Exam 1

Date: February 23, 2017

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
- You have 75 minutes, so allocate your time accordingly.
- Unless otherwise stated, make all I/O accesses friendly.
- Please read the entire exam before starting.

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

[8 points] ***Problem 1a*****:** Represent the following hexadecimal numbers as 8-bit binary, and determine their values as unsigned and signed decimal numbers.

Hexadecimal	Binary	Unsigned decimal	Signed decimal
0x7F	**01111111**	**127**	**127**
0xEF	**11101111**	**239**	**-17**

[12 points] ***Problem 1b*****:** Assume that ***number*** is a 16-bit variable initialized to 0x0088. For the following code snippets indicate which branch is taken.

Code Snippet	Destination label

`LDR  R1, =number`
`LDRB R0, [R1]`
`CMP  R0, #136`
`BNE  next`
`BL   subroutine`
	subroutine

`LDR   R1, =number`
`LDRSB R0, [R1]`
`CMP   R0, #100`
`BLT   next`
`BL    subroutine`
	next

`LDR  R1, =number`
`LDRH R0, [R1]`
`CMP  R0, #120`
`BHI  next`
`BL   subroutine`
	next

`LDR   R1, =number`
`LDRSH R0, [R1]`
`CMP   R0, #136`
`BGE   next`
`BL    subroutine`
	next

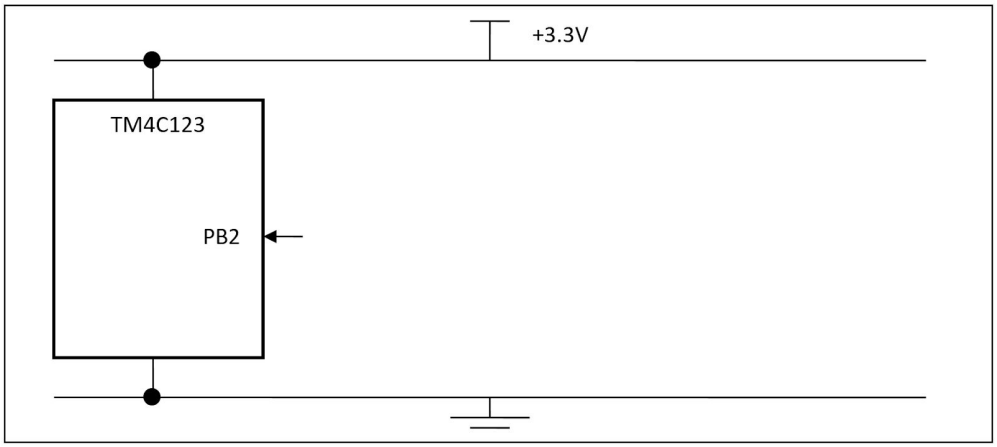
[5 points] ***Problem 2a***: Write friendly C code (function Ports_Init()) to turn the clock ON for two ports, PortA and PortE. Also configure and digital enable their pins as follows: PortA: All pins are input; PortE: pins 4,3,2,1,0 are outputs. Assume that the following device register declarations are given (only these need manipulating for initialization): `SYSCTL_RCGCGPIO_R`, `GPIO_PORTA_DIR_R`, `GPIO_PORTA_DEN_R`, `GPIO_PORTA_DATA_R`, `GPIO_PORTE_DIR_R`, `GPIO_PORTE_DEN_R`, `GPIO_PORTE_DATA_R` (some may not be needed).

```
void Ports_Init(void) {
  volatile uint32_t delay;

  SYSCTL_RCGCGPIO_R |= 0x11;  // Turn both clocks ON
  delay = 42;
  GPIO_PORTA_DIR_R &= ~0xFF; // PA7-0 are inputs so write 0
  GPIO_PORTA_DEN_R |= 0xFF;  // PA7-0 are digital enabled
  GPIO_PORTE_DIR_R |= 0x1F;  // PA4-0 are outputs so write 1
  GPIO_PORTE_DEN_R |= 0x1F;  // PA4-0 are outputs so write 1

Or <because these are all the pins of these two ports>
  GPIO_PORTA_DIR_R = 0x00; // PA7-0 are inputs so write 0
  GPIO_PORTA_DEN_R = 0xFF;  // PA7-0 are digital enabled
  GPIO_PORTE_DIR_R = 0x1F;  // PA4-0 are outputs so write 1
  GPIO_PORTE_DEN_R = 0x1F;  // PA4-0 are outputs so write 1









}
```

