# Exam 1

**Date:** February 23, 2018

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 books, no papers, no data sheets (other than the last two pages of this Exam)
- No devices other than pencil, pen, eraser (no calculators, no electronic devices), please 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. **Do Not write answers on back of pages as we will not be scanning the back of your exam sheets.**
- You have 75 minutes, so allocate your time accordingly.
- Unless otherwise stated, make all I/O accesses friendly and all subroutines AAPCS compliant
- Please read the entire exam before starting.

| | | |
|---|---|---|
| **Problem 1** | 22 | |
| **Problem 2** | 8 | |
| **Problem 3** | 20 | |
| **Problem 4** | 20 | |
| **Problem 5** | 10 | |
| **Problem 6** | 20 | |
| **Total** | 100 | |

**(4)** ***Problem 1a.*** The code below is intended to calculate the sum of first 32 natural numbers and send it to output. Natural numbers are integers starting with 1. Fix the bugs in the code and write your code in the right column:

```
uint16_t i = 0;
uint16_t sum = 0;
while (i<32)
      sum +=i;
Output(sum);
```

1. i should be initialized to 1 (1 pt) [Will work with 0 too)
2. i<32 must be i < 33 OR i <= 32 (2pts)
3. Missing i++ and braces in loop (1pt)

**(2)** ***Problem 1b.*** What is the value of sum when the following code sequence is executed?

```
uint32_t arrA[3] = {1,2,3};
uint32_t arrB[3] = {5,6,7};
uint32_t sum = arrA[1] + arrB[2];
```

**2+7 = 9**

**(4)** ***Problem 1c.*** Debug this *asm* routine where a Callee is provided addresses of two `uint16_t` numbers as its two arguments and intends to return the larger of the two numbers. Write the fixed subroutine in assembly on the right. Make sure it is AAPCS compliant.

```
Callee    LDR R1, [R0]    <-
          LDR R3, [R2]    <-
          CMP R1, R3
          BGT return1     <-
          MOV R3, R0
          BX  LR
return1   MOV R1, R0      <-
          BX  LR
```

**Needs to be LDRH for 16-bit(1)**
**Second input in R1 not R2 (1)**

**Comparison must be BHI (1)**

**Result in R0 not R1 (1)**

**(4)** ***Problem 1d.*** Translate the C code below to AAPCS-compliant assembly.

```
int16_t func(uint8_t shift,
             int16_t vara) {
  int16_t  varb = vara >> shift;
  return varb;
}
```

```
func
   ASR  R0,R1,R0
   BX   LR
```

**(8)** ***Problem 1e.*** A Caller function calls a Callee function (see below). The Callee takes four inputs.

```
Caller
    MOV  R0, #14
    MOV  R1, #10
    MOV  R2, #12
    MOV  R4, #11
    PUSH {R4}
    BL   Callee
    BX   LR
```

```
Callee
    CMP R0, R1
    BHI No1
    MOV R0, R1
No1 CMP R0, R2
    BHI No2
    MOV R0, R2
No2 POP {R4}
    CMP R0, R4
    BHI No3
    MOV R0, R4
No3 BX  LR
```

Answer the following:
(i) What does the Callee function return (expressed as a function of inputs)?

Return the largest of 4 inputs

(ii) Circle (≥0) which of the following AAPCS rules does the Callee function violate?

a. Does not save and restore registers that it is supposed to
b. Expects inputs in wrong registers
c. Returns output in wrong register
d. Improper use of stack

(ii) Circle (≥0) which of the following AAPCS rules does the Caller function violate?

e. Does not save and restore registers that it is supposed to
f. Sends inputs in wrong registers
g. Expect returned output in wrong register
h. Improper use of stack

**(8)** ***Problem 2.*** The Stack Pointer (SP) is at address **`0x2000.0F00`** when the code below starts executing.

```
main
    ...  ; registers R0 to R3 are initialized here to
         ; store uint16_t values 30,20,10,4 respectively.
    push {R0, R2, R1, R3}   ; LOCATION A
    ...                     ; some code
    pop {R3, R1}            ; LOCATION B
    pop {R0, R15}           ; LOCATION C
    ...                     ; some code
```

