# *EXPERIMENT-2 REPORT*

*KARNIKA SAMYUKTHA C U - EE24B114*

## AIM

This experiment introduces assembly programming and interaction with peripherals in Atmel Atmega8 microcontroller.

1. Wire the microcontroller along with the given peripherals in a breadboard, to make it work. For instance, after wiring the peripheral viz, the LED (an output device), write an AVR assembly program to blink an LED. Try another peripheral (input device), the push button and DIP switch.

2. Program the microcontroller to read the DIP switch values and display it in an LED using assembly programming.

3. Program the microcontroller to perform the addition and multiplication of two four-bit numbers which are read from the DIP switches connected to a port and display the result using LED's connected to another port.

## APPARATUS REQUIRED

1. Atmel AVR (Atmel8L) Chip - 1

2. A breadboard with microprocessor socket

3. 8-bit DIP switches

4. 5 LEDs

5. Capacitors, resistors and wires

6. AVR Programmer (USB-ASP)

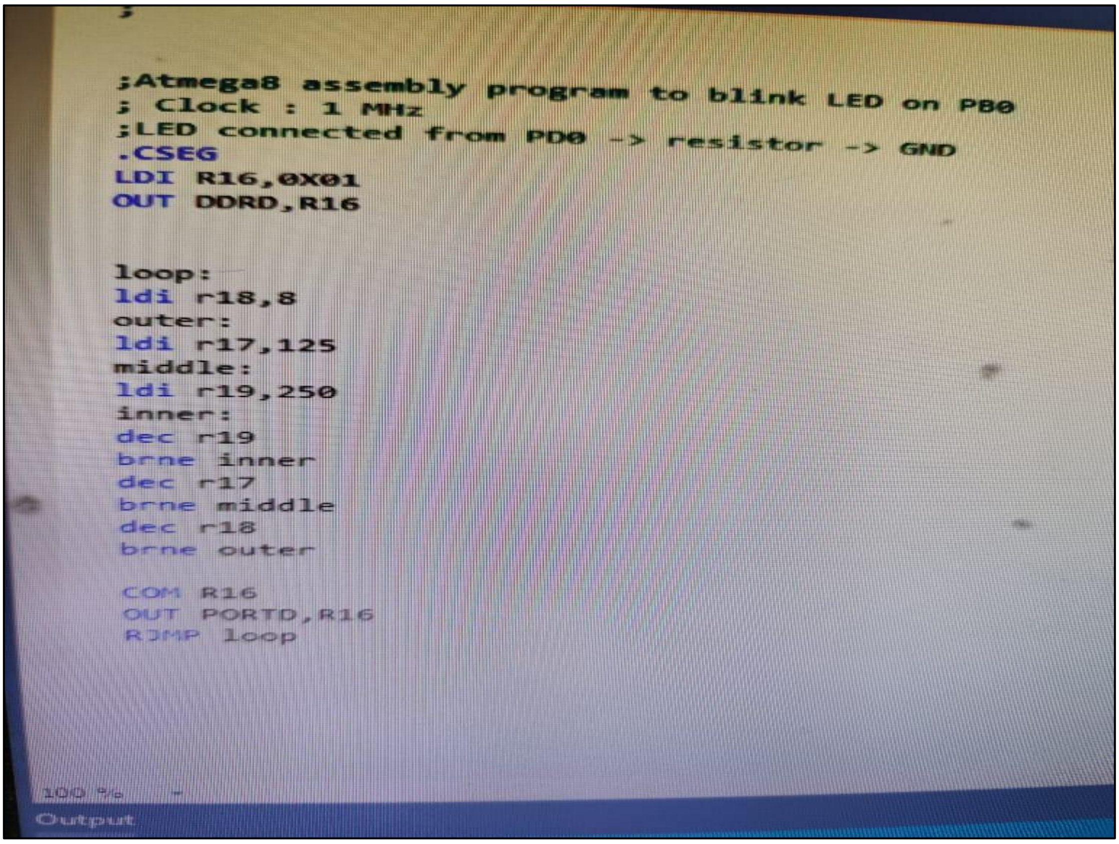
7. A windows PC loaded with Microchip Studio 7 and AVR Burn-O-MAT (for burning_asm)

## IMPLEMENTATION OF PROGRAMS

### 1. Program 1 – Blinking LED

- Objective: Make an LED connected to Port D (e.g. PD0) blink at 1 Hz frequency.
- Setup:
    - Atmega8 microcontroller wired with basic power, clock, reset, and ISP connections.
    - LED connected to PD0 with a 330 Ω resistor to ground.

