Exam 1

Date: February 20 , 2025

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 the back of pages as we will not be scanning the back of your exam sheets.**
- You have **75+5** 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	20	
Problem 2	15	
Problem 3	15	
Problem 4	12	
Problem 5	12	
Problem 6	10	
Problem 7	16	
Total	100	

(10) ***Problem 1a.*** Give the value stored in the destination register after performing each operation. The initial values are `R0 = 0x23, R1 = 0x02,` and `R4 = 0x32`. Give your answers in Hex.

Operation	**Register**
`LDR R3, #=64`	`R3=`
`ADDS R2, #`	`R2=`
`ANDS R2, R0, R1`	`R2=`
`LSLS R0, R1`	`R0=`
`CMP  R4, R1`	`R4=`

(10) ***Problem 1b.*** Assume the contents of `R0` are `0x00004000`. Relevant memory contents are shown below right. Assume `R6` has `0` and `R7` has `2` in it. What are the 32-bit contents (in hex) of each of the registers after the instructions below are executed:

```
LDR R1, [R0]
LDRH R2, [R0]
LDRSB R3, [R0, R6]
LDRSH R4, [R0, R7]
LDRB  R5, [R0]
```

`0x00004000`	`0x12`
	`0xA1`
	`0x00`
	`0xC2`

`R1:`
`R2:`
`R3:`
`R4:`
`R5:`

(15) ***Problem 2.*** Consider the following C code in the left column. The function **`Pow`** computes the power of **`x`** raised to **`n`** (e.g., **`Pow(x=2,n=3)`** is **`8`**). The function assumes that the input **`x≥1`** and input **`n ≥ 0`**. The last box shows a C call to this function. Convert the C code to assembly code in the right column, so that it implements the same functionality in each box (for example, the C local variable **`power`** can simply be a register in assembly). AAPCS is in effect.

`uint32_t Result=0;` `uint8_t a, num;`	`.data` `.align 2`

`uint32_t Pow(uint8_t x,`
`uint8_t n){`
`  uint32_t power = 1;`
	`.text`
	`Pow:`

`  while (n != 0){`
`  power = power * x;`
`  n = n-1;`
`  }`	

`  return power;` `}`	`BX LR`

`// Assume a and num are set`
`// to some 8-bit values`
`// before the call below`
`Result = Pow(a, num);`	

(15) ***Problem 3.*** This program performs an operation on global variables **a** & **b** (signed 16-bit numbers) and stores the result in **res** (signed 32-bit). The assembly code is shown on the left and your task is to write the equivalent in C.

(4) i. What is the value of **res** if **a** were initialized to **-5** and **b** to **3**.

(9) ii. Complete the missing lines of the C code to match the functionality of the assembly code. You are **not required to do a literal translation line by line**.

`    .data`
`a:   .short -5`
`b:   .short 3`
`res: .long 0`
	`// Declarations`

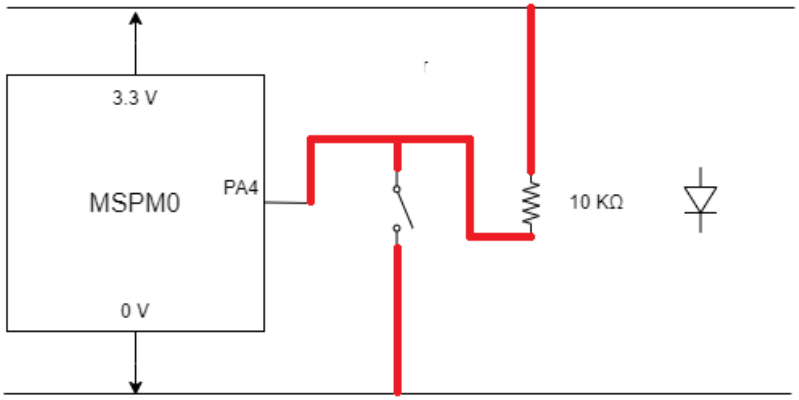
`    .text`
`    .align 2`
`main:`
`     MOVS R0, #0`
`     LDR  R1, =a`
`     LDR  R2, =b`
`     MOVS R3, #0`
`     LDRSH R5, [R2,R3]`
`Loop:`
`     LDRSH R4, [R1,R3]`
`     CMP  R4, #0`
`     BEQ  DONE`
`     BLT  NEG`
`     ADDS R0, R0, R5`
`     SUBS R4, R4, #1`
`     STRH R4,[R1,R3]`
`     B Loop`
`NEG:`
`     SUBS R0,R0, R5`
`     ADDS R4, R4, #1`
`     STRH R4,[R1,R3]`
`     B Loop`
 	
`DONE:`
`    LDR R1, =res`
`    STR R0, [R1]`
`ForEver:`
`     B ForEver`
	`// main function`
	`int main(){`
	
	`}`

(2) iii. In 8 words or less describe what the program does for arbitrary signed values of a and b.

(12) ***Problem 4.***

(2) i. In your own words explain what it means to have a Negative Logic Switch.

