**Department of Electrical Engineering and   
Computer Science**

**Faculty Member:** Dr. Arbab Latif  **Dated:** 28/02/2022

**Semester:** 4th **Section:** BEE 12C

**EE-222: Microprocessor Systems**

Lab 4: Digital Input / Output

Group Members

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Reg. No** | **Report**  **Marks / 10** | **Viva**  **Marks / 5** | **Total**  **Marks / 15** |
| Danial Ahmad | 331388 |  |  |  |
| Muhammad Umer | 345834 |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Experiment

# Functions, Branches and Delays

## Objectives

1. Control GPIOs of ATmega16A
2. Implement branches in assembly language
3. Implement calculated delays
4. Break the code down to modular functions

## Equipment

Software

* *Atmel Studio*

Hardware

* ATmega16A microcontroller unit
* Universal Programmer
* Seven Segment Display
* Resistance 47Ω
* LEDs (may use from trainer kit)
* Switch or Button (may use from trainer kit)

## Introduction

 Atmel Studio IDE is a free development environment for programming Atmel MCUs, sourced by Microchip Technology Inc. It provides us with the means to simulate assembly language codes on specific Microcontrollers and provides an easy and intuitive way of producing .HEX files, which are what makes burning the code on the hardware possible.

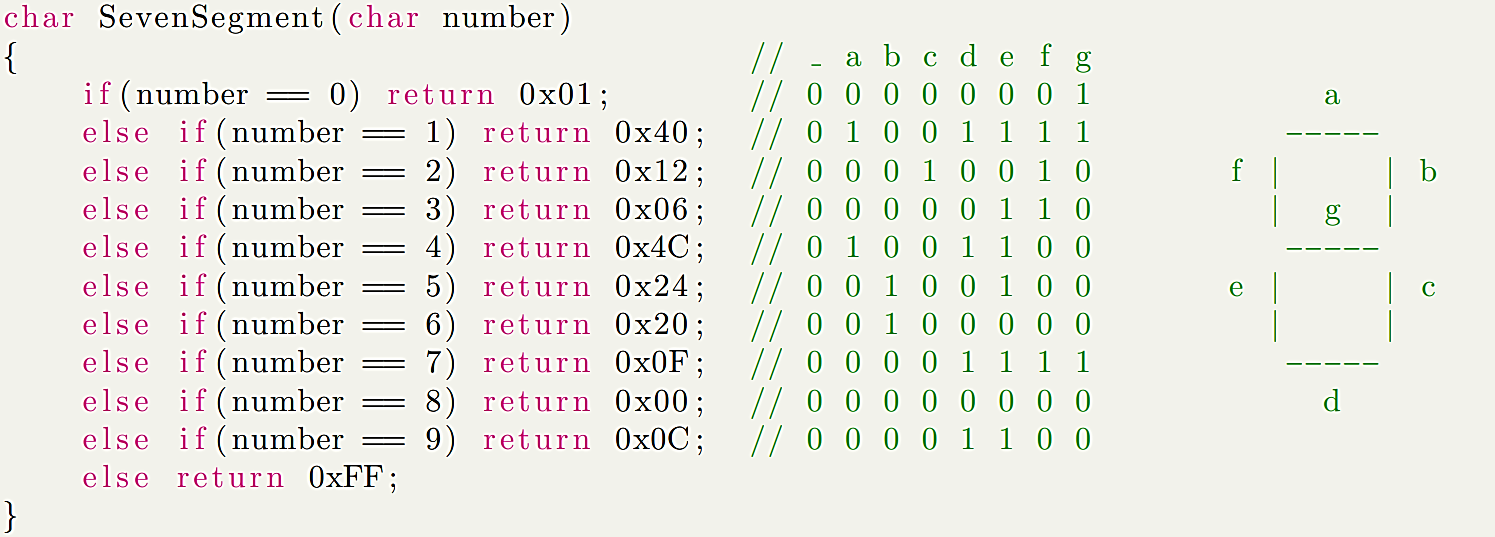
In this specific lab, we familiarize ourselves with branches and jumps, both conditional and unconditional, and utilize them to make a BCD up-down counter. We also interface a common anode BCD – 7 Segment decoder with our ATmega16A microcontroller using I/O Port Programming.

