2024 EXAMINATIONS

Lancaster University

Part I

COMPUTING AND COMMUNICATIONS

SCC.131 – Digital Systems – In-Person, Online [1.5hrs]

---

*Candidates are asked to answer **THREE** questions from **THREE**; each question is worth a total of 25 marks.*

Question 1

1.a. [MCQ; single answer] Different machine architectures may organise multi-byte words differently in memory. Select which statement is true with respect to memory organisation. (Select one)

i. Big-endian: within a multi-byte word (e.g. 4 bytes for a 32-bit integer) the location (byte) with the lowest memory address holds the least-significant-byte (LSByte)
Little-endian: within a multi-byte word (e.g. 4 bytes for a 32-bit integer), the location (byte) with the lowest memory address holds the most-significant-byte (MSByte)

ii. Big-endian: within a multi-byte word (e.g. 4 bytes for a 32-bit integer), the location (byte) with the lowest memory address holds the most-significant-byte (MSByte)
Little-endian: within a multi-byte word (e.g. 4 bytes for a 32-bit integer) the location (byte) with the lowest memory address holds the least-significant-byte (LSByte)

iii. Big-endian: within a multi-byte word (e.g. 4 bytes for a 32-bit integer), the location (byte) with the highest memory address holds the most-significant-byte (MSByte)
Little-endian: within a multi-byte word (e.g. 4 bytes for a 32-bit integer) the location (byte) with the lowest memory address holds the least-significant-byte (LSByte)

[1 mark]

1.b. [MCQ; single answer] Of the following, which one is the most important distinguishing feature of the Harvard Architecture? (Select one)

i. There is a control unit
ii. Instructions and data are held in the same memory
iii. Instructions and data are held in separate memories
iv. There are input and output mechanisms

[1 mark]

1.c. [MCQ; multiple answer] Consider the following statements about the DAPLink and **select those that are true**. False statements carry negative marks.

i. The DAPLink is a hardware component of micro:bit.
ii. The DAPLink is a bus that connects the interface MCU to the application MCU.
iii. The DAPLink has an interface mode and a bootloader mode.
iv. The DAPLink provides micro:bit with drag-and-drop programming and debugging features.
v. The DAPLink is firmware that runs on the interface MCU of the micro:bit.
vi. The DAPLink is software written in C/C++ that abstracts hardware components as software components.
vii. The DAPLink is firmware that runs on the application MCU of the micro:bit.

[1 mark]

Question 1 continues on next page…

Question 1 continued.

1.d. [MCQ; single answer] Fill the gap:

"One of the De Morgan's laws says that the negation of AND statement is logically equivalent to the _____ statement in which each component is negated." (Select one)

i. AND
ii. OR
iii. NOT
iv. XOR

[1 mark]

1.e. [MCQ; single answer] Which of the following belongs to the control lines: (Select one)

i. Audio card.
ii. Memory address register.
iii. Keyboard.
iv. Latch.

[1 mark]

1.f. [MCQ; single answer] Assume that you have updated the source file main.cpp, compiled it and obtained the executable file MICROBIT.hex, which you are about to transfer to a micro:bit device. Which **one** of the following statements is true?

i. MICROBIT.hex has been dynamically linked against shared objects.
ii. MICROBIT.hex has been statically linked against archives of objects.
iii. Either option (a) or (b) can be chosen for MICROBIT.hex. In both cases, MICROBIT.hex can run on micro:bit.

[1 mark]

1.g. In ASCII code, the character B is represented by the value 66 in decimal. The character H is represented by the value 72 in decimal.

i. Represent the character B in 7-bit binary code
ii. Represent the character H in hexadecimal code
iii. Represent the character H in octal code

[3 marks]

1.h. Convert the following decimal numbers to 8-bit binary using Excess 127:

i. -3
ii. -126
iii. 0

[3 marks]

Question 1 continues on next page…

Question 1 continued.

1.i. Convert the following decimal numbers to 8-bit 2's complement binary and then add them (giving the answer also in 8-bit 2's complement binary). Show your working.

-23 + -15

[3 marks]

1.g. Use a Karnaugh map to find a minimum expression of the following function. Show the map and your final answer.

