UNIVERSITY OF CAPE TOWN
DEPARTMENT OF ELECTRICAL ENGINEERING

EEE2039W (Module D), EEE2026S (Module D), EEE3070S
Introduction to Microcontrollers

FINAL EXAMINATION NOVEMBER 2014

TIME: 90 minutes

MARKED OUT OF: 88
AVAILABLE MARKS: 92

INSTRUCTIONS:

1. Attempt all questions.
2. This is a closed book exam.

Examiner: James Gowans
External Examiner: John Frost

KEEP YOUR ANSWERS SHORT AND TO THE POINT.
Questions apply to the STM32F051 on the UCT development board.

Question 1 **(11)**

a) Our system has 8 KiB of RAM. (Note: KiB not KB).
Assuming no other RAM usage, how many times could a register be pushed onto the stack before we ran out of RAM? (1)

b) Briefly discuss the main difference between the Thumb and Thumb-2 instruction sets. (2)

c) What is the block of memory which holds the addresses of the exception handlers known as? (1)

d) What two functions are provided by the NVIC. (2)

e) How does the ARM Cortex-M0 relate to the STM32F0? (1)

f) Give two advantages of reducing code duplication. (2)

g) Explain how the LDR instruction (which is only 16-bits wide) is able to implement a 32-bit register load, such as: `LDR R0, =0xAABBCCDD` (2)

Question 2 **(10)**

a) How many cycles would the following block of code take to execute? (3)

```
      LDR R0, =0xFFFF0000    (2 cycles)
loop: ADDS R0, R0, #1        (1 cycle)
      BNE loop               (3 cycles if taken, 1 if not taken)
```

b) Which two registers are affected by the `BL` instruction? What happens to them? (3)

c) Assume the SP contains the value 0x2000 1670 before the following block of code executes.
What values are contained in SP and R0 after execution of the following code? (2 + 2 = 4)

```
      LDR R0, =0x48001800
      LDR R1, =0x00112233
      PUSH {R0}
      PUSH {R0}
      PUSH {R1}
      POP  {R0}
      SUB SP, #4
      LDR R1, [SP]
      ADDS R0, R0, R1
```

Question 3 **(10)**

a) Write a block of assembly code which will: (4)

- set bits 0, 1 and 5 of the contents of the **byte** at address 0x4800 0140
- clear bits 2, 3 and 7 of the contents of the byte at address 0x4800 0140
- leave the other bits of the memory address unchanged.

b) Implement the above in C. (3)

c) Consider the attached page from the Cortex-M0 reference manual.
What binary string does the following instruction compile to? (3)

```
LDR  R0,  [SP, #8]
```

Question 4 **(9)**

In the context of a timer, describe the following (2 x 3 = 6)

a) Prescaler
b) Count Register
c) Overflow

d) Assume the timer module is being clocked at 48 MHz and the ARR is holding the value 1000. What value must be placed into the prescaler register to cause an overflow frequency of 1 kHz. (2)

e) Discuss 1 reason why we might prefer a timer rather than a delay loop for creating a delay. (1)

Question 5 **(12)**

a) Describe two advantages which C has over assembly. (2)

b) Contrast automatic variables and static variables. Make reference to how they are defined, their lifetime and their location in memory. (6)

c) What is the value of foo_ptr after the following has executed? (1)

```
int16_t  *foo_ptr  =  0x08004100;
foo_ptr  -=  4;
```

d) Assuming the above code has executed, what is the data type of `&foo_ptr`? (1)

e) Agan, assuming the code in (c) has just executed, is the following line legal? (2)
Why or why not? (marks for reason)

```
&foo_ptr  = 0x20000400;
```