# Lab Tasks

## Task A

Write a function “SevenSegment” in assembly. Assume that the function will receive input in R16. It operates on the input and returns the result in R17 as described by the C program below.

The purpose of function is to take in an 8-bit value between 0-9 and produce its corresponding 8-bit “Common Anode Seven Segment Display Code” in which the actual code is in lower 7 bits and the 8th bit or MSB is just ignored and kept zero. Logic low at pin should turn off the LED while logic high should immediately turn it on.



**Code**

**;**

**; Lab4.asm**

**;**

**; Created: 11/03/2022 1:00:52 pm**

**; Author : Danial and Umer**

**;**

**; initializing stack pointer**

**.ORG 0x00**

**LDI R31, HIGH(RAMEND)**

**OUT SPH, R31**

**LDI R31, LOW(RAMEND)**

**OUT SPL, R31**

**CLR R31**

**; to test our subroutine**

**LDI R16, 0**

**CALL SevenSegment**

**END:**

**RJMP END**

**; Subroutine for sending BCD decoder output upon input**

**SevenSegment:**

**CPI R16, 0**

**BRNE A**

**LDI R17, 0X01**

**RET**

**A:**

**CPI R16, 1**

**BRNE B**

**LDI R17, 0X4F**

**RET**

**B:**

**CPI R16, 2**

**BRNE C**

**LDI R17, 0X12**

**RET**

**C:**

**CPI R16, 3**

**BRNE D**

**LDI R17, 0X06**

**RET**

**D:**

**CPI R16, 4**

**BRNE E**

**LDI R17, 0X4C**

**RET**

**E:**

**CPI R16, 5**

**BRNE F**

**LDI R17, 0X24**

**RET**

**F:**

**CPI R16, 6**

**BRNE G**

**LDI R17, 0X20**

**RET**

**G:**

**CPI R16, 7**

**BRNE H**

**LDI R17, 0X0F**

**RET**

**H:**

**CPI R16, 8**

**BRNE I**

**LDI R17, 0X00**

**RET**

**I:**

**CPI R16, 9**

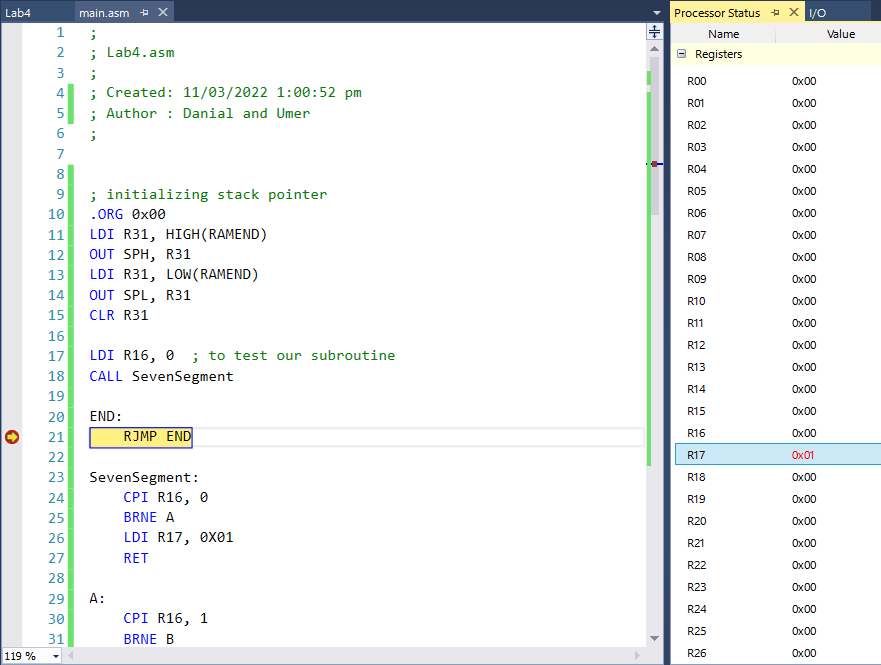
**BRNE J**

**LDI R17, 0X0C**

**RET**

**J:**

**RET**



Exemplar Output

## Task B

Write an assembly program that integrates a switch with seven segment display such that when the microcontroller is powered on, it;

1. Displays zero on a seven segment display.
2. Create a delay of 500ms
3. Check the state of switch.
4. If switch is high(1) it displays next number in ascending order.

