Exam 1

Date: February 21, 2013; 9:30-10:45am

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).
- You must put your answers on pages 2-6 only.
- You have **75** minutes, so allocate your time accordingly.
- Show your work, and put your answers in the boxes.
- *Please read the entire quiz before starting.*

Question 1 (10 points)

Consider the following 8-bit addition (assume registers are 8 bits wide)

```
Load 0x40 into R1
Load 0x4B into R2
Add  R3 = R1+R2
```

a. What will be the 8-bit result in Register R3 (in hex)?

0x8B

b. What is 8-bit result in Register R3 (as unsigned decimal)?

128+11=139

c. What is 8-bit result in Register R3 (as signed decimal)?

-128+11= -117

d. What will be the value of the carry (C) bit?

answer is correct, C=0

e. What will be the value of the carry (V) bit?

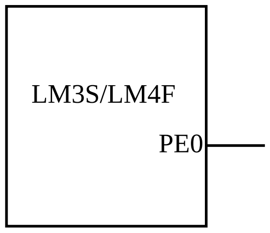
answer is incorrect, V=1

Question 2 (10 points).

Interface the LED to PE0 using negative logic. The desired LED operating point is 1.8V at 1.2 mA. The output voltage V_{OL} of the microcontroller is 0.5 V. You may also use 3.3 power and ground, as you need them. Show the equation used to calculate the resistor value and the circuit diagram showing the connections.

3.3V ---R---LED---PE0,

R = (3.3-1.8-0.5)/1.2mA = (1.5-0.5)/1.2mA =(1V)/1.2mA = 833 ohms



Question 3

Part a) (10 points). Write an assembly subroutine, called **ROL**, that rotates a 32-bit number given in R1 a number of times given in R2. That is, the input is passed by value in Registers R1 and R2, and the output is returned in Register R3. You may use Registers R3 and R12 as scratch registers without saving and restoring them. Notice that *rotating* an n-bit number k places to the *left* is the same as rotating it n-k places to the *right*. (Check the **ROR** instruction)

```
;---------- ROL -------
; Inputs: R1 has number to rotate; R2 has places to rotate
; Output: R3 has result (R1 rotated left R2 places)
ROL
     MOV  R3,#32
     SUB  R3,R3,R2 ;32-R2
     ROR  R1, R1, R3      ; rotate right
     BX   LR
```

Part b) (5 points). What specific modifications must be done to the subroutine in part (a) so that the subroutine is AAPCS compliant?

```
; Inputs: R0 has number to rotate; R1 has places to rotate
; Output: R0 has result (R0 rotated left R1 places)
ROL  MOV  R3,#32
     SUB  R3,R3,R1 ;32-R1
     ROR  R0, R0, R3      ; rotate right
     BX   LR
```

Question 4 (15 points)
Assume the stack pointer (SP) is initialized to 0x2000.0408. Registers R0, R1, R2 and R12 are initialized to 12, 3, 8 and 5 respectively. Answer the following:
Part a) Show is the content of the stack and the SP after the following sequence of operations:

```
PUSH {R0}
PUSH {R1-R2}
PUSH {R12}
```

SP->	0x200003F8	5
	0x200003FC	3
	0x20000400	8
	0x20000404	12
	0x20000408	

Part b) Given the state of the stack after part a), show the content of the stack, the SP and registers R3, R4 after the following operation:

```
POP {R3-R4}
```

	0x200003F8	5
	0x200003FC	3
SP->	0x20000400	8
	0x20000404	12
	0x20000408	

R3: 5   R4: 3

Part c) Given the state of the stack after part b), show the content of the stack, the SP and registers R0-R3 after the following operation:

```
PUSH {R0-R1}
POP {R0-R3}
```

	0x200003F4	
	0x200003F8	12
	0x200003FC	3
	0x20000400	8
	0x20000404	12
SP->	0x20000408	

R0: 12   R1: 3   R2: 8   R3: 12

Question 5 (20 points)

You may use the following definitions or add others if you need. Bit-specific addressing is allowed but not required (use the dashed block below to make declarations for bit-specific addressing if needed)

```
#define GPIO_PORTB_DATA_R       (*((volatile unsigned long *)0x400053FC))
#define GPIO_PORTB_DIR_R        (*((volatile unsigned long *)0x40005400))
#define GPIO_PORTB_AFSEL_R      (*((volatile unsigned long *)0x40005420))
#define GPIO_PORTB_DEN_R        (*((volatile unsigned long *)0x4000551C))
#define SYSCTL_RCGCGPIO_R       (*((volatile unsigned long *)0x400FE608))
#define SYSCTL_RCGCPGIO_GPIOB   0x00000002  // port B Clock Gating Control
```

Part (a) Write a C function `Init` that initializes Port B, pins 0 and 7 as outputs, and pins 3 and 4 are to be made input. (Ignore pull-up, pull-down resistors or high-amp drivers for outputs).

```
//Initializes GPIO port B with pins 0,7 as outputs and pins 3,4 as inputs
void Init(void) {
      SYSCTL_RCGCGPIO_R |= SYSCTL_RCGCGPIO_GPIOB; //Enable Clock
      GPIO_PORTB_DIR_R |= 0x81;             //pins 0,7 are output
      GPIO_PORTB_DIR_R &= ~0x18;            //pins 3,4 are input
      GPIO_PORTB_AFSEL_R &= ~0x99;          // Regular function
      GPIO_PORTB_DEN_R |= 0x99;             // Enable 0,3,4 and 7 pins
}
```

Part b) Assume the initialization in part a) has been executed. Write C function that toggles PB0 and PB7 only when both inputs PB3 and PB4 are high. Otherwise PB0, PB7 are both off.

```
//Checks if both PB3 and PB4 are high and only then toggles PB0 and PB7
// otherwise turns both off.
void doIt(void) {
      if((GPIO_PORTB_DATA_R & 0x18) == 0x18) {
            GPIO_PORTB_DATA_R ^= 0x81;
      } else {
            GPIO_PORTB_DATA_R &= ~0x81;
      }
}
```

Question 6 (30 points)

a. What are the three registers that are initialized on Reset?

PC, LR and SP, virtually every I/O register is initialized

b. In C what is the size in bytes for each of the following data types:

Data Type	Size(in bytes)
`uint16_t`	2
`int32_t`	4
`int8_t`	1

c. Register R1 has the value 0x80008001, what is its value after the following operations are performed independently:

Operation	R1
`LSR R1,R1,#3`	0x10001000
`LSL R1,R1,#4`	0x00080010
`ASR R1,R1,#1`	0xC0004000

d. What is the purpose of the following statements in your assembly code:

Statement	Purpose
`EXPORT Start`	Put a symbol in global symbol table
`THUMB`	Use Thumb (rather than ARM) instructions
`AREA DATA ALIGN=2`	Put stuff in RAM

e. Answer the following fill in the blank or True/False questions:

The Cortex M4 architecture is a Von-Neumann architecture	False
The ARM ISA does not support Indirect addressing	True
In Thumb mode all instructions are 2 bytes long	False
ARM architecture supports bi-endian representation	True
The remainder operation in C is ______%__________	
When a C Program can calls an assembly subroutine the 2nd parameter it passes can be found in register____R1______________	

Same appendix as HW4