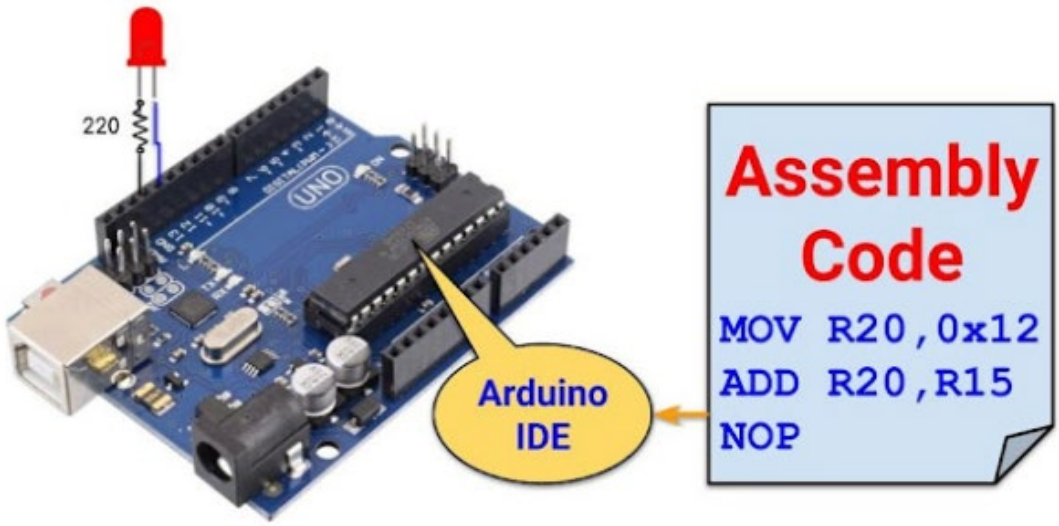
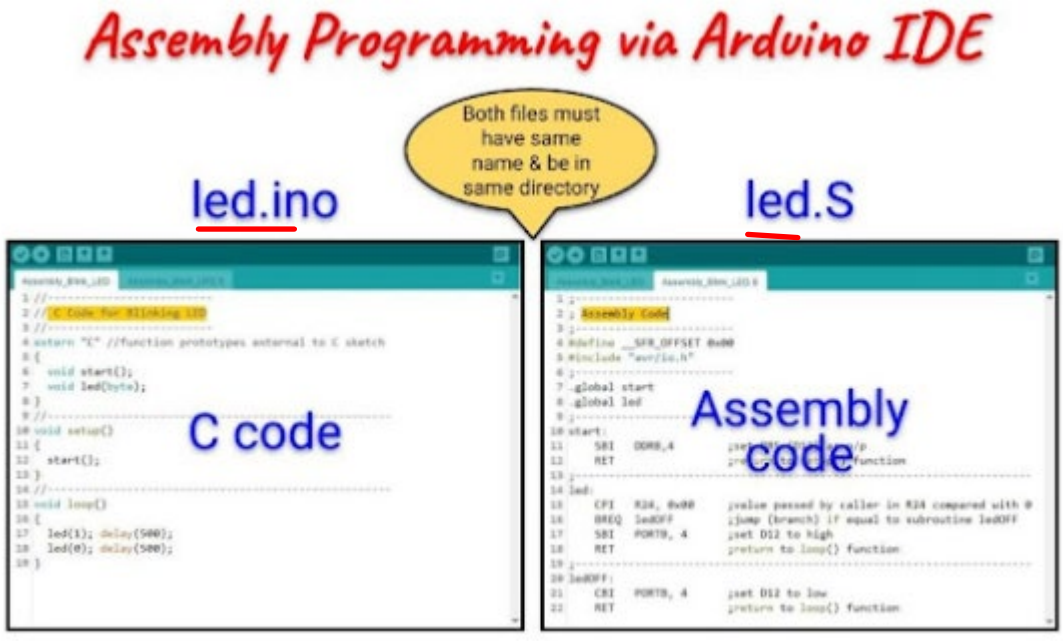
American International University- Bangladesh
Faculty of Engineering (EEE)
EEE 4103: Microprocessor and Embedded Systems Laboratory

Title: Familiarization of assembly language program in a microcontroller.

