ECE 375 LAB 3

Introduction to AVR Simulation with Atmel Studio

Lab Time: Wednesday 10-12

Tu Lam

ADDITIONAL QUESTIONS

1) What is the initial value of DDRB?

The initial value of DDRB is 0b00000000 (0x00).

2) What is the initial value of PORTB?

The initial value of PORTB is also 0b00000000 (0x00) [This is the same as DDRB's value].

3) Based on the initial values of DDRB and PORTB, what is Port B's default I/O configuration?

Based on the initial vales of DDRB and PORTB, Port B's default I/O configuration is that the DDRB set to 0 meaning they are configure to be as the input and PORTB is not reading any input from any pin from PIN 0-7.

4) What 16-bit address (in hexadecimal) is the stack pointer initialized to?

The 16-bit address of the stack pointer that it has initialized to is 0x10FF.

5) What are the contents of register r0 after it is initialized?

The content in the register r0 is holding after it is initialized is 0xFF (0b11111111).

6) How many times did the code inside of LOOP end up running?

The number of times inside the LOOP end up running is four times before it exited the LOOP.

7) Which instruction would you modify if you wanted to change the number of times that the loop runs?

The instruction I would modify if I wanted to change the number of times that the loop runs is on the instruction where: (brne LOOP).

This instruction that is controlling and setting how many times the program must loop.

8) What are the contents of register r1 after it is initialized?

The content of register r1 after it is initialized is 0xAA (0b10101010)

9) What are the contents of register r2 after it is initialized?

The content of register r2 after it is initialized is 0x0F (0b00001111).

10) What are the contents of register r3 after it is initialized?

The content of register r3 after it is initialized is 0x0F (0b00001111).

11) What is the value of the stack pointer when the program execution is inside the FUNCTION subroutine?

The value of the stack pointer is 0x10FD when the program is inside the FUNCTION subroutine.

12) What is the final result of FUNCTION? (What are the hexadecimal contents of memory locations $0105:$0104)?

The final result of FUNCTION which is the contents of memory locations $0105:$0104 is the value 0x0E for $0104 & 0xBA for $0105.

CONCLUSION

Lab #3 introduces the user to get familiarize with using the AVR Assembly Debugging tool. The whole lab process walk you through how breakpoint work and how it would help in the future when it comes to debugging. Also, the lab helps you follow through how to read the content and which PORTx is being configure and how the address, and content value is being store inside the register in the memory.

SOURCE CODE

Provide a copy of the source code. Here you should use a mono-spaced font and can go down to 8-pt in order to make it fit. Sometimes the conversion from standard ASCII to a word document may mess up the formatting. Make sure to reformate the code so it looks nice and is readable.

```
;***********************************************************
;*
;*       Lab3Sample.asm
;*
;*       This is a sample ASM program, meant to be run only via
;*       simulation. First, four registers are loaded with certain
;*       values. Then, while the simulation is paused, the user
;*       must copy these values into the data memory. Finally, a
;*       function is called, which performs an operation, using
;*       the previously-entered values in memory as input.
;*
;***********************************************************
;*
;*        Author: Tu Lam
;*          Date: October 18, 2020
;*
;***********************************************************
```

```
.include "m128def.inc"                          ; Include definition file

;***********************************************************
;*      Internal Register Definitions and Constants
;***********************************************************

.def    mpr = r16
.def    i = r17
.def    A = r18
.def    B = r19

;***********************************************************
;*      Start of Code Segment
;***********************************************************
.cseg                                           ; Beginning of code segment

;***********************************************************
;*      Interrupt Vectors
;***********************************************************
.org    $0000                                   ; Beginning of IVs
                rjmp    INIT                    ; Reset interrupt

.org    $0046                                   ; End of Interrupt Vectors

;***********************************************************
;*      Program Initialization
;***********************************************************
INIT:                                           ; The initialization routine
                ldi             mpr, low(RAMEND)        ; initialize Stack Pointer
                out             SPL, mpr
                ldi             mpr, high(RAMEND)
                out             SPH, mpr

;***********************************************************
;*      Main Program
;***********************************************************
MAIN:
                clr             r0                      ; *** SET BREAKPOINT HERE *** (#1)
                dec             r0                      ; initialize r0 value


                clr             r1                      ; *** SET BREAKPOINT HERE *** (#2)
                ldi             i, $04
LOOP:           lsl             r1                      ; initialize r1 value
                inc             r1
                lsl             r1
                dec             i
                brne    LOOP                            ; *** SET BREAKPOINT HERE *** (#3)


                clr             r2                      ; *** SET BREAKPOINT HERE *** (#4)
                ldi             i, $0F
LOOP2:          inc             r2                      ; initialize r2 value
                cp              r2, i
                brne    LOOP2                           ; *** SET BREAKPOINT HERE *** (#5)

                                                        ; initialize r3 value
                mov             r3, r2                  ; *** SET BREAKPOINT HERE *** (#6)

                ;               Note: At this point, you need to enter several values
                ;               directly into the Data Memory. FUNCTION is written to
                ;               expect memory locations $0101:$0100 and $0103:$0102
                ;               to represent two 16-bit operands.
                ;
                ;               So at this point, the contents of r0, r1, r2, and r3
                ;               MUST be manually typed into Data Memory locations
                ;               $0100, $0101, $0102, and $0103 respectively.
```

```
                                                    ; call FUNCTION
                rcall   FUNCTION                    ; *** SET BREAKPOINT HERE *** (#7)

                                                    ; infinite loop at end of MAIN
 DONE:  rjmp    DONE

;***********************************************************
;*      Functions and Subroutines
;***********************************************************

;-----------------------------------------------------------
; Func: FUNCTION
; Desc: ???
;-----------------------------------------------------------
FUNCTION:
                ldi             XL, $00
                ldi             XH, $01
                ldi             YL, $02
                ldi             YH, $01
                ldi             ZL, $04
                ldi             ZH, $01
                ld              A, X+
                ld              B, Y+
                add             B, A
                st              Z+, B
                ld              A, X
                ld              B, Y
                adc             B, A
                st              Z+, B
                brcc    EXIT
                st              Z, XH
EXIT:
                ret                                 ; return from rcall
```