CEE-345 Microprocessor System Design Spring 2019

Midterm Exam

WRITE YOUR NAME HERE: **Jacob Hillebrand**

Please use the drop box on D2L to turn in this test

1. **(12 pts)** There are 6 assembly instructions used for Atmel processors in the table below. Place an ‘**x**’ mark in an appropriate row to indicate an illegal instruction and mark ‘**o**’ to indicate a legal instruction.

|  |  |
| --- | --- |
| Instruction | Illegal (x) or Legal (o) instruction |
| LDI XH,0x0135 | **x** |
| ST Y, R16 | **o** |
| LDI R15, 0x01 | **x** |
| LDI R16,$255 | **o** |
| LDI R16, ‘A’ | **x** |
| OUT DDRB, 0xFF | **o** |

1. Assume the program below is built and run in Atmel Studio (AS). Find the hexadecimal value in that specific register as you step through each instruction execution in AS.

.ORG 0

LDI R16, 0xDF

MOV R17, R16

ROR R17

LDI R18, 0xCB

MOV R19, R18

ROL R19

STS 0x114, R19

LDI XL, 0x14

LDI XH, 0x01

LD R20, X

1. **(3 pts)**

|  |  |
| --- | --- |
| Register | Value in hexadecimal format |
| R17 value after the execution of ROR instruction | 0xEF |

1. **(3 pts)**

|  |  |
| --- | --- |
| Register | Value in hexadecimal format |
| R19 value after the execution of the ROL instruction | 0x97 |

1. **(4 pts)**

|  |  |
| --- | --- |
| Register | Value in hexadecimal format |
| R20 value after the execution of the LD instruction | 0x79 |

\*Note: Because we loaded R19 into the X register backwards

1. **(12 pts)** Find each data value in the specific register as you step through the execution of the following code.

#include <avr/io.h>

x1 = 0b00000000; **0x00**

x2 = 0b11111111**; 0xFF**

x3 = 0b01111010; **0x7A**

x4 = 0b10101010; **0xAA**

int main (void)

{

DDRB = 0xFF;

**PORTA = 0x74 & 0xFF;**

PORTB = 0x31 | 0x84;

**PORTC = 0x34 ^ 0x88;**

PORTD = ~0x55;

**x1 |= (1 << 3);**

**x2 &= ~(1 << 5);**

**x3 ^= (1 << 4);**

**x4 = (x4 >> 2) & 0x0F**

}

|  |  |
| --- | --- |
| Data Register | Value in hexadecimal format |
| PORTA | 0x74 |
| PORTB | 0xB5 |
| PORTC | 0xBC |
| PORTD | 0xAA |
| x1 | 0x08 |
| x2 | 0xDF |
| x3 | 0x6A |
| x4 | 0x0A |

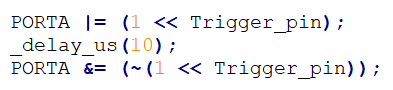
1. The Atmega32 microcontroller has a10-bit analog to digital converter (ADC). Please answer the following questions.
   1. **(5 pts)** How many discrete analog output levels can a 10-bit ADC detect?

**1024 discrete analog output levels (2^10)**

* 1. **(5 pts)** A 8-bit ADC with 0 to 5V **voltage** range is used to sample an analog sensor's **voltage.** What is voltage value for each step size of this ADC?

**5/(2^8)=0.01953125 V or 19.5 mV**

1. **(6 pts)** Use your own words to describe what the code does below.



Ans:

**This operation first shifts the value 1 (0x01) by the value of “Trigger\_pin” to the left, then logical “or’s” it with the current value of “PORTA” and assigns the result to “PORTA”. Then, a 10 (microsecond, I’m guessing?) delay is used before the final part. In the final part, 1 (0x01) is again bit-shifted left by the value of “Trigger\_pin”. A logical “NOT” is applied to that value, which is then logical “AND’ed” with the value of “PORTA”, and that final value is assigned to “PORTA”.**

* 1. 

Ans:

**This instruction will set the Global interrupt flag in the status register, which will force the instruction directly following this one to be executed before any other pending system interrupts**.

