Programme Code: TU856
Module Code: CMPU1019

TECHNOLOGICAL UNIVERSITY DUBLIN

Grangegorman

TU856 - Bachelor of Science (Honours) in Computer Science

Year 1

SEMESTER 2 EXAMINATIONS 2022/23

CMPU1019 Microprocessor Systems

Internal Examiner(s):
Frank Duignan

Dr. Paul Doyle

Instructions To Candidates:
Answer 3 of the following 4 questions

Exam Duration: 2 hours

Special Instructions /Handouts/ Materials Required:

Numbers prefixed by 0x are in hexadecimal (base 16)

Question 1

(a) Listing Q1a shows a C program that prints out ascending values of the variable **count**. The output from the program is as follows:

```
32765
32766
32767
-32768
-32767
-32766
```

Using binary notation, explain why the output value changes sign.

```
#include <stdio.h>
#include <stdint.h>
int main()
{
   int16_t count=32765;
   for (int i=0; i < 6;i++)
   {
      printf("%d\n",count);
      count++;
   }
}
```

Listing Q1a

[5 marks]

(b) Table Q1b shows a section of the ASCII character set. Listing Q1b contains a C function to convert a number to a string representing this value in hexadecimal. Assuming the value passed to the function is 0x1B4, show how the contents of the **HexString** are filled during the first three passes of the **while** loop.

[6 marks]

0 48	: 58	A 65	K 75	U 85
1 49	; 59	B 66	L 76	V 86
2 50	< 60	C 67	M 77	W 87
3 51	= 61	D 68	N 78	X 88
4 52	> 62	E 69	O 79	Y 89
5 53	? 63	F 70	P 80	Z 90
6 54	@ 64	G 71	Q 81	
7 55		H 72	R 82	
8 56		I 73	S 83	
9 57		J 74	T 84	

Table Q1b

```
void printHex(uint16_t value)
{
   char HexString[5];
   int digit;
   int index;
   HexString[4] = 0;
   index=3;
   while(index >= 0)
   {
      digit = value % 16;
      value = value / 16;
      if (digit <= 9)
      {
         digit = digit + 48;
      }
      else
      {
         digit = digit + 55;
      }
      HexString[index]=digit;
      index--;
   }
   puts(HexString);
}
```

Listing Q1b

(c) What is the 16 bit hexadecimal result of the following C-language calculations:

i. 0x1A9D & ~0xB28C

[2 marks]

ii. 0xAA55 ^ 0x55AA

[2 marks]

(d)

i. What role does a UART typically play in a microprocessor system?

[4 marks]

ii. A function which waits for a character to arrive on a UART in the STM32F031 is shown in Listing Q1d. **Line A** continues to loop until a character is received on USART1. What calculation is carried out in the **while** statement and when will it exit?

[6 marks]

```
char egetchar()
{
    while( (USART1->ISR & (1 << 5)) == 0); // Line A
    return (char)USART1->RDR;
}
```

Listing Q1d

(e) A serial communications link operates with odd parity checking and at a speed of 19200 bits per second.

i. Will the parity bit be a 1 or 0 when the character 'K' is transmitted (see ASCII table in question Q1b)?

[4 marks]

ii. Assuming an overhead of 3 bits per byte and no delay between each transmission, how long will it take to send a message of 1500 bytes?

[4 marks]

Question 2

Figure Q2a shows a circuit sketch showing how a button and LED are connected to an STM32F031 microcontroller. Assuming that the input/output (I/O) pins have been configured at boot time, write C functions that do the following:

(a) Turn the LED on without changing any other I/O port bits.

[4 marks]

(b) Turn the LED off without changing any other I/O port bits

[4 marks]

(c) Read the state of the button. A value of 1 should be returned if the button is pressed, otherwise the function should return 0.

[5 marks]

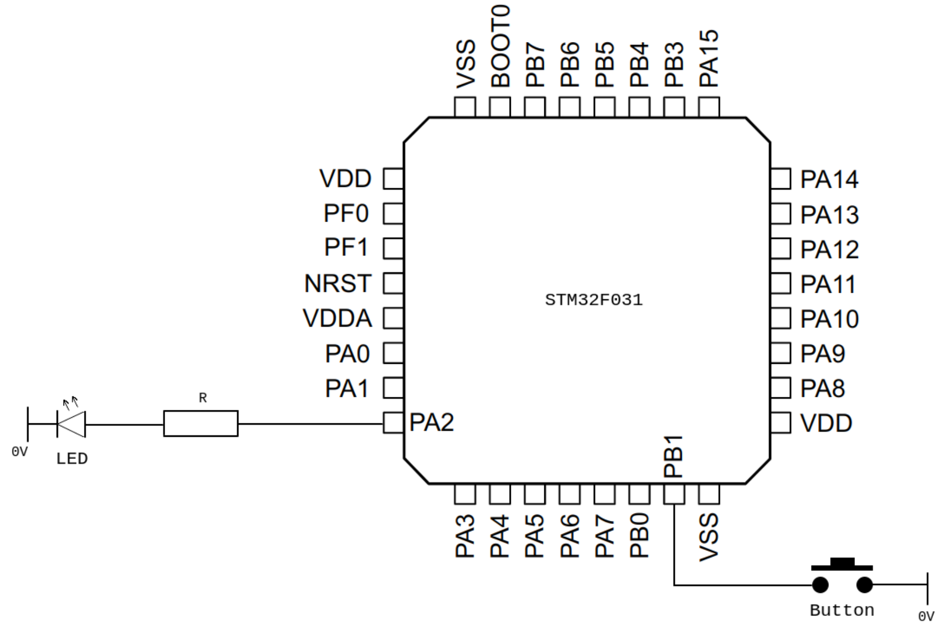

Figure Q2a

(d)

i. Outline the operation of the SysTick timer in ARM Cortex M microcontrollers such as the STM32F031

[6 marks]

ii. The clock source for a SysTick timer is 48MHz. The timer is required to interrupt the CPU at a rate of 10000Hz (10kHz). What value should be placed in the Auto-Reload Register (SysTick->Load) to achieve this?

