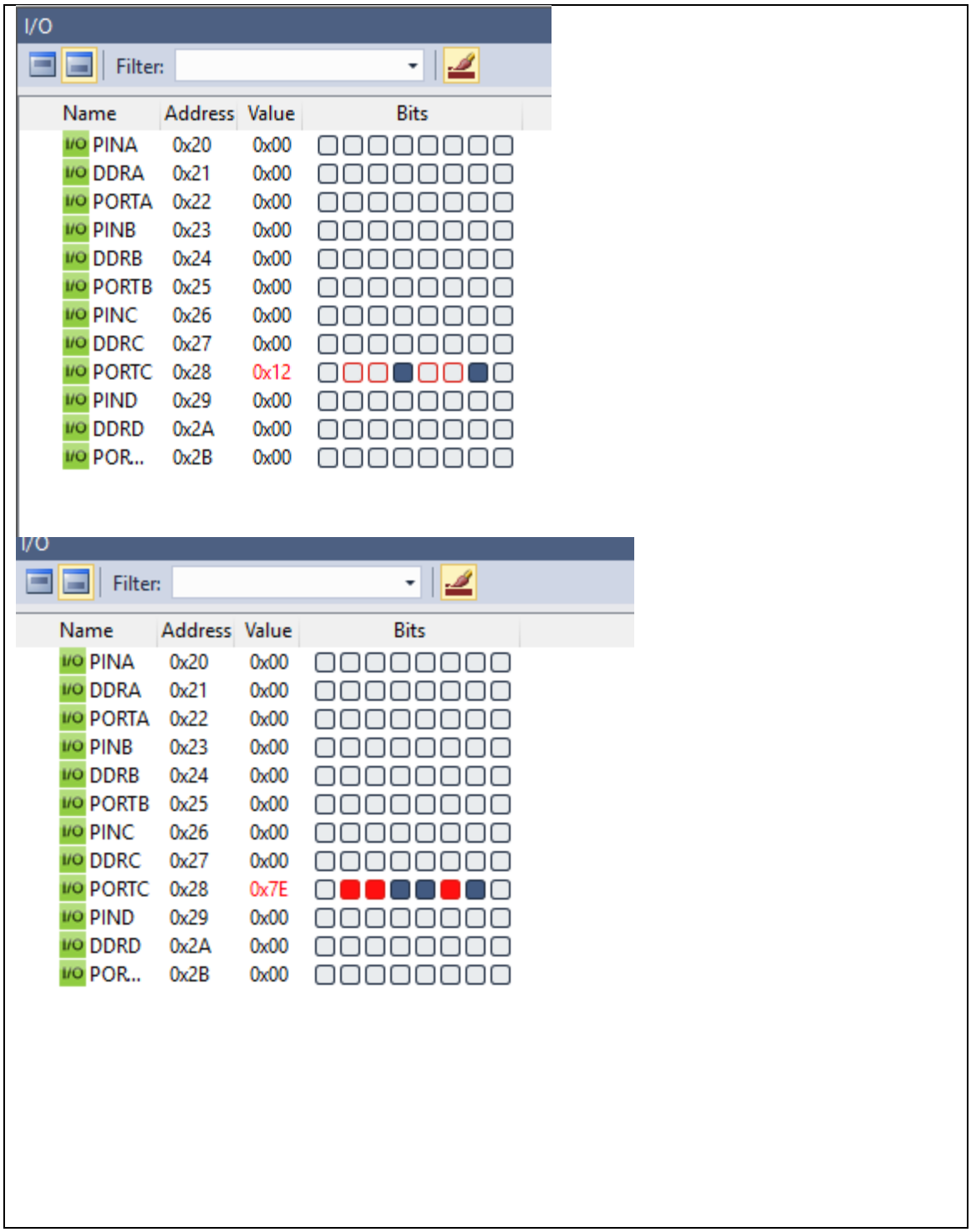
```
DELAY:
        ldi  r18, 21
        ldi  r19, 199
        L1: dec  r19
           brne L1
           dec  r18
           brne L1
         RET
```

2. Show the stopwatch value between one output and the next output



3. Show all outputs of PORTC


I/O
Filter:
Name
Address
Value
Bits
I/O PINA 0x20 0x00
I/O DDRA 0x21 0x00
I/O PORTA 0x22 0x00
I/O PINB 0x23 0x00
I/O DDRB 0x24 0x00
I/O PORTB 0x25 0x00
I/O PINC 0x26 0x00
I/O DDRC 0x27 0x00
I/O PORTC 0x28 0x1A
I/O PIND 0x29 0x00
I/O DDRD 0x2A 0x00
I/O POR... 0x2B 0x00
I/O PINA 0x20 0x00
I/O DDRA 0x21 0x00
I/O PORTA 0x22 0x00
I/O PINB 0x23 0x00
I/O DDRB 0x24 0x00
I/O PORTB 0x25 0x00
I/O PINC 0x26 0x00
I/O DDRC 0x27 0x00
I/O PORTC 0x28 0x08
I/O PIND 0x29 0x00
I/O DDRD 0x2A 0x00
I/O POR... 0x2B 0x00


I/O
Filter:
Name
Address
Value
Bits
I/O PINA 0x20 0x00
I/O DDRA 0x21 0x00
I/O PORTA 0x22 0x00
I/O PINB 0x23 0x00
I/O DDRB 0x24 0x00
I/O PORTB 0x25 0x00
I/O PINC 0x26 0x00
I/O DDRC 0x27 0x00
I/O PORTC 0x28 0x12
I/O PIND 0x29 0x00
I/O DDRD 0x2A 0x00
I/O POR... 0x2B 0x00
I/O
Filter:
Name
Address
Value
Bits
I/O PINA 0x20 0x00
I/O DDRA 0x21 0x00
I/O PORTA 0x22 0x00
I/O PINB 0x23 0x00
I/O DDRB 0x24 0x00
I/O PORTB 0x25 0x00
I/O PINC 0x26 0x00
I/O DDRC 0x27 0x00
I/O PORTC 0x28 0x7E
I/O PIND 0x29 0x00
I/O DDRD 0x2A 0x00
I/O POR... 0x2B 0x00

ECE3613 Processor System Laboratory Rubric
Lab #: 5
Section: 001 / 002
Name: _____

Activity	Section	Task	Full Points	Earned Points	Comment
1	1.1	Output 1	5		
		Output 2	10		
	Code section for Act.1.1		5		
Subtotal			20		
	1.2	Output 1	10		
		Output 2	10		
		Output 3	10		
Subtotal			30		
	Flowchart		10		
Activity 1 Total			60		
2	Code sections	Procedure (1)	5		
		Procedure (3)	5		
		Procedure (4)	5		
	Output	1	2		
		2	8		
		3	5		
		4	5		
		5	5		
	Flowchart		10		
Activity 2 Total			50		
Total			110		
Extra Credit			20		