COMP2300/6300 Final Exam 2019

Student ID:

u							

Reading time: 15 minutes

Writing time: 180 minutes

Make sure you read each question carefully. Some words have footnotes[1] to clarify what they mean in the context of the question.

Questions are not equally weighted, and the size of the answer box is not necessarily related to the length of the expected answer or the number of marks given for the question.

All answers must be written in the boxes provided in this booklet. You will be provided with scrap paper for working, but only the answers written in this booklet will be marked. Do not remove this booklet from the examination room. There is additional space at the end of the booklet in case the boxes provided are insufficient. If you use these extra pages, make sure you clearly label which question the answer refers to.

Greater marks will be awarded for answers that are short and specific rather than long, vague, or rambling. Marks may be deducted for providing information that is irrelevant to a question. If a question ask for you to "explain your answer", make sure both your *answer* (e.g. yes/no) and your *explanation* are clearly indicated. If a question has several parts, you may answer the later parts even if you cannot answer the earlier ones.

Where you are asked to write assembly code programs, marks will not be deducted for minor syntax errors.

[1] like this one!

For examiner use

Question 1 Logic, Bits, and Instructions (25 marks total)

Part 1 2 marks

Suppose the following code is executed. What will the final value of `r0` be? Choose your answer from the options below.

```
mov r0, 0b10101101
mov r1, 0b00011101
lsl r1, 3
orr r0, r0, r1
```

Answer:

- `0b10101111`
- `0b11101101`
- `0b11111111`
- `0b10101101`

Part 2 3 marks

LSR (register)

Logical Shift Right (register) shifts a register value right by a variable number of bits, shifting in zeros, and writes the result to the destination register. The variable number of bits is read from the bottom byte of a register. It can optionally update the condition flags based on the result.

Encoding T1 All versions of the Thumb instruction set.

`LSRS <Rdn>,<Rm>`	Outside IT block.
`LSR<c> <Rdn>,<Rm>`	Inside IT block.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	1	1	Rm			Rdn		

```
d = UInt(Rdn);  n = UInt(Rdn);  m = UInt(Rm);  setflags = !InITBlock();
```

Using the T1 encoding for the `lsr` instruction as shown above, fill out (in the boxes provided) the 16-bit bit pattern (0s and 1s) which represents the following line of assembly code:

```
lsr r1, r6
```

Answer:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Part 3 5 marks

What will the values of the NZCV status bits be after these instructions are executed? What value will be in `r0`? Fill in your answers in the spaces below.

```
mov r0, 0xfffffff0
mov r1, 0x10
adds r0, r0, r1
```

Answer:

- N:
- Z:
- C:
- V:
- `r0`:

Part 4 5 marks

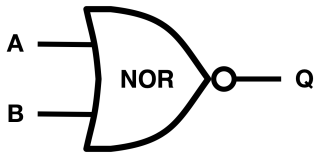
You want to implement a pseudo-instruction called `not` that will invert a register's value in a bit-wise manner. The pseudo instruction's behaviour will be defined as follows:

```
not Rn, Rm        @ Rn := ~Rm
```

What set of instructions could be used to implement `not`? Try to use the fewest instructions possible, and note down if any extra registers, other than `Rn` and `Rm` mentioned above, are needed.

Part 5 2 marks

You're responsible for designing a new CPU that only uses only NOR gates (defined as follows):



A	B	Q
F	F	T
F	T	F
T	F	F
T	T	F

Draw diagrams to define a circuit that is equivalent to a logical NOT gate using only NOR gates.

Answer:

Part 6 3 marks

Draw diagrams to define a circuit that is equivalent to a logical OR gate using only NOR gates (as defined above).

Answer:

Part 7 5 marks

What are flip-flop circuits, and how could they be used as part of a CPU? Be as specific as you can. You may include pictures/diagrams in your answer.

Question 2 Control Structures and Functions (25 marks total)

Part 1 2.5 marks

Suppose a function `f` obeys the ARM Architecture Procedure Call Standards (AAPCS). `f` takes three parameters and returns one 32-bit value.

After `f` completes, in which of the following places will the return value be found?

Answer:

- in `r0`
- in `r1`
- in `r2`
- in `r12`
- in `pc`
- in `fp`
- on the stack
- in the `.data` section

Part 2 2.5 marks

Suppose the following assembly code has stored the values 32, 7, 84, and 128 on the stack.

```
mov     r3, #32
stmdb  sp!, {r3}
mov     r3, #7
stmdb  sp!, {r3}
mov     r3, #84
stmdb  sp!, {r3}
mov     r3, #128
stmdb  sp!, {r3}
```

Which of the following instructions will load the value 84 into `r3`?

