Programme Code: TU857
Module Code: CMPU2013

TECHNOLOGICAL UNIVERSITY DUBLIN

Grangegorman

TU857 - BSc. (Honours) Degree in Computer Science (Infrastructure)

Year 2

SEMESTER 1 EXAMINATIONS 2022/23

CMPU2013 Microprocessors

Internal Examiners: Frank Duignan
Dr. Paul Doyle

External Examiner(s):
Ms. Caroline McEnroy

Instructions to Candidates: Answer **3** of the following 4 questions

Exam Duration: 2 hours

Special Instructions: Numbers with the prefix **0x** are in hexadecimal

Question 1:
(a) Using your calculator or otherwise, determine the 16 bit results of the following C-language calculations. Express your answer in hexadecimal.

(i) 0xfe00 & (1 << 9)

[3 marks]

(ii) 0x1a23 | (0x9210)

[3 marks]

(b) What value will X have after the following C-code fragment?

```
int16_t X;  // X is a signed 16 bit number
X = -32767;
X = X – 2;
```

[5 marks]

(c) A computer system is required to store unsigned numbers in the range 0 to 16 million. What is the minimum number of bits required to store one of these numbers?

[4 marks]

(d) Comment on the roles played by each of the following in computer systems:
(i) The address bus

[2 marks]

(ii) The data bus

[2 marks]

(iii) The instruction pipeline

[2 marks]

(e) A serial communications link operates at 38400 bits per second and uses odd parity checking.
(i) Will the parity bit be 0 or 1 when the value 0x49 is transmitted?

[3 marks]

(ii) Assuming an overhead of 3 bits per byte, how long will it take to send a block of 16384 bytes over this link?

[3 marks]

(f)
The code shown in Listing Q1a converts a 24 bit colour value to an 16 bit (RGB-565) value:

```
uint16_t RGBToWord(uint16_t R, uint16_t G, uint16_t B)
{
   uint16_t Colour=(R & 0xf8) << 8;
   Colour = Colour + ( (G & 0xfc) << 3);
   Colour = Colour + (B >> 3);
   return Colour;
}
```

Listing Q1a

Show that the output is **0x8410** when the function is called as follows:

```
RGBToWord(128,128,128);
```

[6 marks]

Question 2:

(a) Port A of the STM32F031 is associated with three registers:

GPIOA->MODER
GPIOA->ODR
GPIOA->IDR

(i) State the function of each of these registers.

[6 marks]

(ii) An STM32F031 microcontroller program is required to set BIT2 of GPIOA->ODR without affecting the other bits. Show how you would do this in a single line of C-code.

[4 marks]

(iii) An STM32F031 microcontroller program is required to wait for bit 4 of GPIOA->IDR to become 0. The states of the other bits are not known in advance. Show how you would program this in C.

[4 marks]

(iii) Explain how "Pull-up" resistors are commonly used to convert mechanical switch or button movements into electrical signals suitable for a microcontroller's digital inputs.

[4 marks]

(b) Listing Q2a contains C-code for a function that can be used to set a particular bit (bitnumber) in a memory location

```
void setBit(volatile uint32_t *locn, uint32_t bitnumber)
{
  uint32_t value;
  uint32_t mask;
  value = *locn;  // read current value
  mask = (1 << bitnumber); // create the mask
  value = value | mask; // change the value (OR)
  *locn = value;  // write back to memory;
}
```

Listing Q2a

(i) Explain the operation of the **setBit** code.

[4 marks]

(ii) Write the compliment of this function **clearBit** which makes a particular bit in a memory location zero

[6 marks]

```
typedef struct
{
  volatile uint32_t MODER;
  volatile uint32_t OTYPER;
  volatile uint32_t OSPEEDR;
  volatile uint32_t PUPDR;
  volatile uint32_t IDR;
  volatile uint32_t ODR;
  volatile uint32_t BSRR;
  volatile uint32_t LCKR;
  volatile uint32_t AFRL;
  volatile uint32_t AFRL;
  volatile uint32_t BRR;
} GPIO_TypeDef;

#define GPIOA ((GPIO_TypeDef *) 0x4800 0000)
```

Listing Q2b

Listing Q2b shows how the symbol GPIOA can be created in C. This symbol can be used to access the various elements of General Purpose Input/Output port A in the STM32F031.

(iii) What is the meaning of the **volatile** keyword used in the structure definition?

[3 marks]

(iv) At what address is the symbol **GPIOA->IDR** ?

[2 marks]

Question 3:

(a) The contents of some of the STM32F031 core registers are as shown below. Also shown are the contents of some memory locations. What number goes where when each of the following instructions is executed one after another in a program?

PUSH R1

[3 marks]

POP R0

[3 marks]

POP R2

[3 marks]

Contents of Registers

Register	**Contents**
R0	0x00000006
R1	0x00000007
R2	0x00000008
SP	0x20000ffc

Contents of RAM

Address	**Contents**
0x20000fec	0x00000001
0x20000ff0	0x00000002
0x20000ff4	0x00000003
0x20000ff8	0x00000004
0x20000ffc	0x00000005

(b) The following assembly language instruction can be used to place a 32 bit value in an ARM Cortex-M0 register. How is this encoded in 16 bits?

```
LDR R0,=0x12345678
```

[7 marks]

(c) Listing Q3a shows an ARM Thumb assembler listing for a function that copies one block of memory to another

(i) What happens when Lines A, B and C are executed?

[9 marks]

(ii) Write a few lines of C code that calls this function.

[4 marks]

(iii) What is meant by the Arm Architecture Procedure Call Standard?

[4 marks]

```
AREA THUMB, CODE, READONLY
EXPORT my_memcpy
; void my_memcpy(uint8_t *dest, uint8_t * src, uint32_t n);
; on entry, R0 = dest, R1 = src, R2 = n
my_memcpy
my_memcpy_loop
CMP R2,#0
BEQ my_memcpy_exit ;----   LINE A
LDRB R3,[R1]
STRB R3,[R0] ;----------    LINE B
SUBS R2,R2,#1
ADDS R0,R0,#1
ADDS R1,R1,#1
B my_memcpy_loop
my_memcpy_exit
BX LR ;----------  LINE C

end
```

Listing Q3a

Question 4:

(a) What is the principal function of the following ARM Cortex M0 registers?

(i) PC

[2 marks]

(ii) LR

[2 marks]

(b) What ARM Cortex M0 Arithmetic flags are set by the following instructions sequences

(i)

```
EORS R0,R0
```

[3 marks]

(ii)

```
LDR R0,=0xfffffffe
ADDS R0,R0,#2
```

[3 marks]

(iii)

```
MOVS R0,#4
SUBS R0,#8
```

[3 marks]

(iv)

```
LDR R0,=0x80000000
SUBS R0,#2
```

[3 marks]

(c) Using suitable examples, explain the use of the following GNU ARM Thumb assembler directives:

(i) SPACE

[2 marks]

(ii) EXPORT

[2 marks]

(iii) AREA

[2 marks]

(d) The STM32F031 uses a load/store architecture. What is this and how does it affect how it performs the addition of two numbers in memory?

[6 marks]

(e) Using suitable examples explain the operating of the following addressing modes in the STM32F031:

(i) Immediate addressing

[2 marks]

(ii) Register indirect with offset addressing

[3 marks]