A'B'C'D'+A'B'CD'+ABC'D+ABCD

[2 marks]

1.k. Using the theorems of Boolean algebra, simplify the following Boolean expression, showing your working.

F= A'B+AB'+AB

[1 mark]

1.l. The source file sphere.c below contains two functions that calculate the surface area and the volume of a sphere for a given radius:

```
1  #include <stdio.h>
2
3  float pi = 3.14159;
4
5  float sphereArea(float radius) {
6    float calcArea = 4*pi*(radius*radius);
7    return calcArea;
8  }
9
10 float sphereVolume(float radius) {
11   float calcVol = sphereArea(radius)*(radius/3);
12   return calcVol;
13 }
14
15 int main () {
16   float r = 18;
17   printf("Surface area for r=%0.2f: %0.2f.\n", r, sphereArea(r));
18   printf("Volume for r=%0.2f: %0.2f.\n", r, sphereVolume(r));
19 }
```

i. After the source file sphere.c is compiled, the **object file** sphere.o is obtained. What will **the most likely size of each section of the object file** be?

i. TEXT: 502 bytes, DATA: 0 bytes, BSS: 0 bytes
ii. TEXT: 498 bytes, DATA: 0 bytes, BSS: 4 bytes
iii. TEXT: 498 bytes, DATA: 4 bytes, BSS: 0 bytes

[1 mark]

Question 1 continues on next page…

ii. **[MCQ; single answer]** The **executable file** sphere is then created and debug information is added to it. GDB is used to debug sphere and **breakpoints are introduced in lines 7 and 12**. Carefully read the source code and **specify the index of the frame in the stack that is associated with function** main(), whenever a breakpoint is reached during runtime.

When breakpoint in **line 7** is reached **for the first time**, main() is:

i. associated with frame 0
ii. associated with frame 1
iii. associated with frame 2
iv. not in the stack.

[1 mark]

iii. **[MCQ; single answer]** When breakpoint in **line 7** is reached **for the second time**, main() is:

i. associated with frame 0
ii. associated with frame 1
iii. associated with frame 2
iv. not in the stack.

[1 mark]

iv. **[MCQ; single answer]** When breakpoint in **line 12** is reached, main() is:

i. associated with frame 0
ii. associated with frame 1
iii. associated with frame 2
iv not in the stack.

[1 mark]

v. The **instruction pointer** of the frame associated with function sphereArea() points to the address in main(), where control will return after sphereArea() completes execution. The address (in hexadecimal representation) of the instruction pointer is 0x7fffffffdf28. GDB reports that the following **four** 32-bit words (also in hexadecimal representation) have been stored in the memory region starting from address 0x7fffffffdf20:

```
0x7fffffffdf20: 0xffffdf40  0x00007fff  0x555551e6  0x00005555
```

Based on this information, determine the **64-bit value** stored at 0x7fffffffdf28, assuming that the system is **little endian**. Provide the stored 64-bit value in hexadecimal format including leading zeros.

[1 mark]

Question 1 continues on next page…

Question 1 continued.

m. The source file below uses C preprocessing language to create and display a string:

```
1  #include <stdio.h>
2
3  #define ab "The future"
4  #define cd "Next year"
5  #define CONCAT(c,d) c##d
6
7  int main () {
8    char *a = "20";
9    char *b = "25";
10    printf("%s", CONCAT(a,b));
11  }
```

Determine the string that will be displayed give it as it would be displayed on the screen.

[2 marks]

[Total 25 marks]

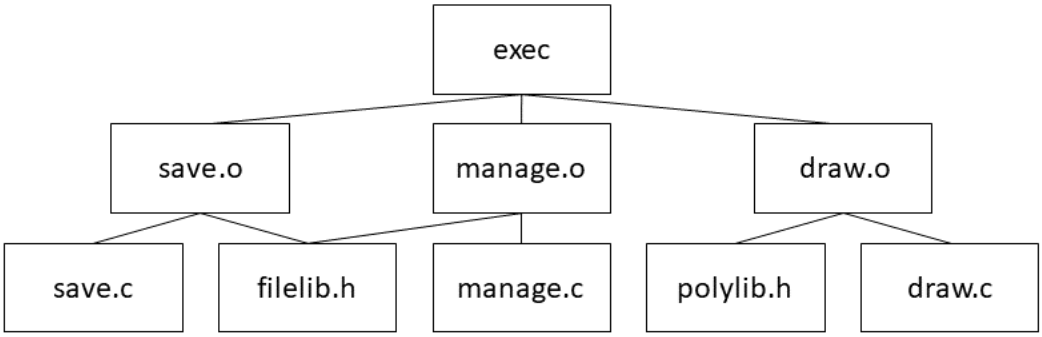
Question 2

2.a. [MCQ; multiple answer] Which of the following registers should be preserved when calling a function:

i. r0
ii. r6
iii. r2
iv. r12
v. r4

[2 marks]