- Program:
  - Configure Port D as output (DDRD = 0xFF).
  - Toggle PD0 HIGH/LOW with delay in between.
  - Delay generated using software loop ≈ 500 ms, i.e. ~5,00,000 clock cycles at 1 MHz.
- Observation: LED blinks continuously with ~1 second period.



```
;
;Atmega8 assembly program to blink LED on PB0
; Clock : 1 MHz
;LED connected from PD0 -> resistor -> GND
.CSEG
LDI R16,0X01
OUT DDRD,R16

loop:
ldi r18,8
outer:
ldi r17,125
middle:
ldi r19,250
inner:
dec r19
brne inner
dec r17
brne middle
dec r18
brne outer

COM R16
OUT PORTD,R16
RJMP loop
```

Video Link: [https://drive.google.com/file/d/1uMKy09WJa7YjVpo8JIM4S0UVjrBagXg4/view?usp=drivesdk](https://drive.google.com/file/d/1uMKy09WJa7YjVpo8JIM4S0UVjrBagXg4/view?usp=drivesdk)

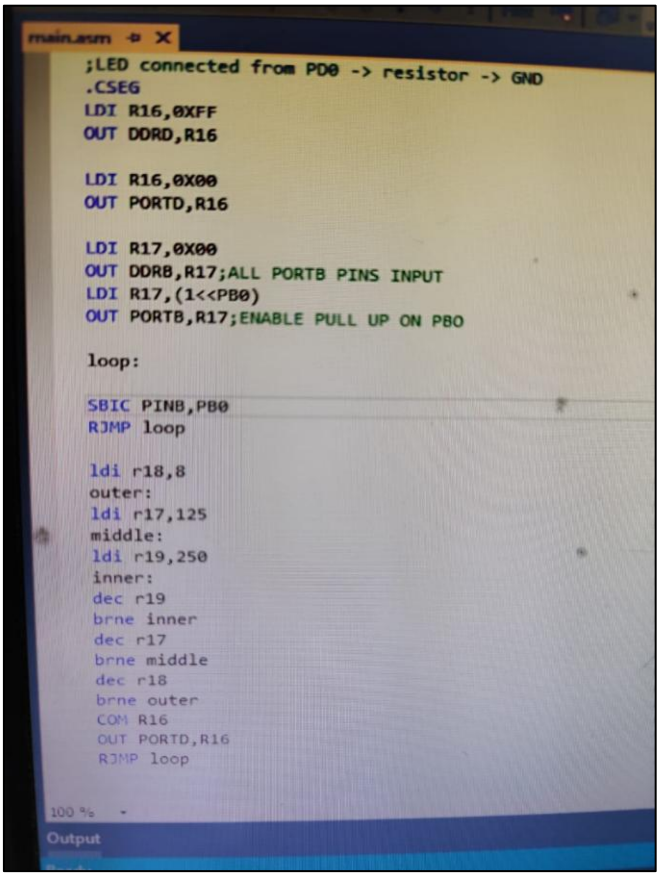
2. **Program 2 – LED output controlled by Push Button**

- Objective: Control LED blinking on Port D with a push button input on Port B.
- Setup:
  - Push button connected to PB0, with one end to ground and a 10 kΩ pull-up resistor to ground.
  - LED on PD0 through 330 Ω resistor to ground.
- Program:
  - Configure Port D as output, Port B as input with pull-up enabled (DDRB = 0x00, PORTB = (1<<PB0)).
  - Poll PB0 using SBIC PINB, PB0 instruction.

