LAB 3-C

COUNTER

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

Test the operation of an "up-counter" from $00 - $FF on picsimlab. Connect the LEDs to each pin of the PORTD and perform the following steps:

a) Assemble the following code:

```
;========Press your up-counter code here==========
RESET:
    ;Configure PORTD as Output
    LDI R16, 0xFF
    OUT DDRD, R16

    ;Initialize Counter Register
    LDI R16, 0x00

MAIN_LOOP:
    ; Output
    OUT PORTD, R16

    ; Call Delay
    CALL DELAY

    ; Increment Counter
    INC R16

    ; Loop back

    RJMP MAIN_LOOP
```

LAB 3-C

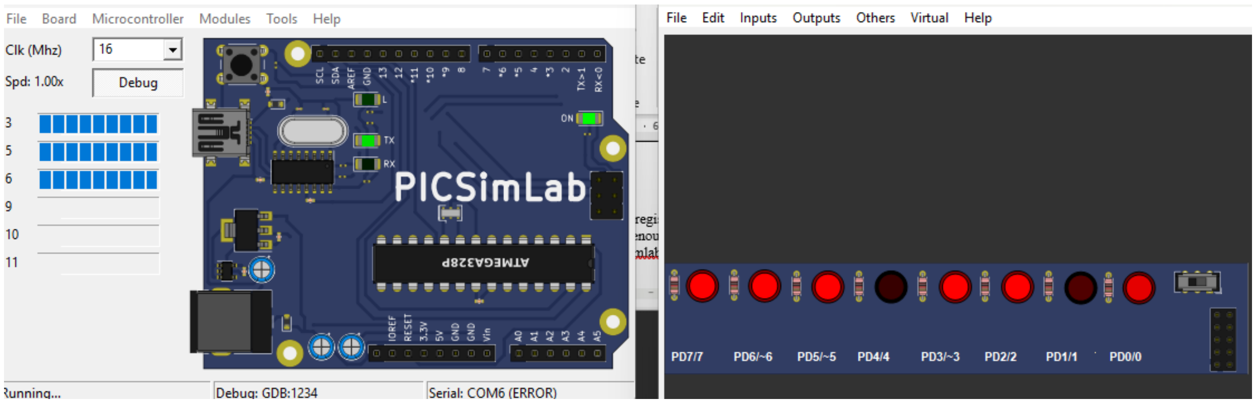
COUNTER

```
;==================================================
DELAY:  LDI     R21, 32
DL1:    LDI     R22, 200
DL2:    LDI     R23, 250
DL3:    NOP
        NOP
        DEC     R23
        BRNE    DL3
        DEC     R22
        BRNE    DL2
        DEC     R21
        BRNE    DL1
        RET
```

b) Upload the hex file to the picsimlab.

c) Observe the LEDs counting up from $00 to $FF.

d) Change the time delay in between the counts by changing the value of R21 register (or increase/decrease the number of NOPs) but make sure the time delay is long enough that you can observe the LEDs counting up. Create a hex file then upload to picsimlab to see what differences.



Running...
Debug: GDB:1234
Serial: COM6 (ERROR)

the LEDs on PORTD display a binary count where the lights appear to move to the left as the value increases from $00 toward $FF. Once the counter reaches the maximum 8-bit value of 255, it automatically rolls over and loops back to 0 to start again. By adjusting the R21 register in the delay subroutine, the time between counts changes, causing the LEDs to flicker and increment either faster or slower.

