LAB 3-B

I/O PORTS

OBJECTIVES:

- To examine the I/O port operation using a simulator.
- To trace through a CALL subroutine using a simulator.

MATERIAL:

- Atmel Studio
- https://lcgamboa.github.io/js/picsimlab.html?../picsimlab_examples/ (Simulator)

WEB SITES:

- www.microchip.com for Atmel Studio Software

ACTIVITY 1

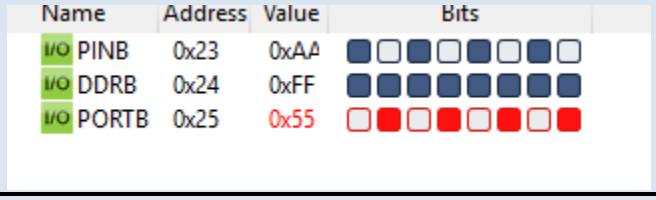
Write and assemble a program to toggle all the bits of PORTB, PORTC, and PORTD continuously by sending $55 and $AA to these ports. Put a time delay (between the "on" and "off" states. Then using the simulator, single-step through the program and examine the ports. **Do not single-step through the time delay call.**

```
.EQU DELAY_INNER = 100      ; วนลูปรอบใน
.EQU DELAY_OUTER = 255      ; วนลูปรอบนอก (แก้จาก 800 เป็น 255 เพื่อให้ไม่ Error)

RESET:
    ; 1. ตั้งค่า Stack Pointer (จำเป็นมากสำหรับการใช้ CALL)
    LDI R16, HIGH(RAMEND)
    OUT SPH, R16
    LDI R16, LOW(RAMEND)
    OUT SPL, R16

    ; 2. ตั้งค่า PORT B, C, D ให้เป็น Output (ส่งข้อมูลออก)
    LDI R16, 0xFF       ; โหลดค่า 11111111 (All Output)
    OUT DDRB, R16       ; ตั้งค่าทิศทาง PORTB
    OUT DDRC, R16       ; ตั้งค่าทิศทาง PORTC
    OUT DDRD, R16       ; ตั้งค่าทิศทาง PORTD

MAIN_LOOP:
    ; 3. ส่งค่า $55 (01010101) ไปที่พอร์ต - ไฟติดสลับดวง
    LDI R16, 0x55
    OUT PORTB, R16
    OUT PORTC, R16
```

LAB 3-B

I/O PORTS

```
    OUT PORTD, R16

    ; 4. เรียก Delay (หน่วงเวลา)
    CALL DELAY_SUB

    ; 5. ส่งค่า $AA (10101010) ไปที่พอร์ต - ไฟดิดสลับอีกแบบ
    LDI R16, 0xAA
    OUT PORTB, R16
    OUT PORTC, R16
    OUT PORTD, R16

    ; 6. เรียก Delay อีกครั้ง
    CALL DELAY_SUB

    RJMP MAIN_LOOP       ; วนกลับไปทำข้อ 3 ใหม่เรื่อยๆ

; Activity 2:
    LDI R18, DELAY_OUTER    ; โหลดค่ารอบนอก
L1:
    LDI R19, DELAY_INNER    ; โหลดค่ารอบใน
L2:
    NOP                     ; No Operation (ถ่วงเวลา 1 cycle)
    DEC R19                 ; ลดค่ารอบในลง 1
    BRNE L2                 ; ถ้ายังไม่เหลือ 0 ให้วนกลับไป L2

    DEC R18                 ; ลดค่ารอบนอกลง 1
    BRNE L1                 ; ถ้ายังไม่เหลือ 0 ให้วนกลับไป L1
    RET                     ; กลับไปทำงานต่อที่ Main Loop
```



ACTIVITY 2

Examine the registers of the delay subroutine and make the delay shorter or longer by changing the DELAY_INNER or DELAY_OUTTER value.

```
.EQU DELAY_INNER = 10
.EQU DELAY_OUTER = 70
```

it flickering faster than previous delay

LAB 3-B

I/O PORTS

ACTIVITY 3

Using a simulator, write a program to get a byte of data from PORTD (Change the value of PORTD during debugging when getting data from it) and send it to PORTB. Also, give a copy of it to registers R20, R21, and R22. Single-step the program and examine the ports and registers.

a) Upon reset, all the ports of the AVR are configured as ___**Input**__ (input, output).

b) To make all the bits of a port an input port we must write __$00_ hex to DDRx.

c) Write a program to monitor port B.0 continuously. When it becomes low, it sends $55 to PORTB.

```
MONITOR_LOOP:
SBIC PINB, 0       ; Skip next instruction if Bit = 0
RJMP MONITOR_LOOP ;keep looping  If Bit = 1

LDI R16, 0x55
OUT PORTB, R16
```

LAB 3-B

I/O PORTS

ACTIVITY 4

Test the AVR's ports by using picsimlab for input operation as follows.

a) Connect the pins of PORTx.4-PORTx.7 (PORTD for example) of the AVR to DIP switches. Also connect the pins of PORTy.4-PORTy.7 (e.g. PORTB) to LEDs.

b) Then, write and run a program to get data from PORTx.4-PORTx.7 and send it to PORTy.4-PORTy.7, respectively. Any change of status of the switches connected to PORTx will be instantly reflected on LEDs which are connected to PORTy.

<u>Note:</u> The main program functions must be in the infinite loop to keep the controller working

```
RESET:
    ;PORTB
    LDI R16, 0xFF   ; Set all pins on Port B to Output
    OUT DDRB, R16

    ;PORTD
    LDI R16, 0x00   ; Set all pins on Port D to Input
    OUT DDRD, R16

    ; Enable PORTD
    LDI R16, 0xFF
    OUT PORTD, R16

; LOOP
MAIN_LOOP:

    IN R16, PIND
    OUT PORTB, R16

    RJMP MAIN_LOOP
```

Name	Address	Value	Bits
I/O PIND	0x29	0x1E	▢▢▢■■■■▢
I/O DDRD	0x2A	0x00	▢▢▢▢▢▢▢▢
I/O POR..	0x2B	0xFF	■■■■■■■■

Name	Address	Value	Bits
I/O PINB	0x23	0x1E	▢▢▢■■■■▢
I/O DDRB	0x24	0xFF	■■■■■■■■
I/O PORTB	0x25	0x1E	▢▢▢■■■■▢

The program continuously reads the DIP switch states from the PIND register and writes them to the PORTB data register . This infinite loop ensures that any change in the switches (bits 4-7) is instantly reflected on the LEDs connected to PORTB.