1. **(18 pts)** Find the values in the data registers in the table below. Record each register value immediately after the execution of each instruction. Assume the stack memory register RAMEND has the address at 0x085f

**.equ C1 = 6**

**.equ C2 = 4**

**.equ C3 = 0x80**

**.equ C4 = 0x40**

**.equ C5 = 0x14**

**.equ RAMEND = 0x085f**

**LDI R20, (2<<C1) | (3<<C2)**

**LDI R21, 0b101<<3**

**LDI R22, 0b100>>2**

**LDI R16, (C3&C4)| C5**

**LDI R17, 0x54^0x78**

**LDI R18, LOW(RAMEND)**

**LDI R19, HIGH(RAMEND)**

**LDI R20, ~(1<<6)**

**LDI R21, 0xff & ~(1<<4)**

|  |  |
| --- | --- |
| Register | Value in binary or Hex |
| R20 | 0xB0 |
| R21 | 0b101 |
| R22 | 0b001 |
| R16 | 0x14 |
| R17 | 0x2C |
| R18 | 0x5F |
| R19 | 0x08 |
| R20 | 0xBF |
| R21 | 0xEF |

1. Assume there are 8 low-enabled LEDs and a C program outputting a low signal turns on a LED on the STK-600 board with LED7 being the most significant bit position of the LED array. Also, assume the buttons interface to the STK-600 use the circuit shown in Fig. 1.
   1. **(10 pts)** Create C code including a processor’s header file and its port connections that light up the rightmost LED0 of the 8 LED array on the STK-600 board when the buttons (SW0) (SW1) are both pressed and stay pressed. Releasing either one of the buttons will turn off the LED.

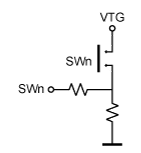
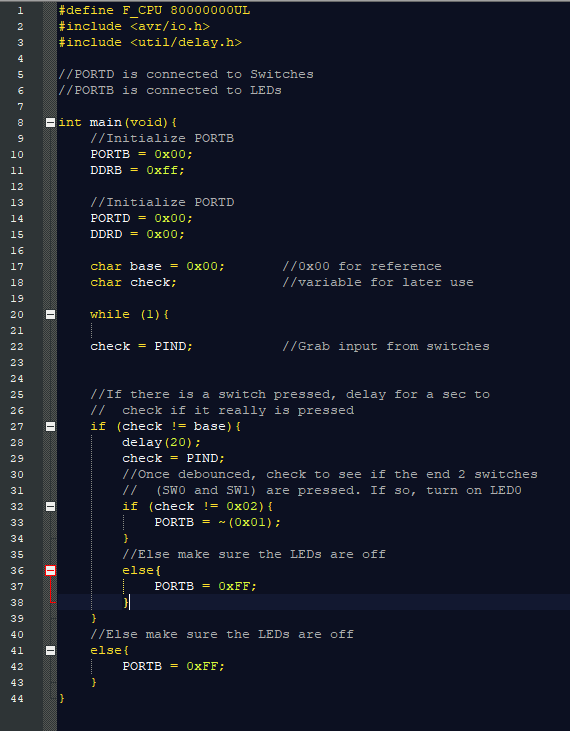
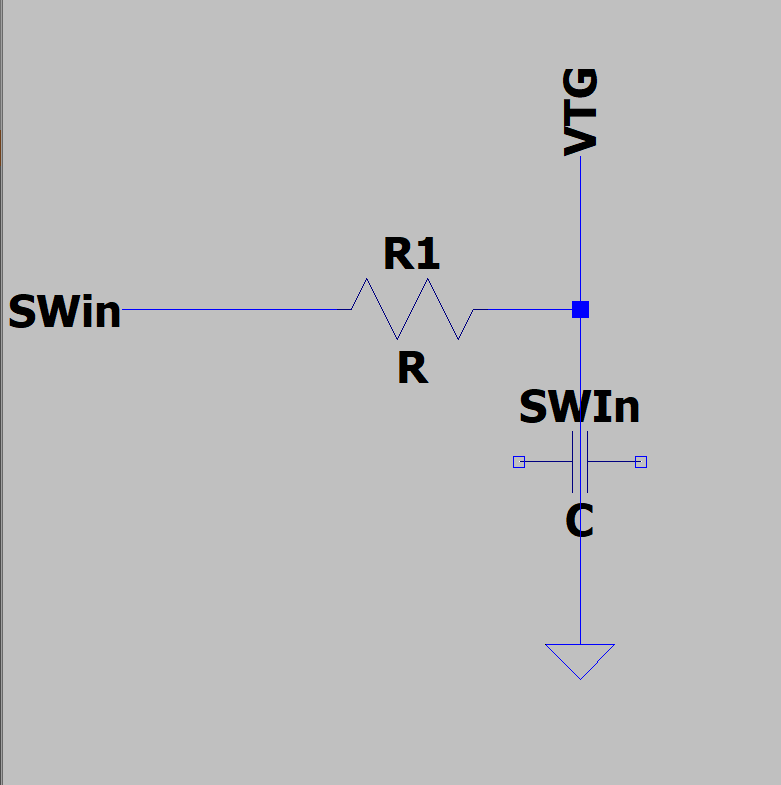
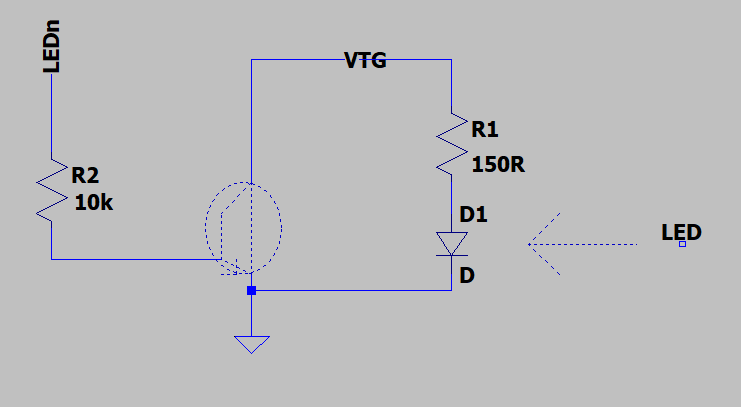