LAB 3-C

COUNTER

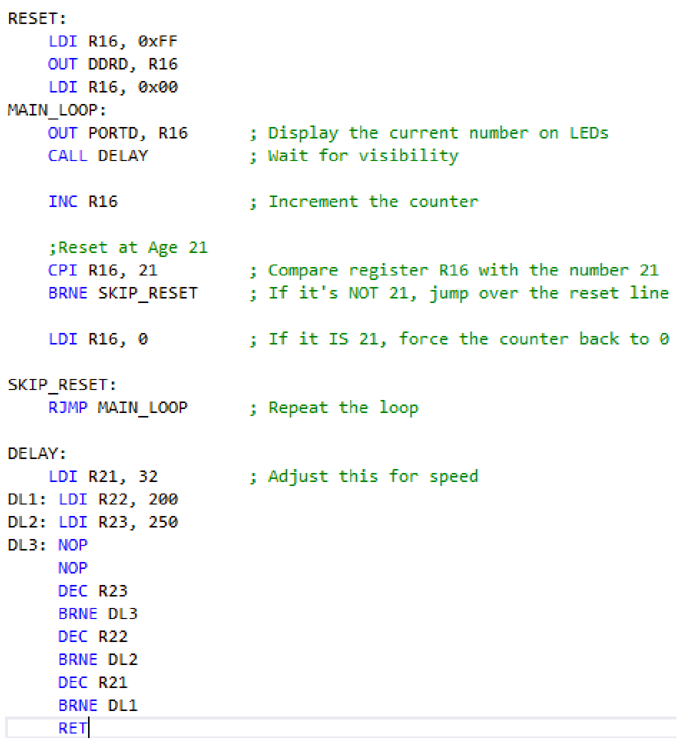
ACTIVITY 2

In Activity 1, the maximum count was $FF (or 255). Modify the above program to set maximum count to 10.

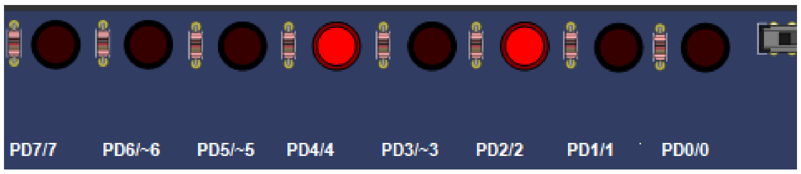
e) Upload the hex file into the AVR simulator.

f) Observe the LEDs counting up from $00 to $09 (00001001 binary) continuously.

g) Change the maximum count to the value of your age and observe the LED counting up to that number.



```
RESET:
    LDI R16, 0xFF
    OUT DDRD, R16
    LDI R16, 0x00
MAIN_LOOP:
    OUT PORTD, R16      ; Display the current number on LEDs
    CALL DELAY          ; Wait for visibility

    INC R16             ; Increment the counter

    ;Reset at Age 21
    CPI R16, 21         ; Compare register R16 with the number 21
    BRNE SKIP_RESET     ; If it's NOT 21, jump over the reset line

    LDI R16, 0          ; If it IS 21, force the counter back to 0

SKIP_RESET:
    RJMP MAIN_LOOP      ; Repeat the loop

DELAY:
    LDI R21, 32         ; Adjust this for speed
DL1: LDI R22, 200
DL2: LDI R23, 250
DL3: NOP
     NOP
     DEC R23
     BRNE DL3
     DEC R22
     BRNE DL2
     DEC R21
     BRNE DL1
     RET
```



the LEDs on PORTD display a binary count from 1 to 20. When the counter reaches your age of 21 ($00010101_2$), the LEDs at D4 (PD4), D6 (PD2), and D8 (PD0) light up briefly before the comparison logic triggers a reset to 0. Because the count is limited to 21, the "left" LEDs from D1 to D3 (PD7–PD5) remain off as the sequence never reaches those higher binary values.

LAB 3-C

COUNTER

ACTIVITY 3

For this activity, connect the DIP switches to pins of port B. Now, use the DIP switches on picsimlab to set the maximum count for an up-counter instead of using constant as Activity 1 and 2.

a) Upload the hex file into the AVR simulator.

