Midterm 1

Date: February 21, 2013

UT EID: _____

Printed Name: _____
Last, First

Your signature is your promise that you have not cheated and will not cheat on this exam, nor will you help others to cheat on this exam:

Signature: _____

Instructions:

- Closed book and closed notes.
- No calculators or any electronic devices (turn cell phones off).
- Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space (boxes) provided.
- *Anything outside the boxes will be ignored in grading.*
- For all questions, unless otherwise stated, find the most efficient (time, resources) solution.

Problem 1	10	
Problem 2	10	
Problem 3	15	
Problem 4	20	
Problem 5	30	
Problem 6	15	
Total	100	

Problem 1 (10 points): Numbers

(a) (5 points) How many bits are needed to represent all days in a year? What C data type should be used to store such values?

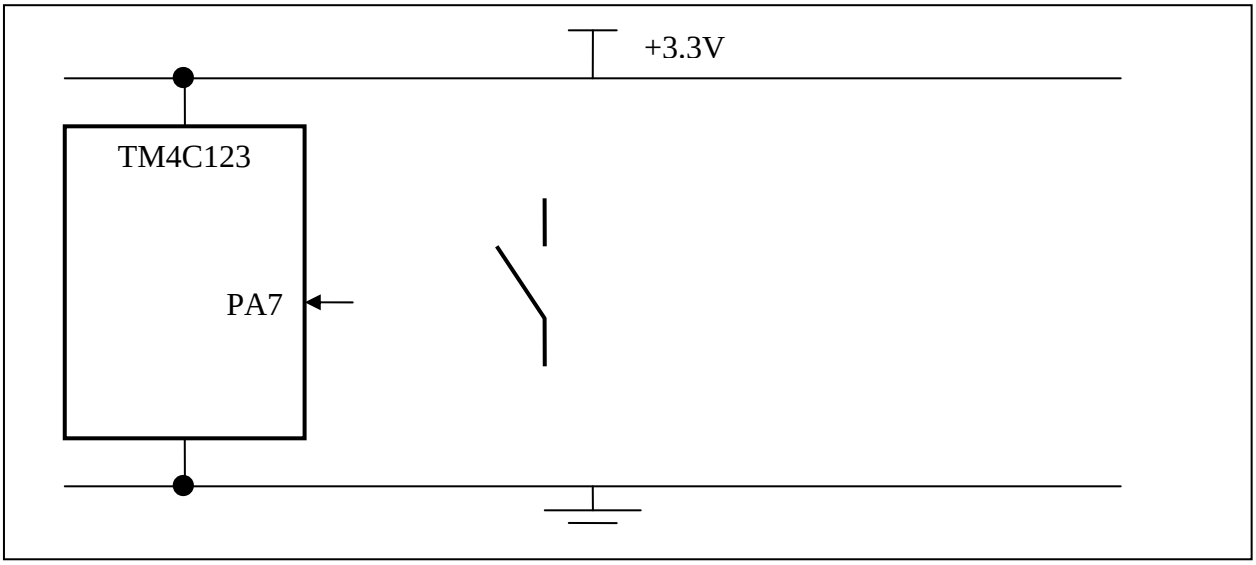
Number of Bits	
C Data Type	

(b) (5 points) What values has the 8-bit number 0x70 when converted to decimal and binary representations?

Signed decimal	Unsigned decimal	Binary

Problem 2 (10 points): Interfacing

Interface a switch to (input) port PA7 of the TM4C123 using negative logic. Assuming that no current can flow in or out of the TM4C123 and that the switch is perfect (zero resistance when closed), what current will flow through the switch when it is closed?



Problem 3 (15 points): Arithmetic and Addressing

(a) (5 points) For the following operation sequence, what will be the value of register R0 and condition code bits N, Z, V and C after execution of the sequence. Assume all values and registers are 8-bit wide:

8-bit sequence	R0	N	Z	V	C
R1 ← -111 R2 ← 221 R0 ← R1 + R2					

(b) (5 points) Consider the following operation sequence:

```
LDR   R1,=#-168
ASRS  R2,R1,#2
CMP   R1,R2
```

Mark which of the following branches will be taken after executing the above sequence:

Branch	Taken	Not taken
BEQ		
BHS		
BMI		
BLO		
BLT		

(c) (5 points) Consider the following assembly program:

```
       AREA CODE  ;assume this starts at address 0x0000.1000
num    DCD   0x87654321
Start  LDR   R0,=num
       LDRSH R1,[R0]
       LDRSB R2,[R0,#3]
       ADD   R3,R1,R2
```

What is the value in register R3 at the end of execution?

