**Exam 1**

**Date:** Oct 3, 2019

UT EID: Professor: Valvano

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. *Anything outside the boxes/blanks will be ignored in grading*. You may use the back of the sheets for scratch work.
* You have 75 minutes, so allocate your time accordingly.
* For all questions, unless otherwise stated, find the most efficient (time, resources) solution.
* Unless otherwise stated, make all I/O accesses friendly and all subroutines AAPCS compliant.
* *Please read the entire exam before starting.*

**(10) Question 1.**

**(2) Part a)** Which equation describes the power dissipated in a resistor? …………

D) P=I2R

A) P = V/I B) P = V2\*R

C) P = I2\*R D) P = V/R2

E) Some of A – D F) None of A – D

n+1

**(2) Part b)** If you add an *n*-bit signed number to an *m*-bit signed number,

what is the maximum number of bits in the sum? Assume *n* ≥ *m*. ………………………….

C-bit is set for unsigned overflow

**(2) Part c)** Assume you add two unsigned 32-bit numbers using the

**ADDS R2,R1,R0** instruction. Which bit is set on overflow? ………………..………

**(2) Part d)** Sketch an approximate plot of LED current as a function of LED voltage:



A) Restricted, accessible only within the function

**(2) Part e)** What is the scope of a local variable in C? …………………………….

C) Permanent, they always exist, in RAM

**(2) Part f)** Where are global variables allocated in C? …………………….

**(12) Question 3.** There is a 16-bit signed global variable called **Stuff**.

 **.data**

 **.align 2**

**Stuff: .space 2**

Write Cortex M assembly subroutine that performs the same operation as this C function

**ShiftandAdd:**

 **LDR R0,=Stuff**

 **MOVS R3,#0**

 **LDRSH R1,[R0,R3]**

 **ASRS R1,R1,#2**

 **MOVS R3,#5**

 **ADDS R1,R1,R3**

 **STRH R1,[R0]**

 **BX LR**

**int16\_t Stuff;**

**void ShiftandAdd(void){**

 **Stuff = (Stuff>>2)+5;**

**}**

**(5) Question 4.** There is a 32-bit constant called **Thing**.

 **.text**

C0xFFFFFF9A (little endian)

**Thing: .long 0xF012349A**

**(12) Question 5.** Show the declaration of a C function that finds the minimum value of an array. The length of the array is fixed at 1000. A pointer to the array is passed by reference into the function. The function returns the smallest value in the array. The function prototype is

**int8\_t min(int8\_t data[1000]);**

**int8\_t min(int8\_t data[1000]){int8\_t result;**

 **result = data[0];**

 **for(int i=1; i<1000; i++){**

 **if(result > data[i]){**

 **result = data[i];**

 **}**

 **}**

 **return result;**

**}**

**(10) Question 6.** **Part a)** use pulldown

**(4) Part b)** For full credit, design the hardware interface that uses the fewest number of external components (resistors, LEDs, ULN2003B).



**(4) Part c)**

 **LDR R1,=GPIOB\_DIN31\_0 // input register**

 **LDR R0,[R1] // all of input**

 **MOVS R2,#0x0FF // bit 7-0 mask**

 **ANDS R0,R0,R2 // just PB6**

 **BX LR**

.

**(10) Question 7.** Interface an LED to PA7 using positive logic.

**(5) Part a)** The desired LED operating point is 2V, 1mA. The microcontroller output high voltage is 3.2V, the microcontroller output low voltage is 0.2V. The ULN2003 output low voltage is 0.3V. For full credit, design an interface that uses the fewest number of external components. Hardware only, no software. For any resistor(s) you use, show your work for determining the resistor value(s). **R = (3.2V-2V)/1mA = 1.2kΩ**



**(5) Part b)** Assume Port A is initialized so PA7 is an output. Write a function in C that accepts an input parameter (0 or 0x80) and writes to Port A in a friendly manner. Include both the prototype and the declaration of the function.