[5 points] ***Problem 2b***: A 319K student attempted to write code that swapped the bytes of a 16-bit unsigned Input number and wrote them into Output. So, if Input was `0xABCD` then Output after the program is run should be `0xCDAB`. His flawed solution is given below. Fix it by providing the correct solution on the right (Cite the line and the correction).

1: AREA DATA, ALIGN=2 2: Input SPACE 1 3: Output SPACE 1 4: AREA |.text|, CODE, READONLY, ALIGN=2 5: THUMB 6: EXPORT Start 7: Start 8: LDR R0,=Input 9: LDR R1,=Output 10: LDR R3,[R0] 11: LDR R4,[R0,#2] 12: STR R4,[R1] 13: STR R3,[R1,#2] 14: ALIGN 15: END	**2: Input SPACE 2** **3: Output SPACE 2** **10: LDRB R3,[R0]** **11: LDRB R4,[R0,#1]** **12: STRB R4,[R1]** **13: STRB R3,[R1,#1]**

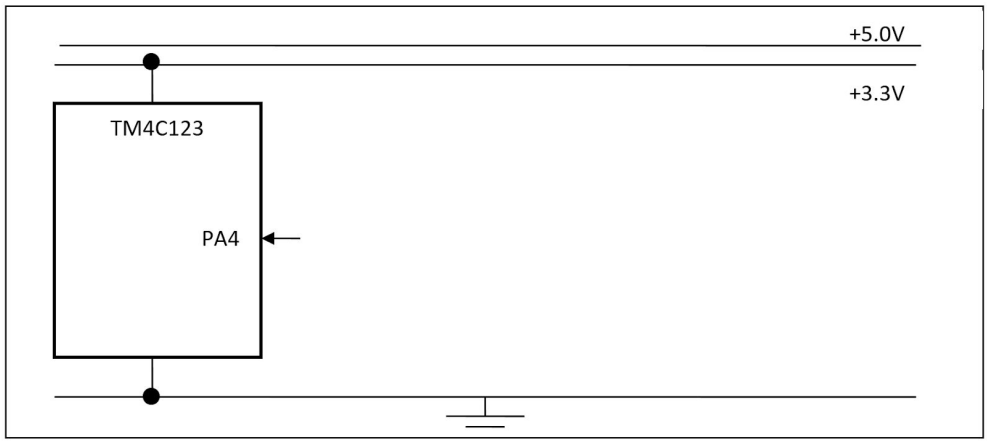
[10 points] ***Problem 3a***: Interface a switch through a 10 kΩ resistor to port PB2 (which has been configured as an input port) using negative logic. Show the circuit diagram with all components properly labeled. Assuming that no current can flow into and out of the port pin and the switch is ideal, find the current through the switch and the voltage across the resistor. Complete the table below .



Switch configuration	Current through switch	Voltage across the resistor
Switch open	**0A**	**0V**
Switch closed	**0.33 mA**	**3.3V**

[10 points] ***Problem 3b***: Interface an LED through a resistor to port PA4 (which has been configured as an output port) using positive logic. The LED's desired brightness needs an operating point as 3.0V, 15 mA. The V_{OL} and V_{OH} of the TM4C123 is 0.3V and 3.3V respectively. The 7406 driver has also been provided to you whose V_{OL} is 0.5V. Find the value of the resistor R that needs to be connected, and show the circuit diagram. Two supply voltages have also been provided

having values 3.3V and 5V. What is the current through the LED and the voltage across the resistor? Complete the table below the figure.



