Name (2 pts.):

Problem 1. Warmups (Circle all answers that apply).

a) (5pts). Where does malloc allocate memory?
   (a) On the stack
   (b) On the heap
   (c) In global memory
   (d) In SRAM

b) (5pts). What clock speed does our Arduino operate at?
   (a) 8MHz
   (b) 12MHz
   (c) 16MHz
   (d) 20MHz

c) (5pts). If an LED connected to our Arduino drops 1.0V and there is a 100 ohm resistor connected in series with it, what current will be flowing through it?
   (a) 1mA
   (b) 10mA
   (c) 20mA
   (d) 40mA

d) (5pts). To enable the pullup on pin 7/PD7 when it is already configured as an input you would do:
   (a) pinMode(PD7,INPUT_PULLUP);
   (b) digitalWrite(7,HIGH);
   (c) PORTD |= (1<<7);
   (d) DDRD &= ~(1<<7);
(e) (5pts). If the Atmel processor on our Arduino is using 10mA, how much power is it using?
   (a) 50 mW
   (b) 10 Joules
   (c) 33 mW
   (d) 2 mW

(f) (5pts). What causes a button to “bounce” when pressed?
   (a) No pullup resistor
   (b) Button that is incorrectly connected
   (c) Reading the pin in analog mode
   (d) Inductance and capacitance of the wires/buttons/etc

(g) (5pts). How do you write the value 0x23 to memory address 0x36?
   (a) (*((volatile uint8_t *) 0x36)) = 0x23
   (b) (*((volatile uint8_t &)) 0x36)) = 0x23
   (c) (*((volatile uint8_t *) 0x36)) = 0x23
   (d) (*((volatile uint8_t &)) 0x36)) = 0x23

(h) (5pts). If an operation with an 8-bit variable takes 10 clock cycles, approximately how many clock cycles would it take with a 16-bit variable?
   (a) 10
   (b) 20
   (c) 22
   (d) 30

(i) (5pts). An Instruction Set Architecture defines which of the following for a processor:
   (a) The maximum number of CPU registers
   (b) The maximum number if I/O pins supported
   (c) The number of clock cycles an operation will take
   (d) The type of operations available
Problem 2. Hex and bit operations (all references to bit locations are zero referenced). For each bit operation subproblem write a single line of C code to achieve the desired result.

a) (5 pts.) Clear bit 2 in the variable var.

b) (5 pts.) What is the value of ((((0xA>>2) | (3<<5)) + 2) in hex?

c) (5 pts.) Set bits 3-5 (inclusive) in the variable var to bits 5-7 in the 16-bit variable config. As with all of these problems, do this in a single line of code (e.g. var=...). The bit references are zero indexed.
Problem 3. Memory operations. Refer to the following code example for this question:

```c
uint16_t globalVar = 0x236;
uint8_t data[] = {0,1,2,3,4,5,6,7};
uint8_t *globalPtr;

int8_t main(void){
    uint8_t var = 0x10;
    uint8_t *p1 = data + 4;

    p1[1] = 4;
    *(p1+4) = data[0]++;
    var += globalVar;
    data[3] = p1[2];

    //Draw Memory Map Here
    return 0;
}
```

a) (10 pts.) Fill in the below memory map for the above code after execution has reached the statement `Draw Memory Map Here`. Assumptions you should make: 1) the compiler allocates the memory in the order the statements appear; 2) global variables are allocated starting with the high address; 3) the stack starts at zero; 4) each memory location stores 8 bits; and 4) the memory is little endian (least significant byte stored at lowest address).

<table>
<thead>
<tr>
<th>Address</th>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
b) (5 pts.). What is Harvard memory architecture? What are two advantages of using a Harvard architecture over von Neumann?

Problem 4. Timer/Counter. For these questions, see the provided register description pages for Timer1 and assume this is running on the Arduino we use in class. The timer is configured as:

```
TCCR1A = 0;
TCCR1B = (1 << CS11) | (1 << CS10);
TCCR1C = 0;
TCNT1 = 0;
```

a) (5 pts.). How many seconds will it take before the timer will roll over (reset to zero). Leave your answer as a fraction.

b) (5 pts.). Write the code to configure the registers to count the number of times a button on pin T1 has been released.
Problem 5. Digital I/O

a) (5 pts.). Write the C code to set pin PB5 to an input pin with the internal pullup resistor enabled by directly writing to the processor registers (recall the I/O registers have the general names DDRx, PORTx, PINx).

b) (5 pts.). Describe two different approaches to debouncing a button? Which approach is better and why?

Problem 6 (3 pts.). This is a harder problem that is worth only 3 points, so only attempt it once you are happy with the rest of your answers. Make sure to explain your answer, otherwise you will not get any points for this. There is also no partial credit for this problem, it is either right or wrong.

There is a function that you need to measure exactly (to the clock cycle) how long it takes during each execution. The challenge is that this function will have different runtimes each execution since it is also responding to external events and you cannot add any extra hardware to your Arduino. You have determined that the longest it ever takes to execute is 6 milliseconds. Write the code to measure this function and also describe how this works. **Hint:** The 16-bit timer can only measure about 4 milliseconds in its highest resolution mode (tick per clock cycle). In addition, approaches like using millis(), micros(), or other methods that rely on interrupts will not work since an interrupt will disturb the function we are trying to measure.