ECE 375
Computer Organization and Assembly Language Programming
Winter 2022
Assignment #3

[25 pts]

1- Consider the AVR code segment shown below that initializes and handles interrupts.
   (a) Explain in words what the code accomplishes when it is executed. That is, explain what it does and how it does it.
   (b) Write and explain the interrupt initialization code (lines 1-7) necessary to make the interrupt service routine (starting at ISR:) work properly. More specifically,
      (i) Fill in lines (1-2) with the necessary code to set the interrupt in question to detect an interrupt on a rising edge.
      (ii) Fill in lines (3-4) with the necessary code to mask out all other interrupts except the interrupt in question.
      (iii) Fill in lines (5-6) with the necessary code to set the port in question for input.
      (iv) Fill in line (7) to enable interrupt.

```
.include “m128def.inc”
.def   mpr = r16                ; Multi-purpose register
.def count = r17                ; Assume R17 is initially 0

.ORG   $0000
START: RJMP INIT
.ORG   $0002
       JMP ISR
.ORG   $0046
INIT:  ________________________(1)
       ________________________(2)
       ________________________(3)
       ________________________(4)
       ________________________(5)
       ________________________(6)
       ________________________(7)
       LDI   XH, high(CTR)
       LDI   XL, low(CTR)
       LDI   YH, high(DATA)
       LDI   YL, low(DATA)
WAIT:  RJMP  WAIT
.ORG   $100F
ISR:   IN    mpr, PINA
       ST    Y+, mpr
       INC   count
       ST    X, count
       RETI
.DSEG
CTR:   .BYTE 1
DATA:  .BYTE 256
```

[25 pts]

2- The AVR code below (with some missing information) is designed to initialize and service interrupts from three I/O devices (DevA, DevB, and DevC).

(a) There are 8 external interrupt pins (INT7-INT0) in AVR. Which three interrupt pins are these I/O devices connected to?

(b) Which I/O device's interrupt is detected on a falling edge?

(c) The interrupt pins referred to in part (a) are connected to two of the 7 ports (PORTA-PORTG) in AVR. Which ports are they?

(d) There are important instructions missing in lines (1-2) of the code. Fill in the missing instructions in lines (1-2) so that the code will work correctly.

(e) Suppose DevA requires that no interrupts are detected while it is being serviced. Fill in lines (3-4) with the necessary code to clear any latched interrupts at the end of `ISR_DevA`.

(f) Suppose interrupts are detected from all three interrupt pins at the same time. Which subroutine (`ISR_DevA`, `ISR_DevB`, or `ISR_DevC`) will be executed first?

```
.include “m128def.inc”
.def   mpr = r16
START:
.org   $0000
   RJMP   INIT
.org   $0002
   JMP    ISR_DevA
.org   $0008
   JMP    ISR_DevB
.org   $000C
   JMP    ISR_DevC

INIT:
   ldi    mpr, 0b10000011
   sts    EICRA, mpr
   ldi    mpr, 0b00001100
   out    EICRB, mpr
   ldi    mpr, __________
   out    EIMSK, mpr
   ldi    mpr, $00
   out    DDRD, mpr
   ________________________(1)
   ldi    mpr,0b00001000
   ________________________(2)
   sei
      …
MAIN:
   {  …
      …do something…
   }

ISR_DevA:
   …
   ________________________(3)
   ________________________(4)
   RETI

ISR_DevB:
   {…
   …
   RETI
   }
ISR_DevC:
   {…
   …
   RETI
   }
```

[25 pts]

3- Write an AVR assembly code that waits for 1 sec using the 8-bit Timer/Counter0 with the system clock frequency of 16 MHz operating under Normal mode. This is done by doing the following:

(1) Timer/Counter0 is initialized to count for 10 ms and then interrupts on an overflow;
(2) The main part of the program simply loops, and for each iteration, tests whether the interrupt has occurred 100 times; and
(3) On each interrupt, Timer/Counter0 is reloaded to interrupt again in 10 ms.

Use the skeleton code shown below:

```
.include “m128def.inc”
.def mpr = r16
.def counter = r17
…
.ORG     $0000
    RJMP  Initialize
.ORG     $0020          ; Timer/Counter0 overflow interrupt vector
    JMP   Reload_Counter

.ORG     $0046          ; End of interrupt vectors
Initialize:
    …
    …Your code goes here…
    …

LOOP:
    …
    …Your code goes here…
    …

Reload_counter:
    …
    …Your code goes here…
    …
    RETI
```

[25 pts]

4- Write a subroutine `initUSART1` to configure ATmega128 USART1 to operate as a transmitter and sends a data every time USART1 Data Register Empty interrupt occurs. The transmitter operates with the following settings:

- 8 data bits, 2 stop bits, and even parity
- 9,600 Baud rate
- Transmitter enabled
- Normal asynchronous mode operation
- Interrupt enabled

Assume the system clock is 16 MHz. The skeleton code is shown below:

```
.include “m128def.inc”
.def mpr = r16
.ORG $0000
     RJMP  initUSART1
…
.ORG $003E
     JMP   SendData
…
.ORG $0046
initUSART1:
     …
     …Your code goes here…
     …
Main:
     ld   mpr, X+         ; Send first data
     sts   UDR1, mpr
Loop:
     RJMP Loop

SendData:
     ld    mpr, X+          ; Send next data
     sts   UDR1, mpr
     reti
```