ECE 375 Lab 3

Introduction to AVR Simulation with Atmel Studio

Lab Time: Wednesday 12-2

Bradley Martin

QUESTIONS

1. *What is the initial value of DDRB?*

   The value of DDRB is set to 0x00.

2. *What is the initial value of PORTB?*

   The Value of PORTB is set to 0x00.

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

   Based on the answers above this is telling me that PORT B's default I/O configuration is NULL/zero.

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

   The stack pointer is initialized to 0x10FF.

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

   Register r0 has 0xFF

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

   The code inside LOOP ran 4 times.

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

   I would modify ldi of i which was set to $04.

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

   The contents of r1 are 0xFF.

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

   The contents of r2 are 0x0F.

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

    The contents of r3 are 0x0F

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

    The stack pointer now has the value 0x10FD.

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

    The contents of $0104 are 0x0e and the contents of $0105 are 0x0f.

SOURCE CODE

```
;***********************************************************
;*
;*      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: Taylor Johnson
;*         Date: January 15th, 2016
;*
;***********************************************************

.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
```