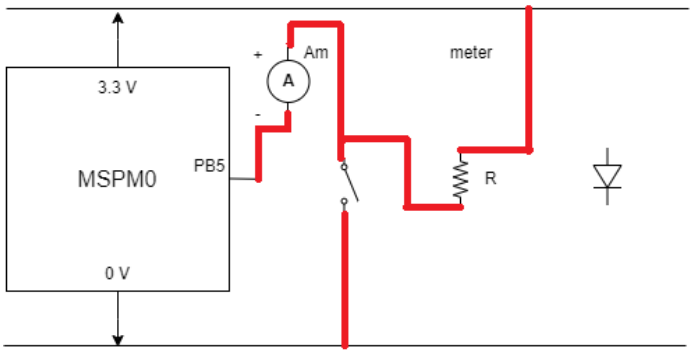
(4) ii. Given the components in the figure below connect what you need to interface Port A pin 4 to a *Negative Logic switch*.



(4) iii. Write a function **`ReadSwitch`** in assembly code to read the value of pin PA4. The function should return 0 if the switch is pressed and 1 if it is not pressed. Assume that the pin is already properly initialized. Follow AAPCS

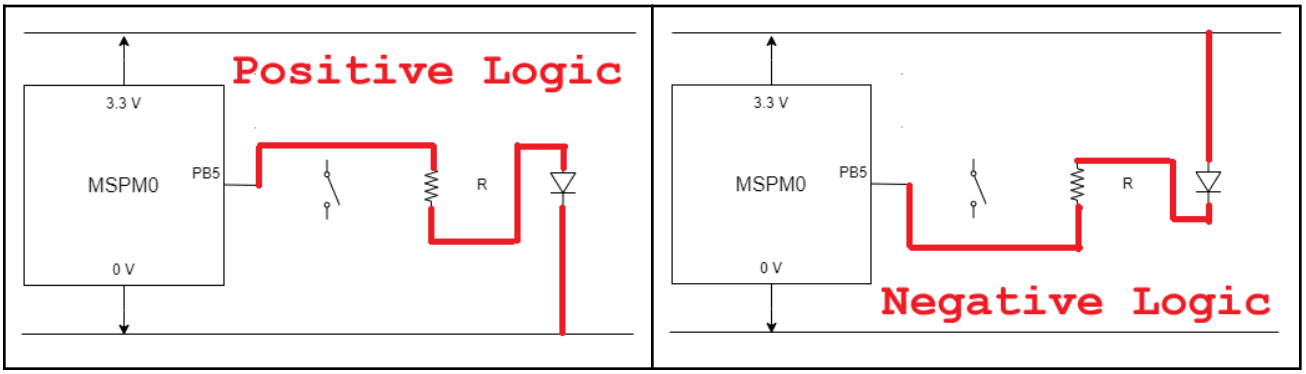
```
ReadSwitch:


  BX LR
```

(2) iv. You want to measure the current going into pin PA4. Redraw your circuit diagram from Part (ii) to include the provided Ammeter (a Multimeter with the knob set to measuring current).



(12) ***Problem 5.***

(4) i. Use the following parts to interface an LED to PB5 anyway you wish. Assume PB5 is initialized as output, with V_{OH}=3.2V, V_{OL}=0.1V.



(1) ii. For the circuit you've drawn, what output voltage on the pin from the MSPM0 will cause the LED to turn on?

- ☐ 3.2 V Check for Positive Logic
- ☐ 0.1 V Check for Negative Logic

(1) iii. Is the LED you interfaced a positive or negative logic LED (check one)?

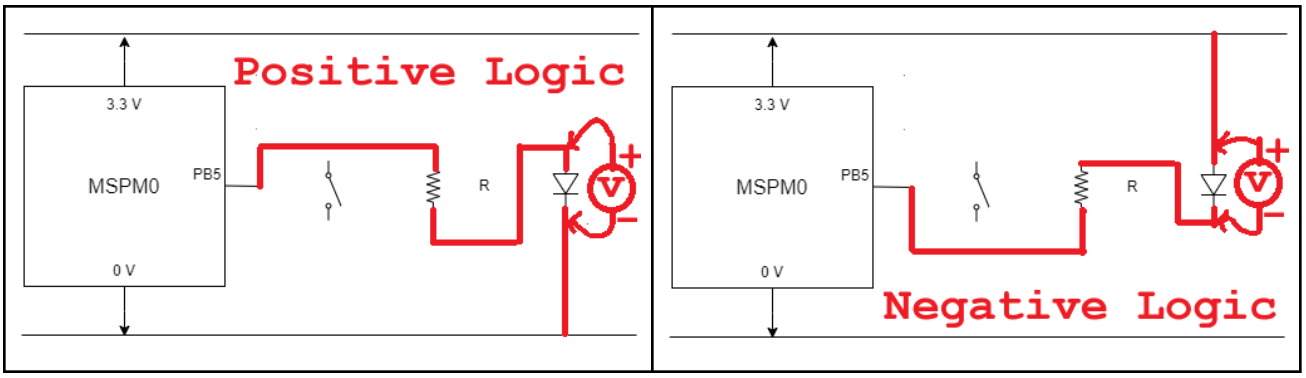
- ☐ Positive Logic (Left)
- ☐ Negative Logic (Right)

(2) iv. Which registers will need to be initially configured to set PB5 correctly? Check all that apply.

