

COMP2300/6300 Final Exam 2018

Student ID:

u							
---	--	--	--	--	--	--	--

Reading time: 15 minutes

Writing time: 180 minutes

Permitted materials: one A4 page with notes on both sides

Make sure you read each question carefully. Some words have footnotes¹ 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 short, specific answers than long, vague/rambling ones. 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.

¹like this one!

--	--	--	--

For examiner use

Question 1 Instructions & encoding (25 marks total)

Part 1 2.5 marks

True or false: the Arithmetic and Logic Unit (ALU) is responsible for reading and writing data in RAM memory.

Answer:

true

false

Part 2 2.5 marks

```
mov r0, 0

maybe_infinite_loop:
    adds r0, r0, 1
    beq somewhere_else
    b maybe_infinite_loop
```

If program execution enters the `maybe_infinite_loop` loop, will it ever exit (branch out of) the loop, or will it stay in the loop forever? You may assume that `somewhere_else` is a valid branch destination.

Answer:

- program execution will stay in the `maybe_infinite_loop` forever (i.e. it **is** an infinite loop)
- program execution will exit the `maybe_infinite_loop` eventually (i.e. it **is not** an infinite loop)

Part 3 5 marks**REV**

Byte-Reverse Word reverses the byte order in a 32-bit register.

Encoding T1 ARMv6-M, ARMv7-M

REV<C> <Rd>, <Rm>

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

Using the T1 encoding for the **rev** 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:

rev r2, r6

Answer:

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

Part 4 5 marks**initial r0**

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

final r0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Assuming an **initial** bit pattern in **r0** as shown in the diagram, write a sequence of one (or more) ARMv7 assembly instructions which will transform the **initial r0** bit pattern into the **final r0** bit pattern shown in the diagram.

Part 5 10 marks

In your own words, explain how the **fetch-decode-execute cycle** works on your discoboard. Be as specific as you can. You may include pictures/diagrams in your answer.

Question 2 Functions (25 marks total)

Part 1 2.5 marks

$$foo(a, b, c) = \sqrt{(4a + b)^c} \quad (1)$$

Assume there is a (ARM) function **foo** which calculates the (mathematical) function above and obeys the *ARM Architecture Procedure Call Standard* (AAPCS)².

On entry to the function (i.e. immediately after the instruction “**bl foo**”) where will the value of the argument *c* be found?

Answer:

- in **r0**
- in **r1**
- in **r2**
- in **r3**
- in **r4**
- in **r5**
- on the stack
- in the **.data** section

Part 2 2.5 marks

You inherit a broken discoboard where the branch-with-link instruction **bl** does not work—if the program attempts to execute a **bl** instruction the discoboard will explode.

You come up with an idea for a workaround—a **call** assembler macro which could be used to make a function call like so:

```
call my_func @ equivalent to "bl my_func" on a working discoboard
```

²which is the “standard” ARM calling convention, which we have been using all along in the course.

Which **one** of the following definitions of the **call** assembler macro is correct? That is, for which of these macros is **call my_func** equivalent³ to **bl my_func**?

Answer:

```
@ macro (a)
.macro call function_name
    mov lr, pc
    b \function_name
.endm
```

```
@ macro (b)
.macro call function_name
    add lr, pc, 4
    b \function_name
.endm
```

```
@ macro (c)
.macro call function_name
    add lr, pc, 4
    bx lr
.endm
```

```
@ macro (d)
.macro call function_name
    mov r0, lr
    bx lr
.endm
```

³In this question you can ignore any ARM/Thumb mode interworking issues, i.e. the Thumb bit. And if you don't know what I'm talking about, don't worry. The macro you're looking for is the one which best represents (conceptually) what the **bl** instruction actually does.