Answer:

- `ldr r3, [sp]`
- `ldr r3, [sp, #4]`
- `ldr r3, [sp, #8]`
- `ldr r3, [sp, #12]`
- `ldr r3, [sp, #16]`

Part 3 5 marks

Implement the following "for"-loop in ARMv7 assembly code.

```
int acc = 0;
for (int i = 0; i < 10; i = i+1) {
    acc = acc + i;
}
```

Part 4 10 marks

Write a **recursive** assembly function (starting at the label **pow**) which calculates the power function:

$$pow(x, y) = x^y \tag{1}$$

for positive integers x and y.

Add comments to explain how the parameters are passed into the function and where the result will be stored. For this question, you can assume that the result will fit into a 32-bit unsigned integer, so do not worry about numerical overflow in your function.

Part 5 5 marks

The output of our pow function is likely to overflow, for instance, when calculating $pow(2, 33)$. If you were given the task of optimising the pow function to express numbers that are as large as possible what changes could you make? Explain your answer.

Be as specific as you can. You may include pictures/diagrams in your answer.

Question 3 Asynchronism & data (25 marks total)

Part 1 2 marks

Select the **best definition** of the term "mutual exclusion" below.

Answer:

- A method to save data in memory so that no process can modify it.
- A method to ensure that only one program can access a shared resource at a time.
- A method to determine whether two programs are accessing memory at once.
- A method to exclude programs from accessing the private memory of other programs.
- A method to stop multiple programs from changing CPU registers.
- A method for saving data in between context switches.

Part 2 3 marks

Of the program tasks below, select the **three** that are **most likely** to be accomplished with interrupts in a discoboard program. Your first three selections will be considered to be your answer.

Answer:

- Turning on an LED.
- Branching based on the value of a register.
- Sending data over a network.
- Receiving data over a network.
- Calculating the frequency of a musical pitch.
- Scheduling a regularly timed event.
- Playing MIDI notes.
- Loading data from memory.
- Setting peripheral control registers.
- Responding to unexpected errors.

Part 3 5 marks

You've been asked to write a simple encryption program to obscure **lower-case text data** on a discoboard by shifting each letter **one position backwards** (e.g., "b" should be encoded as "a").

The program should **not affect punctuation, spaces or upper-case letters**. The letter "a" should be wrapped to "z".

Your first task is to write a function to apply this encryption scheme to a **single lower-case letter stored in memory**.

Write a function called `encode_letter`, that takes a memory location as its argument, **loads and encodes** the letter, and finally **stores** the encoded letter back in the same memory location.

You can assume that the character's memory location is passed to your function in `r0`.

You can use the ASCII encoding scheme below to assist you.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	*[NULL]*	32	20	*[SPACE]*	64	40	@	96	60	`
1	1	*[START OF HEADING]*	33	21	!	65	41	A	97	61	a
2	2	*[START OF TEXT]*	34	22	"	66	42	B	98	62	b
3	3	*[END OF TEXT]*	35	23	#	67	43	C	99	63	c
4	4	*[END OF TRANSMISSION]*	36	24	$	68	44	D	100	64	d
5	5	*[ENQUIRY]*	37	25	%	69	45	E	101	65	e
6	6	*[ACKNOWLEDGE]*	38	26	&	70	46	F	102	66	f
7	7	*[BELL]*	39	27	'	71	47	G	103	67	g
8	8	*[BACKSPACE]*	40	28	(	72	48	H	104	68	h
9	9	*[HORIZONTAL TAB]*	41	29	)	73	49	I	105	69	i
10	A	*[LINE FEED]*	42	2A	*	74	4A	J	106	6A	j
11	B	*[VERTICAL TAB]*	43	2B	+	75	4B	K	107	6B	k
12	C	*[FORM FEED]*	44	2C	,	76	4C	L	108	6C	l
13	D	*[CARRIAGE RETURN]*	45	2D	-	77	4D	M	109	6D	m
14	E	*[SHIFT OUT]*	46	2E	.	78	4E	N	110	6E	n
15	F	*[SHIFT IN]*	47	2F	/	79	4F	O	111	6F	o
16	10	*[DATA LINK ESCAPE]*	48	30	0	80	50	P	112	70	p
17	11	*[DEVICE CONTROL 1]*	49	31	1	81	51	Q	113	71	q
18	12	*[DEVICE CONTROL 2]*	50	32	2	82	52	R	114	72	r
19	13	*[DEVICE CONTROL 3]*	51	33	3	83	53	S	115	73	s
20	14	*[DEVICE CONTROL 4]*	52	34	4	84	54	T	116	74	t
21	15	*[NEGATIVE ACKNOWLEDGE]*	53	35	5	85	55	U	117	75	u
22	16	*[SYNCHRONOUS IDLE]*	54	36	6	86	56	V	118	76	v
23	17	*[ENG OF TRANS. BLOCK]*	55	37	7	87	57	W	119	77	w
24	18	*[CANCEL]*	56	38	8	88	58	X	120	78	x
25	19	*[END OF MEDIUM]*	57	39	9	89	59	Y	121	79	y
26	1A	*[SUBSTITUTE]*	58	3A	:	90	5A	Z	122	7A	z
27	1B	*[ESCAPE]*	59	3B	;	91	5B	[	123	7B	{
28	1C	*[FILE SEPARATOR]*	60	3C	<	92	5C	\	124	7C	|
29	1D	*[GROUP SEPARATOR]*	61	3D	=	93	5D	]	125	7D	}
30	1E	*[RECORD SEPARATOR]*	62	3E	>	94	5E	^	126	7E	~
31	1F	*[UNIT SEPARATOR]*	63	3F	?	95	5F	_	127	7F	*[DEL]*

Answer on the next page.

