Directions  There are 6 problems worth a total of 100 points. The number of points for each question is indicated. Make sure that you show all work since partial credit will be given. This exam is closed book, closed notes, and no calculators are allowed. You may not communicate in any way with anyone other than exam proctors during the exam time. Don’t forget to put your name down on the first page.

You have 3 hours to finish the exam

Name: __________________________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points possible</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Problem 2</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Problem 5</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Problem 6</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
1. (15 points) Appropriately Interface a switch and an LED to pins PA0 and PA1 respectively so when the following code snippet is run the relationship between the switch and LED is a NOT logic. That is when the switch is on the LED is off and vice-versa. The LED’s operating point is $V_d = 1.6V, I_d = 1mA$. You may or may not need all the circuit elements provided.

```c
#define PA0 (*((volatile unsigned long *)0x40004004))
#define PA1 (*((volatile unsigned long *)0x40004008))
int main() {
    ... // Port A pin 0 is input and pin 1 is output
    while (1) {
        PA1 = PA0; // write switch logic state to LED
    }
}
```

(a) What is the value of resistance $R$ used to interface the LED? 
(b) What interface logic does did you use to connect the switch to the microcontroller? 
(c) What interface logic does did you use to connect the LED to the microcontroller?
2. (15 points) There are two parts to this question. First, complete the C subroutine below that takes a string of characters that are null terminated and returns the checksum of these characters. The checksum of a list of characters is defined as the bit-wise Exclusive-Or of all the characters together.

**Hint:** To perform the XOR of multiple characters you can do a bit-wise XOR of the first two characters and then perform a bit-wise XOR of the result with the third character and so on.

```c
unsigned char ChkSum(unsigned char *array) {
    // Note: In C, arrays are implemented using pointers where an array is just a
    // pointer to the first element. array[i] access the i’th element of the array
}
```

For the second part, you are required to write assembly code that calls (call-by-reference) the above C subroutine from assembly. Include appropriate IMPORT and/or EXPORT statements and follow the AAPCS calling convention.

```assembly
input SPACE 60 ; Storage for a null-terminated char array

; Assume input is initialized here and null terminated (you don’t have to do)

; Setup call to ChkSum by passing it the appropriate information (call-by-reference)

JSR ChkSum
```


3. (20 points) In the following code, the stack is used for passing an input parameter, a return value, local variables, and, to save registers that the subroutine modifies. The subroutine makes use of two local variables, which are accessed using the stack pointer (SP) as the frame pointer. Read the comments to understand the code and pay attention to instructions to figure out the sizes of input/output parameters and variables. Note, the code below does not use AAPCS convention.

```assembly
AREA DATA ; Memory address 0x2000.0000
Result SPACE 4
AREA \text{.text}, CODE, READONLY, ALIGN=2 ; Memory address 0x0000.0100
Count DCW 10

main
LDR R0,=Count
LDRH R1,[R0]
STRH R1,[SP],#-2 ; Pass input on stack and post-decrement SP by 2
SUB SP, #4 ; Allocate space for return value on stack
JSR Sub
LDR R1, [SP, #nnn] ; get return value into reg R1
LDR R0,=Result
STR R1, [R0] ; put returned value into global variable Result
ADD SP, #6 ; De-Allocate input and output space
lp B lp ; Loop indefinitely

;****************Sub*************
inp EQU aaa ; binding for input parameter
out EQU bbb ; binding for return value
sum EQU ccc ; binding for local variable sum
num EQU ddd ; binding for local variable num

Sub
PUSH {R0,R1} ; save registers
SUB SP, #6 ; Allocate space for sum and num
LDRH R0, [SP, #inp] ; get a copy of input parameter into R0
STRH R0, [SP, #num] ; set local variable num to passed input
MUL R1, R0, R0
STR R1, [SP, #sum] ; change sum
LDR R0, [SP, #sum]
----> STR R0, [SP, #out] ; store sum into out to return
ADD SP, #6 ; De-allocate locals
POP {R0,R1} ; restore registers
BX LR
ALIGN
END
```

(a) (10 points) What are the values of \texttt{aaa}, \texttt{bbb}, \texttt{ccc}, \texttt{ddd} and \texttt{nnn}?
(b) (10 points) Give the contents of the stack after line indicated by the arrow --- > is executed, along with where the Stack pointer (SP) points to. The stack state immediately preceding the subroutine call is shown.  

**Notes:** (i) Each entry in the table is 2-bytes, (ii) you may or may not need all the locations in the table, (iii) if a variable is 4 bytes then fill its value in multiple locations

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>SP - &gt;</td>
<td>out</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
</tbody>
</table>

4. (15 points) Answer the following

(a) (5 points) Given two boards communicating using UART0 with a baud-rate set to 40 Kbps (kilo bits per sec). Assuming the one start bit and one stop bit are used per byte, how much time will it take to transmit a text file of size 10Mbytes?

(b) (5 points) Assuming the Bus cycle speed of 50MHz, what values must the registers UART0_IBRD_R and UART0_FBRD_R be set to for a baud-rate of 40 Kbps?

(c) (5 points) Assuming the Bus cycle speed of 50MHz, what should the register NVIC_ST_RELOAD_R be set to for the Systick interrupt to occur every 100ms?
5. (20 points) Given the following Mealy FSM:

![FSM Diagram]

**Part a** (10 points): Fill the blanks in the C code below that implements the FSM:

```c
#define PA0 (*((volatile unsigned long *)0x40004004))
#define PA21 (*((volatile unsigned long *)0x40004018))
const struct State{
    ______________Out[4]; // Output to PA0
    unsigned short wait; // Wait time
    const struct State ______________; // Next states
};
typedef const struct State StateType;
typedef StateType * StatePtr;
#define S0 ______________
#define S1 ______________
fsm[_______] = {
    {______________________________}, // S0 Transition table
    {______________________________}  // S1 Transition table
};
_____________Pt; // Pointer to Current State
```

**Part b** (10 points): Assuming that the two bit input comes from pins PA1 and PA2 and the following Delay routine is given for you to call:

```c
void Delay(unsigned short); // Input is millisecs to wait
```

Complete the following FSM controller routine (delay->input->output->state_change):

```c
void FSMController()
{
    while(1) {
        
    }
}
```
6. (15 points) Answer the following

(a) (4 points) What is the purpose of the E pin in the HD44780 protocol for the LCD interface? What changes does one have to make to it when writing a character to the LCD?

(b) (4 points) In Lab8 where we had the background thread writing to the global variable ADCMail and the main program reading ADCMail. Why did this not cause a race condition?

(c) (4 points) If the input voltage is 1.0V, what value will the LM3S 10-bit ADC return?

(d) (2 points) If we want 25 unique levels from an n-bit DAC, what is the minimum n that can accomplish this?

(e) (1 point) We enable the ADC clock by setting bit 16 of the SYSCTL_RCGC0_R register. [True/False]