Lab 4: Digital Input Output

EE222: Microprocessor Systems

Contents

1 Acknowledgements

This lab exercise is prepared by Mohammad Azfar Tariq and Muhammad Usman under the supervision of Dr. Rehan Ahmed for the course EE-222 Microprocessor Systems. Reporting any error or discrepancy found in the text is appreciated.

2 Administrivia

2.1 Objective

By the end of this lab you will be able to

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

2.2 Deliverable

You are required to submit

- Appropriately Commented Code
- Image of assembled hardware
- Comment on the issues you faced in Developing the Solution.

in the beginning of next lab

2.3 Hardware Resources

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

3 Introduction

3.1 ATmega16A Clock

ATmega16A has many options for clock. We can provide clock to it through external crystal or RC oscillator. However, an internal RC oscillator is also present which can be calibrated to provide clock of 1,2,4 or 8 MHz. The factory default configurations have already set the internal RC oscillator to provide a **"1MHz"** clock. In this lab we will be using ATmega16A with factory default configurations. For more information see chapter 8 in data sheet[1].

3.2 Delay Calculation

In delay calculation one must keep in view two things,

1. CPU clock cycle frequency (F_{cpu})
2. Instruction cycles consumed

$$Delay\ time = (no.\ of\ instruction\ cycles)/F_{cpu}$$

3.3 Digital I/O

There are four I/O ports in ATmega16, A, B, C and D. Three registers are specific to each port,

- Data Direction Register (`DDRx`).
- Port Output Register (`PORTx`).
- Port Input Register (`PINx`)

`DDRx` register decides wether the port pins will act as input or output. If any bit of `DDRx` is loaded with 1, the corresponding pin of the port will be activated in output mode and vice versa for input mode. For example to activate pins of Port B as output pins, we have to load 0xFF in `DDRB`.
To send some binary value out at the pins (which are set as output by `DDRx`), we have to store it in `PORTx` register. For example to send 0xAA out at the pins of port B, we have to store it in `PORTB`.
To read the status of pins (high or low) which are set as input by `DDRx`, we have to read register `PINx`. For example, if we have to read what logic values are present at the pins of port B, we must read the values in `PINB` register, they will be same.
Under normal condition a pin in a port cannot perform dual operations (–that is input and output at the same time).
Use specific instructions "`IN` and `OUT`" to read and deliver data to I/O registers.

[1] http://ww1.microchip.com/downloads/en/devicedoc/atmel-8154-8-bit-avr-atmega16a_datasheet.pdf

3.4 Modelling Control Structures

3.4.1 If-Else

$a \longrightarrow R14$, $b \longrightarrow R15$, $c \longrightarrow R16$

```
if(a == b){
    c = 1;
}
else{
    c = 2;
}
```

```
    LDI   R16, 2
    CP    R14, R15
    BRNE  done
    LDI   R16, 1
done:
    ;rest of the program
```

3.4.2 While-Loop

$sum \longrightarrow R16$

```
while(sum < 10)
{
    sum = sum + 1;

}
```

```
Loop:
    SUBI  R16, -1
    CPI   R16, 10
    BRLT  Loop
    ;rest of the program
```

3.4.3 If-Else if-Else

$number \longrightarrow R20$, $a \longrightarrow R16$

```
if(number == 1)
{
    a = 10;
}
else if(number == 2)
{
    a = 20;
}
else
{
    a = 30;
}
```

```
    LDI   R16, 10
    CPI   R20, 1
    BREQ  Done
    LDI   R16, 20
    CPI   R20, 2
    BREQ  Done
    LDI   R16, 30
Done:
    ;rest of the program
```

4 Lab Tasks

4.1 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.

```
char SevenSegment(char number)
{                                       // . a b c d e f g
    if(number == 0) return 0x01;        // 0 0 0 0 0 0 0 1
    else if(number == 1) return 0x40;   // 0 1 0 0 1 1 1 1
    else if(number == 2) return 0x12;   // 0 0 0 1 0 0 1 0
    else if(number == 3) return 0x06;   // 0 0 0 0 0 1 1 0
    else if(number == 4) return 0x4C;   // 0 1 0 0 1 1 0 0
    else if(number == 5) return 0x24;   // 0 0 1 0 0 1 0 0
    else if(number == 6) return 0x20;   // 0 0 1 0 0 0 0 0
    else if(number == 7) return 0x0F;   // 0 0 0 0 1 1 1 1
    else if(number == 8) return 0x00;   // 0 0 0 0 0 0 0 0
    else if(number == 9) return 0x0C;   // 0 0 0 0 1 1 0 0
    else return 0xFF;
}
```

```
      a
    -----
f  |     | b
   |  g  |
    -----
e  |     | c
   |     |
    -----
      d
```

`BRNE`, `BREQ` and `CPI` are your friends in this problem ☺

4.2 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, ...`).
5. 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 ...`).
6. Move to step 2 and repeat.

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