LAB 4-A

LOGIC INSTRUCTIONS AND BIT MANIPULATIONS

OBJECTIVES:

- To write a program to perform bit manipulations.

MATERIAL:

- Atmel Studio

WEB SITES:

- www.microchip.com for Atmel Studio Software

ACTIVITY 1

Write a program that copies bits 1, 2, and 3 of PIND to bits 0, 1, and 2 of port B.

```
.ORG $0000
    RJMP RESET

RESET:
    ;Configuration
    LDI R16, 0xFF
    OUT DDRB, R16    ; Set PORTB as Output
    LDI R16, 0x00
    OUT DDRD, R16    ; Set PORTD as Input

MAIN:
    ;  Read Input
    IN R16, PIND     ; Read value from PIND

    ; Masking (Keep only bits 1, 2, 3)
    ; Binary: 0000 1110 (Hex: 0x0E)
    ANDI R16, 0x0E

    ; Shifting (Move bits 1,2,3 -> 0,1,2)
    LSR R16          ; Logical Shift Right (divides by 2)

    ; Output
    OUT PORTB, R16   ; Send result to PORTB

    RJMP MAIN        ; Loop forever
```

LAB 4-A

LOGIC INSTRUCTIONS AND BIT MANIPULATIONS

ACTIVITY 2

Write a program that copies bits 1 and 6 of PINC to bits 0, and 2 of port B, respectively.

```
.ORG $0000
    RJMP RESET

RESET:
    ;Configuration
    LDI R16, 0xFF
    OUT DDRB, R16   ; Set PORTB as Output
    LDI R16, 0x00
    OUT DDRC, R16   ; Set PORTC as Input

MAIN:
    ; Read Inputs
    IN R16, PINC    ; Read all bits from PINC
    MOV R17, R16    ; Make a copy in R17 so we can work on Bit 6 separately

    ; Process Bit 1 -> Bit 0 (Using R16)
    ANDI R16, 0x02  ; Mask: Keep only Bit 1 (0000 0010)
    LSR R16         ; Shift Right 1 time -> Moves Bit 1 to Bit 0

    ; Process Bit 6 -> Bit 2 (Using R17)
    ANDI R17, 0x40  ; Mask: Keep only Bit 6 (0100 0000)
    LSR R17         ; Shift 1 (Bit 5)
    LSR R17         ; Shift 2 (Bit 4)
    LSR R17         ; Shift 3 (Bit 3)
    LSR R17         ; Shift 4 (Bit 2) -> Now Bit 6 is at Bit 2 position

    ; Combine and Output
    OR R16, R17     ; Combine the two results
    OUT PORTB, R16  ; Send to PORTB

    RJMP MAIN       ; Repeat
```

LAB 4-A

LOGIC INSTRUCTIONS AND BIT MANIPULATIONS

ACTIVITY 3

1) Write a program that inverts bit 3 of port C and sends it to bit 5 of port B.

```
.ORG $0000
    RJMP RESET

RESET:
    ;  Configuration
    LDI R16, 0xFF
    OUT DDRB, R16   ; Set PORTB as Output
    LDI R16, 0x00
    OUT DDRC, R16   ; Set PORTC as Input

MAIN:
    ;  Read Input
    IN R16, PINC    ; Read from PORTC

    ; Isolate Bit 3 (0000 1000 = 0x08)
    ANDI R16, 0x08  ; Keep only Bit 3

    ;  Invert Bit 3
    LDI R17, 0x08   ; Load mask for Bit 3
    EOR R16, R17    ; Toggle (Invert) Bit 3

    ;  Shift Left to Bit 5 position
    LSL R16         ; Shift 1 (Now at Bit 4)
    LSL R16         ; Shift 2 (Now at Bit 5)

    ;  Output Result
    OUT PORTB, R16  ; Send to PORTB

    RJMP MAIN       ; Repeat loop
```

LAB 4-A

LOGIC INSTRUCTIONS AND BIT MANIPULATIONS

2) Find the value in R16 after the following code.

```
LDI   R16, $45
ROR   R16
ROR   R16
ROR   R16
```

R16 = $48 in hex

Initial State

- **Instruction**: LDI R16, $45
- **Hex $45** in Binary: 0100 0101
- **Carry Flag (Initial)**: Assume $C = 0$ (default).

Step-by-Step Rotation

The ROR instruction shifts all bits one position to the right. The bit that "falls off" the right side (Bit 0) moves into the **Carry flag**, and the current value of the **Carry flag** moves into the left side (Bit 7).