- ☐ IOMUXPB5 For PinCM
- ☐ DOE For Output Enable
- ☐ DIN
- ☐ DOUT

(3) v. Assume you want a current of 1.2mA through the LED and the drop across the LED is 2V. Using the output voltage from part (ii), what should R be?

(1) vi. You want to measure the Voltage across the LED. Redraw your circuit diagram from Part (i) to include the provided Voltmeter.



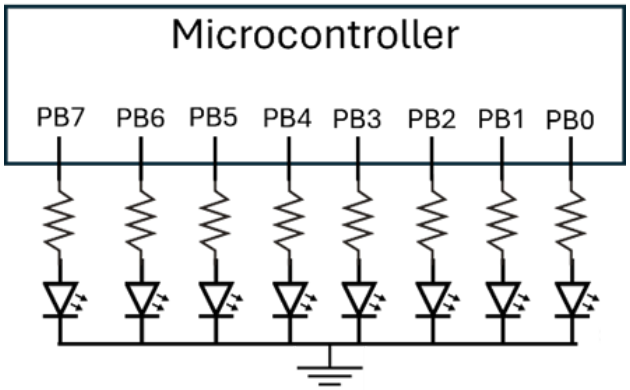
10) Problem 6. The following assembly program has been written to comply with AAPCS. When main enters the infinite loop, register R4 contains the value "0xDEADBEEF". Fill in the missing Registers for the PUSH/POP instructions to make the program comply with AAPCS, and as a result produce the correct output in register R4. Additionally, show the contents of the stack (addresses and values) when the line with the comment "**SHOW STACK**" is executed. Assume the initial Stack Pointer is 0x20208000.

```
main:
 .req n,_____
    LDR n, =0xDEAD0000
    LDR R4,=0xBEEF
    BL lawls
    ADDS R4, R4, n //The address of this instruction is 0x000000CA
Loop: B Loop

//This function takes no inputs and produces no outputs.
lawls:
    PUSH {___,___}
    LDR R0, =0xBEEF0000 //SHOW STACK
    LDR R4, =0xDEAD
    POP {___,___}
```

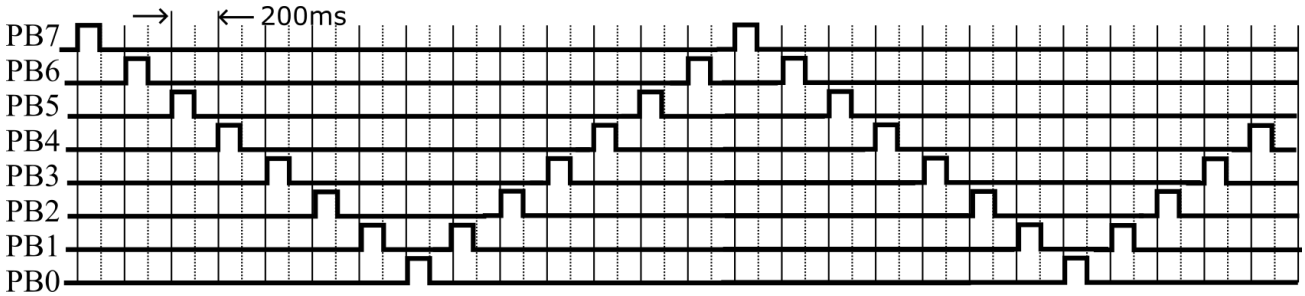
	Address	Value
Initial SP –>	**0x20208000**	

(16) ***Problem 7***. The following C functions are available for you to call from Assembly (using the `BL` instruction). Assume AAPCS rules on how to pass the input that the function expects. X is a pin from 0 to 7. E.g., X=5 means PB5

```
void LED_On(uint8_t X);
void LED_Off(uint8_t X);
void Delay_ms(N); // Delays for N ms
```



Suppose you have 8 LEDs interfaced like shown in the figure above. Write an Assembly language program (this is the main program), which repeatedly flashes the LEDs back and forth in the following order:



The implementation is based on iterating the array called **`Pattern`** that is predeclared for you. Your solution **must** use this array.

(6) i. Let `R5=i` be the array index from 0 to 13, write code (not a function) to get the value of `Pattern[i]` into `R0`.

(10) ii. Complete the program.

Hint: Use the code you wrote in part (i) for array access.

```
Pattern: .byte 7,6,5,4,3,2,1,0,1,2,3,4,5,6 // Pattern array

main:
       BL Initialize  // Does initialization of output pins PB7-0
       // Your code goes here









            .end
```