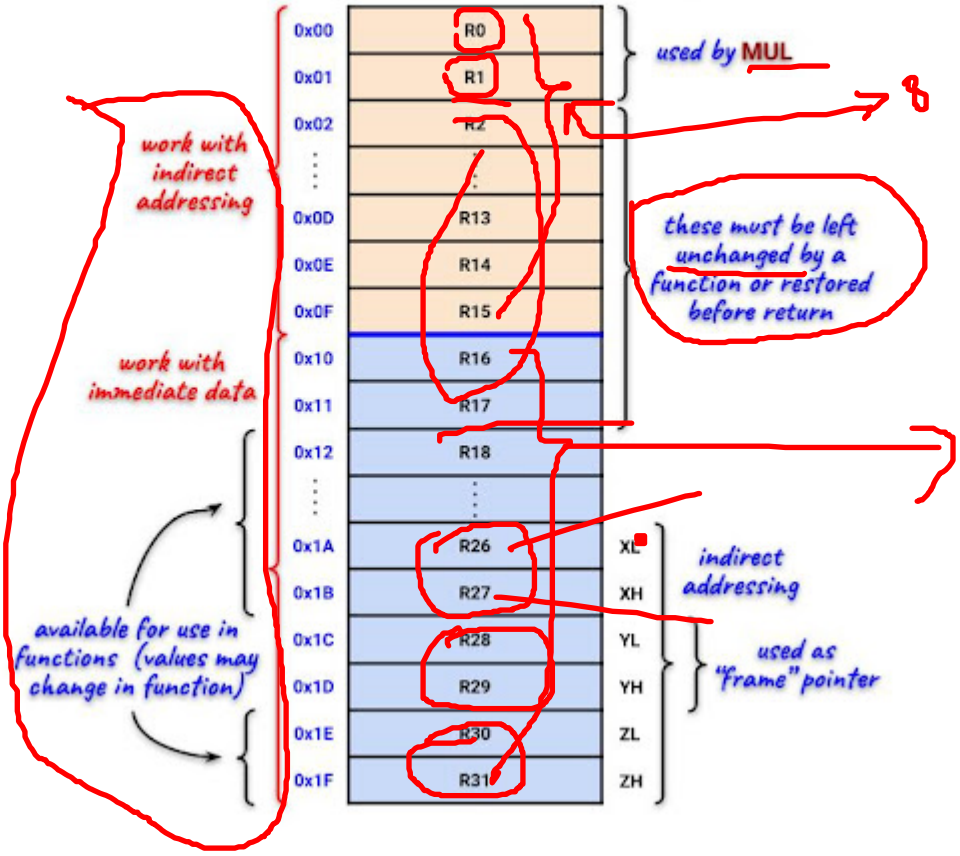
Introduction: In this experiment, the main objective is to learn how to write an assembly program for a blink LED program in a microcontroller.

Theory and Methodology: Assembly language programming using Arduino IDE.

