LAB 1-1
I/O INTERFACE AND CALCULATION INSTRUCTIONS

Name: **LƯƠNG THÀNH VỸ (2151280)**
TRẦN TRUNG TOÀN (2151265)

OBJECTIVE:

- Implement I/O Port interface and calculation instructions.

REFERENCES:

- Lab Manual Chapter 1-2

EXPERIMENT 1:

a) Connect one AVR port (e.g., PORT A) to a dip switch. Connect another port to a LED bar (e.g., PORT B).

a) Write a program to continuously read the state of the dip switch and send it to the LED. If the switch is in the OFF state, the corresponding LED will turn off.

EXPERIMENT 2:

a) Write a program to read the value of the port connected to the dip switch, add 5 to it, and send it to the port connected to the LED bar.

b) Change the state of the dip switch and observe the status of the LED bar.

EXPERIMENT 3:

a) Connect and implement a program to calculate the product of two nibbles (high and low) of PORT A and send it to PORT B. Consider these two nibbles as unsigned numbers.

Example: PORT A = 0b0111_1111, then PORT B = 3 * 15.

b) Change the state of the dip switch and observe the status of the LED bar.

EXPERIMENT 4:

a) Connect and implement a program to calculate the product of two nibbles (high and low) of PORT A and send it to PORT B. Consider these two nibbles as signed numbers.

Example: PORT A = 0b0111_1111, then PORT B = 3 * (-1).

b) Change the state of the dip switch and observe the status of the LED bar.

LAB 1-1
I/O INTERFACE AND CALCULATION INSTRUCTIONS

EXPERIMENT 5:

c) Connect PA0 to a single switch and PA1 to a single LED on the LED block (note that they are from the same port).

d) Write a program to turn on the LED if the switch is pressed and turn it off if the switch is released.

LAB REPORT

Class group: *Group:*
Subject:

EXPERIMENT 1

1. Answer the following questions:

a. How do you retrieve values from the two nibbles of PORT A?

Answer:

The state of each pin or bit in PORTA can be read from the PINA register. To retrieve the values of the two nibbles from PORTA, we can simply load the content of PINA into a register (e.g., R17) with the following instruction:

```
IN    R17,  PINA
```

To manipulate the individual nibbles of PORTA, we can perform a bitwise AND operation on the lower nibble by masking the first four bits of PINA with a logic 1 mask (PINA & 0b00001111). For the upper nibble, we use the SWAP instruction to exchange the lower and upper nibbles, then perform the same bitwise AND operation to isolate the lower nibble (which is now the higher nibble after the swap).

- Exmaple code to retrieve lower nibble to R17:

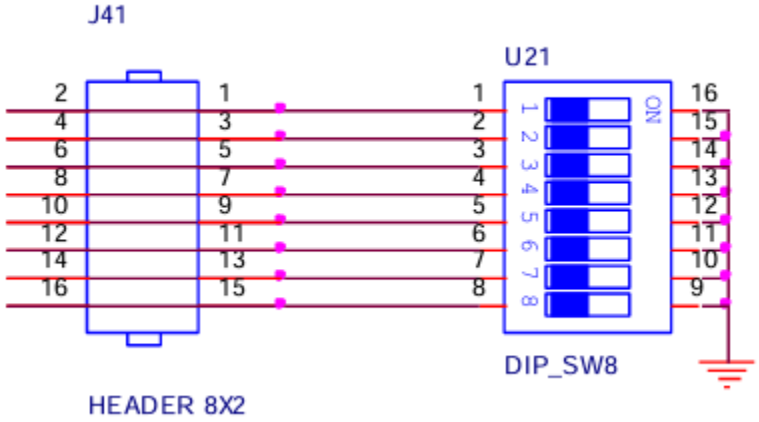
```
IN      R17, PINA
ANDI  R17,  0x0F
```

- Exmaple code to retrieve upper nibble R17:

```
IN      R17, PINA
SWAP  R17
ANDI  R17,  0x0F
```

b. When the switch is in the ON/OFF state, what is the pin value of the port?

Answer:



The DIP switch on the experiment kit connects the header (PORT pins) to ground. When a switch on the 8-bit DIP switch is turned ON, the corresponding GPIO pin is connected to ground, causing the value of the PORT on the AVR chip to be LOW (logic 0). When the switch is OFF, the PORT pin is disconnected from ground, and due to the pull-up resistors, the value of the PORT becomes HIGH (logic 1).

c. When the port pin is in state 1, is the BAR LED on or off?