**(2) 2a.** What is the value stored in SP after executing the instruction at LOCATION A?

| |
|---|
| **SP = SP-16 = 0x2000.0F00 - 16 = 0x2000.0EF0** |

**(4) 2b.** What are the content of registers R3 and R1 immediately after executing the instruction at LOCATION B?

| | |
|---|---|
| **R3:** 20 | **R1:** 30 |

**(2) 2c.** At what address will the processor start executing immediately after executing the instruction at LOCATION C?

| |
|---|
| **R15 = PC = 0x0000.0004** |

**(10)** ***Problem 3a.*** Consider the following C function that implements a piece-wise linear equation. Convert the C code to assembly code so that it implements the same functionality.

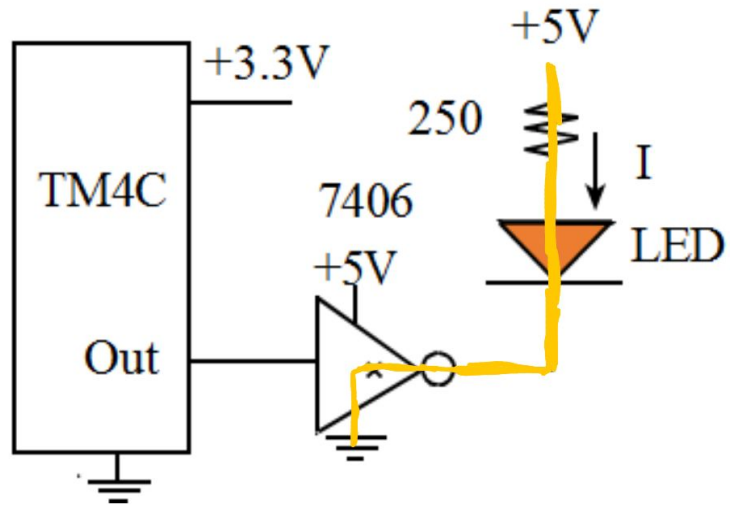
| | |
|---|---|
| `int16_t x,y;` | `AREA DATA, ALIGN=2`<br>`x SPACE 2`<br>`y SPACE 2` |
| `void fun(void){` | `AREA |.text|, CODE, READONLY, ALIGN=2`<br>`fun` |
| `if(x < 100){`<br>`  y = 0;`<br>`}` | `LDR R0,=x`<br>`LDRSH R0,[R0] ; 16-bit signed`<br>`CMP R0,#100`<br>`BGE ge100`<br>`MOV R0,#0`<br>`B done` |
| `else {`<br>`  y = x/2 - 200;`<br>`}` | `ge100 ASR R0,R0,#1 ;shift once to divide by 2`<br>`SUB R0,R0,#200` |
| `}` | `done LDR R1,=y`<br>`STRH R0,[R1]`<br>`BX LR` |

**(10)** ***Problem 3b.*** Consider the following assembly function that performs input/output on Port A. You may assume Port A has been initialized so PA7 is an input, and PA3 is an output. Convert the assembly function to a C code so that it implements the same functionality .

```
toggleWhen
   LDR  R0,=GPIO_PORTA_DATA_R
a1 LDR  R1,[R0]
   ANDS R1,R1,#0x80
   BEQ  a1
   MOV  R2,#20
a2 CMP  R2,#0
   BEQ  a3
   LDR  R1,[R0]
   EOR  R1,R1,#0x08
   STR  R1,[R0]
   SUB  R2,R2,#2
   B    a2
a3 BX   LR
```

