EE2016: Microprocessor Lab

Experiment 8: Assembly Programming and Interaction with Peripherals in Atmel Atmega8 Microcontroller

Ajeet E S, EE22B086
Amogh Agrawal, EE22B087

June 5, 2024

Contents

1 Aim 1

2 Observations: Google Drive Link 1

3 Problem 1: Control an LED using a DIP switch 1
3.1 Problem Statement 1
3.2 Approach 1
3.3 Flowchart 1
3.4 Code 2
3.5 Inferences 2
3.5.1 Register Transfers 2
3.5.2 Ports and Pins 2
3.5.3 Pull-Up Resistor 2

4 Problem 2: 4- Bit Addition 2
4.1 Problem Statement 2
4.2 Approach 3
4.3 Flowchart 3
4.4 Code 3
4.4.1 Register Transfers 4
4.4.2 Ports and Pins 4
4.4.3 Pull-Up Resistor 5

5 Problem 3: 4- Bit Multiplication 5
5.1 Problem Statement 5
5.2 Approach 5
5.3 Flowchart 6
5.4 Code 6
5.5 Inferences 7
5.5.1 Register Transfers 7
5.5.2 Ports and Pins 7
5.5.3 Pull-Up Resistor 8

List of Listings

1 Code to Take Input and Display through LEDs 2
2 Code to Perform 4- Bit Addition 4
3 Code to Perform 4- Bit Multiplication 7

1 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.
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 LEDs connected to another port.

2 Observations: Google Drive Link

The link to the codes and video for the Experiment are uploaded here:
Code and Video

3 Problem 1: Control an LED using a DIP switch

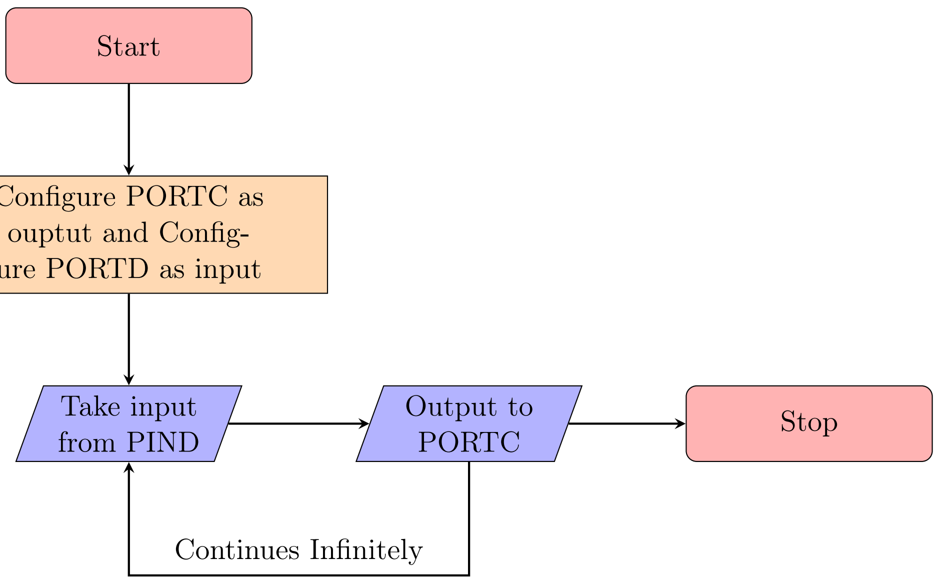
3.1 Problem Statement

Take input using a DIP Switch and use it to light up an LED Connected to an Output Port, say take input from PB0 and light the LED connected to PD0.

3.2 Approach

We set the value of DDRx to 0x00 and 0xFF for D and C respectively to make them input and output respectively. Then we take input through a DIP Switch and output the data taken in to pin D using the OUT statement.

3.3 Flowchart



3.4 Code

The code is given below:

```
.include "m8def.inc"
LDI R16, 0xFF
OUT DDRC, R16 ; Set All pins of C as output

LDI R16, 0x00
OUT DDRD, R16 ;Set all pincs of D as input

NOP
IN R16, PIND ; Take input from Pin D
OUT PORTC, R16 ; Put the value read into C
```

Listing 1: Code to Take Input and Display through LEDs

3.5 Inferences

3.5.1 Register Transfers

1. Data is transferred to R16 using LDI.
2. The values in R16 are put to DDRC and DDRD pins using OUT command.
3. The value is taken as input to R16 using IN command.
4. Data is outputted to PORTC using the OUT command.

3.5.2 Ports and Pins

1. PORTC is used to write value onto the output.
2. PIND is used to take input from the D pins.
3. DDRC and DDRD are used to set C and D as output and input respectively.