2.b. The diagram in this question describes the way an executable file called exec depends on three object files which, in turn, depend on three C source files and two header files:



Write the Makefile below into your answer booklet, adding the keywords below into the empty boxes to create a makefile, which contains rules that correctly reflect the dependencies shown in the diagram:

```
# Makefile for exec
[          ] = gcc

exec: save.o [          ] draw.o
        $(CC) save.o [          ] draw.o -o [          ]

save.o: [          ] filelib.h
        [          ] -c [          ] -o save.o

[          ] : manage.c [          ]
        [          ] -c manage.c -o [          ]

[          ] : draw.c [          ]
        [          ] -c draw.c -o [          ]
```

Available Keywords:
exec, filelib.h, CC, save.c, draw.o, manage.o, polylib.h, $(CC)

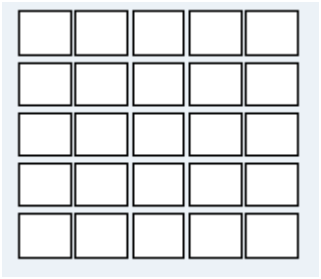
[3 Marks]

Question 2 continues on next page…

2.d. The source code in this question turns on pixels of the micro:bit display at specific coordinates:

```
1  #include "MicroBit.h"
2
3  MicroBit uBit;
4
5  int main () {
6
7    uBit.init();
8
9    for (int y = 0; y <= 4; y++)
10     uBit.display.image.setPixelValue(y, y, 255);
11
12   for (int y = 0; y <= 4; y++)
13     uBit.display.image.setPixelValue(4-y, y, 255);
14
15   uBit.display.image.setPixelValue(2, 2, 0);
16 }
```

The array below represents the 5x5 display of the micro:bit. Draw a 5x5 table in your answer booklet and write keywords "ON" and "off" into the 5x5 array to indicate which pixels will turn on and which pixels will remain off, when execution of the program is completed. Fill in all of the 25 elements of the array (empty elements are not assumed to be in the off state).

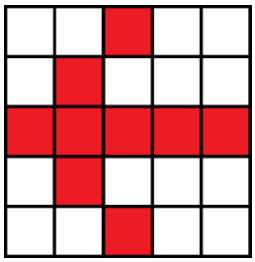


[3 marks]

Question 2 continues on next page…

Question 2 continued.

2.e. Let arrow be a MicroBitImage that represents the following 5x5 bitmap picture:



Copy the code below and fill in **all of the empty gaps**, so that the tip of the arrow initially appears on the **right side** of the micro:bit display, and then gradually shifts to **left side** of the display until the arrow comes into **full view**.

```
for (int      ;      ;      )
{
   uBit.display.image.paste(arrow, x,    );
   uBit.sleep(200);
}
```

[2 marks]

2.f. A colleague of yours has developed source code, in which uBit of type MicroBit and an array b of type PacketBuffer that contains a single byte have been declared. Function main(), which initialises the radio component of micro:bit and sets up event listeners for button A, button B and button AB (when both buttons A and B are pressed together), has also been completed. You have been tasked to create **event handlers** for the following events:

- If **button AB** is pressed, the **radio** component of micro:bit should **send** the contents of array b over the air as a **datagram**.
- If **button A** is pressed, the value of b[0] should **reduce by 1** but should not drop below 0. If button A is pressed when b[0] is equal to 0, the value of b[0] should change to 9. The final value of b[0] should be printed on the display.
- If **button B** is pressed, the value of b[0] should **increase by 1** but should not go above 9. If button A is pressed when b[0] is equal to 9, the value of b[0] should change to 0. The final value of b[0] should be printed on the display.