Write your answer for Part 3 here:

Part 4 5 marks

Now use your **`encode_letter`** function to encode a string of characters in memory. You can assume that the string is stored at the label **`string_location`** in the `.data` section and that it is zero-terminated as shown:

Address Offset from `string_location`	0x00	0x01	0x02	0x03	0x04	0x05
Hex:	0x48	0x65	0x6c	0x6c	0x6f	0x00
ASCII:	H	e	l	l	o	(End of String)

Make sure your answer accounts for strings of different lengths.

Part 5 10 marks

Explain using diagrams and text what happens to a discoboard program before, during, and after handling an interrupt.

Make sure to indicate what happens to the program's execution process as it transitions from normal execution to the interrupt handler, and then back again.

Question 4 Networks, OS, & Architecture (25 marks total)

Part 1 5 marks

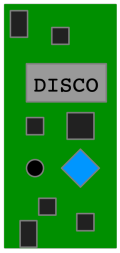
You have been asked to build a traffic light controller for a 4-way intersection using discoboards. You're responsible for designing the **physical connections** between the discoboards, and **software** to run on them.

Each of the four traffic lights contains one discoboard and can display three signal lights (red, yellow, and green).

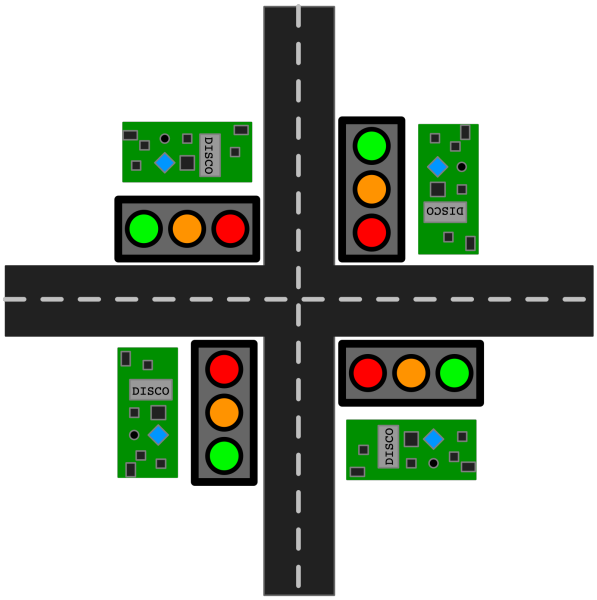
A fifth discoboard (the control board) will be used as a remote controller for the four traffic light discoboards.

The traffic light setup might look something like this:

Controller Discoboard



Traffic Light Discoboards



Your first task is to describe a protocol for the **control discoboard** to control the lights on **just one traffic light board**. Your protocol must be able to turn each signal colour on and off. Discuss the physical connections needed and whether your protocol is serial or parallel.

Be as specific as you can. You may include pictures/diagrams in your answer.

Answer on the next page.

Write your answer for Part 1 here.

Part 2 5 marks

Now you need to extend your network and protocol to allow you to control all **four traffic light discoboards** at the four-way intersection illustrated. Describe how the discoboards will be connected and how your new protocol allows you to control the signals of each traffic light discoboard independently.

Describe the topology of your network, and discuss how the traffic light discoboards are addressed independently.

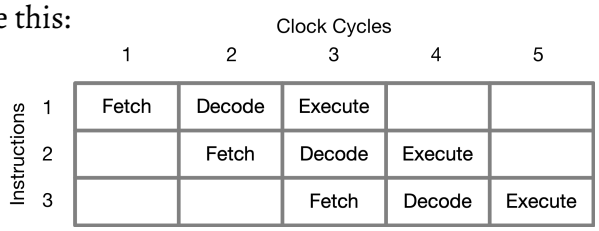
Be as specific as you can. You may include pictures/diagrams in your answer.

Part 3 10 marks

Explain the main roles of the operating system in a computer system. Be as specific as you can. You may include pictures/diagrams in your answer.

Part 4 5 marks

Computer processors, including the discoboard's ARM Cortex-M4, use pipelining to accelerate execution by overlapping the calculation of instructions. For example, a three-stage pipeline might look like this:



What hazards can occur during instruction pipelining, and what workarounds can be applied to mitigate these? Be as specific as you can. You may include pictures/diagrams in your answer.

Note: you don't have to use all of the following pages for your answer—the extra pages are included in case you need them for other questions (as described on the title page).