**Exam 1**

**Date:** Oct 4, 2017

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.
* *Please read the entire exam before starting.*

|  |  |  |
| --- | --- | --- |
| **Problem 1** | 12 |  |
| **Problem 2** | 8 |  |
| **Problem 3** | 10 |  |
| **Problem 4** | 15 |  |
| **Problem 5** | 10 |  |
| **Problem 6** | 15 |  |
| **Problem 7** | 20 |  |
| **Problem 8** | 10 |  |
| **Total** | 100 |  |

**(12) Question 1.** Short answers.

Voltage when the output is low. E,g., the microcontroller output pin is 0.3V when output is low

**(2) Part a)** What does *VOL* mean?

Yes, this is true for all electrical devices

**(2) Part b)** Does the equation **power = voltage\*current** apply to both resistors and LEDs? Answer yes or no.

Information is not lost, it is retained, if power is removed and then restored

**(2) Part c)** What does nonvolatile mean (in context of computer memory?

It always branches; all values of R0 will branch; ORRS sets bit 2, so it will be nonzero

**(2) Part d)** Under what conditions does this code branch?

**ORRS R0,R0,#4**

**BNE FunTimes**

**(2) Part e)** Considering R0 as input and R1 as output,

(R1 is 16\*R0)

R1 is 15\*R0

what is the mathematically relation between R1 and R0?

**LSLS R1,R0,#4**

**SUBS R1,R1,R0**

n+1; think of example

63+15 = 78, 6bit+4bit => 7bit

**(2) Part f)** 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*.

**(8) Question 2.** Consider the shift operation.

ASRS is for signed numbers, it preserves bit 31, so negative numbers remain negative and positive numbers remain positive. LSRS is for unsigned numbers, it shifts 0’s into bit 31 as it shifts. Both shift rights effectively divide by a power of 2

LSLS works for both signed and unsigned numbers. It shifts 0’s into bit 0 as it shifts. Shift left effectively multiplies by a power of 2

**(10) Question 3** Assume **Data** is an 8-bit unsigned variable in RAM.

**uint8\_t Data;**

Write assembly code that performs the following C code (no function, just assembly code),

**if(Data >= 32){**

**Data = 255; // ceiling**

**}else{**

**Data = Data<<3;**

**}**

**LDR R1,=Data**

**LDRB R0,[R1]**

**CMP R0,#32 //8\*32 would have been 256, causing overflow**

**BLO ok**

**MOVS R0,#255 // ceiling**

**B set**

**ok: LSLS R0,R0,#3 // unsigned**

**set: STRB R0,[R1]**

**LDR R1,=Data**

**LDRB R0,[R1]**

**CMP R0,#32 //8\*32 would have been 256, causing overflow**

**BHS ceil**

**ok: LSLS R0,R0,#3 // unsigned**

**B set**

**ceil: MOVS R0,#255 // ceiling**

**set: STRB R0,[R1]**

**(15) Question 4.** Consider the following C function with two inputs and one output.

**uint32\_t x;**

**uint32\_t func(uint32\_t in1, uint32\_t in2){**

**uint32\_t out;**

**out = 1;**

**while(in1 >= in2){**

**out = out\*in2;**

**in2 = in2 + 1;**

**}**

**return out;**

**}**

**(5) Part a)** If we were to execute **x=func(6,4);** what would be the value of x?

x = 4\*5\*6=120

**(10) Part b)** Write **func** in assembly using AAPCS

**// AAPCS: inputs in R0,R1**

**func: MOVS R3,#1 //R3 is out**

**loop: CMP R0,R1 //R0 is in1, R1 is in2**

**// while loop has test first**

**BLO done**

**MULS R3,R3,R1 //out=out\*in2**

**ADDS R1,R1,#1 //in2=in2+1**

**B loop**

**done: MOVS R0,R3**

**// AAPCS: output in R0**

**BX LR**

**(10) Question 5.**

Write both C and assembly main programs that

1) Read the input pins

2) Count the number of input pins that are high.

