EXAM #3

- You may use a non-programming calculator only. You may use only the provided reference materials.
- Recall that the ‘d’ bit in a machine word indicating a destination is ‘0’ if the destination is W, is ‘1’ if the destination is the file register. For the ‘a’ bit, use the assumptions we have used in class (‘a’ bit is ‘0’ if address in ACCESS RAM, ‘a’ bit is ‘1’ if not in ACCESS RAM). Addresses 0x000 – 0x07F and 0xF80-0xFFF, which lie in access RAM, are automatically assigned an a = 0 by the assembler, while all other addresses lie in banked memory and are therefore assigned a = 1 by the assembler.
- All instructions which require a ‘d’ bit MUST end with a “, w” or “, f”. Unlike the assembler, no default destination will be assumed and this portion of your solution will be marked as incorrect.
- Unless stated otherwise, all multi-byte data values are stored in little-endian ordering.
- Please note the relative value of each problem in the table below.
- Answers should be clearly indicated. Placing them in a BOX would be ideal.
- Be as neat and well organized as possible. This is in your grade’s best interest.
- If you need additional space to work, do so on the backside of the page. Make sure it is clear where your work continues.
- Absolutely NO cheating is allowed. If you are caught in the attempt of, the act of, or the past action of academic dishonesty, you will receive the maximum punishment allowed by University policy.
- No panicking allowed!

<table>
<thead>
<tr>
<th>Page</th>
<th>Maximum</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
Part I: (84 pts)

a. (4 pts) Write C code that configures PORTB for use with an LED on RB7 and RB5 and a pushbutton switch on RB3 and RB1. The internal weak pullup must be enabled. Do not assume any default bit values.

b. (6 pts) How many bytes can be sent in 5 seconds assuming a baud rate of 9600, and an asynchronous data format of 1 start bit, 8 data bits, and 1 stop bit assuming the bytes are sent as fast as possible?

c. (6 pts) Write C code that implements the `char getch(void)` function (read one character from the serial port). No interrupts are enabled.

d. (8 pts) Explain what happens in the code below by specifying what appears on the console after the PIC is powered on, and justify your answer by explaining the sequence of events. You must clearly specify if output continually appears on the console or if at some point it stops.

```c
main() {  
    char c;  
    serial_init(95,1);  
    if (!POR) {  
        POR = 1;  
        SWDTEN = 1;  
        printf("Hello!");pcrlf();  
    } else  
        SWDTEN = 0;  
    if(SWDTEN) {  
        asm("sleep");  
    }  
    while (1) {  
        printf("Looping");pcrlf();  
    }  
} //end main()
```
e. (8 points) Implement the \textit{intIncCopy()} function given below. Assume FSR0 already contains the pointer value for \textit{int *dest} on function entry but that the pointer value for \textit{int *src} is passed in the CBLOCK. In the subroutine, use FSR1 to implement the pointer operations for \textit{int *src}. Note that \textit{dest} and \textit{src} are ints.

\begin{verbatim}
void intIncCopy(unsigned int* dest,
                unsigned int* src, unsigned char length)
{
    while (length)
    {
        *dest = *src + 1;
        dest++;
        src++;
        length--;
    }
}
\end{verbatim}

f. (8 points) Implement the \textit{main()} code below in PIC assembly. Pass the value for \textit{int *src} directly in FSR0; pass the value for \textit{int *src} and \textit{char length} using the CBLOCK space for \textit{intIncCopy}.

\begin{verbatim}
main()
{
    int array1[5], array2[5];

    // Code to initialize array2
    intIncCopy(array1, array2, 5);
}
\end{verbatim}
For the following instructions, assume the memory contents BEFORE THE EXECUTION OF EACH CODE SNIPPET is given in the box below.