Copy the code and fill in **all of the empty gaps** in the code below.

Question 2 continues on next page…

```
1   void onButtonA(MicroBitEvent e) {
2       b[0] = (b[0]>0) ?                  :                ;
3       uBit.display.print(b[0]);
4   }
5
6   void onButtonB(MicroBitEvent e) {
7       b[0] = (b[0]<9) ?                  :                ;
8       uBit.display.print(b[0]);
9   }
10
11  void onButtonAB(MicroBitEvent e) {
12      uBit.                                               ;
13  }
```

[3 marks]

2.g. Your program encodes a subtract instruction (SUB Rd, Rn, Rm) as the following 32-bit value:

0xeba8060c

Select which **registers** are used as the source (Rn, Rm) and destination of the instruction in the table below.

Hint: The encoding format of the Sub instruction is the following:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	1	1	1	0	1	S	Rn				(0)	imm3			Rd				imm2		stype		Rm			

[3 marks]

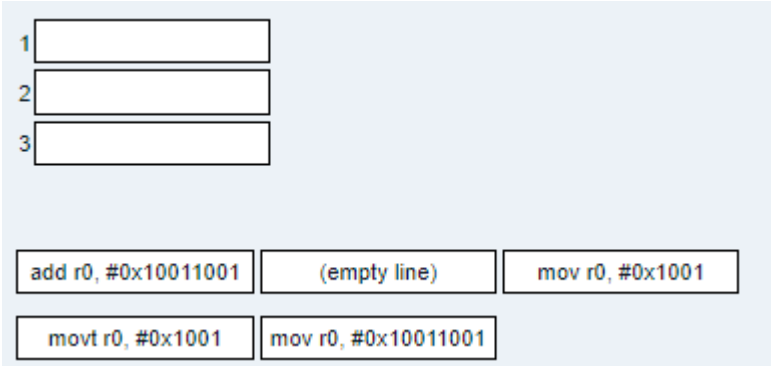
Register map

Operand	Register
Rd	
Rn	
Rm	

Question 2 continues on next page…

Question 2 continued.

2.h. Construct the minimum ARM program that loads the value 0x10011001 into register r0. Assume the register is set to the value 0x0 at the beginning. Your instructions should be in the correct order. You can use up to 4 instructions. If you need less than four instructions, use the empty line option to fill in the empty lines.



Note: The following program layout allows up to four different instructions. If you believe your solution requires less instruction, fill in the empty lines with an empty value.

[3 marks]

2.i. You are given the following C code.

```
int j = 0;
for (int i = 0; i < 10; i++) {
   j = j + i;
}
```

Write a block of ARM Assembly that will match the functionality of the previous C block. Provide a mapping of C variable into program registers. You are allowed to use any register from the ARM CPU.

Note: Code will be manually marked, and you will not be marked down for minor syntax errors.

[6 marks]

[Total 25 Marks]

On the next page you can find information about ARM assembly syntax and the format of the instructions that we used during the academic year:

Instruction

Syntax	Flags
adc{s} {Rd,} Rn, Rm {, shift}	NZCV
add{s} {Rd,} Rn, #const	NZCV
adc{s} {Rd,} Rn, #const	NZCV
qadd {Rd,} Rn, Rm	Q
sub{s} {Rd,} Rn, Rm {, shift}	NZCV
sub{s} {Rd,} Rn, #const	NZCV
and{s} {Rd,} Rn, Rm {, shift}	NZCV
bic{s} {Rd,} Rn, Rm {, shift}	NZCV
orr{s} {Rd,} Rn, Rm {, shift}	NZCV
orn{s} {Rd,} Rn, Rm {, shift}	NZCV
eor{s} {Rd,} Rn, Rm {, shift}	NZCV
and{s} {Rd,} Rn, #const	NZCV
bic{s} {Rd,} Rn, #const	NZCV
orr{s} {Rd,} Rn, #const	NZCV
orn{s} {Rd,} Rn, #const	NZCV
eor{s} {Rd,} Rn, #const	NZCV
cmp Rn, Rm {, shift}	NZCV
cmp Rn, #const	NZCV
mov{s} Rd, Rm	NZ
mov{s} Rd, #const	NZC
lsl{s} Rd, Rm, Rs	NZC
lsr{s} Rd, Rm, Rs	NZC
asr{s} Rd, Rm, Rs	NZC
ror{s} Rd, Rm, Rs	NZC
lsl{s} Rd, Rm, #const	NZC
lsr{s} Rd, Rm, #const	NZC
asr{s} Rd, Rm, #const	NZC
ror{s} Rd, Rm, #const	NZC
ldr Rd, label	-
ldr Rd, [Rb, Ri {, LSL n}]	-
str Rs, [Rb, Ri {, LSL n}]	-
ldr Rd, [Rb {, #const}]	-
str Rs, [Rb {, #const}]	-
b<c> label	-
bl label	-
bx Rm	-
blx Rm	-