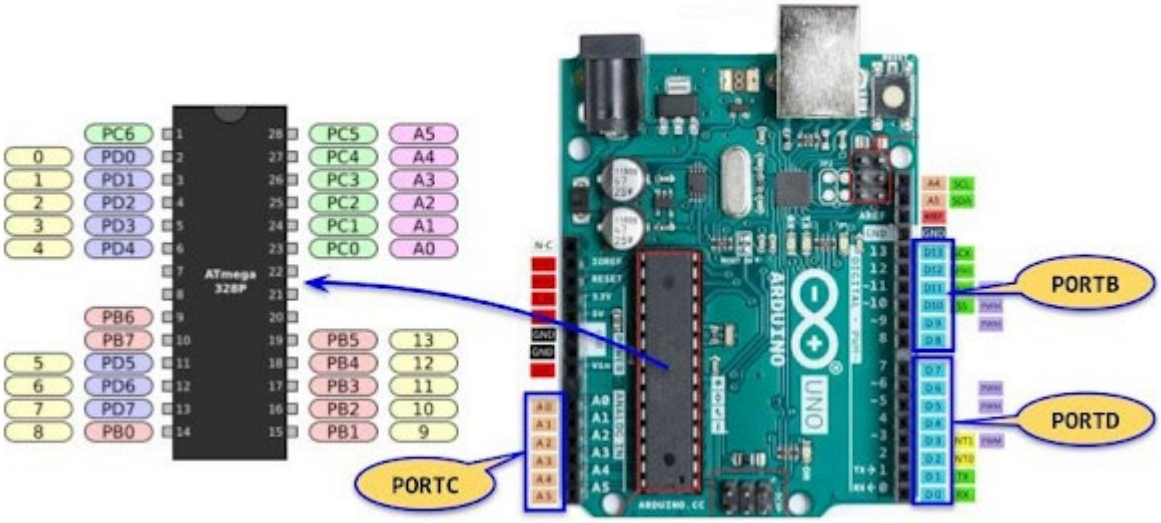


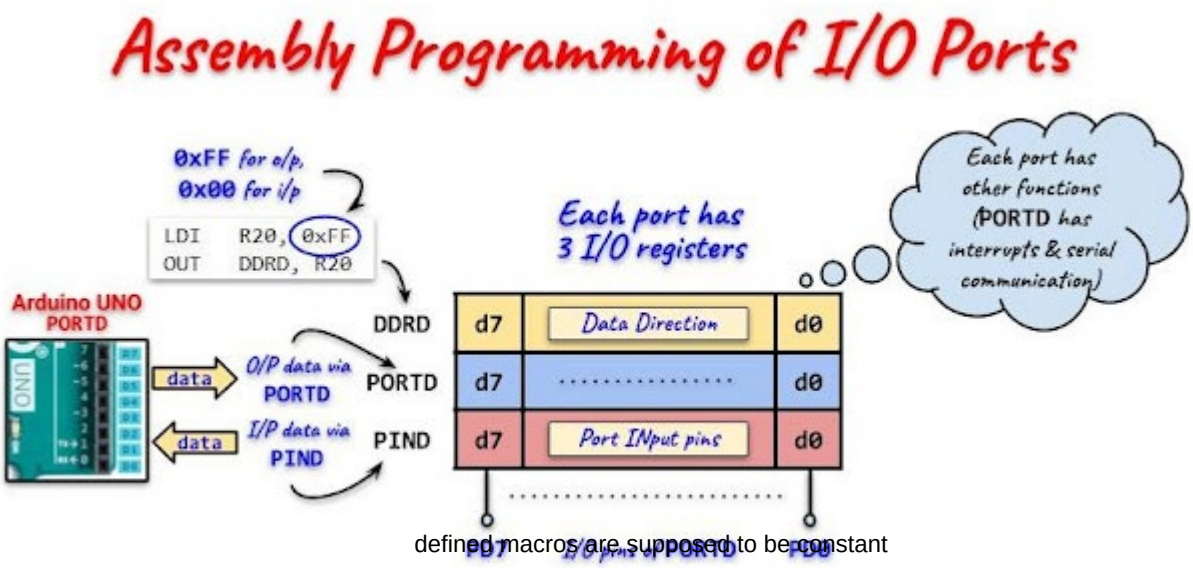


ATMega328P MCU Registers



Programming ATmega328 I/O Ports





defined macros are supposed to be constant

PART 1: Blink a LED

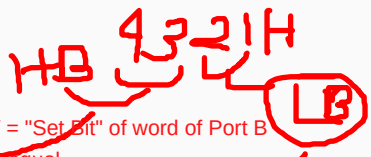
The .ino file:

```
//-------------------------
// C Code for Blinking LED
//-------------------------
extern "C" // Specifying that the function is defined elsewhere and uses the C-language calling convention.
{
  void start();
  void led(byte);
}
//-----------------------------------------------------
void setup()
{
 start();
}
//-----------------------------------------------------
void loop()
{
  led(1);
  led(0);
}
```

