Queen Mary
University of London

Main Examination Period 2019-2020

ECS404U **Computer Systems and Networks** **Duration: 2 hours**

SOLUTIONS AND MARKING SCHEME

YOU ARE NOT PERMITTED TO READ THE CONTENTS OF THIS QUESTION PAPER UNTIL INSTRUCTED TO DO SO BY AN INVIGILATOR.

Instructions: This paper contains FOUR questions. **Answer ALL questions**.
Cross out any answers that you do not wish to be marked.

Calculators are permitted in this examination. Please state on your answer book the name and type of machine used.

Complete all rough workings in the answer book and cross through any work that is not to be assessed.

Possession of unauthorised material at any time when under examination conditions is an assessment offence and can lead to expulsion from QMUL. Check now to ensure you do not have any notes, mobile phones, smartwatches or unauthorised electronic devices on your person. If you do, raise your hand and give them to an invigilator immediately.

It is also an offence to have any writing of any kind on your person, including on your body. If you are found to have hidden unauthorised material elsewhere, including toilets and cloakrooms it will be treated as being found in your possession. Unauthorised material found on your mobile phone or other electronic device will be considered the same as being in possession of paper notes. A mobile phone that causes a disruption in the exam is also an assessment offence.

Exam papers must not be removed from the exam room.

Question:	1	2	3	4	Total
Points:	25	25	25	25	100
Score:					

Leave this table blank.

Examiners: Dr A Khouzani and Prof E Robinson

Question 1

Computer Architecture

(a) In a circuit diagram the symbol in figure 1 represents a transistor. Name the type of transistor and the different connections to it, and explain how it functions as a switch. You are not asked for a physical explanation, just an explanation in terms of the roles of the different connections. **[5 marks — basic]**

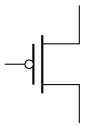

Figure 1: a transistor

Solution: This is a pmos transistor. The connections to it are gate, source and drain. The gate controls the transistor when it functions as a switch. If the potential at the gate is low, then the source and drain are connected (the circuit across the transistor is closed). If the potential at the gate is high, then the source and drain are not connected (the circuit across the transistor is open).

Marking scheme: 1 mark for pmos
1 mark for names of connections
2 marks for getting the facts about functionality right
1 mark for expressing it clearly.

(b) Figure 2 below depicts a logic gate.

(i) Detailing which transistors give open or closed circuits, explain the output at C when the input at A=0 and the input at B=1.

(ii) Identify the connective being computed by this logic gate and explain your answer.

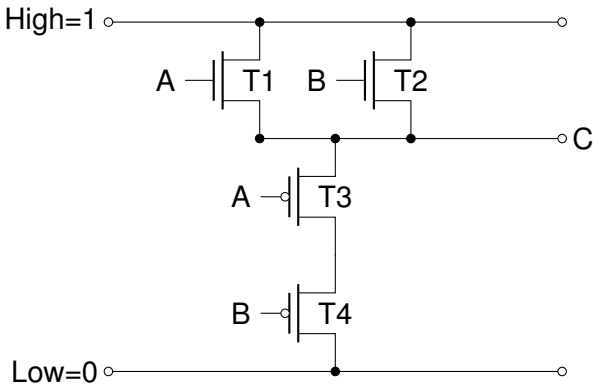

Figure 2: a logic gate

[8 marks — medium]

Question continues on next page

Solution:

(i) Nmos are closed when the potential at the gate is high (1) and pmos are closed when the potential at the gate is low. Therefore when A=0 and B=1, T2 and T3 are closed, while T1 and T4 are open. It follows that C is connected to High through T2, but is not connected to low because T4 is open. The output at C is therefore High (1).

(ii) This is OR. If either A or B is 1, then the corresponding nmos T1 or T2 is closed connecting C to High. And the corresponding T3 or T4 is open, disconnecting C from Low. If both A and B are 0, then both T1 and T2 are open, disconnecting C from HIgh, while both T3 an dT4 are closed, connecting C to low. This output corresponds to the truth table for OR.

Marking scheme:

(i) 4 total: 2 for a good account of whether transistors are open or closed. 2 for correct use of that to deduce output.

(ii) 4 total: 2 for making clear that there is a correspondence with the values computed by OR (eg by truth table, ie for making it clear that the student understands what it is for a logic gate to compute a connective, and 2 for a reasonable explanation of why this is OR in particular.

(c) Figure 3 below depicts a DRAM cell.

(i) What is the function of a *capacitor*?

(ii) Explain how this cell operates to store a single bit, making clear the roles of the capacitor, transistor, word line and bit line.

(iii) Explain why the information in this cell is lost when the power to it is cut off.

