EMBEDDED SYSTEMS LAB 1

Fall Semester 2023

Lab Experiment Lab 1 – Blink LED in Assembler

Instructor: Dr. Prof. Fangning Hu

Author of the report: Faraz Ahmed and Sohaib Salman

Experiment conducted by: Faraz Ahmed, Sohaib Salman

Place of execution: Research 1

Date of Execution: 5th September 2023

Introduction

The objective of this laboratory experiment is to gain a comprehensive understanding of how to effectively manage the digital I/O (Input/Output) ports of the ATmega328 microcontroller. The ATmega328 microcontroller is equipped with three distinct digital ports, denoted as PORTB, PORTC, and PORTD, each comprising eight individual pins. These pins can be dynamically configured to function either as output or input ports, providing a high degree of versatility in interfacing with external devices.

To exert control over these ports, three essential registers are employed:

1. **Data Register (PORTx):** This register maintains the current state of the port, storing the present output values.
2. **Data Direction Register (DDRx):** Responsible for dictating the direction of each port (input or output).
3. **Port Input Pin Register (PINx):** This register exclusively holds the input values of the port, offering a read-only interface.

By comprehensively understanding and manipulating these registers, we aim to explore the full spectrum of capabilities offered by the ATmega328 microcontroller.

Pre Lab-Tasks

2. Assembly language

- **LDI** (Load Immediate): LDI loads an immediate value into a register. For example, **LDI R16, 0x0A** loads the value **0x0A** into register **R16**.
- **OUT** (Store Register to I/O Space): OUT stores the contents of a register into an I/O address. For instance, **OUT 0x3F, R16** stores the value of register **R16** into I/O address **0x3F**.
- **SBI** (Set Bit in I/O Register): SBI sets a specific bit in an I/O register. For example, **SBI 0x1A, 5** sets bit 5 in the I/O register at address **0x1A**.
- **CBI** (Clear Bit in I/O Register): CBI clears a specific bit in an I/O register. For instance, **CBI 0x1B, 3** clears bit 3 in the I/O register at address **0x1B**.

- JMP/RJMP (Jump/Relative Jump): JMP is an unconditional jump to a specified address, while RJMP performs a relative jump. For example, JMP Label jumps to the location labeled Label.
- CALL/RCALL (Call/Subroutine Call): CALL is used to call a subroutine, which is essentially a function in assembly language. RCALL is a relative call, similar to RJMP. For instance, CALL Subroutine calls the subroutine labeled Subroutine.
- RET (Return): RET is used to return from a subroutine to the main program. It retrieves the return address from the stack.
- DEC (Decrement): DEC decreases the value of a register by one. For example, DEC R16 decrements the value in register R16.
- BRNE (Branch if Not Equal): BRNE performs a conditional branch. It jumps to a specified label if a specified condition is met. For instance, BRNE Label branches to Label if the Zero Flag is not set.
- CLI (Clear Global Interrupt Flag): CLI disables interrupts, allowing the program to execute without interruption.

These instructions provide the foundational building blocks for writing programs in AVR assembly language, allowing for precise control over the microcontroller's operations. Keep in mind that these descriptions are simplified, and specific usage may vary based on the microcontroller's architecture and the assembler being used.

3. Clock cycles for 1 second delay

```
Delay:  LDI  R17, 0x02
loop:   DEC  R17
        BRNE loop
        RET
```

In this loop, we are decrementing R17 in each iteration until it becomes zero. This loop will take 2 * (N- 1) clock cycles, where N is the initial value of R17 (in this case, N = 2).

Now, we want to modify the delay loop so that it produces a 1-second delay with an 8MHz clock. To do this, we need to calculate the new value of R17. Since the loop duration should be 1 second and each iteration takes 2 * (N- 1) cycles, we have:

2 * (N- 1) = 8,000,000 (1 second at 8MHz)

N = 4,000,000

Therefore, we should load R17 with 0x3D0900 in order to achieve a 1-second delay with an 8MHz clock.

Here is the modified code:

```
Delay:  LDI  R17, 0x09
        LDI  R18, 0xD0
        LDI  R19, 0x3D
loop:   DEC  R17
        BRNE loop
        DEC  R18
        BRNE loop
        DEC  R19
        BRNE loop
        RET
```

Lab Assignments

1. Debugged Example 1:

```
Col_1    Col_2    Col_3        Col_4
        CLI              ; Clear the I-bit
        DEC     R17      ; Minus 1 from R17
        ADD     R16,R17  ; Add R17 to value in R16
        MOV     R12,R16  ; Copy the value in R16 to R12
END:  JMP     END        ; Jump to the label END
```