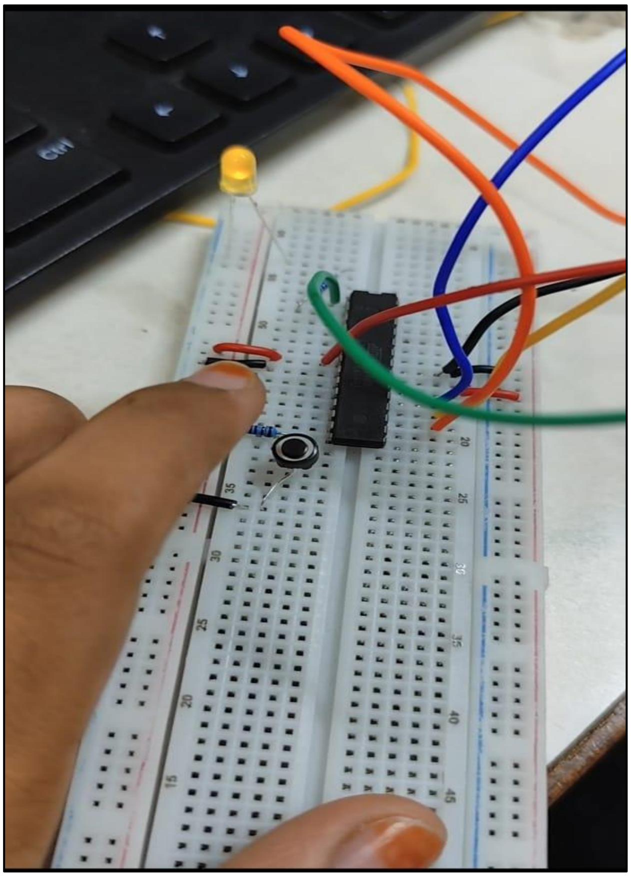
- Only when PB0 is pressed (logic 0), LED blinks with delay loop.

Observation:

- LED turns ON when button is pressed.
- LED remains OFF while button is not pressed.



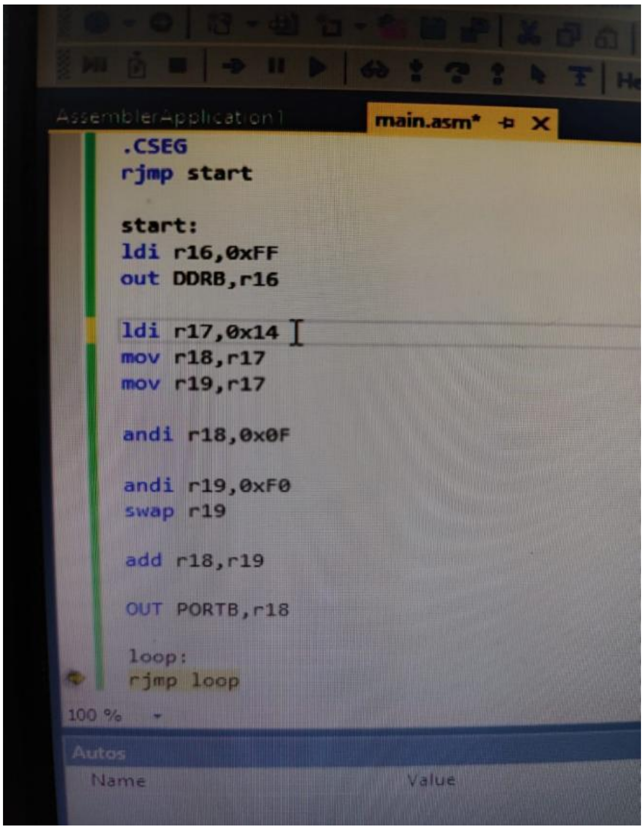
```
;LED connected from PD0 -> resistor -> GND
.CSEG
LDI R16,0XFF
OUT DDRD,R16

LDI R16,0X00
OUT PORTD,R16

LDI R17,0X00
OUT DDRB,R17;ALL PORTB PINS INPUT
LDI R17,(1<<PB0)
OUT PORTB,R17;ENABLE PULL UP ON PBO

loop:

SBIC PINB,PB0
RJMP loop

ldi r18,8
outer:
ldi r17,125
middle:
ldi r19,250
inner:
dec r19
brne inner
dec r17
brne middle
dec r18
brne outer
COM R16
OUT PORTD,R16
RJMP loop
```