Answer:

The objective of this experiment is to program the AVR so that when a switch is in the OFF state, the corresponding LED will turn off.

Since the PORT pins are HIGH (logic 1) when the switch is OFF, the LED will be turned off when the PORT pin is in a HIGH state.

⇨ In other words, the state of the LED bar is the inverse of the values on the PORT pins. Therefore, to control the LED bar, we can simply assign the bitwise complement (inverted values) of PINA to the LED bar.

d. Source code with comments.

Answer:

```
; DIP switch OFF => Input HIGH  => LED off
; DIP switch ON  => Input LOW   => LED on
; PORTA = Switch  (Input)
; PORTB = LED     (Output)
```

```
.ORG 00
        LDI     R16,   0x00      ; Config PORTA as Input   (DIP Switch)
        OUT     DDRA,  R16
        LDI     R16,   0xFF      ; Config PORTB as Output  (LED bar)
        OUT     PORTA,  R16       ; Enable Pull up resistors
        OUT     DDRB,  R16

        LDI     R16,   0x00      ; Clear PORTB
        OUT     PORTB, R16

LOOP:   IN      R17,   PINA      ; R17 = PORTA
        COM     R17              ; State of LED bar is inverse of PORT pins
        OUT     PORTB, R17       ; Assign COM(PORTA) to LED bar
        RJMP    LOOP
```

RESULT ON EXPERIMENT KIT:

https://drive.google.com/file/d/1U7VcOFA8pdpcvtlTyJk3xYiAQXhCBUl4/view?usp=sharing

EXPERIMENT 2

1. Answer the following questions:

a. Source code with comments.

Answer:

In this program, we add 5 to the binary value read from the DIP switch. Because the switch is active-low, we need to complement the value from the switch before adding 5.

```
; DIP switch OFF => Input HIGH  => LED off
; DIP switch ON  => Input LOW   => LED on
; PORTA = Switch  (Input)
; PORTB = LED     (Output)

.ORG 00
            LDI     R16,   0x00      ; Config PORTA as Input  (DIP switch)
            OUT     DDRA,  R16
            LDI     R16,   0xFF      ; Config PORTB as Output (LED bar)
            OUT     PORTA, R16       ; Enable Pull up resistors
            OUT     DDRB,  R16

            LDI     R16,   0x00      ; Clear PORTB
            OUT     PORTB, R16

            LDI     R20, 0x05        ; R20 = 5, for adding 5 to PORT pins
```

```
LOOP:   IN      R17,   PINA     ; R17 = PORTA
        COM     R17             ; State LED bar is inverse of PORT pins
        ADD     R17,   R20      ; Add 5 to the value of PORT pins
        OUT     PORTB, R17      ; Send the value to LED bar
        RJMP    LOOP
```

RESULT ON EXPERIMENT KIT:

https://drive.google.com/file/d/1hg2IkBo6TJCpmZEKUf16UxngvKhIk0ZV/view?usp=sharing

EXPERIMENT 3

1. Answer the following questions:

a. How do you retrieve values from the two nibbles of PORT A?

Answer:

The state of each pin or bit in PORTA can be read from the PINA register. To retrieve the values of the two nibbles from PORTA, we can simply load the content of PINA into a register (e.g., R17) with the following instruction:

```
IN    R17,  PINA
```

To manipulate the individual nibbles of PORTA, we can perform a bitwise AND operation on the lower nibble by masking the first four bits of PINA with a logic 1 mask (PINA & 0b00001111). For the upper nibble, we use the SWAP instruction to exchange the lower and upper nibbles, then perform the same bitwise AND operation to isolate the lower nibble (which is now the higher nibble after the swap).

- Exmaple code to retrieve lower nibble to R17:

```
IN      R17, PINA
ANDI    R17,  0x0F
```

- Exmaple code to retrieve upper nibble R17:

```
IN      R17, PINA
```

```
SWAP  R17
ANDI  R17,  0x0F
```

b. Source code with comments.