3.5.3 Pull-Up Resistor

1. A $10K\Omega$ Resistor is connected to the input pin to keep the input in PIN D to logic High (1) when the switch is open.
2. So, the value read will be Logic 1 when the switch is open and Logic 0 when the switch is closed

4 Problem 2: 4- Bit Addition

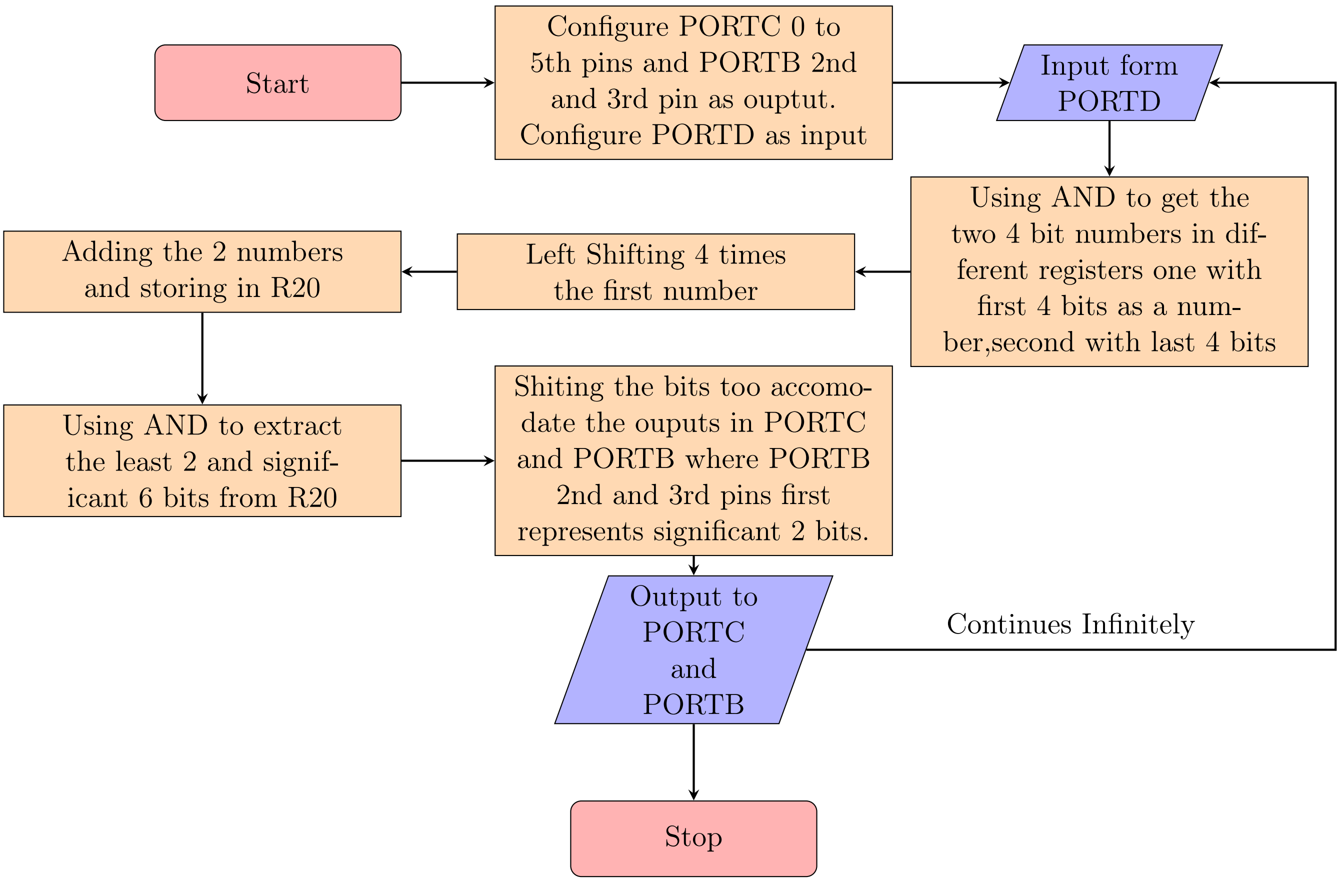
4.1 Problem Statement

Perform 4-bit addition of two unsigned nibbles from an 8-bit dip input switch (set by TAs) and display the result obtained in LEDs.

4.2 Approach

1. We use D0-D7 for Input, and C0-C5, B1 and B2 for Output. This is because the other pins are in use already.
2. We take one nibble from the lower 4 bits of D and the next nibble from the higher 4 bits.
3. We display the two Least Significant Bits of the Result through B2 and B1. The next six bits are displayed from C0 through C5 (MSB at C5 and LSB at C0).