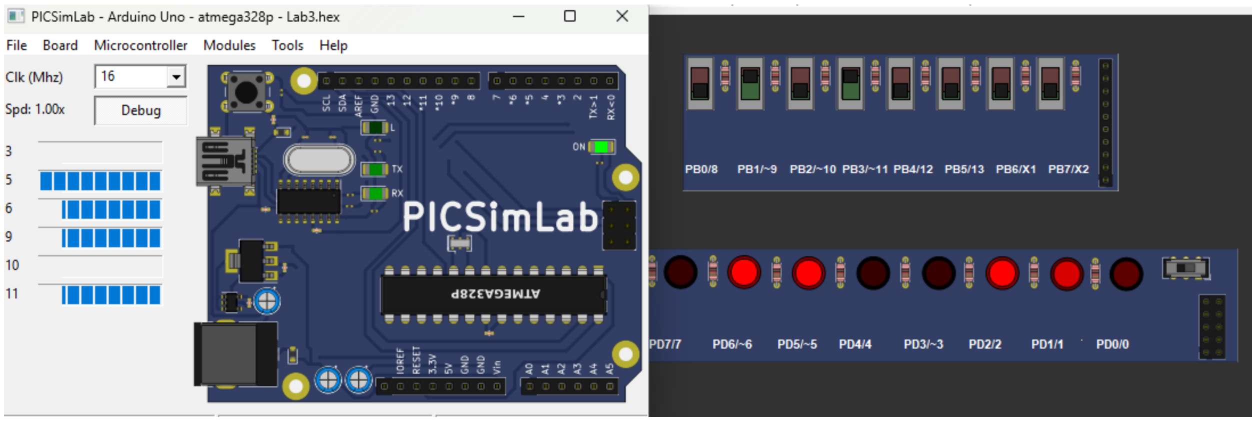
b) Observe the LEDs counting up from 00 to the value set by the 8 switches continuously.

Answer question

1) What is the maximum count for register R20?

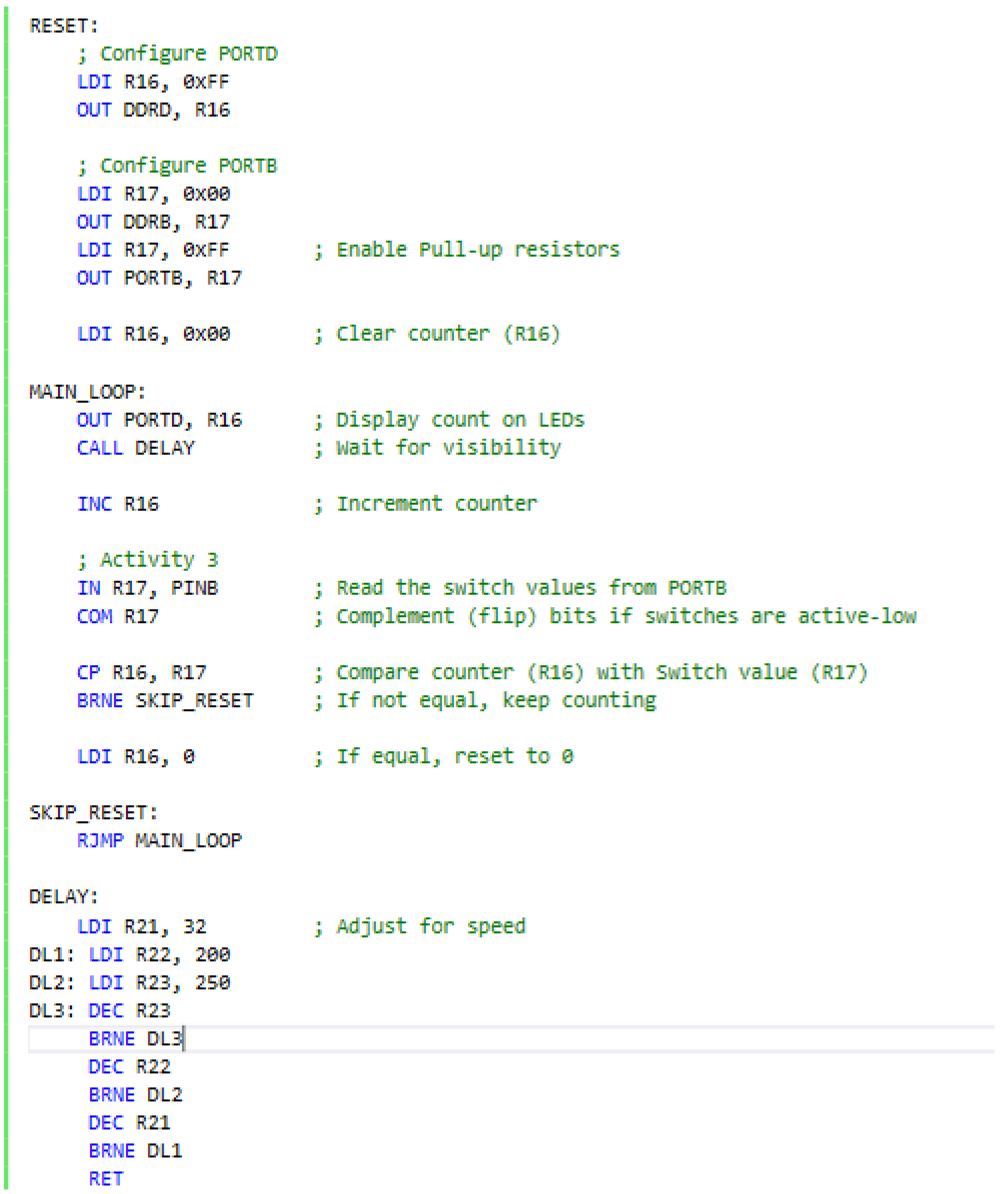
2) In this Lab, which port was used to display the count? Which one was used to set the maximum count? Can we use one port for both (inputting the maximum count and displaying the count)?