<table>
<thead>
<tr>
<th>Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x059</td>
<td>0xA8</td>
</tr>
<tr>
<td>0x05A</td>
<td>0x30</td>
</tr>
<tr>
<td>0x05B</td>
<td>0xF2</td>
</tr>
<tr>
<td>0x05C</td>
<td>0x6E</td>
</tr>
<tr>
<td>W register</td>
<td>0x02</td>
</tr>
</tbody>
</table>

**g.** (6 pts) Give the final contents of any changed REGISTERS or MEM locations.

```
    lfsr   FSR0, 0x05A  FSR0 = ________
    movff  PREINC0, 0x059
```

Location _____ = _______

**h.** (6 pts) Give the final contents of any changed REGISTERS or MEM locations.

```
    movlw  0x59    FSR0 = ________
    movwf  FSR0L
    clrf   FSR0H
    movff  0x05C, INDF0  Location _____ = _______
```

**i.** (6 pts) Give the final contents of any changed REGISTERS or MEM locations.

```
    lfsr   FSR0, 0x05A  FSR0 = ________
    movff  POSTDEC0, 0x059
```

Location _____ = _______

**j.** (6 pts) Give the final contents of any changed REGISTERS or MEM locations.

```
    lfsr   FSR0, 0x05A  FSR0 = ________
    decf   PLUSWO, f   Location _____ = _______
```
h. (20 points) After the execution of ALL of the C code below, fill in the memory location values. Assume little-
endian order for multi-byte values.

```c
signed char a[2];
signed int b;
signed long c;
signed int *ptrb;

a[0] = 3;
a[1] = -7;
ptrb = &a[0];
b = *ptrb + 2;
ptrb++;
c = *ptrb >> 1;  // Computes *ptr >> 1 as int, then converts to a long
```

<table>
<thead>
<tr>
<th>Location</th>
<th>Contents (MUST be given in hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1A0</td>
<td></td>
</tr>
<tr>
<td>0x1A1</td>
<td></td>
</tr>
<tr>
<td>0x1A2</td>
<td></td>
</tr>
<tr>
<td>0x1A3</td>
<td></td>
</tr>
<tr>
<td>0x1A4</td>
<td></td>
</tr>
<tr>
<td>0x1A5</td>
<td></td>
</tr>
<tr>
<td>0x1A6</td>
<td></td>
</tr>
<tr>
<td>0x1A7</td>
<td></td>
</tr>
<tr>
<td>0x1A8</td>
<td></td>
</tr>
<tr>
<td>0x1A9</td>
<td></td>
</tr>
</tbody>
</table>

CBLOCK 0xA0
a:2, b:2, c:4, ptrb:2
ENDC
Part II: (16 points) Answer the 4 out of the next 5 questions. Each question is worth 4 points. Cross out the question you do not want graded.

a. Write PIC18 assembly code for the C statement below assuming that FSR0 already contains the address of “ptr”.

```c
long *ptr;
ptr++;
```

b. Draw the waveform for the value 0x37 using 1 start bit, 1 stop bit, and 8 data bits as transmitted by the PIC from its TX pin using asynchronous serial IO. Label each bit and indicate which bit is the start bit and which bit is the stop bit.

c. Give the value of the SPBRG register required to communicate at 9600 baud with Fosc = 10 MHz. State the value of the BRGH bit used to achieve this baud rate.

d. What inputs cause a tri-state buffer and an open-drain buffer to produce the same output (i.e. drive to Vdd, drive to ground, high-impedence)? Label the inputs and outputs in the diagram below.

```
<table>
<thead>
<tr>
<th>EN =</th>
<th>IN =</th>
<th>OUT =</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Tri-state buffer

```
<table>
<thead>
<tr>
<th>EN =</th>
<th>IN =</th>
<th>OUT =</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Open-drain buffer

```
<table>
<thead>
<tr>
<th>EN =</th>
<th>IN =</th>
<th>OUT =</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Same output

e. Draw a pushbutton connected to a PIC18 port that will provide a logic ‘1’ when the switch is pressed, and a logic ‘0’ when the switch is released. Your connection must be electrically viable, i.e, you cannot produce a short between VDD and GND or any other undesirable condition.