4.3 Flowchart



4.4 Code

The code to perform 4- bit addition is given below:

```
.include "m8def.inc"
LDI R16 , 0x3F
OUT DDRC, R16; Set 6 LSB of C as outputs

LDI R16, 0x06
OUT DDRB, R16; Set  Pins 1 and 2 of B as Output

LDI R16 , 0x00
OUT DDRD, R16; Set all pins of D as Input

```

```
    NOP
    IN R16 , PIND; Take Input from D
    LDI R20, 0x0F
    AND R20, R16; Extract the Lowest 4 Bits(1st Number)

    LDI R21, 0xF0
    AND R21, R16; Extract the highest 4 Bits(2nd Number)

    LDI R22, 0x04
Label:
    LSR R21
    DEC R22
    BRNE Label; Shift R21 by 4 bits

    ADD R20,R21; Add the 2 numbers

    LDI R21, 0x03
    AND R21, R20
        LSL R21; Extract the lowest 2 bits (Output through B)

    LDI R22, 0xFC
    AND R22, R20
        LSR R22; Extract the other bits (Output through C)

    OUT PORTC, R22
    OUT PORTB, R21; Set output
```

Listing 2: Code to Perform 4- Bit Addition

4.4.1 Register Transfers

1. Data is transferred to R16 using LDI.
2. The values in R16 are put to DDRB, DDRC and DDRD pins using OUT command.
3. The value is taken as input to R16 using IN command.
4. Data is outputted to PORTB and PORTC using the OUT command.
5. The AND command does AND of the inputs and stores that to the first register
6. The nibbles are stored to R20 and R21. R16 is used for moving values to DDRx and also to take the input.

4.4.2 Ports and Pins

1. PORTC and PORTB are used to write value onto the output.
2. PIND is used to take input from the D pins.
3. DDRB, DDRC and DDRD are used to set B, C and D as output, output and input respectively.

4.4.3 Pull-Up Resistor

1. A $10K\Omega$ Resistor is connected to the input pin to keep the input in PIN D to logic High (1) when the switch is open.

2. So, the value read will be Logic 1 when the switch is open and Logic 0 when the switch is closed

3. This prevents any unpermitted states in the registers

5 Problem 3: 4- Bit Multiplication

5.1 Problem Statement

Perform 4-bit multiplication of two unsigned nibbles from an 8-bit dip input switch (set by TAs) and display the result obtained in LEDs.