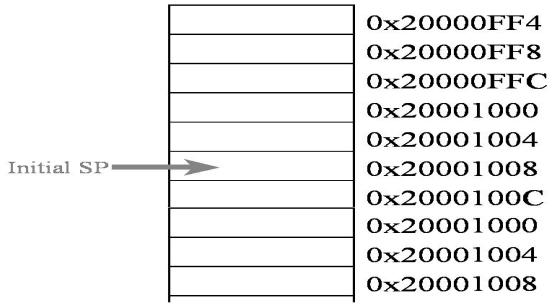
Voltage at PA4	Current through LED	Voltage across the resistor
Logic level 0	**0A**	**0V**
Logic level 1	**15mA**	**1.5V**

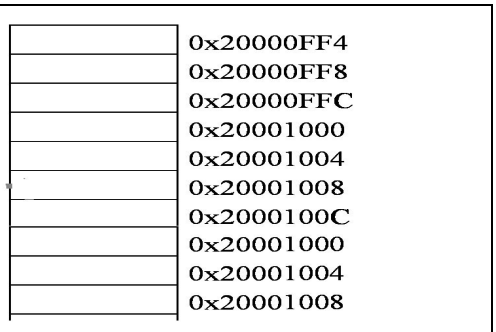
[10 points] ***Problem 4***: Work through the assembly code below to identify the contents in the stack. Also, describe why CalleR and Callee are not AAPCS compliant.

```
00000000 CalleR
00000000   MOV   R0,#1
00000004   MOV   R1,#7
00000008   MOV   R2,#2
0000000C   MOV   R3,#5
00000010   PUSH  {R3, R1, R2, R0}
      ;----A-----
00000012   BL              Callee
00000016   POP             {R5}
```

```
00000018 Callee
00000018   POP   {R2}
0000001A   MOV   R0,#3
0000001E Again
0000001E   POP   {R1}
00000020   CMP   R1,R2
00000022   BLT   Next
00000024   MOV   R2, R1
00000026 Next
00000026   SUBS  R0,#1
00000028   BNE   Again
0000002A   PUSH  {R2}
0000002C     ; ----B----
```

[5 points] ***Part (a)***: Give the state of the stack (SP and contents) at execution points **A** and **B**





At Execution Point B

At Execution Point A	
Final SP (at A): 0x2000.0FF8	**Final SP (at B): 0x2000.1004**

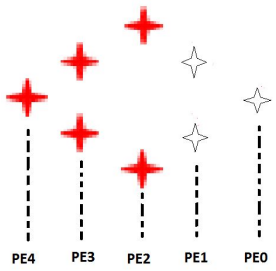
[5 points] ***Part (b)***:What makes CalleR and Callee non-compliant with AAPCS

CalleR
- **Pushed R0--R3 into stack; should be in regs.**
- **Modifies R5 without saving it.**

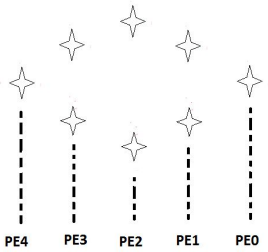
Callee
- **Pops first four arguments from stack instead of regs. (related to Callee error above).**
- **Returns value in R2.**

Both CalleR and Callee should push and pop LR onto the stack instead of arguments.

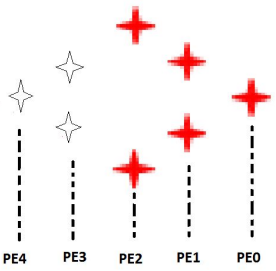
[20 points] ***Problem 5***: Write a software (in assembly) that controls the turn indicators for a bicycle. There are eight indicator LEDS that are interfaced as shown in the figure below to 5 output pins on the TM4C microcontroller. Note, PE3,2,1 control two LEDs each and PE4 and PE0 each control one LED.