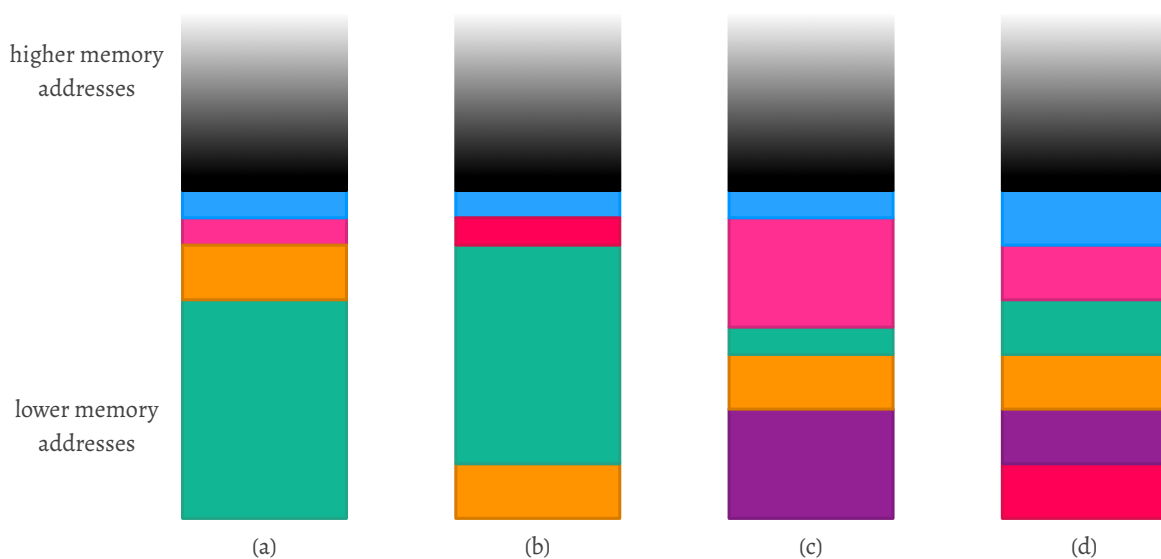
Part 3 5 marks

The following code snippet is from a JavaScript-like language⁴ which compiles (without optimisations) directly to ARMv7 machine code to run on your discoboard.

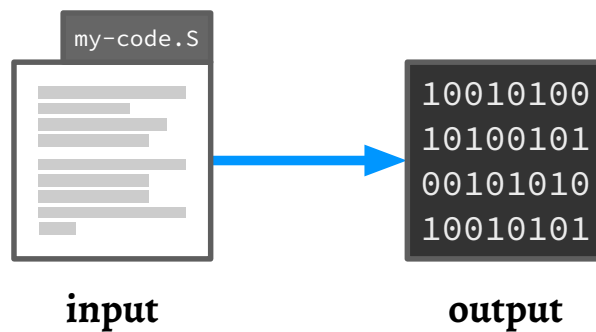
```
function bar() {
  let x = 50;
  let y = 3.14;
  let blockchain = [1, 4, 6, 3, 6, 3, 6, 5, 3, 7, 7, 7, 8, 1, 5, -4];
  let name = "Satoshi";
  // *breakpoint*
  // more code follows...
  //
  //
}
```

When a breakpoint is set on the **breakpoint** line (shown in the comment above) and the function **bar** is called, which picture **best** represents the state of **bar**'s stack frame at the **breakpoint** point in time, assuming that a descending stack is in use?

Answer:



⁴This is **not** JavaScript, it's a simple, direct-compiled language with JavaScript-like syntax.

Part 4 5 marks

A source file containing a valid sequence of ARM assembly instructions (e.g. `my-code.S`) is “assembled” into binary machine code by the assembler program in the usual fashion (as shown in the diagram, although it’s not to scale—there will be *many* more 0s and 1s than that).

If you **cannot** see the original source file `my-code.S` but you **can** see the binary machine code output, is it possible to determine whether the original source file contained a **function call**? Explain your answer—be as specific as you can. You may include pictures/diagrams in your answer.

Part 5 5 marks

In the same situation as Part 4 , is it possible to determine whether the original source file `my-code.S` contained an **assembler macro**? Explain your answer—be as specific as you can. You may include pictures/diagrams in your answer.