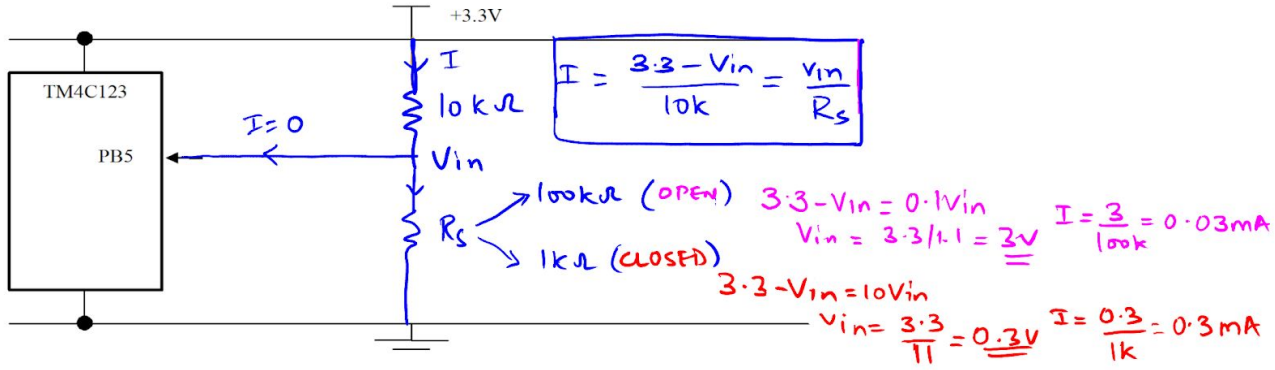
```
while((GPIO_PORTA_DATA_R&0x80)==0){
}

for(int i = 20; i != 0; i = i-2){
  GPIO_PORTA_DATA_R ^= 0x08;
}
int i = 20;
while(i != 0){
  GPIO_PORTA_DATA_R ^= 0x08;
  i = i-2;
}
}
```

**(10)** ***Problem 4a***: This circuit works correctly. The microcontroller's output high voltage is 3.2V. The microcontroller's output low voltage is 0.1V. The LED's operating voltage is 2.6V. The $V_{OL}$ for the 7406 driver is 0.4V. Part 1) Calculate the LED current when on. Part 2) Draw a continuous line on the circuit showing the path of the LED current from source to sink.




**$5-I_d*250-V_d-V_{OL} = 0 \Rightarrow 5-I*250-2.6-0.4 = 0 \Rightarrow 250I = 2 \Rightarrow I = 8mA$**

**(10)** ***Problem 4b***: Interface a switch using an external 10 kΩ resistor to Port B bit 5 using negative logic. You may assume PB5 has been configured as an input port.



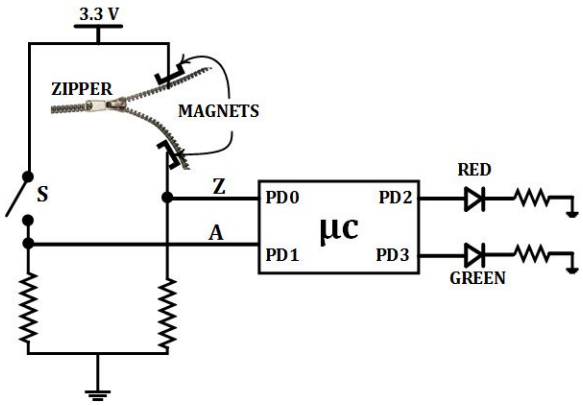

Assume no current can flow into or out of the port pin. The on-resistance of the switch is **not** 0 ohms and the off-resistance is **not** ∞ ohms. Rather, the resistance of this switch is 1 kΩ when closed and 100 kΩ when open. Find the current through the switch and the voltage across the switch for the open and closed conditions. Complete the following table. Show the equation and then do the math to calculate the numbers.

| Switch State | Current through switch | Voltage across switch |
|---|---|---|
| Switch open | **0.03mA** | **3V** |
| Switch closed | **0.3mA** | **0.3V** |