The .S file:

```
;--------------
; Assembly Code
;--------------
#define __SFR_OFFSET 0x00 // defined constant macro, requesting the assembler program to process a predefined sequence of instructions
#include "avr/io.h" // header file (.h) includes the appropriate I/O definitions for the device that has been specified
;----------------------
.global start // ".global" directive exports symbols in your code to where it points in the object code generated
```

```
.global led
;----------------------
start:
   SBI   DDRB, 5          ;set PB5 (D13) as o/p  ; SBI = "Set Bit" of Register B
   RET                    ;return to setup() function
;--------------------------------------------------------------------------
led:
   CPI   R24, 0x00        ;value in R24 passed by caller compared with 0 // CPI = "ComPare Immediate"
   BREQ  ledOFF           ;jump (branch) if equal to subroutine ledOFF // BREQ = "BRanch if EQual"
   SBI   PORTB, 5         ;set D13 to high // SBI = "Set Bit" of Port B
   RCALL myDelay  ; calling a subroutine and jumps to the target address / label ( "myDelay") // RCALL = Relative Call
   RET                    ;return to loop() function
;--------------------------------------------------------------------------
ledOFF:
   CBI   PORTB, 5         ;set D13 to low  // CBI = "Clear Bit" of Port B
   RCALL myDelay    ; calling a subroutine and jumps to the target address / label ( "myDelay") // RCALL = Relative Call
   RET                    ;return to loop() function
;--------------------------------------------------------------------------
.equ  delayVal, 10000     ;initial count value for inner loop The "equate" directive is used to substitute values for symbols or labels.
;--------------------------------------------------------------------------
myDelay:
   LDI   R20, 100          ;initial count value for outer loop // "LDI = Load Immediate"
outerLoop:
   LDI   R30, lo8(delayVal)   ;low byte of delayVal in R30
   LDI   R31, hi8(delayVal)  ;high byte of delayVal in R31
innerLoop:
   SBIW  R30, 1           ;subtract 1 from 16-bit value in R31, R30 // SBIW = "Set Bit" of word of Port B
   BRNE  innerLoop        ;jump if countVal not equal to 0 // "Branch if not equal
   ;-------------
   SUBI  R20, 1           ;subtract 1 from R20 // Subtracts the immediate data
   BRNE  outerLoop        ;jump if R20 not equal to 0
   RET
;--------------------------------------------------------------------------
```

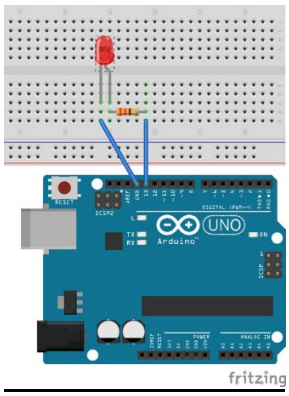


.

Equipment:

1) Arduino Uno
2) Arduino IDE
3) One Led
4) One 220 ohm resistor
5) PC having Intel Microprocessor

Experimental Setup:



Experimental procedure:

1) Create led.ino and led.S files using code given above.
2) Create a folder named led and place the above two files in the led folder.
3) Open led.ino using Arduino IDE.
4) Compile and upload to the hardware.
5) Modify the program to blink a led at digital PIN 12 with a different delay.