Part 6 5 marks

Consider the *triangular number*⁵ function

$$T_n = \sum_{k=1}^n k = 1 + 2 + \dots + n \quad (2)$$

which takes a single positive integer argument n and returns the sum of all the integers from 1 up to n . As an example,

$$T_3 = 1 + 2 + 3 = 6 \quad (3)$$

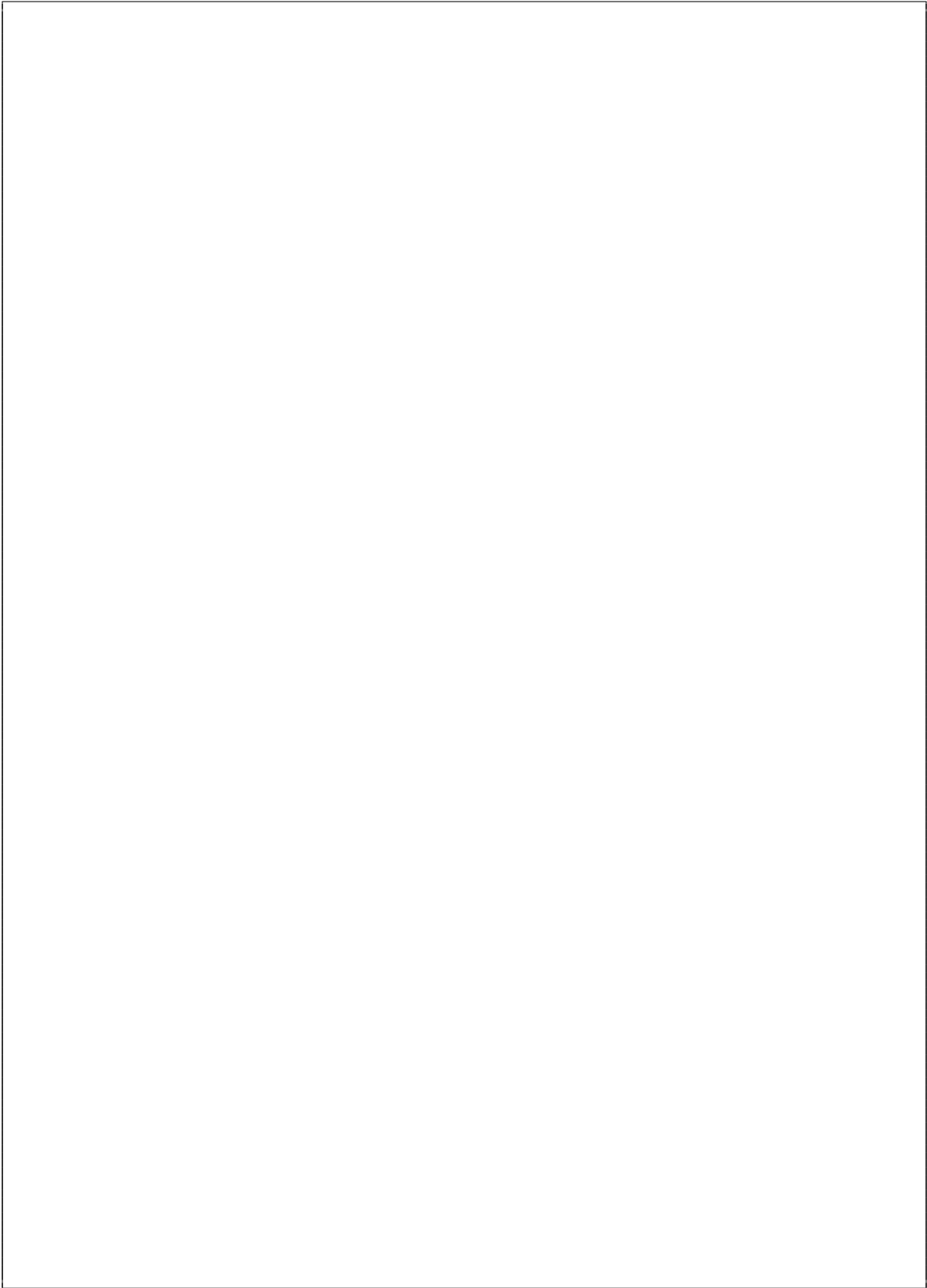
Here is a **recursive** function **triangular** in ARM assembly language which takes a single integer argument n (in **r0**) and returns T_n (also in **r0**):

```
triangular:
    push {lr}
    push {r0}
    cmp r0, 1
    beq end_triangular
    sub r0, r0, 1
    bl triangular
    ldr r1, [sp]
    add r0, r0, r1
end_triangular:
    add sp, sp, 4
    pop {lr}
    bx lr
```