3) If the count is 2 or 3 make the output high, if 0 or 1 make the output low

4) Pull Repeat 1-3 over and over

|  |  |
| --- | --- |
| **main:**  **BL Init**  **MOVS R0,#1 // mask**  **MOVS R7,#8 // mask**  **LDR R4,=DIN**  **LDR R5,=DOUT**  **loop:**  **LDR R3,[R4] // inputs**  **ANDS R2,R2,R0 // count**  **LSRS R1,R3,#1**  **ANDS R1,R1,R0**  **ADDS R2,R2,R1**  **LSRS R1,R3,#2**  **ANDS R1,R1,R0**  **ADDS R2,R2,R1**  **LDR R6,[R5]**  **CMP R2,#2**  **BLO clr // could be BLT**  **ORRS R6,R7 // set**  **B next**  **clr:**  **BICS R6,R7 // clear**  **next:**  **STR R6,[R5] // output**    **B loop**  **.end** | **int main(void){**  **Init();**  **while(1){**  **uint32\_t count;**  **count = DIN&0x01;**  **count += (DIN>>1)&0x01;**  **count += (DIN>>2)&0x01;**  **if(count >= 2){**  **DOUT |= 0x08;**  **}else{**  **DOUT &= ~0x08;**  **}**  **}**  **}** |

**(15) Question 6.** Assume the value of the Stack pointer (SP) is **0x20200FF8** when the following code sequence starts execution (i.e., **PC=0x00001000**). The initial stack contents are given on the right. When drawing the stack contents, you need only to show numbers on the stack that represents valid data.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0x00001000 POP {R0,R1}**  **0x00001004 ADD R2,R0,R1**  **0x00001008 BL Func B**  **0x0000100C ...**  **...**  **0x00002000 Func PUSH {R2,LR} A**  **0x00002004 MOV R2,R1**  **0x00002008 MUL R0,R2**  **0x0000200C ADD 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 = 0x20201000  
**Add R2,R0,R1** causes R2 = 5

**SP = 0x20200FF8**

**BL** causes LR = 0x0000100D

**Push R2,LR**

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 EE319K this bit will always be 1 for Thumb.

|  |  |
| --- | --- |
| **0x20200FF4** |  |
| **0x20200FF8** | **5** |
| **0x20200FFC** | **0x0000100D** |
| **0x20201000** | **4** |
| **0x20201004** | **5** |
| **0x20201008** | **6** |
| **0x2020100C** | **7** |

**(10) Part b)** Give the state of the stack (SP and contents) while executing the instruction at memory location 0x0000100C as shown by the arrow B and the values stored in R0, R1, and R2.

**movs r2,r1** causes r2=3

**muls r0,r2** causes r0 = 2\*3=6

**adds r0,r1** causes r0 = 6+3 = 9

**POP {R2,PC}**restores R2 = 7, SP to 0x20201000

**SP = 0x20201000**

|  |  |
| --- | --- |
| **0x20200FF4** |  |
| **0x20200FF8** |  |
| **0x20200FFC** | **R0 = 9** |
| **0x20201000** | **4**  **R1 = 3** |
| **0x20201004** | **5** |
| **0x20201008** | **6**  **R2 = 5** |
| **0x2020100C** | **7** |

**(20) Question 7.** Assume the microcontroller’s output voltage high is 3.3V. Assume the microcontroller’s output voltage low is 0V. The *VOL* for the 7406 driver is 0.5V. Pick resistors appropriately and assume you have 5V, 3.3V, and ground to which you can connect your components. The symbols for each part are given below for your convenience – *use the minimum number of parts to construct the interface*.

Part a) Interface the LED to Port B bit 7 (PB7) using negative logic (means low => LED on). The LED operating point is 2.3V at 2mA. 2mA<8mA so no 7406 needed

 

Show calculations for selecting resistor value(s)

R = (3.3-2.3)/2mA = 500 ohms

**Part b)** Interface this single pole double throw switch to the microcontroller PB6 input. The switch has two possibilities. The first case is the C pin is connected to the L pin. For this case, make PB6 low. The second case is the C pin is connected to the R pin. For this case, make PB6 high. The L pin is never connected to the R pin, and the C pin is connected to either L or R.



**(10) Question 8.**

**#define T1 80000 // 1 ms**

**#define T2 2\*T1 // 2 ms**

**#define T4 4\*T1 // 4 ms**

**const uint32\_t Times[256] = {T1, T1, T2, T4, T2, T1};**

**void wave(uint32\_t t);**

**int main(void){ int i=0;**

**Clock\_Init80MHz(0);**

**PB18\_Init(); // you do not need to write this**

**while(1){**

**wave(Times[i]);**

**i = (i+1)&0xFF; // 0 to 255 with wrap**

**}**

**}**

**void wave(uint32\_t t){**

**GPIOB->DOUTSET31\_0 = 1<<18;**

**Delay(t);**

**GPIOB->DOUTCLR31\_0 = 1<<18;**

**Delay(t);**

**}**