PART 2: Push button LED control

.ino file:

```
//-------------------------------------
// C Code: RGB LED ON/OFF via Buttons
//-------------------------------------
extern "C"
{
  void start();
  void btnLED();
}
//----------------------
void setup()
{
  start();
}
//----------------------
void loop()
{
  btnLED();
}
```

.S file:

```
;---------------------------------------
; Assembly Code: RGB LED ON/OFF via Buttons
;---------------------------------------
#define __SFR_OFFSET 0x00
#include "avr/io.h"
;----------------------
.global start
.global btnLED
;=============================================================================
start:
    SBI   DDRB, 4          ;set PB4 (pin D12 as o/p - red LED) ; SBI = "Set Bit" of Register B
    SBI   DDRB, 3          ;set PB3 (pin D11 as o/p - green LED)
    SBI   DDRB, 2          ;set PB2 (pin D10 as o/p - blue LED)
    CBI   DDRD, 2          ;clear PD2 (pin D02 as i/p - red button) ; CBI = "Clear Bit" of Register D
    CBI   DDRD, 3          ;clear PD3 (pin D03 as i/p - green button)
    CBI   DDRD, 4          ;clear PD4 (pin D04 as i/p - blue button)
    RET                    ; return to setup )
```

Questions for Report writing:

1. Include all codes printouts following lab report writing template.

```
;-----------------------------------------------------------------------------
btnLED:
    SBIC PIND, 2            ;skip next statement if red button not pressed
    RJMP redledON           ;jump to label redledON
    SBIC PIND, 3            ;skip next statement if green button not pressed
    RJMP greenledON         ;jump to label greenledON
    SBIC PIND, 4            ;skip next statement if green button not pressed
    RJMP blueledON          ;jump to label blueledON
    RJMP btnLED             ;return to label btnLED
;-----------------------------------------------------------------------------
redledON:
    LDI   R21, 10             ;initial value of counter R21
redagain:
    SBI   PORTB, 4            ;turn ON red LED
    RCALL myDelay             ;call subroutine myDelay
    CBI   PORTB, 4            ;turn OFF red LED
    RCALL myDelay             ;call subroutine myDelay
    SUBI  R21, 1              ;decrement counter by 1
    BRNE  redagain            ;loop if counter not zero
    RJMP btnLED               ;return to label btnLED
;-----------------------------------------------------------------------------
greenledON:
    LDI   R21, 10           ;initial counter R21 count value
greenagain:
    SBI   PORTB, 3          ;turn ON green LED
    RCALL myDelay           ;call subroutine myDelay
    CBI   PORTB, 3          ;turn OFF green LED
    RCALL myDelay           ;call subroutine myDelay
    SUBI  R21, 1            ;decrement counter by 1
    BRNE  greenagain        ;loop if counter not zero
    RJMP btnLED             ;return to label btnLED
;-----------------------------------------------------------------------------
blueledON:
    LDI   R21, 10           ;initial counter R21 count value
blueagain:
    SBI   PORTB, 2          ;turn ON blue LED
    RCALL myDelay           ;call subroutine myDelay
    CBI   PORTB, 2          ;turn OFF blue LED
    RCALL myDelay           ;call subroutine myDelay
    SUBI  R21, 1            ;decrement counter by 1
    BRNE  blueagain         ;loop if counter not zero
    RJMP btnLED             ;return to label btnLED
;-----------------------------------------------------------------------------
.equ  delayVal, 10000       ;equate delayVal with initial count value
;--------------------       ;for inner loop
myDelay:
    LDI   R20, 90           ;initial count value for outer loop
outerLoop:
    LDI   R30, lo8(delayVal)  ;low byte of delayVal in R30
    LDI   R31, hi8(delayVal)  ;high byte of delayVal in R31
innerLoop:
    SBIW  R30, 1            ;subtract 1 from 16-bit value in R31, R30
    BRNE  innerLoop         ;jump if countVal not equal to 0
    ;--------------
    SUBI  R20, 1            ;subtract 1 from R20
    BRNE  outerLoop         ;jump if R20 not equal to 0
;=============================================================================
```