Are there any disadvantages to this recursive approach *compared to a non-recursive version of the same function*? Explain your answer—be as specific as you can. You may include pictures/diagrams in your answer.

Answer on the next page.

⁵Don't worry about why it's called a triangular number—the point of this question is to think about how this function might be implemented in ARM assembly language.

A large, empty rectangular box with a thin black border, occupying the central portion of the page. It is intended for the student to write their answers to the exam questions.

Question 3 Memory & data (25 marks total)

Part 1 2.5 marks

After the following code has been executed, what is the value (in hex) in the 32-bit little-endian memory **word** at the address 0x20000004?

```
mov r4, 0x15
mov r5, 0xFF
adds r6, r4, r5
ldr r0, =0x20000000
str r6, [r0, 4]!
```

Answer:

- 0x114
- 0xF5
- 0x1F5
- 0xFF

Part 2 2.5 marks

After all the code in Part 1 has been executed, what is the value (in hex) in the register **r0**?

Part 3 2.5 marks

You spill coffee on your discoboard and the result is that 16 lines of the discoboard's address bus are damaged, so that the **16 least-significant bits** of the address register (i.e. the register holding the memory address) in *any* load or store instruction are always read as 0 (regardless of the value in the register).

True or false: the memory addresses 0x20000000 and 0x20000004 can both still be used independently to load & store data in your programs.

Answer:

true

false

Part 4 2.5 marks

Assuming the same “damaged address bus” discoboard from Part 3 , what has happened to the size of the addressable memory space, i.e. the number of valid memory addresses?

Answer:

- the damaged discoboard has **the same** address space as a correctly-functioning⁶ discoboard
- the damaged discoboard has **half** the address space of a correctly-functioning discoboard
- the damaged discoboard’s address space has been **reduced in size by 16 valid memory addresses** compared to a correctly-functioning discoboard
- the damaged discoboard’s address space has been **reduced in size by a factor of 2^{16}** compared to a correctly-functioning discoboard

⁶a discoboard without a damaged address bus

Part 5 10 marks

The final marks & grades for COMP2300 will be calculated and recorded using a **grading program** running on Ben's discoboard⁷. For every student, the grading program must store the student's

- UID (e.g. u1234567)
- final mark (out of 100)

Describe a data structure which could be used to store this data for *one* student. Be specific about the size, layout and interpretation (e.g. width, signed/unsigned, offsets, etc.) of **all the fields** in the data structure. Your answer must include an example of the data structure in use (i.e. storing one student's data). You may include pictures/diagrams in your answer.