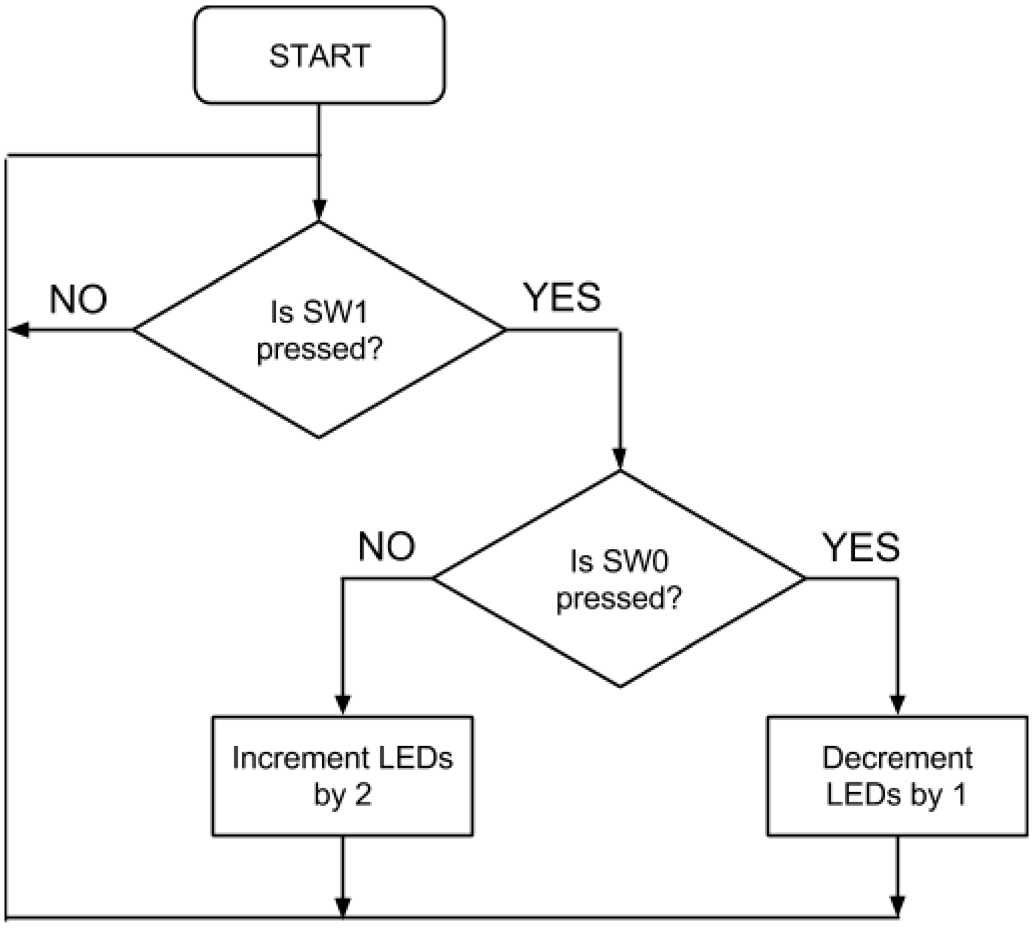
Question 6: **(9)**

a) Implement a C function called array_averager which takes as arguments: (4)

- a pointer to the start of an array of signed 16-bit numbers,
- the length of the array

The function should return the average value (aka: arithmetic mean) of the elements in the array as a signed 16-bit number.

b) Assume that ST's peripheral register definitions header file has been included. (5)
Write a block of **C code** **to implement the following. You don't need to write initialisations, just implement the logic for the flow diagram.**



Question 7: **(7)**

We wish to design a small motorised toy car that is equipped with a microphone and controlled by shouting. The louder you shout, the faster it goes. However, if it detects that it is about to hit a wall, it stops and turns on an LED.

a) In simple "black-box" diagrams, sketch a conceptual microcontroller-based design to meet the above challenge. You are expected to label appropriate hardware, peripherals and signal types. (3)

b) Draw a simple (+-6 blocks) flowchart showing the algorithm you would implement in code on the microcontroller. (4)

Question 8 **(9)**

a) What value will the variable 'i' get set to after the following two lines have run? (3)

```
uint16_t arr[] = {0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0x42};
uint32_t i = *((uint32_t*)arr);
```

b) Assuming the code below has just executed, which memory addresses will be modified if we write to: (3)
ss_ptr -> d[1]
(List all individual byte addresses)

```
struct SomeStruct
{
   uint32_t  a;
   uint16_t  b[8];
   uint16_t  *c;
   uint16_t  d[2];
};

struct  SomeStruct  *ss_ptr  =  (struct  SomeStruct  *)0x20000400;
```

c) When the following line of code is executed, which memory addresses are set to which values? List all individual byte addresses. (3)

```
*(int32_t*)0x20001440  =  -2;      // that's MINUS two.
```

Question 9 **(11)**

a) Assume that an ADC is operating in 10 bit mode, right justified.
What value would be present in the ADC data register after it has converted 1.69 volts when running off of a 3.42 V supply rail? (2)

b) Which parameter of the ADC can we adjust to cause it to take fewer cycles to perform a conversion? (1)

c) For the following line of code, is (1) or (2) a better comment? Justify. (2)
(marks for justification)

```
GPIOB->MODER = 0x5555;
```

1) // set PB0-7 to outputs
2) // write the hexadecimal number 0x55555 to the GPIOB Mode Register

d) What value is displayed on the LEDs after the following block of code is executed? (4)

```
GPIOB->ODR  =  0;
uint16_t  counter  =  0;
for ( ;  counter < 16;  ++counter) {
            GPIOB->ODR += (counter << 3);
}
```

e) What is the value of `i` after the following has executed?
Explain. (marks for explanation) (2)

```
uint8_t  i  =  0;
uint8_t  foo  =  0xAA;
uint8_t  bar  =  0x55;

if (foo  && (bar  & foo))  {
      i  =  100;
}
```

Bonus: **(4)**
Discuss the details of the way our microcontrollers interfaces with the LCD screen.

END