LAB 3-A

STACK MEMORY

OBJECTIVES:

⍰ To examine the stack.

MATERIAL:

⍰ Atmel Studio.

WEB SITES:

⍰ www.microchip.com for Atmel Studio Software

ACTIVITY 1

Write and assemble a program to load values $20, $31, $42, $53, and $64 into each of registers R20 to R24 and then push each of these registers onto the stack. Single-step the program, and examine the stack and the SP register after the execution of each instruction.

```
.INCLUDE "m328pdef.inc" ; Include definition file for ATmega328P

.ORG 0x0000
  ; 1. Initialize Stack Pointer to RAMEND (Required for PUSH to work)
  LDI R16, LOW(RAMEND)
  OUT SPL, R16
  LDI R16, HIGH(RAMEND)
  OUT SPH, R16

  ; 2. Load values into registers
  LDI R20, 0x20
  LDI R21, 0x31
  LDI R22, 0x42
  LDI R23, 0x53
  LDI R24, 0x64

  ; 3. Push registers onto the stack
  PUSH R20        ; SP decrements by 1
  PUSH R21        ; SP decrements by 1
  PUSH R22
  PUSH R23
  PUSH R24

HERE: RJMP HERE    ; Infinite loop to end
```

Lab 3-A

Stack Memory

ACTIVITY 2

Write and assemble a program to:

a) Set SP = $29D,

b) Store (without using push operation) a different value 6, 5, 4, 3, 2 ,1 in RAM locations $29D, $29C, $29B, $29A, $299, and $298, respectively

c) POP each stack location into registers R20 – R24.

d) Use the simulator to single-step and examine the registers, the stack, and the stack pointer.

```
.INCLUDE "m328pdef.inc"

.ORG 0x0000
    ; 1. Set Stack Pointer (SP) = $029D
    LDI R16, 0x9D
    OUT SPL, R16
    LDI R16, 0x02
    OUT SPH, R16

    ; 2. Manually store values in RAM (Mimicking PUSH behavior)
    ; We use the X pointer (R27:R26) to address memory

    ; Store 6 at $029D
    LDI XL, 0x9D
    LDI XH, 0x02
    LDI R17, 6
    ST X, R17

    ; Store 5 at $029C
    LDI XL, 0x9C
    LDI R17, 5
    ST X, R17

    ; Store 4 at $029B
    LDI XL, 0x9B
    LDI R17, 4
    ST X, R17

    ; Store 3 at $029A
    LDI XL, 0x9A
    LDI R17, 3
    ST X, R17

    ; Store 2 at $0299
    LDI XL, 0x99
    LDI R17, 2
    ST X, R17

    ; Store 1 at $0298
    LDI XL, 0x98
    LDI R17, 1
    ST X, R17

    ; 3. Prepare SP for POPPING
    ; The stack grows downwards. We filled up to $298.
    ; To POP correctly using the hardware instruction, SP must point
    ; to the "Next Empty" slot below the data, which is $0297.
    LDI R16, 0x97
    OUT SPL, R16
    LDI R16, 0x02
    OUT SPH, R16

    ; 4. POP into registers R20 - R24
    ; POP increments SP, then reads.
    POP R20        ; Reads from $298 (Value: 1)
    POP R21        ; Reads from $299 (Value: 2)
    POP R22        ; Reads from $29A (Value: 3)
    POP R23        ; Reads from $29B (Value: 4)
    POP R24        ; Reads from $29C (Value: 5)
    ; Note: Value 6 (at $29D) remains on stack as we ran out of registers (R20-R24).

HERE: RJMP HERE
```

From Activity 1 and 2, answer the following questions:

1) Upon reset, what is the value in the SP register?

   RAMEND (For the ATmega328P, this is typically address $08FF).

2) Upon pushing data onto the stack, the SP register is ____decremented______ (decremented, incremented).

3) Upon popping data from the stack, the SP register is _____incremented______ (decremented, incremented).

4) Can you change the value of the SP register? If yes, explain why you would want to do that.

   Yes, you can change the value of the Stack Pointer (SP) register by writing to the SPL and SPH I/O registers using the OUT instruction. You would typically want to do this to initialize the stack at the beginning of a program, ensuring it points to the end of the SRAM (RAMEND) so that the stack has maximum space to grow downwards without overwriting other variables or data stored at lower addresses. Furthermore, you might change the SP to point to a specific memory location to manually manage memory allocation or test stack operations, as demonstrated in Lab 3 Activity 2 where the pointer is explicitly set to $29D