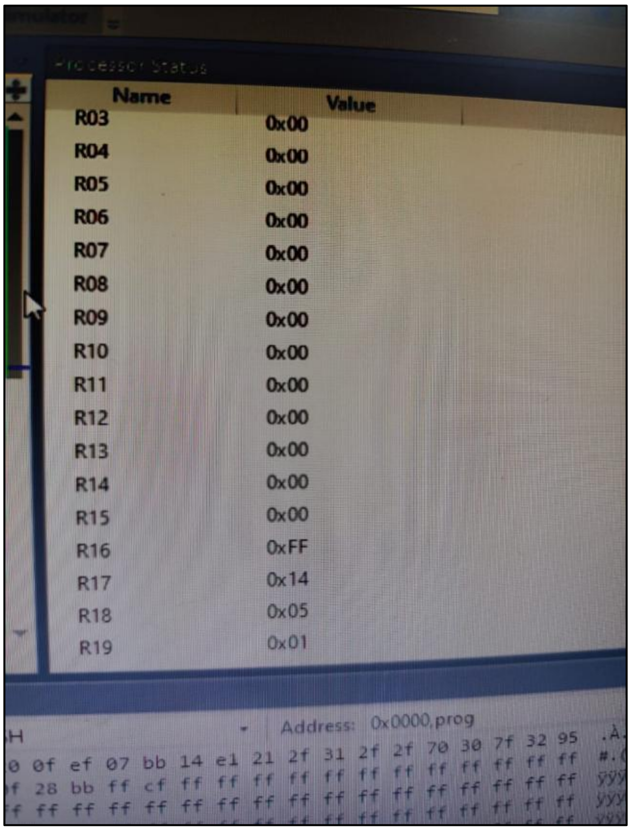
Video Link: https://drive.google.com/file/d/1uOdLLtc9FbPK6V71PlHh8BuZ8sjUGxQB/view?usp=drivesdk

## 3. Program 3 – Roll Number Based Bit Manipulation

- Objective: Perform bit manipulation on roll number digits (last two digits).
- Procedure (for roll no. EE24B114 → "14"):
  - Move decimal value 14 into register R17.
  - Perform SWAP R17 to swap nibbles (upper 4 bits ↔ lower 4 bits).
  - Perform bitwise addition on swapped values.
  - Move lower nibble to R18, upper nibble to R19.
  - Store final result in R18 for output/display.
- Observation: Demonstrates register-level manipulation (shift, swap, add, masking).

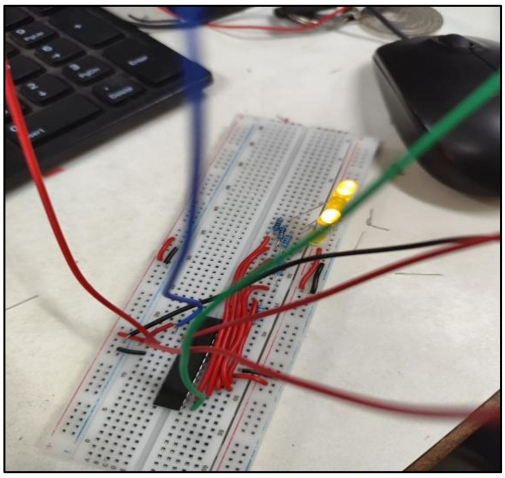


```
.CSEG
rjmp start

start:
ldi r16,0xFF
out DDRB,r16

ldi r17,0x14
mov r18,r17
mov r19,r17

andi r18,0x0F

andi r19,0xF0
swap r19

add r18,r19

OUT PORTB,r18

loop:
rjmp loop
```



| Name | Value |
|---|---|
| R03 | 0x00 |
| R04 | 0x00 |
| R05 | 0x00 |
| R06 | 0x00 |
| R07 | 0x00 |
| R08 | 0x00 |
| R09 | 0x00 |
| R10 | 0x00 |
| R11 | 0x00 |
| R12 | 0x00 |
| R13 | 0x00 |
| R14 | 0x00 |
| R15 | 0x00 |
| R16 | 0xFF |
| R17 | 0x14 |
| R18 | 0x05 |
| R19 | 0x01 |