Problem 4 (20 points): Execution

Given the following ARM assembly program:

```
          AREA DATA ; assume this starts at address 0x2000.0000
  res     DCD  0

          AREA CODE ; assume this starts at address 0x0000.2000
  f       PUSH {LR}
          CMP  R0, #1
          BEQ  done
          PUSH {R0}
          SUB  R0, R0, #1
          BL   f
          POP  {R1}
          MUL  R0, R0, R1
  done    POP  {LR}
          BX   LR

          AREA CODE ; assume this starts at address 0x0000.4000
Start     MOV  R0, #2
          BL   f
          LDR  R1,=res
          STR  R0, [R1]
```

(a) (10 points) Assume the stack pointer SP is initialized to 0x2000.8000. Show the contents of the stack and indicate the location of the stack pointer right after the point when the statement at address 'f' has just been executed for the second time. Hint: every ARM instruction occupies exactly 4 bytes in memory.

0x2000.7FEC	
0x2000.7FF0	
0x2000.7FF4	
0x2000.7FF8	
0x2000.7FFC	

(b) (5 points) What is the value in memory location 'res' at the end of execution?

(c) (5 points) What general functionality does the subroutine 'f' implement?

Problem 5 (30 points): Input/Output

You are asked to develop a software module to control the seatbelt warning lamp as part of a car dashboard. For the part of the system that you are responsible for, the following inputs and outputs are relevant:

- Ports PB3…PB0 are connected to a RPM sensor that reports the current engine speed as a scaled (in units/increments of 500 RPM) unsigned 4-bit integer value, i.e. if the sensor reports a value of 2 on PB3…PB0, the engine speed is 1000 RPM.
- Port PB6 is connected to the seatbelt switch that indicates whether the seatbelt is fastened.
- Port PB7 is connected to the safety warning indicator LED.

Your subsystem is supposed to turn on the LED if the engine is running (RPM >= 1000) and the seatbelt is not fastened.

Since your code is part of a bigger system, make sure to develop subroutines that are friendly, i.e. that do not modify unrelated bits of ports. You can assume that relevant definitions are given:

```
GPIO_PORTB_DATA_R
GPIO_PORTB_DIR_R
GPIO_PORTB_AFSEL_R
GPIO_PORTB_DEN_R
SYSCTL_RCGCGPIO_R
```

(a) (10 points) Write the assembly code for the initialization subroutine of the *Belt* module. The *Belt_Init* subroutine should make PB7 an output, and PB6 and PB3 through PB0 inputs. Fill in the blanks in the code template below. You are not allowed to use bit-specific addressing or the `BIC` instruction.

```
Belt_Init
    LDR R1, =SYSCTL_RCGC2_R
    LDR R0, [R1]

    _______________
    STR R0, [R1]
    NOP
    NOP
    LDR R1, =GPIO_PORTB_DIR_R
    LDR R0, [R1]



    _______________
    STR R0, [R1]
    LDR R1, =GPIO_PORTB_AFSEL_R
    LDR R0, [R1]

    _______________
    STR R0, [R1]
    LDR R1, =GPIO_PORTB_DEN_R
    LDR R0, [R1]

    _______________
    STR R0, [R1]
    BX  LR
```

(b) (20 points) Write a main C program that first calls the *Belt_Init* subroutine from (b) then performs a loop over and over to turn the LED on iff (if and only if)

- the engine is running (RPM >= 1000), and
- the seatbelt is not fastened.

In all other cases, the LED should be off.

```
// declaration of function implemented in assembly
void Belt_Init(void);


// main program
void main(void)
{









}
```

Problem 6 (15 points): C Programming and Parameter Passing

Given below is the C code for a function that checks whether a number is prime. Translate the C code into assembly. Follow the AAPCS calling convention standard, i.e. use register R0 both to pass value 'v' and return the result, and you can freely use registers R0 through R3. Note: in ARM assembly, the modulo operation (A % B) has to be implemented as (A – B * (A / B)).

C code

```
uint32_t prime(uint32_t v)
{
  unsigned int i;

  for(i = 2; i < v; i++) {
    if((v % i) == 0) {
      return 0;
    }
  }

  return 1;
}
```

Assembly code

```
prime
```

Same appendix as HW4