Answer:

```
; DIP switch OFF => Input HIGH  => LED off
; DIP switch ON  => Input LOW   => LED on
; PORTA = Switch  (Input)
; PORTB = LED     (Output)
.ORG 00
            LDI   R16,   0x00      ; Config PORTA as Input   (DIP switch)
            OUT   DDRA,  R16
            LDI   R16,   0xFF      ; Config PORTB as Output  (LED bar)
            OUT   PORTA, R16       ; Enable Pull up resistors
            OUT   DDRB,  R16

            LDI   R16,   0x00      ; Clear PORTB
            OUT   PORTB, R16

LOOP:       IN    R17,   PINA
            COM   R17              ; State LED bar is inverse of PORT pins
            ANDI  R17,   0x0F      ; Mask the lower nibble of PORTA

            IN    R18,   PINA
            COM   R18              ; State LED bar is inverse of PORT pins
            SWAP  R18              ; SWAP the nibble
            ANDI  R18,   0X0F      ; Mask the higher (swapped) nibble of PORTA

                                   ; Multiply Upper Nibble with Lower Nibble
            MUL   R17,   R18       ; 4-bit X 4-bit = 8-bit, Result = {R1, R0}
            OUT   PORTB, R0        ; Only care about R0 (first 8-bit of the result)
            RJMP  LOOP
```

RESULT ON EXPERIMENT KIT:

https://drive.google.com/file/d/1YH_OvcP0VuOnUMhFSAR0iYIA1RbSb9Ae/view?usp=sharing

EXPERIMENT 4

1. Answer the following questions:

a. Source code with comments.

Answer:

For signed multiplication, we use the MULS instruction instead of MUL. The MULS instruction treats the operands (in this case, R17 and R18) as signed integers. This introduces a challenge when storing 4-bit nibbles in 8-bit registers, as we need to preserve the sign of the nibble when retrieving them.

We can preserve the sign by performing an arithmetic right shift, which shifts the bits while maintaining the sign bit.

- **For the lower nibble:** First, we swap or logically left shift the lower nibble to the upper 4 bits of the register so that the sign bit of the 8-bit value aligns with the sign bit of the 4-bit nibble. Afterward, we perform an arithmetic right shift to move the sign-extended value back into the lower 4 bits. This effectively sign-extends the 4-bit nibble into 8 bits for storing in the register.
- **For the upper nibble:** Since the sign bit of the 8-bit register is already aligned with the sign bit of the nibble, we can simply perform an arithmetic right shift to extend the sign and store the signed upper nibble in the lower 4 bits of the register.

```
; DIP switch OFF => Input HIGH  => LED off
; DIP switch ON  => Input LOW   => LED on
; PORTA = Switch  (Input)
; PORTB = LED     (Output)


.ORG 00
          LDI     R16,   0x00      ; Config PORTA as Input   (DIP switch)
          OUT     DDRA,  R16
          LDI     R16,   0xFF      ; Config PORTB as Output  (LED bar)
          OUT     PORTA, R16       ; Enable Pull up resistors
          OUT     DDRB,  R16

          LDI     R16,   0x00      ; Clear PORTB
          OUT     PORTB, R16

LOOP:     IN      R17,   PINA
          COM     R17              ; State LED bar is inverse of PORT pins
          ANDI    R17,   0x0F      ; Mask the lower nibble of PORTA
          SWAP    R17              ; Swap or shift left, so that sign bit is MSB
          ASR     R17              ; Arithmetic right shift 4 times
          ASR     R17
          ASR     R17
          ASR     R17

          IN      R18,   PINA
          COM     R18              ; State LED bar is inverse of PORT pins
```

```
        ANDI    R18,    0XF0        ; Mask the higher (swapped) nibble of PORTA
        ASR     R18                 ; The sign bit is already MSB
        ASR     R18                 ; Arithmetic right shift 4 times
        ASR     R18
        ASR     R18

                                    ; Multiply Upper Nibble with Lower Nibble
        MULS    R17,    R18         ; 4-bit X 4-bit = 8-bit, Result = {R1, R0}
        OUT     PORTB, R0           ; Only care about R0 (first 8-bit of the result)
        RJMP    LOOP
```

RESULT ON EXPERIMENT KIT:

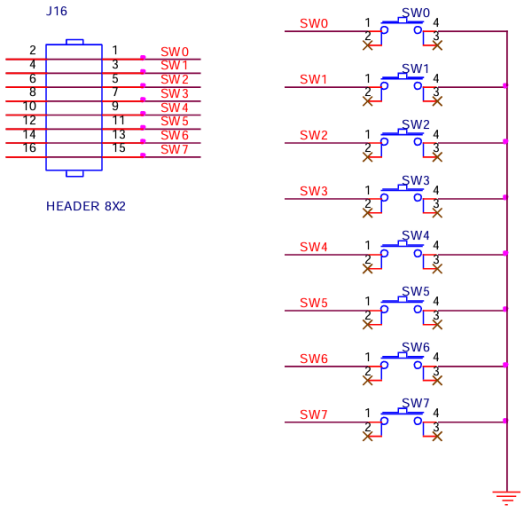
https://drive.google.com/file/d/1qK72hUB5mIdLUn6SdRs_YGVNEQnsVS4C/view?usp=sharing

EXPERIMENT 5

1. Answer the following questions:

a. When the switch is pressed/released, what is the pin value of the port?