[4 marks]

(e) What is meant by each of the following terms:

i. Hardware Interrupt Request

[2 marks]

ii. Interrupt Vector Table

[2 marks]

iii. Interrupt Handler

[2 marks]

(f) Outline the sequence of events that occurs when an STM32F031 receives and processes (handles) an interrupt request

[4 marks]

Question 3

Listing Q3a contains an assembly language program for the STM32F031 which concatenates (joins) two strings together.

(a) Identify two Assembler Directives in the program and explain what they do.

[4 marks]

(b) Identify an example of Immediate addressing in the program and explain what that instruction does.

[2 marks]

(c) Identify an example of Register Indirect addressing in the program and explain what that instruction does.

[2 marks]

(d) Which ALU flag is checked when the instruction on LINE B is executed?

[2 marks]

(e) The operands for the instructions in lines marked LINE A and LINE C are almost the same (**LR** is replaced by **PC**). Why do they differ and what do these instructions achieve?

[6 marks]

(f) The **strcat** function runs the risk of overflowing the target string. A safer alternative is strncat which has the following prototype:

char *strncat(char *dest**, const char** *src**, int** n**);**

Where:

dest is a pointer to the destination string

src is a pointer to the source string

n is maximum number of bytes that should be copied from the source string

i. In what register will the value **n** be passed to this function?

[4 marks]

ii. What instruction would you use to subtract 1 from this register during the execution of the function?

[4 marks]

iii. What instruction(s) would you use to test if this register has reached zero?

[4 marks]

iv. Hence modify the strlwr function in listing Q3a so that it implements **strncat**.

[5 marks]

```
    AREA DATA
Dest SPACE 100

    AREA THUMB, CODE, READONLY

Reset_Handler
    ; put two letters in to the destination string
    ; for testing purposes
    LDR R0,=Dest
    MOVS R2,#'a'
    STRB R2,[R0]
    ADDS R0,R0,#1
    STRB R2,[R0]
    ; insert a NULL (0) to terminate the string.
    MOVS R2,#0
    ADDS R0,R0,#1
    STRB R2,[R0]
    ; add the source string (Src) to the end of the destination
    ; string (Dest)
    LDR R0,=Dest
    LDR R1,=Src
    BL mystrcat
Loop
    B Loop ; while(1);
; char *strcat(char *dest, const char *src);
; on entry:
; R0 points at the destination string
; R1 points at the source string
; on exit:
; R0 points at the destination string.
mystrcat
    PUSH {R0-R7,LR} ; *** LINE A ***
seek_end
    LDRB R3,[R0]
    CMP R3,#0
    BEQ copy_loop ; *** LINE B ***
    ADDS R0,R0,#1
    B seek_end
copy_loop
    LDRB R3,[R1]
    STRB R3,[R0]
    CMP R3,#0
    BEQ exit_copy_loop
    ADDS R0,R0,#1
    ADDS R1,R1,#1
    B copy_loop
exit_copy_loop
    POP {R0-R7,PC} ; *** LINE C ***

Src DCB "HelloWorld",0

    END
```

Listing Q3a

Question 4

(a) State the normal function of the following ARM Cortex M0 registers:

i. PC

[2 marks]

ii. LR

[2 marks]

iii. SP

[2 marks]

(b) Modern microprocessors typically include additional hardware to accelerate their performance. Describe what is meant by each of the following and how they increase performance.

i. Instruction Pipelining

[5 marks]

ii. Cache

[5 marks]

(c) Listing Q4a shows an assembly language function **get_user_data** whose job is to process data entered by a user via the function **egetchar** (in a different program module). The **get_user_data** function makes use of a local string buffer which accumulates the characters entered by the user up until they press the *Enter* key.

i. Is the local string buffer allocated on the heap, stack or in the global memory area?

[4 marks]

ii. Why would the program crash if the line marked LINE A was not included?

[6 marks]

iii. Why will the program likely crash if the user enters more than 16 characters?

[7 marks]

```
get_user_data
        PUSH {LR}
        SUB SP,#16      ; allocate 16 bytes for a buffer
        MOV R1,SP  ; make R1 point at the buffer
        MOVS R2,R1 ; make R2 point at the buffer
        ; Now go and fetch data from the user using egetchar.
        ; exit when the user presses ENTER
get_user_data_loop
        BL egetchar
        CMP R0,#'\r' ; check for ENTER pressed
        BEQ get_user_data_process
        STRB R0,[R2]
        ADDS R2,R2,#1
        B get_user_data_loop
get_user_data_process
        ; Process user data (not shown)
get_user_data_exit
        ADD SP,#16 *** LINE A ***
        POP {PC}
```

Listing Q4a