Fig. 1 Button Interface Design

****

* 1. **(5 pts)** Fig. 1 has the output node (SWn) and the circuit produces a logic ‘HIGH’ at SWn when the push Button (SWn) is pressed. This output node produces a logic ‘LOW’ when the push button is NOT pressed. Modify Fig. 1 circuit such that the output node (SWn) produces a ‘LOW” output when SWn is pressed, and produces a ‘HIGH’ output when SWn is not pressed.

1. **(5 pts)** In Fig. 2, VTG node receives a 5Vdc source. A logic ‘HIGH’ signal at LEDn node will light up the LED. A logic ‘LOW’ signal at LEDn node will turn off the LED. Modify Fig. 2 circuit such that a logic LOW’ signal at LEDn node lights up the LED and a logic ‘HIGH’ at LEDn node turns off the LED.

**Adjusted Circuit**

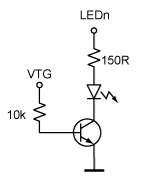


Fig. 2 Active high (high enabled) LED

1. **(12 pts)** Write a complete C program including a processor header files and its port connections using the STK-600 board. You need to use the **switches and LEDs**, and then create a C project using the **shift operators** to implement a Knight-Rider LED blinking patterns as described in the table below. The program allows users to control LEDs such that the three blinking patterns are created. The users will use the two buttons (SW0 and SW1) on the STK-600 board to select the LED patterns.

|  |  |  |
| --- | --- | --- |
| SW1 SW0 | | LED Pattern |
| 1 | 1 | LED shifts from right to left |
| 1 | 0 | LED shifts from left to right and then from right to left |
| 0 | 0 | LED shifts from left to right |

Pattern 1: LEDs light up from the leftmost LED (LED7) to the rightmost LED (LED0) similar to the Knight-Rider lab but LEDs are shifting in a single direction when the button (SW0) and (SW1) are both pressed and stay pressed on the STK-600 board. The left-to-right rotating LEDs should wrap around but turn off when the button (SW0) and (SW1) are both released. Only one LED allowed to be lit at a time when the LEDs are running.

Pattern 2: similar to the pattern 1 from above, the LEDs light up from the leftmost LEDs to the rightmost LED, pause a few seconds (required to be observable), and then LEDs light up from rightmost LED to the leftmost LED when SW1 button is pressed down and stay pressed. This is similar to the Knight-Rider lab and adding two push buttons (SW0 and SW1) to display the Knight-Rider LEDs. Only one LED allowed to be lit at a time when the LEDs are running. The rotating LEDs should wrap around but the LEDs stay off when buttons (SW0) and (SW1) are both released. Only one LED allowed to be lit at a time when the LEDs are running.

Pattern 3: similar to pattern 1 above, the LEDs light up from the rightmost LED (LED0) to leftmost LED (LED7) and then wrap around when button SW0 and SW1 are not pressed. Only one LED allowed to be lit at a time when the LEDs are running.

Work space