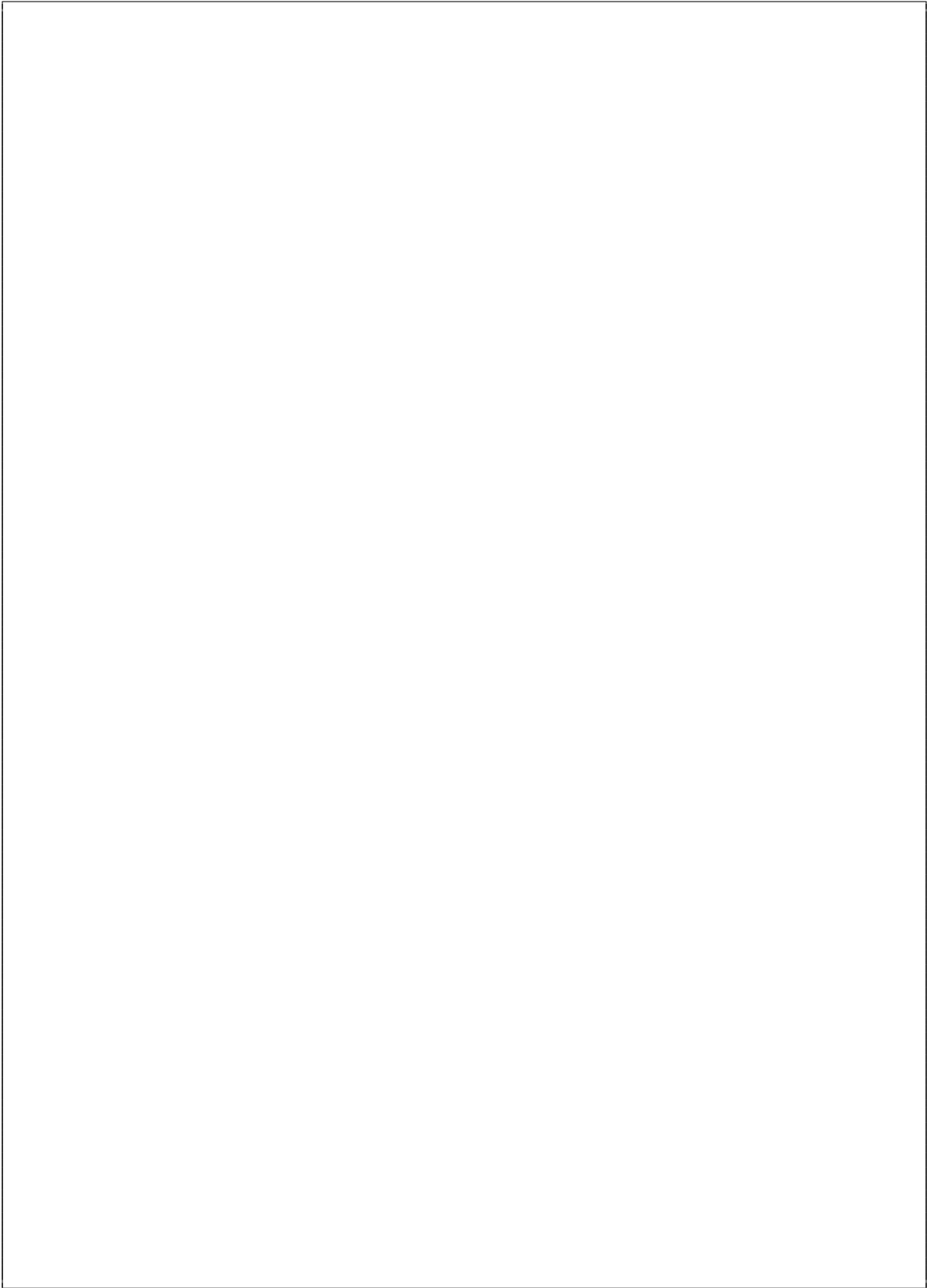
You are not optimising for any particular criteria (e.g. minimum size, maximum read/write performance). You can design the data structure however you like, as long as it stores the UID & mark information as described above. The following ASCII table may be helpful, although you do not have to use it.

Answer on the next page.

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]

⁷This isn't *actually* true, but let's just pretend it is for the purposes of this question.

A large, empty rectangular box with a thin black border, occupying the central portion of the page. It is intended for students to write their answers to the exam questions.

Part 6 5 marks

The grading program (using your “student mark” data structure from Part 5) gives correct results, but runs too slowly. If you were given the task of optimising the program for maximum performance⁸ what changes would you make? You are able to make changes to the data structure and/or the grading program itself (i.e. Ben’s code). Be as specific as you can. You may include pictures/diagrams in your answer.

⁸i.e. so that it finishes in as few cycles as possible

Question 4 Networks & Operating Systems (25 marks total)

Part 1 2.5 marks

When an interrupt is triggered, where is the value in the program counter (**pc**) stored so that it can be restored after the interrupt handler(s) returns?