Left Indication



All Off



Right Indication

An accelerometer interfaced externally to input pins PA7-PA0 gives the accelerometer reading in the x-direction. The reading is a signed number between -90 and +90. The accelerometer is mounted on the rider's helmet, so when the rider tilts her head to the left the readings are negative [-90,0) and when she tilts her head to the right the readings are positive (0,90]. You can consider any value between -20 and 20 to be false indicators of rider's intent to turn. Indication of left or right has to flash (on/off) the corresponding LEDS (see figure) at 4Hz with 50% duty cycle.

You can assume that all GPIO port initialization is already done for you and you only need to access the Data registers (GPIO_PORTE_DATA_R, GPIO_PORTA_DATA_R) for completing the software module. Your task is split into two parts.

[8 points] ***Part (b)***: Write a delay routine that delays in units of 1ms. You may assume the following: (i) the microcontroller is operating at the default clock speed of 16MHz, (ii) all

add/sub/logic operations take 1 clock cycle and all conditional branches take 3 clock cycles.

```
; DelayMS - Delays for R0 ms
; Input: R0 has the number of milliseconds to delay for
; Output: None
DelayMS
      LDR R1,=4000    ; Count * 4 *(1/16x10^6)= 10^-3 => Count = 4000
Dloop SUBS R1,#1      ; 4 cycle loop
      BNE  Dloop
      SUBS R0,#1
      BNE  DelayMS
      BX LR
```

[8 points] ***Part (b)***: Complete the main loop of the software. Your code should turn the appropriate LEDs On/Off to get the flashing effect when indicating a turn, otherwise (to keep straight) the LEDS are all off. You must call the Delay routine you wrote above in parts (a).

```
; Turn Indicator Program
; All AREA declarations and EQU statements here (assume done for you)
; Make your own pseudo-op statements if needed

Start
       ; All initialization code here (assume done for you)
      LDR R4,=GPIO_PORTA_DATA_R
      LDR R5,=GPIO_PORTE_DATA_R
mLoop
      LDR R2,[R4]
      CMP R2,#20
      BGT Right      ; Could be BGE
      CMP R2,#-20
      BLT Left       ; Could be BLE
      MOV R3,#0      ; All OFF
      STR R3,[R5]
      B  Done
Left
      LDR R3,[R5]
      EOR R3,#0x1C   ; PE4,PE3,PE2
      STR R3,[R5]
      B  Done
Right
      LDR R3,[R5]
      EOR R3,#0x07   ; PE2,PE1,PE0
      STR R3,[R5]
Done
      MOV R0,#125    ; 4Hz => 4 times per second =>ON/Off for 1/8s (125ms)
      BL DelayMS
      B  mloop
```

[20 points] ***Problem 6***: The two columns below represent the same program with C on the left and ASM on the right. Some rows are missing from each column. Fill any row needed to make both columns completely equivalent. You also need to fill in the parameters for the function `foo` and the condition for the C `while` loop. You also should fill in the value of the variable `sY` where specified. You may not need all boxes. No more than one statement/op per box. No extra boxes.
THIS PROBLEM HAD A BUG. DO NOT SOLVE FOR PRACTICE. LOOK AT NEXT PAGE.
GRADING WAS LENIENT AND MANY SORT-OF-OK ANSWERS ACCEPTED.
ORANGISH COLORS INDICATE NOT-QUITE-RIGHT, BUT ACCEPTED ANSWERS