**#define GPIO\_PORTA\_DATA\_R (\*((volatile uint32\_t \*)0x400043FC))**

**void LED\_Set(uint32\_t value);**

**void LED\_Set(uint32\_t value){ uint32\_t old;**

 **old = GPIOB->DOUT31\_0; // previous data**

 **old = old&(~0x80); // clear bit 7**

 **GPIOB->DOUT31\_0 = old|value; // new value in bit 7**

**}**

**void LED\_Set(uint32\_t value){**

 **if(value == 0x80){**

 **GPIOB->DOUTSET31\_0 = 0x80; // turn on**

 **} else{**

 **GPIOB->DOUTCLR31\_0 = 0x80; // turn off**

 **}**

**}**

**(15) Question 8.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0x00001000 POP {R0,R1}** **0x00001004 ADD R2,R0,R1** **0x00001008 BL Func** **0x0000100C ...****...** **0x00002000 Func PUSH {R2,LR} A****0x00002004 MOVS R2,R1****0x00002008 MULS R0,R2****0x0000200C ADDS R0,R1****0x00002010 POP {R2,PC}** |

|  |  |
| --- | --- |
| **0x20200FF4** | **1** |
| **0x20200FF8** | **2** |
| **0x20200FFC** | **3** |
| **0x20201000** | **4** |
| **0x20201004** | **5** |
| **0x20201008** | **6** |
| **0x2020100C** | **7** |

 |

**(6)** **Part a)** Give the state of the stack (SP and contents) after executing of the **PUSH** instruction, as shown by arrow A: **pop (R0,R1}**causes R0=2, R1=3, SP = 0x20001000
**Add R2,R0,R1** causes R2 = 5

**SP = 0x20200FF8**

**BL** causes LR = 0x0000100D

**Push R2,LR**

|  |  |
| --- | --- |
| **0x20200FF4** |  |
| **0x20200FF8** | **5****R2 = 5****R1 = 3****R0 = 2** |
| **0x20200FFC** | **0x0000100D** |
| **0x20201000** | **4** |
| **0x20201004** | **5** |
| **0x20201008** | **6** |
| **0x2020100C** | **7** |

We give full credit for 0x0000100C. On the ARM/Thumb processors, the PC is 32 bits with bit 0 always clear. The processor uses this bit to specify if the destination code is ARM (0) or Thumb (1). For ECE319K this bit will always be 1 for Thumb.

 **(25) Question 9.** In this question there are two microcontrollers, such that the two Port B’s are connected (PB7 to PB7, PB6 to PB6,…PB0 to PB0). The goal is to send a three bit value from one microcontroller to the other. The transmit software will be on the left microcontroller. At all times the transmitter must have exactly one of the Port B pins high. E.g., the 8-bit Port B data must be 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40 or 0x80. You may assume the transmitter on the left has initialized all of Port B to be output, and the receiver on the right has initialized all of Port B to be input. On the transmitter, design an assembly function that accepts a 3-bit value in R0 (let ***n*** be the value in R0, you may assume 0 ≤ ***n*** ≤ 7) and write to Port B the value 2***n***. On the receiver design an assembly function that reads Port B (knowing the value will be restricted to 2***n*** for some 0 ≤ ***n*** ≤ 7) and returns in R0 the value ***n***.

**GPIO\_PORTB\_DATA\_R EQU 0x400053FC**

**//Output: R0 is 0 to 7**

**Receive:**

 **LDR R2,=GPIOB\_DIN31\_0**

 **LDR R1,[R2]**

 **MOVS R0,#0 //n**

**loop2: LSRS R1,#1 //bit goes into C**

 **BCS done2**

 **ADDS R0,#1 //n=n+1**

 **B loop2**

**done2: BX LR**

**//Input: R0 is 0 to 7**

**Transmit:**

 **MOVS R1,#1**

**loop: CMP R0,#0**

 **BEQ done**

 **LSLS R1,#1 //1,2,4,… 0x80**

 **SUBS R0,#1**

 **B loop**

**done: LDR R0,=GPIOB\_DOUT31\_0**

 **STR R1,[R0]**

 **BX LR**

**Transmit2:**

 **LDR R2,=** **Table**

 **LDRB R1,[R2,R0]**

 **LDR R0,=GPIOB\_DOUT31\_0**

 **STR R1,[R0]**

 **BX LR**

**Table: .byte 0x01,0x02,0x04,0x08**

 **.byte 0x10,0x20,0x40,0x80**

**Transmit3:**

 **MOVS R1,#1**

 **LSLS R2,R1,R0**

 **LDR R3,=GPIOB\_DOUT31\_0**

 **STR R2,[R3]**

 **BX LR**

**DCB 1,2,3 ; allocates three 8-bit byte(s)**

**DCW 1,2,3 ; allocates three 16-bit halfwords**

**DCD 1,2,3 ; allocates three 32-bit words**

**SPACE 4 ; reserves 4 bytes**