Answer:

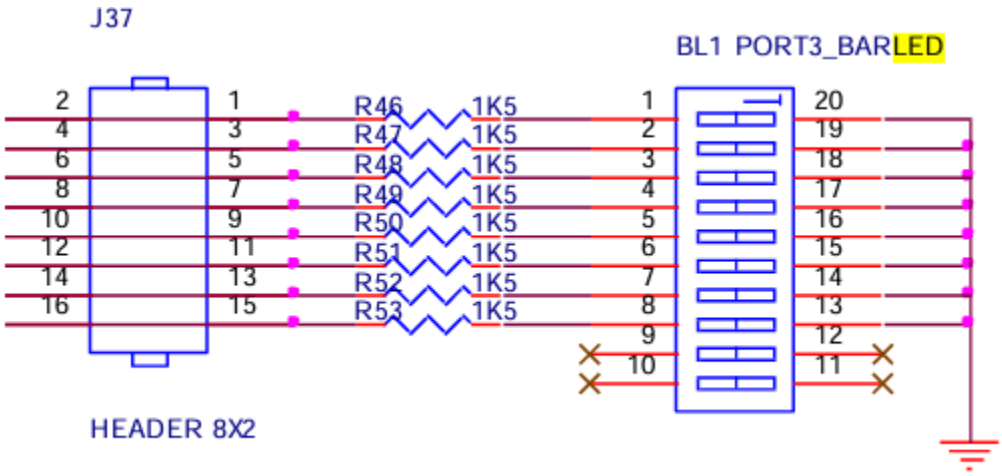


The push button on the experiment kit connects the PORT pin to ground when pressed, which results in the PORT pin being in a LOW state (state 0). To ensure the PORT pin is in a HIGH state (1) when the button is not pressed, pull-up resistors are required. These resistors drive the PORT pin to a HIGH state when the button is released. Therefore, the push button is an Active

Low button, pressing it pulls the pin LOW by connecting it to ground, and releasing it allows the pull-up resistor to set the PORT pin to HIGH.

b. To make the LED light up, what logic level should the port pin output?

Answer:



The LED bar on the experiment kit has one terminal connected to Ground (hard wired), while the other terminal is connected to the AVR's PORT through a 1.5K resistor. To light up the LED, a logic 1 must be output from the PORT pin, creating a voltage difference between the two terminals of the LED, 5V from the PORT pin and 0V from the Ground. To turn off the LED, a logic 0 is output from the PORT pin, which results in no voltage difference, as both the logic 0 and Ground are at 0V.

c. Source code with comments.

Answer:

This program ensures that the output PORT pin is asserted only once when a button press is detected or holding the button. The LED turns off only once the button is released.

```
; PA0 = button (active LOW)
; PA1 = LED    (one of the LED bar)
; Press and hold => LED turns ON
; Release        => LED turns OFF

.ORG 00
                CBI     DDRA,  0        ; Config PA0 as Input   (Switch)
                SBI     PORTA, R16      ; Enable Pull up resistors
```

```
                SBI     DDRA,  1        ; Config PA1 as Output  (LED)

                CBI     PORTA, 1        ; clear PA1

CHECK_PRESS:    SBIS    PINA,  0        ; Button is ACTIVE-LOW
                RJMP    LED_ON
                RJMP    CHECK_PRESS     ; Wait for the button to be pressed

CHECK_RELEASE:  SBIC    PINA,  0
                RJMP    LED_OFF         ; Looping while holding the button
                RJMP    CHECK_RELEASE   ; Wait for the button to be released

LED_ON:         SBI     PORTA, 1        ; Turn on LED then check for release
                RJMP    CHECK_RELEASE

LED_OFF:        CBI     PORTA, 1        ; Turn off LED then check for next press
                RJMP    CHECK_PRESS
```

RESULT ON EXPERIMENT KIT:

https://drive.google.com/file/d/1VH67t1ACGk8QdxwQtzpMDsEVu0puRXPy/view?usp=sharing