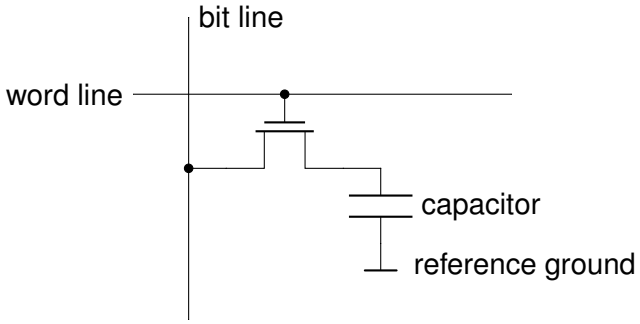

Figure 3: DRAM

[6 marks — advanced]

Question continues on next page

Solution:

(i) A capacitor stores charge.

(ii) The word line controls access to the cell. When it is high, the transistor is closed giving access to the cell, and when it is low, the access is cut off, keeping the level of charge in the capacitor constant. The bit line provides the access to the cell. It is used to store and read information. When it is driven, it will store information in the cell. When it is not driven its potential will be changed by the charge in the cell, allowing a read.

(iii) When the power is cut off, the small charge that is in the capacitor leaks out even though the transistor is open. This does not take long.

Marking scheme:

(i) 1 mark.

(ii) 3 marks, 2 for getting the basics and the third for a good account of the overall functionality.

(iii) 2 marks

(d) The chip in Figure 4 has the Intel Kaby Lake architecture. Some of it marked as being L3S, which stands for Level 3 Cache.

(i) In the context of Computer Architecture, what is a cache?

(ii) Why do computer and chip designers use cache?

(iii) What is the function of this particular cache?

(iv) Why is it described as Level 3?

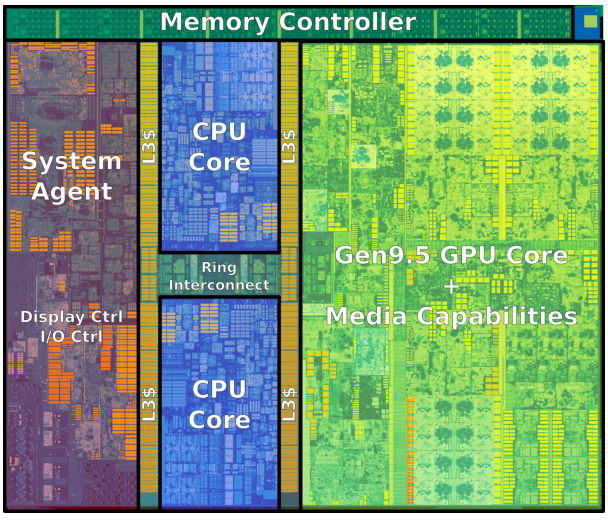

Figure 4: Intel Kaby Lake chip (Question 1-(d)).

Question continues on next page

[6 marks — medium]

Solution:

(i) A cache is a piece of memory that is used to store information from a lower level of the memory hierarchy that might be required by a higher level.

(ii) They use cache because it improves the latency of memory accesses, giving the impression that the lower level of memory is faster than it in fact is, and hence improving performance.

(iii) This particular cache stores information from main memory that might be required by the cpu cores.

(iv) It is Level 3 because there are two further levels of cache internal to the cores, and therefore higher up the memory hierarchy.

Marking scheme:

(i) 2 marks for a reasonable explanation.

(ii) 2 marks for a reasonable explanation

(iii) 1 mark

(iv) 1 mark

Question 2

Digital Representation

Simply giving an answer will never achieve full marks. You must always explain your methods.

(a) These bit sequences represent numbers in 8-bit 2's complement. Explain which numbers they are and how you know whether the numbers are positive or negative.

(i) 11110011

(ii) 00000110

[5 marks — basic]

Solution:

(i) -13, eg $-128 + 64 + 32 + 16 + 2 + 1 = -13$

(ii) 6, eg $4 + 2 = 6$.

The leading bit determines whether the number of positive or negative. If 0, it is 0 or positive. If 1, then the number is negative.

Marking scheme:

(i) 2 marks: 1 for answer 1 for method

(ii) 2 marks: 1 for answer 1 for method

1 mark for answering about positive or negative.

(b) (i) Show that you understand how to do long multiplication in binary by multiplying `11110011` and `00000110`.

(ii) What are the rightmost 8 bits of your answer?

(iii) What do they represent as a number in 8-bit two's complement.

[6 marks — medium]

Solution:

(i)

11110011	
00000110	$*$
111100110	
1111001100	
11110001000	carries
10110110010	

(ii) 10110010

Question continues on next page

(iii) $-128 + 32 + 16 + 2 = -78$

Marking scheme:

(i) 3 marks: 2 for method and 1 for accuracy

(ii) 1 mark

(iii) 2 marks: 1 for answer and 1 for method

(c) This part is about floating point representation. The bit sequence below represents a number in 64-bit IEEE floating point format:

```
11000000 00100101 00000000 00000000 00000000 00000000 00000000 00000000
```

(i) The first bit represents the sign of the number. Is this number positive or negative?