Condition Flags

Flag	Meaning
N	Negative
Z	Zero
C	Carry
V	Overflow

Condition

Condition	Meaning
eq	Equal
ne	Not equal
cs, hs	Carry set, unsigned higher or same
cc, lo	Carry clear, unsigned lower
mi	Minus, negative
pl	Plus, positive or zero
vs	Overflow set
vc	Overflow clear
hi	Unsigned higher
ls	Unsigned lower or same
ge	Signed greater or equal
lt	Signed less
gt	Signed greater
le	Signed less or equal

Question 3

3.a. [MCQ; multiple answer] Which **two** of the following are the primary purposes of a sound, graphics, or network card?

i. As a way to support multiple programs using that sub-system at the same time by installing multiple cards of the same type.
ii. As the only way to enable functionality for the sub-system.
iii. As internal decoration of the computer case with coloured LEDs.
iv. As a way to reduce wear-and-tear on the main CPU.
v. As an interface to physically connect to the outside world.
vi. As an accelerator to off-load work from the CPU.

[2 marks]

3.b. [MCQ; single answer] What is a graphics shader?

i. A Generative AI LLM (large language model) that generates memes.
ii. A program the runs on the GPU for each primitive of a specified type.
iii. An analogue connector to a tv or monitor output.
iv. A tool that makes the windows of a program darker for accessibility reasons.

[1 mark]

3.c. [MCQ; multiple answer] For a program running on a computer on the Internet to make a connection to another program on another computer, what are the **two** things the program starting the connection needs to know?

i. The operating system that the remote machine is running.
ii. The IP address of the remote machine the target program is running on.
iii. The type of network card the remote machine is running.
iv. What programming language the target program was written in on the remote machine.
v. The port of the remote machine that the target program is running on.
vi. What connection speed the remote machine is capable of.

[2 marks]

3.d. [MCQ; single answer] Which one of the following doesn't **NOT** use a virtual machine?

i. WebAssembly
ii. JVM
iii. P-code
iv. x86

[1 marks]

3.e. [MCQ; single answer] WebAssembly uses which of the following architectures?

i. Harvard architecture
ii. von Neumann architecture

[1 marks]

Question 2 continues on next page…

Question 2 continued.

3.f. [MCQ; single answer] What is the maximum **signed integer** that can be represented by a register in the micro:bit CPU?

i. $2^{31} - 1$
ii. $2^{32} - 1$
iii. $2^{63} - 1$
iv. 2^{32}
v. 2^{31}
vi. 2^{64}

[1 mark]

3.g. Consider the following program:

```
.data
test: .word 0x00112233, 0x44556677, 0x8899aabb, 0xccddeeff
ldr r0, =test
ldr r1, [r0, 4]
```

Write the content of register r1 after the code is executed.

Warning: Your answer should be expressed as a 32-bit (8 digit) hex value without a prefix, for example 00000000

[1 marks]

3.h. Audio in the physical world is analogue. Explain how we could represent an audio signal in the digital world, making reference to the key terms of *sample rate* and *bit depth*, and the effect on the *quality* of the representation.

[4 marks]

3.i. Briefly (a few sentences) explain what a *Direct Memory Access (DMA) controller* is and why it may be advantageous to use one?

[3 marks]

3.j. Explain what an *interrupt* is and briefly explain **one** example of how an interrupt can be used by an operating system.

[4 marks]

3.k. What is an *operating system kernel* and give **two** things it is responsible for?

[4 marks]

[Total 25 marks]

---End of Paper---