Ascending order is (0,1,2,3,4,5,6,7,8,9,0,1,2, ...).

1. Else if switch is low(0), it displays next number in descending order.

Descending order is (0,9,8,7,6,5,4,3,2,1,0,9,8 ...).

1. Move to step 2 and repeat.

*You may use the delay function from previous lab and SevenSegment function from previous task.*

**;**

**; Lab4.asm**

**;**

**; Created: 11/03/2022 1:00:52 pm**

**; Author : Danial and Umer**

**;**

**; initializing stack pointer**

**.ORG 0x00**

**LDI R31, HIGH(RAMEND)**

**OUT SPH, R31**

**LDI R31, LOW(RAMEND)**

**OUT SPL, R31**

**CLR R31**

**; Declaring I/O Ports**

**LDI R31, 0xFF**

**OUT DDRC, R31**

**LDI R31, 0x00**

**OUT DDRB, R31**

**; Initial BCD Value = 0**

**LDI R16, 0**

**CALL SevenSegment**

**OUT PORTC, R17**

**CALL DELAY**

**CLR R30**

**; Main Loop**

**START:**

**IN R30, PINB**

**BST R30, 0**

**BRTS INCREMENT**

**BRTC DECREMENT**

**DECREMENT: ; decreases R16 from 9 - 0**

**DEC R16**

**CALL LIMITERL**

**CALL SevenSegment**

**OUT PORTC, R17**

**CALL DELAY**

**RJMP START**

**INCREMENT: ; increases R16 from 0 - 9**

**INC R16**

**CALL LIMITERH**

**CALL SevenSegment**

**OUT PORTC, R17**

**CALL DELAY**

**RJMP START**

**; Subroutine for sending BCD decoder output upon input**

**SevenSegment:**

**CPI R16, 0**

**BRNE A**

**LDI R17, 0X01**

**RET**

**A:**

**CPI R16, 1**

**BRNE B**

**LDI R17, 0X4F**

**RET**

**B:**

**CPI R16, 2**

**BRNE C**

**LDI R17, 0X12**

**RET**

**C:**

**CPI