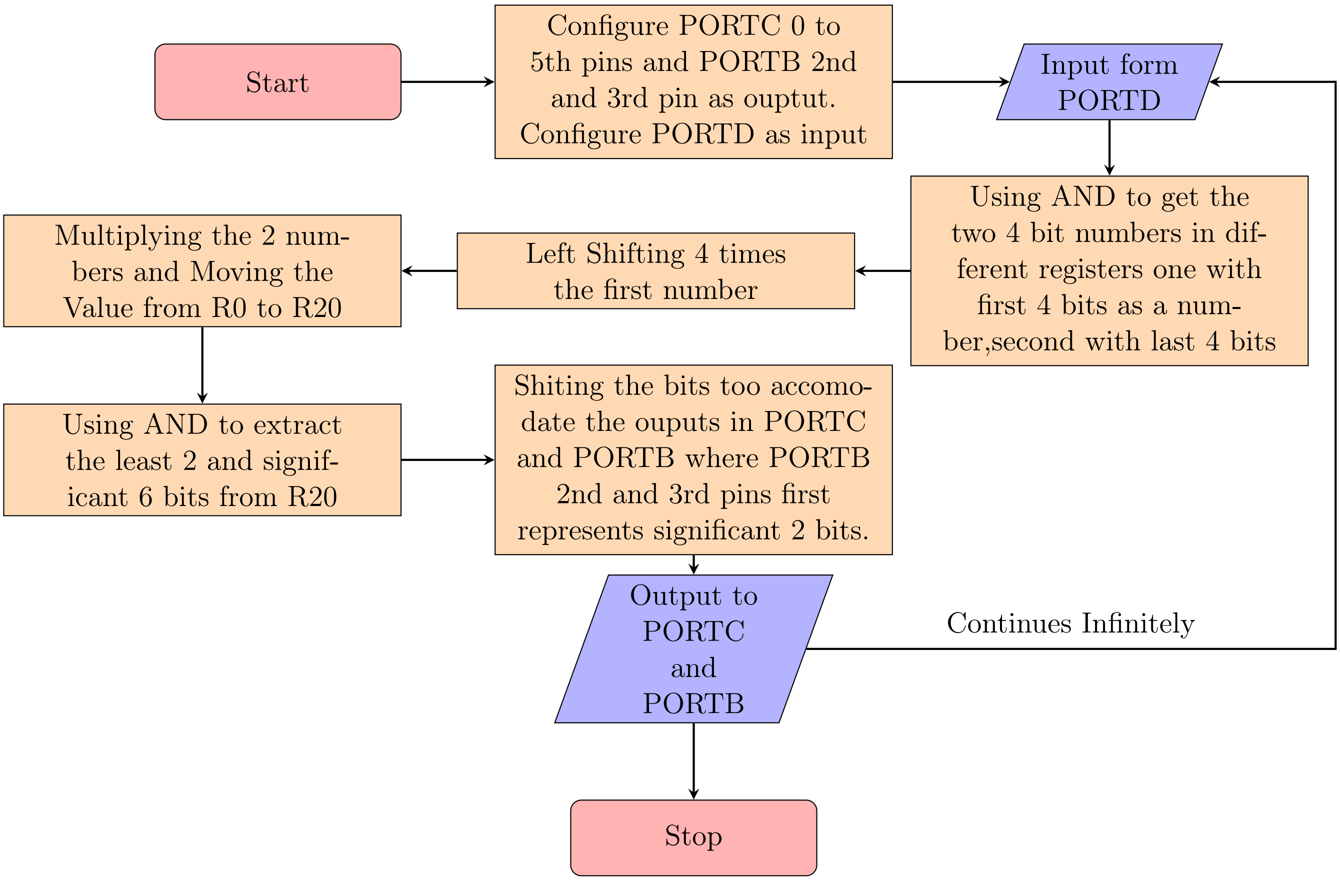
5.2 Approach

1. We use D0-D7 for Input, and C0-C5, B1 and B2 for Output. This is because the other pins are in use already.

2. We take one nibble from the lower 4 bits of D and the next nibble from the higher 4 bits.

3. We then multiply the two nibbles and display the two Least Significant Bits of the Result through B2 and B1. The next six bits are displayed from C0 through C5 (MSB at C5 and LSB at C0).

5.3 Flowchart



5.4 Code

The code to perform 4- bit multiplication is given below:

```
    .include "m8def.inc"
    LDI R16 , 0x3F
    OUT DDRC, R16; Set 6 LSB of C as outputs

    LDI R16, 0x06
    OUT DDRB, R16; Set  Pins 1 and 2 of B as Output

    LDI R16 , 0x00
    OUT DDRD, R16; Set all pins of D as Input

    NOP
    IN R16 , PIND; Take Input from D
    LDI R20, 0x0F
    AND R20, R16; Extract the Lowest 4 Bits(1st Number)

    LDI R21, 0xF0
    AND R21, R16; Extract the highest 4 Bits(2nd Number)

    LDI R22, 0x04
Label:
```

```
    LSR R21
    DEC R22
    BRNE Label; Shift R21 by 4 bits

    MUL R20,R21; Multiply the 2 numbers

    MOV R20, R0; Take the result

    LDI R21, 0x03
    AND R21, R20
        LSL R21; Extract the lowest 2 bits (Output through B)

    LDI R22, 0xFC
    AND R22, R20
        LSR R22; Extract the other bits (Output through C)

    OUT PORTC, R22
    OUT PORTB, R21; Set output
```

Listing 3: Code to Perform 4- Bit Multiplication

5.5 Inferences

5.5.1 Register Transfers

1. Data is transferred to R16 using LDI.
2. The values in R16 are put to DDRB, DDRC and DDRD pins using OUT command.
3. The value is taken as input to R16 using IN command.
4. Data is outputted to PORTB and PORTC using the OUT command.
5. The AND command does AND of the inputs and stores that to the first register
6. The nibbles are stored to R20 and R21. R16 is used for moving values to DDRx and also to take the input.
7. MUL stores the result in R0. Data is transferred from R0 to R20 using the MOV command.
8. The outputs are put to PORTC and PORTB using R22 and R21 seperately using the OUT Command.

5.5.2 Ports and Pins

The Ports and Pins are used in the same way as the previous Problem.

1. PORTC and PORTB are used to write value onto the output.
2. PIND is used to take input from the D pins.
3. DDRB, DDRC and DDRD are used to set B, C and D as output, output and input respectively.

5.5.3 Pull-Up Resistor

1. A $10K\Omega$ Resistor is connected to the input pin to keep the input in PIN D to logic High (1) when the switch is open.

2. So, the value read will be Logic 1 when the switch is open and Logic 0 when the switch is closed

3. This prevents any unpermitted states in the registers