`// Don't fill in this line`	`AREA DATA ; assume starts at 0x2000.0000`
`uint32_t uX=9; // const int32_t uX=#9`	`uX DCD 0x09`
`int32_t sY;`	**`sY SPACE 4 / sY DCD 0 / sY DCD 0xDC`**
`// Don't fill in this line`	`AREA CODE ; assume starts at 0x0000.2000`
`int32_t foo(int32_t a) { // fill param`	`foo ; rest of this line is empty`
`  int32_t b = a;`	`  MOV R1, R0`
`  int32_t c = 0;`	`  MOV R2, #0`
`  while (c < uX) { // or c < 9`	`  LDR R3, =uX`
`    b = b * a;`	`  LDR R3, [R3]`
`    ++c;`	`loop CMP R2, R3`
`  } // only thing in this box is }`	`  BHS cont`
`  return (b);`	**`  MUL(S) R1, R1, R0`**
`} // only thing in this box is }`	**`  ADD(S) R2, R2, #1`**
	`  B loop`
	`cont MOV R0, R1`
	`  BX LR`
`void bar(int32_t a) {`	`bar ; do not add branches below this line`
`// Don't fill in this line`	**`  PUSH {LR, …} ; accepted variants`**
`  if ((a=foo(a)) < sY) // accepted lots of //other if statements`	`  MOV R1, =sY`
`    sY = a; // nothing else on this line`	`  LDR R2, [R1]`
`  } // only thing in this box is }`	`  BL foo`
`} // only thing in this box is }`	`  CMP R0, R2`
	`  BLE next`
	`  STR R0, [R1] ; not quite and accepted ; variants`
	`next; rest of this line is empty`
	`  BX LR`
`int main() { // ignore main() in ASM`	`main  ; ignore main and this entire box ; in ASM`
`  sY = 0xDC;  // ignore this in ASM`	
`  bar(2);  // ignore this in ASM`	
`  // sY at this point is: 2 or 1024 or 512`	
`  return (0); // ignore this in ASM`	
`}  // ignore this in ASM`	

[20 points] ***Problem 6***: The two columns below represent the same program with C on the left and ASM on the right. Some rows are missing from each column. Fill any row needed to make both columns completely equivalent. You also need to fill in the parameters for the function `foo` and the condition for the C `while` loop. You also should fill in the value of the variable `sY` where specified. You may not need all boxes. No more than one statement/op per box. No extra boxes.
THIS PROBLEM IS CORRECTED FROM VERSION GIVEN IN EXAM.

`// Don't fill in this line`	`AREA DATA ; assume starts at 0x2000.0000`
`uint32_t uX; / const uint32_t uX;`	`uX DCD 0x09`
`int32_t sY;`	**`sY SPACE 4 / sY DCD 0`**
`// Don't fill in this line`	`AREA CODE ; assume starts at 0x0000.2000`
`int32_t foo(`**`int32_t a`**`) { `**`// fill param`**	`foo ; rest of this line is empty`
`  int32_t b = a;`	`     MOV R1, R0`
`  int32_t c = 0;`	`     MOV R2, #0`
`  while (`**`c < uX`**`) {`	`     LDR R3, =uX`
`    b = b * a;`	`     LDR R3, [R3]`
`    ++c;`	`loop CMP R2, R3`
`  } // only thing in this box is }`	`     BHS cont`
`  return (b);`	**`     MUL(S) R1, R1, R0`**
`} // only thing in this box is }`	**`     ADD(S) R2, R2, #1`**
	`     B loop`
	`cont MOV R0, R1`
	`     BX LR`
`void bar(int32_t a) {`	`bar ; do not add branches below this line`
`// Don't fill in this line`	**`     PUSH {LR, R4, R5, R6}; R6 for ; alignment`**
`  if (foo(a) < sY) / if ((a=foo(a)) < sY)`	`     MOV R4, =sY`
`    sY = a; // or sY = foo(a);`	`     LDR R5, [R4]`
`  } // only thing in this box is }`	`     BL foo`
`} // only thing in this box is }`	`     CMP R0, R5`
	`     BLE next`
	`     STR R0, [R4]`
	`next; rest of this line is empty`
	`     POP {LR, R4, R5, R6}`
	`     BX LR`
`int main() { // ignore main() in ASM`	`main    ; ignore main and this entire box ; in ASM`
`  sY = 0xDC;  // ignore this in ASM`	
`  bar(2);     // ignore this in ASM`	
`  // sY at this point is: 2 or 1024`	
`  return (0); // ignore this in ASM`	
`}             // ignore this in ASM`	