Answer:

- in the stack pointer register (**sp**)
- on the stack (in memory)
- in the link register (**lr**)

Part 2 2.5 marks

The network protocols **P1** and **P2** are **exactly the same** *except* for the way they use their data line:

- in **P1**, a *rising-edge* on the data line is interpreted by the physical layer as a 1 and a *falling-edge* as a 0
- in **P2**, a *falling-edge* on the data line is interpreted by the physical layer as a 1 and a *rising-edge* as a 0

Assuming the exact same hardware and environment, which **one** of the following statements is true:

Answer:

- **P1** will transfer data faster⁹ than **P2**
- **P2** will transfer data faster than **P1**
- **P1** will transfer data at the exact same rate as **P2**

⁹faster == more bits-per-second

Part 3 2.5 marks

You are writing an assembly program which will run as one task managed by a multitasking OS on your discoboard, and it needs to store some “private”¹⁰ data in memory (RAM).

True or false: the load/store exclusive instructions **ldrex** and **strex** can prevent other tasks from reading and writing your private data in memory.

Answer:

true

false

Part 4 2.5 marks

True or false: an operating system must be at least 100Mb in size when compiled to machine code.

Answer:

true

false

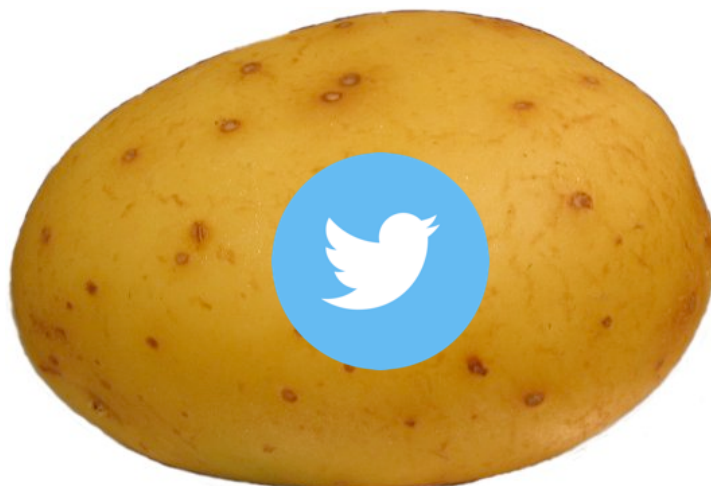
¹⁰This is just regular data—0s and 1s—but it’s data that you don’t want any of the other tasks running on the discoboard to be able to read or modify.

Part 5 5 marks

What are the advantages of a **serial** (single-wire) compared to a **parallel** (multiple-wire) network protocol? In which situation(s) would you prefer a serial protocol over a parallel one? Be as specific as you can. You may include pictures/diagrams in your answer.

Part 6 5 marks

Ok, now the opposite of Part 5 : what are the advantages of a **parallel** (multiple-wire) compared to a **serial** (single-wire) network protocol? In which situation(s) would you prefer a parallel protocol over a serial one? Be as specific as you can. You may include pictures/diagrams in your answer.