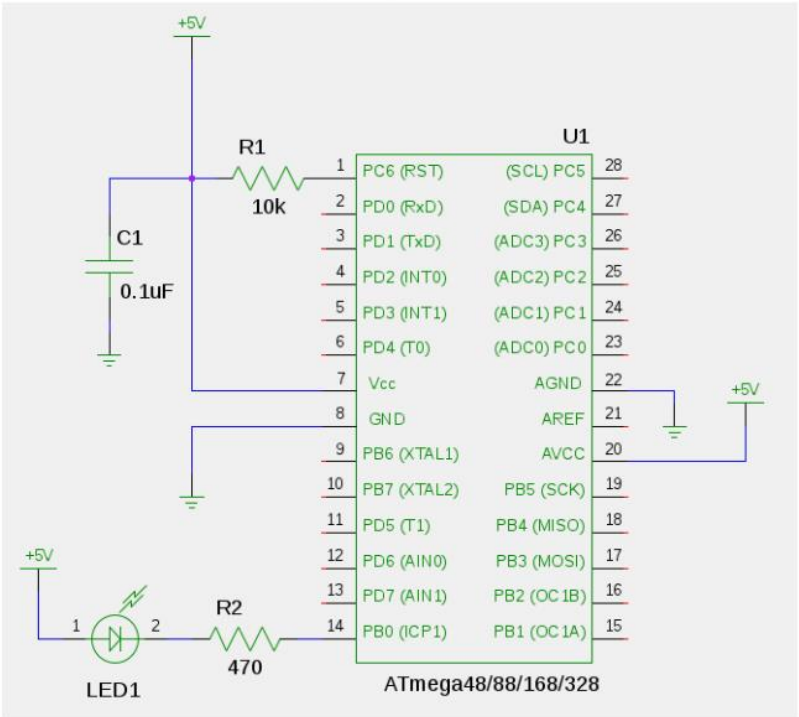
2. Code to toggle PORTD and delay 1 second:

```
.include "m328pdef.inc"
.cseg
.org 0x00
   clr r17                    ; Clear led register
   ; Arduino PIN D8
   ldi r16, (1<<PINB0)     ; Load 00000001 into r16 register
   out DDRB, r16           ; Set PINB0 to output

   .equ inner_loop_val = 28168

start:                        ; Start code
   eor r17, r16               ; Toggle PINB0 in led register
   out PORTB, r17          ; Write led register to PORTB

   ldi r18, 71                ; Initialize outer loop count

   ; Delay loops
outerLoop:
   ldi r24, LOW(inner_loop_val)    ; Initialize inner loop count in inner
   ldi r25, HIGH(inner_loop_val)   ; loop high and low registers

innerLoop:
   sbiw r24, 1              ; Decrement inner loop registers
   brne innerLoop          ; Branch to innerLoop if innerLoop registers != 0

   dec r18                    ; Decrement outer loop register
   brne outerLoop          ; Branch to outerLoop if outer loop register != 0

   rjmp start                 ; Jump to start
```

1. Code Explanation:
    - .include "m328pdef.inc": This includes the definition file for the ATmega328P microcontroller, which provides symbolic names for registers, memory locations, and other hardware-specific details.
    - .cseg: Indicates that the following code is placed in the Code Segment (program memory).
    - .org 0x00: Sets the program counter to address 0x00, indicating the start of the program.
    - clr r17: Clears register r17, which is used as a flag for the LED state.
    - ldi r16, (1<<PINB0): Loads the value 00000001 (binary) into register r16. This value is used to configure PINB0 (Digital Pin 8) as an output.

- **out DDRB, r16**: Configures PINB0 as an output by writing the value of **r16** to the Data Direction Register (**DDRB**).
- **eor r17, r16**: Exclusive OR (**xor**) operation between **r17** and **r16** toggles the LED state in **r17**.
- **out PORTB, r17**: Writes the value in **r17** (representing the LED state) to the PORTB register, toggling the LED.
- **ldi r18, 71**: Initializes the outer loop counter with a value of 71.
- **outerLoop** label:
  - Initializes the outer loop.
- **innerLoop** label:
  - Initializes the inner loop.
  - **sbiw r24, 1**: Subtracts one from the 16-bit value formed by **r24** and **r25**. This effectively decrements the inner loop counter.
  - **brne innerLoop**: Branches back to **innerLoop** if the result of the subtraction is not zero.
  - After the inner loop completes, **r18** (outer loop counter) is decremented.
  - **brne outerLoop**: Branches back to **outerLoop** if **r18** is not zero.
- **rjmp start**: Jumps back to the **start** label to repeat the process.

3. Diagram



4. Modified code that does not need CALL/RCALL

```
.cseg
.org 0x00
   clr r17                  ; Clear led register
   ; Arduino PIN D8
   ldi r16, (1<<PINB0)      ; Load 00000001 into r16 register
   out DDRB, r16            ; Set PINB0 to output

   .equ inner_loop_val = 28168

start:                      ; Start code
   eor r17, r16             ; Toggle PINB0 in led register
   out PORTB, r17           ; Write led register to PORTB

   ldi r18, 71              ; Initialize outer loop count

   ; Delay loops
outerLoop:
   ldi r24, LOW(inner_loop_val) ; Initialize inner loop count in inner
   ldi r25, HIGH(inner_loop_val); loop high and low registers

innerLoop:
   sbiw r24, 1              ; Decrement inner loop registers
   brne innerLoop           ; Branch to innerLoop if innerLoop registers != 0

   dec r18                  ; Decrement outer loop register
   brne outerLoop           ; Branch to outerLoop if outer loop register != 0

   rjmp start               ; Jump to start
```

Resources

1. Atmega328-datasheet
2. Atmel-0856-AVR-Instruction Set Manual
3. The Basic of the AVR Assembly Language (Website)