# **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](http://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 ; Inner loop count

.equ DELAY\_OUTER = 800 ; Outer loop count

delay:

LDI R18, DELAY\_OUTER ; Load outer loop counter (800)

L1:

LDI R19, DELAY\_INNER ; Load inner loop counter (100)

L2:

NOP ; Each NOP takes 1 cycle

DEC R19 ; Decrement inner loop counter

BRNE L2 ; Branch if not zero, takes 2 cycles if branch taken, 1 cycle if not

DEC R18 ; Decrement outer loop counter

BRNE L1 ; Branch if not zero

RET ; Return from subroutine

# **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.

# **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.

1. Upon reset, all the ports of the AVR are configured as \_\_\_\_\_\_ (input, output).
2. To make all the bits of a port an input port we must write \_\_\_\_ hex to DDRx.
3. Write a program to monitor port B.0 continuously. When it becomes low, it sends $55 to PORTB.

# **ACTIVITY 4**

Test the AVR’s ports by using [picsimlab](https://lcgamboa.github.io/js/picsimlab.html?../picsimlab_examples/) for input operation as follows.

1. 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.
2. 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.

**Note: The main program functions must be in the infinite loop to keep the controller working**