**(10)** ***Problem 5***: Write a friendly assembly language subroutine, called Ports_Init, to configure the pins on Port D as follows: pins 0,1 are inputs and 2,3 are outputs. Assume that the following device register declarations are given (only these need manipulating for initialization): **SYSCTL_RCGCGPIO_R**, **GPIO_PORTD_DIR_R**, **GPIO_PORTD_DEN_R**.

```
Ports_Init

   LDR R0, SYSCTL_RCGCGPIO_R  ; Turn ON clock
   LDR R1, [R0]
   ORR R1, #0x08
   STR R1, [R0]
   NOP
   NOP                         ; Clock is stable now

   LDR R0,GPIO_PORTD_DIR_R
   LDR R1, [R0]
   BIC R1, #0x03              ; PD0, PD1 are inputs
   ORR R1, #0x0C              ; PD2, PD3 are outputs
   STR R1, [R0]

   LDR R0,GPIO_PORTD_DEN_R
   LDR R1, [R0]
   ORR R1, #0x0F             ; PC0-4 are Digital
   STR R1, [R0]

   BX LR                     ; Return to caller
```

**(20)** ***Problem 6***: You are hired to write a piece of software for a project called *Backpack Zipper Breach Detector* (BZBD). The purpose of the project is to detect when a zipper on a backpack is breached (opened). Two magnets that together act like a switch are attached to the backpack zipper. The microcontroller reads (Z) an open zipper as an open switch and a closed zipper as a closed switch on PD0. The zipper itself does not conduct current.




A switch S (interfaced on PD1, creating signal A), serves as an Off/On switch to disable or enable the system. The microcontroller reads an open switch S as disable and a closed switch S as enable of the BZBD system. There are two LEDs interfaced (red on PD2 and green on PD3) that indicate whether the zipper is breached (Red ON, Green Off) or is secure (Red Off, Green On). When the system is disabled both LEDs must be Off.

In addition to showing the status, the microcontroller sends a message to a Bluetooth enabled smartphone. The code to enable Bluetooth, connect to a smartphone and send messages has been written by somebody else. You simply have to call the `BT_Update` function to send updates to the smartphone. The prototype for the function is given below:

```
        void BT_Update(uint8_t code);
// The code is 0 for disabled;
//             1 for enabled and zipper breached;
//             2 for enabled and zipper secure
```

**At the beginning, send one update, and then an update must only be sent if there is a change in the status of the system.**

You can assume that all GPIO port initialization is already done for you and you only need to access the Data register (GPIO_PORTD_DATA_R) for completing the software module. Calling a C function follows the AAPCS convention.

**(5)** ***Problem 6a.*** Give a truth-table showing the relationship between the two inputs (A, Z) and the two outputs (Red, Green).

| A | Z | Red | Green | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | ← System Off |
| 0 | 1 | 0 | 0 | ← System Off |
| 1 | 0 | 1 | 0 | ← Zipper Breached |
| 1 | 1 | 0 | 1 | ← System Secure |

**(15)** ***Problem 6b.*** Complete the software in assembly (not C).

```
; BZBD Project Code
; All AREA declarations and EQU statements here (assume done for you)
    IMPORT BT_Update
; Make your own pseudo-op statements if needed (can declare variables too)
State RN 4


Start
      BL Ports_Init ; All initialization code here (assume done for you)
      BL BT_Init    ; Initializes Bluetooth and connects to a Smartphone
; Your code starts here
   MOV State, #0
   MOV R0, State
   BL  BT_Update    ; Set initial state to System Off

mLoop
   LDR R1,=GPIO_PORTD_DATA_R
   LDR R2,[R1]
   AND R3,R2,#2   ; Check Switch
   BNE On
   MOV R0,#0      ; Here means disabled
   B   Update
On
   AND R2,#1     ; Get Zipper state
   LSL R0,R2,#3  ; Zipper state is state of Green LED
   EOR R2,#1     ; Red LED state is complement of Zipper
   LSL R2,#2
   ORR R0,R0,R2  ; R0 has both LEDs state
   STR R0,[R1]
   LSR R0,#2     ; R0 now has Code: 01(Breach) or 10(Secure)
Update
   CMP R0,State
   BEQ mLoop     ; No change in state
   BL  BT_Update ; Update Smartphone
   MOV R4,R0     ; Save new state


   B mLoop


** Code in BOLD is for Update (5 pts)
```