3) In this Lab, we used BRNE (Branch Not Equal) instruction. Explain how it works.


PICSimLab - Arduino Uno - atmega328p - Lab3.hex
File Board Microcontroller Modules Tools Help
Clk (Mhz) 16
Spd: 1.00x
Debug
3
5
6
9
10
11
PICSimLab
ATMEGA328P
PB0/8 PB1/~9 PB2/~10 PB3/~11 PB4/12 PB5/13 PB6/X1 PB7/X2
PD7/7 PD6/~6 PD5/~5 PD4/4 PD3/~3 PD2/2 PD1/1 PD0/0

Lab 3-C

Counter



```
RESET:
    ; Configure PORTD
    LDI R16, 0xFF
    OUT DDRD, R16

    ; Configure PORTB
    LDI R17, 0x00
    OUT DDRB, R17
    LDI R17, 0xFF       ; Enable Pull-up resistors
    OUT PORTB, R17

    LDI R16, 0x00       ; Clear counter (R16)

MAIN_LOOP:
    OUT PORTD, R16      ; Display count on LEDs
    CALL DELAY          ; Wait for visibility

    INC R16             ; Increment counter

    ; Activity 3
    IN R17, PINB        ; Read the switch values from PORTB
    COM R17             ; Complement (flip) bits if switches are active-low

    CP R16, R17         ; Compare counter (R16) with Switch value (R17)
    BRNE SKIP_RESET     ; If not equal, keep counting

    LDI R16, 0          ; If equal, reset to 0

SKIP_RESET:
    RJMP MAIN_LOOP

DELAY:
    LDI R21, 32         ; Adjust for speed
DL1: LDI R22, 200
DL2: LDI R23, 250
DL3: DEC R23
     BRNE DL3
     DEC R22
     BRNE DL2
     DEC R21
     BRNE DL1
     RET
```

the LEDs on **PORTD** display a dynamic binary count determined by the limit set on the **PORTB** switches. By reading the state of the switches with the IN and COM instructions, the program compares the current count in **R16** with the user-defined limit in **R17**. This allows the counter to reset to zero at any chosen value, providing a visual representation of how software logic interacts with physical input to control program flow.