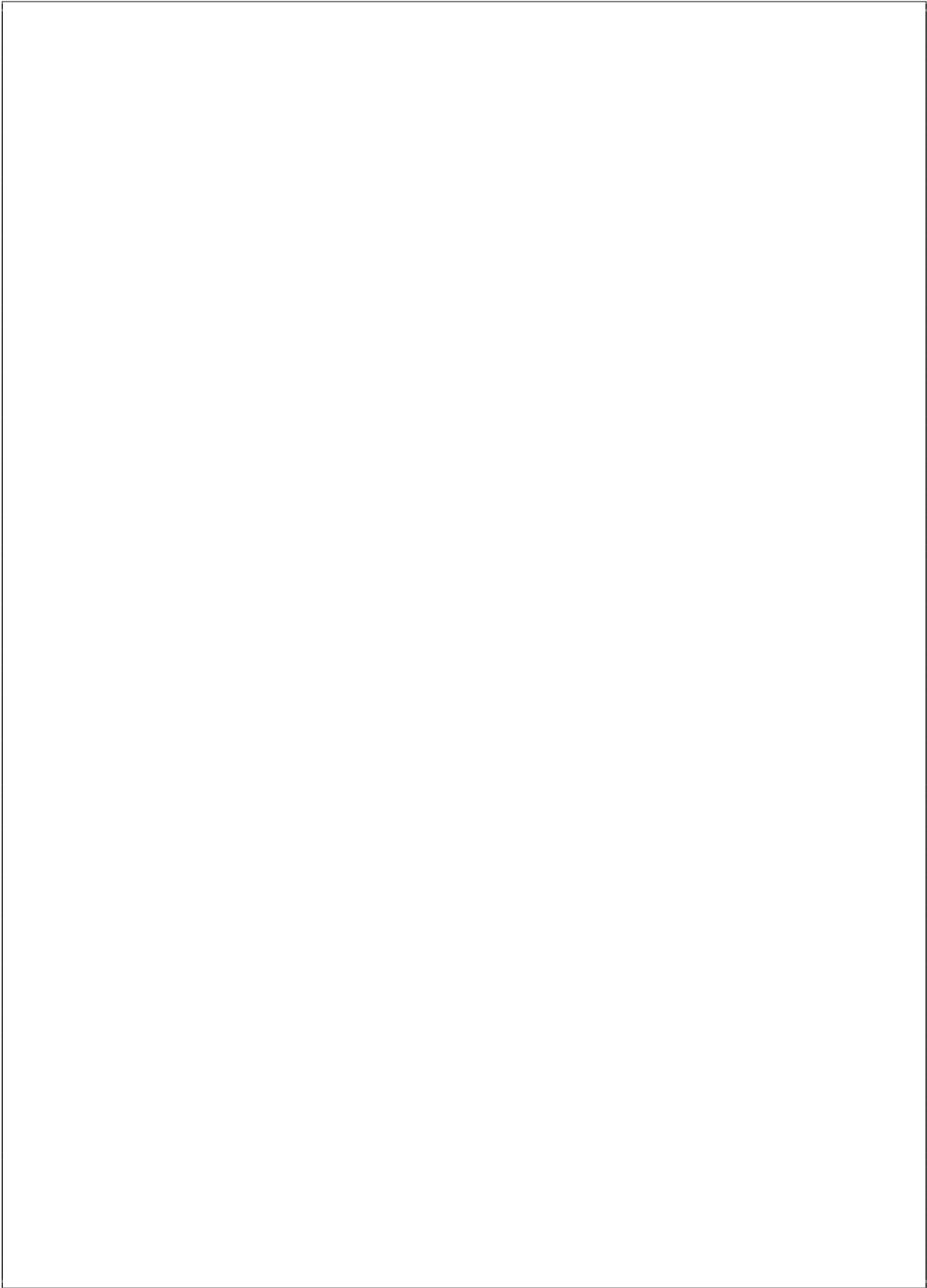
Part 7 5 marks

A company sells a product called **TweetyPotato™**: a potato with a discoboard inside that is connected to the internet (each TweetyPotato has its own twitter account). The company's engineers have written a new version of the TweetyPotato software and uploaded it to their deployment server ready to be uploaded to all the TweetyPotatoes in the world.

You are a **bad person** and have hacked into the company's deployment server. You cannot change all of the code for the new version of the TweetyPotato software, but you can modify anything which will be mapped in the memory region from `0x0` to `0x07FFFFFF` of the discoboard's address space (up to—but not including—the executable code region which starts at `0x80000000`).

Is it possible to modify the software to compromise¹¹ the TweetyPotato devices? If so, how? If not, why not? Explain your answer—be as specific as you can. You may include pictures/diagrams in your answer.

¹¹In this question, “compromise” means to make the TweetyPotato device execute *any* code **you** (the hacker) want, not just the code written by the company's software engineers



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).