1st ROR Operation

- **Before**: 0100 0101, $C = 0$
- **Process**: Bit 0 (1) moves to C. Current C (0) moves to Bit 7.
- **Result**: 0010 0010, $C = 1$ (Hex: **$22**)

2nd ROR Operation

- **Before**: 0010 0010, $C = 1$
- **Process**: Bit 0 (0) moves to C. Current C (1) moves to Bit 7.
- **Result**: 1001 0001, $C = 0$ (Hex: **$91**)

3rd ROR Operation

- **Before**: 1001 0001, $C = 0$
- **Process**: Bit 0 (1) moves to C. Current C (0) moves to Bit 7.
- **Result**: 0100 1000, $C = 1$ (Hex: **$48**)

LAB 4-A

LOGIC INSTRUCTIONS AND BIT MANIPULATIONS

3) Find the value in R16 after the following code.

```
LDI   R16, $45
ROL   R16
ROL   R16
ROL   R16
```

R16 = $29 in hex

Initial State

- **Hex Value**: $45
- **Binary**: 0100 0101
- **Carry Flag (C)**: Initialized as **0**.

Step-by-Step Rotation

In a ROL instruction, all bits shift left. The bit that "falls off" the left side (Bit 7) moves into the **Carry flag**, while the current value of the **Carry flag** moves into the right side (Bit 0).

1. First ROL Operation

- **Before**: 0100 0101, $C = 0$
- **Shift**: Bit 7 (0) moves to C. Current C (0) moves to Bit 0.
- **Result**: 1000 1010, $C = 0$ (Hex: **$8A**)

2. Second ROL Operation

- **Before**: 1000 1010, $C = 0$
- **Shift**: Bit 7 (1) moves to C. Current C (0) moves to Bit 0.
- **Result**: 0001 0100, $C = 1$ (Hex: **$14**)

3. Third ROL Operation

- **Before**: 0001 0100, $C = 1$
- **Shift**: Bit 7 (0) moves to C. Current C (1) moves to Bit 0.
- **Result**: 0010 1001, $C = 0$ (Hex: **$29**)

LAB 4-A

LOGIC INSTRUCTIONS AND BIT MANIPULATIONS

4) In the absence of the "SWAP Rn" instruction, how does the operation perform?

It would need to perform 4 Rotate operations (either ROL or ROR) to swap the nibbles. Alternatively, you could mask the upper nibble, shift it right 4 times, mask the lower nibble, shift it left 4 times, and then OR them together.

5) Can the SWAP instruction work on any register?

Yes, the SWAP instruction works on any of the 32 General Purpose Registers (**R0 to R31**).

6) Find the value in R2 after the following code.

```
CLR   R2
LDI   R21, $FF
EOR   R2, R21
```

R2 = $FF in hex

7) Find the value in A after the following code.

```
CLR   R10
COM   R10
LDI   R16, $AA
EOR   R10, R16
```

R10 = $55 in hex

Question 6: XOR Operation with R2

Goal: Find the value in **R2** after the EOR (Exclusive OR) operation.

- **Step 1: Clear R2**
  - CLR R2 sets all bits in the register to zero.
  - **R2** = 0000 0000 (Binary).
- **Step 2: Load R21 with $FF**
  - LDI R21, $FF loads the register with all ones.
  - **R21** = 1111 1111 (Binary).
- **Step 3: Perform EOR**

  - EOR R2, R21 compares the bits. In XOR, if the bits are different, the result is **1**. If they are the same, the result is **0**.
  - 0000 0000 (R2)
  - 1111 1111 (R21)
  - ---------
  - 1111 1111 (Result)
- **Final Result**: The binary 1111 1111 equals **$FF** in hex.

Question 7: Complement and XOR Operation

Goal: Find the value in **R10** after the COM and EOR operations.

- **Step 1: Clear and Complement R10**
  - CLR R10 sets the register to 0000 0000.
  - COM R10 performs a One's Complement, flipping all bits (0 to 1).
  - **R10** = 1111 1111.
- **Step 2: Load R16 with $AA**
  - LDI R16, $AA loads hex **$AA**.
  - **R16** = 1010 1010 (Binary).
- **Step 3: Perform EOR**
  - EOR R10, R16 performs the XOR comparison:
  - 1111 1111 (R10)
  - 1010 1010 (R16)
  - ---------
  - 0101 0101 (Result)
- **Final Result**: The binary 0101 0101 converts to **5** (left nibble) and **5** (right nibble), resulting in **$55** in hex.