Electrical and Computer Engineering
EE3612Microprocessor Systems Laboratory

Lab 5: Assembly Language Programming – Stack Operations and Subroutine

Fall 2019

OBJECTIVES:

- To learn how to use subroutines in AVR assembly program
- To learn how to use the stack pointer for the simple memory operations
- To investigate the functions of the Stack for calling subroutines

REFERENCES:

Mazidi and Naimi, “The AVR Microcontroller and Embedded Systems,” 2nd Ed. Chapters 1 and 2.

MATERIALS:

Atmel Studio 7

MAP OF THIS LAB:

- Activity 1: Stack operation and IO ports outputs
- Activity 2: Stack operation and Subroutine (CALL/RET)

LAB REPORT INSTRUCTIONS:

This lab consists of three activities. Use the given report frame to write your report and submit to Canvas assignment. **No full report is required.**

Submission Type: short assignment report in doc and pdf (use the given frame)
Due: section 01 – 10/08/2019 2:59pm, section 02 – 10/10/2019 8:59am

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).

Activity 1.1

Sore the given values in the memory locations using Stack Pointer (SP). (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:

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

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

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:

(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)

(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)

(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)

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	

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 40 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 1 (10 points).

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.

(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.

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

(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:

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

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

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

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

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

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
2. Show the stopwatch value between one output and the next output
3. Show all outputs of PORTC