R16, 3**

**BRNE D**

**LDI R17, 0X06**

**RET**

**D:**

**CPI R16, 4**

**BRNE E**

**LDI R17, 0X4C**

**RET**

**E:**

**CPI R16, 5**

**BRNE F**

**LDI R17, 0X24**

**RET**

**F:**

**CPI R16, 6**

**BRNE G**

**LDI R17, 0X20**

**RET**

**G:**

**CPI R16, 7**

**BRNE H**

**LDI R17, 0X0F**

**RET**

**H:**

**CPI R16, 8**

**BRNE I**

**LDI R17, 0X00**

**RET**

**I:**

**CPI R16, 9**

**BRNE J**

**LDI R17, 0X0C**

**RET**

**J:**

**RET**

**DELAY: ; Subroutine to add a delay of 500ms**

**LDI R22, 4**

**DELAY\_1:**

**LDI R21, 125**

**DELAY\_2:**

**LDI R20, 250**

**DELAY\_3:**

**DEC R20**

**NOP**

**BRNE DELAY\_3**

**DEC R21**

**BRNE DELAY\_2**

**DEC R22**

**BRNE DELAY\_1**

**RET ; RETURNS back to Call**

**; Following two limits provide us the ability**

**; to stay within the range of 0 - 9**

**LIMITERL:**

**CPI R16, 255**

**BREQ JAY**

**RET**

**JAY:**

**LDI R16, 9**

**RET**

**LIMITERH:**

**CPI R16, 10**

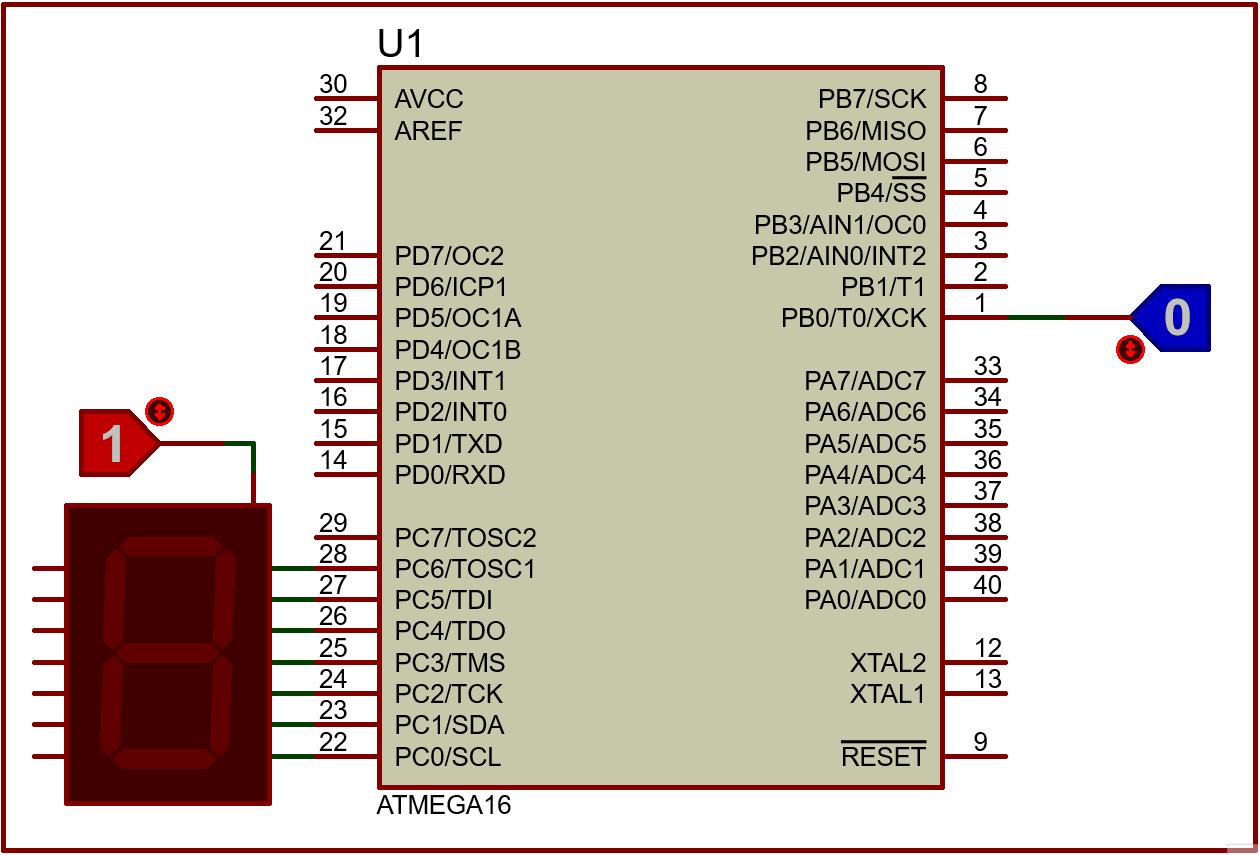
**BREQ EMM**

**RET**

**EMM:**

**LDI R16, 0**

**RET**



Exemplar Output

## Conclusion

After the conduction of this lab, we have learnt how to utilize both conditional and unconditional jumps in our assembly code and combining this knowledge with I/O port interfacing with discrete hardware components, such as the BCD – 7 Segment Decoder, we can accomplish a multitude of unique tasks. For the current case, we used a common anode decoder to make a BCD Up-Down counter.