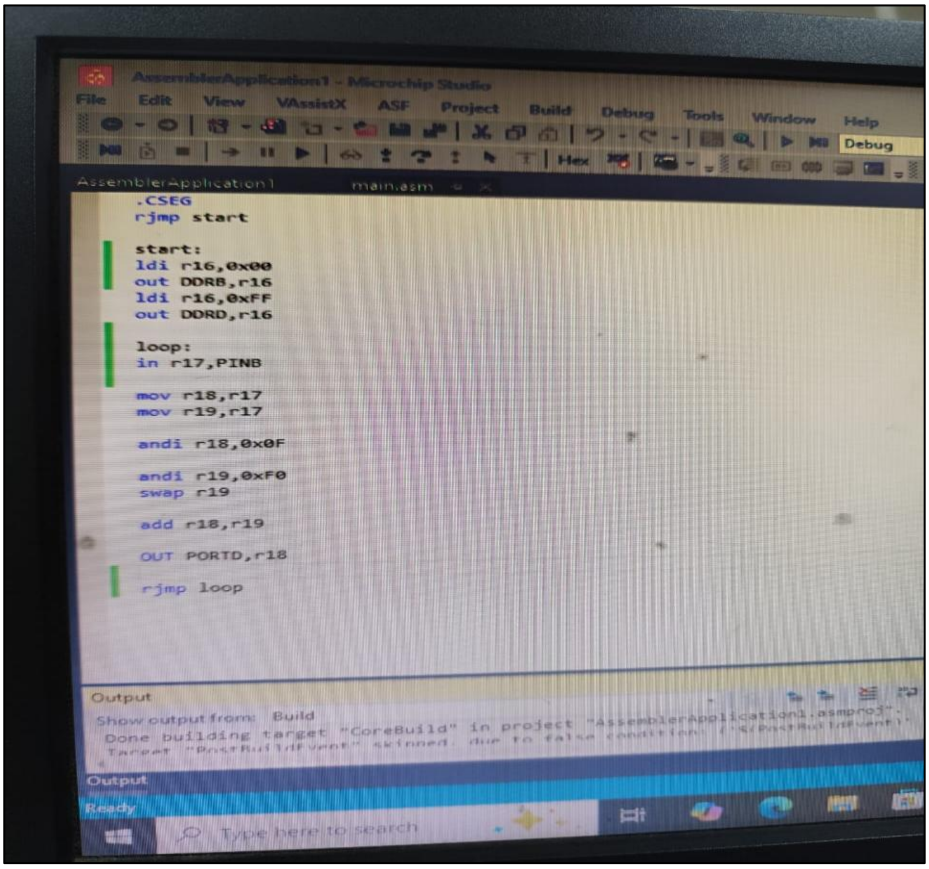
```
H                          Address: 0x0000,prog
0 0f ef 07 bb 14 e1 21 2f 31 2f 2f 70 30 7f 32 95  .À.
f 28 bb ff cf ff ff ff ff ff ff ff ff ff ff ff ff  #.(
f ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿ
                                                   ÿÿÿ
```



## 4. Program 4 – 4-bit Addition using DIP Switches

- Objective: Add two unsigned 4-bit numbers from DIP switches and display result on LEDs.
- Setup:
  - Two DIP switches used for 4-bit numbers.
  - DIP switches connected to Port B pins PB0–PB7 (first nibble = first number, second nibble = second number).
  - LEDs connected to Port D pins PD0–PD4 through 330 Ω resistors.
  - Each DIP switch S1–S4 connected via 1 kΩ resistors to pull-ups.

- Program:
  - Read 8-bit value from PINB.
  - Split into two nibbles: lower 4 bits (N1), upper 4 bits (N2).
  - Add the two numbers.
  - Output result to Port D LEDs (OUT PORTD, Rxx).
- Observation: LEDs display binary sum of the two DIP switch numbers.


AssemblerApplication1 - Microchip Studio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug
Hex
AssemblerApplication1
main.asm
Output
Ready
Type here to search

```
.CSEG
rjmp start

start:
ldi r16,0x00
out DDRB,r16
ldi r16,0xFF
out DDRD,r16

loop:
in r17,PINB

mov r18,r17
mov r19,r17

andi r18,0x0F

andi r19,0xF0
swap r19

add r18,r19

OUT PORTD,r18

rjmp loop
```

```
Show output from: Build
 Done building target "CoreBuild" in project "AssemblerApplication1.asmproj".
 Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)'
```

Video Link: https://drive.google.com/file/d/1uKvm8VFYC31iaoBRNoiw9x2zF_9qNypd/view?usp=drivesdk

*DIP SWITCH 1: (for first num)*

```
VCC-- [1k]----+---- PB0 (pin14)----+---- Switch A1---- GND
              |                    |
VCC—[1k]----+---- PB1 (pin15)----+---- Switch A2---- GND
              |                    |
VCC—[1k]----+---- PB2 (pin16)----+---- Switch A3---- GND
              |                    |
VCC—[1k]----+---- PB3 (pin17)----+---- Switch A4---- GND
```

*DIP SWITCH 2: (for second num)*

```
VCC-- [1k]----+---- PB4 (pin18)----+---- Switch B1---- GND
              |                    |
VCC--[1k]--- +---- PB5 (pin19)----+---- Switch B2---- GND
              |                    |
VCC-- [1k]---- +---- PB6 (pin9) ----+---- Switch B3---- GND
              |                    |
VCC--  [1k]----+---- PB7 (pin10)----+---- Switch B4---- GND
```