(ii) The next 11 bits represent the exponent. In this format, we get the exponent by subtracting 1023 from the number represented by this bit sequence. What is the actual exponent?

(iii) The remaining bits give the part after the point in the significand. What is the significand in binary format and what is it as a decimal number?

(iv) Explain how these components are put together to give the final number represented, and give the number.

[6 marks — advanced]

Solution:

(i) The bit is 1 so the number is negative.

(ii) The seequence is 1000000 0010 representing $1026 = 1024 + 2$, so the exponent is $1026 - 1023 = 3$.

(iii) In binary the significand is 1.0101 represent $1 + 0.25 + 0.0625 = 1.3125$ in decimal.

(iv) The number is $-1.3125 * 2^3 = -1.3125 * 8 = -10.5$.

Marking scheme:

(i) 1 mark

(ii) 2 marks: 1 for method and 1 for accuracy

(iii) 2 marks: 1 for binary and one for decimal.

(iv) 1 marks

Question continues on next page

(d) The MAC address of my device is: 2c:8a:72:bd:ef:38

(i) Explain the format being used and how it represents a bit sequence.

(ii) What is the bit sequence?

[4 marks — medium]

Solution:

(i) It is a byte sequence represented in hex, with each hex digit representing 4 bits.

(ii) 00101100 10001010 01110010 10111101 11101111 00111000

Marking scheme:

(i) 2 marks: 1 for hex, 1 for explanation of representation

(ii) 2 marks: 1 for accuracy, 1 for general method

(e) In ASCII, the decimal code for the character 'A' is 65, and for lower case 'a' it is 97.

(i) Put these into binary and explain why it makes sense to have the upper and lower case characters differ by 32.

(ii) Explain the limitations with ASCII that prompted the development and adoption of Unicode.

(iii) How many bytes are used to represent an ASCII character and how many bytes to represent one in UTF-8?

[4 marks — advanced]

Solution:

(i) 'A' is 01000001 and 'a' is 01100001, so both are at the beginning of significant sequences in binary. Upper and lower case differ by exactly one bit.

(ii) ASCII only represents 128 characters, which is not enough for international alphabets. Unicode was developed to allow characters from all languages to be represented.

(iii) One byte for ASCII, between one and four (variable) for UTF-8.

Marking scheme:

(i) 2 marks: 1 for correct binary and the other for getting the single bit point.

(ii) 1 marks for understanding it is to get round the limitation on character numbers.

(iii) 1 mark for getting that UTF-8 is variable length.

Question 3

Instruction Set Architecture, Assembly Language

(a) Can we use compilers that are designed for different architectures interchangeably? For instance, can we use an Intel/AMD C++ compiler to compile our code to be used on a CPU with ARM or MIPS architecture? Your answer should be supported by brief reasoning.

[4 marks — basic]

Solution: No: the compiler converts (translate) a high level code to machine language, that is, the language that is actually implemented in the hardware architecture, and is presented to the compiler (by the chip designer) as a set of instructions, called Instruction Set Architecture (ISA). Different architectures have different (and in general incompatible) ISAs.

Marking scheme: 1 point for correct identification, 3 points for correct reasoning: 2 points if either different ISA or different hardware implementation is mentioned. Capped at 4 points in total.

(b) Explain why "branching" in the programmes (that is, when there is a conditional jump) can cause a problem for "pipelining" and reduce its efficiency.

[5 marks — medium]

Solution: Pipelining works by breaking down the instructions into stages, and feeding the instructions continuously to the CPU, with the idea that while a stage is cleared, the next instruction can start being processed at that stage, instead of waiting while an instruction clears out entirely before feeding the next instruction. However, if the current instruction is a conditional branch, then in order to know what instruction to feed to the pipeline, we have to wait to see the outcome of the conditional, or just guess what is the likely outcome and feed that one in. However, if our guess is wrong, then the whole pipeline has to flushed and the correct instruction fed in. That clearly undermines the overall efficiency.

Marking scheme: Full 5/5 mark if the fact that the wrong instruction might be fed and hence the need for pipeline flush is mentioned. 3/5 if it is argued that pipelining is impossible when dealing with branching. If only jargon words like pipelining hazard is used without clarification, or (generously) if other types of hazards is explained instead, capped at 1/5 point.

(c) A 32-bit MIPS instruction is `0x21490023` (that is, `21490023` in HEX representation).

Question continues on next page

(i) What is the actual bit-sequence of this instruction?

> **Solution:** `0010 0001 0100 1001 0000 0000 0010 0011`
>
> *Marking scheme:* 2 point: -1 point for each mistake, capped from below at 0.

(ii) The **`opcode`** of this instruction corresponds to **`ADDI`** (add-immediate). Using the following reminder about the 3 different formats of the 32-bit MIPS instructions (Figure 5) and its register names (Table 1), along with your answer to the previous part, fully identify the instruction. Namely, identify the operands of the instruction, express it in MIPS Assembly mnemonics, and briefly describe in words what operation it performs.

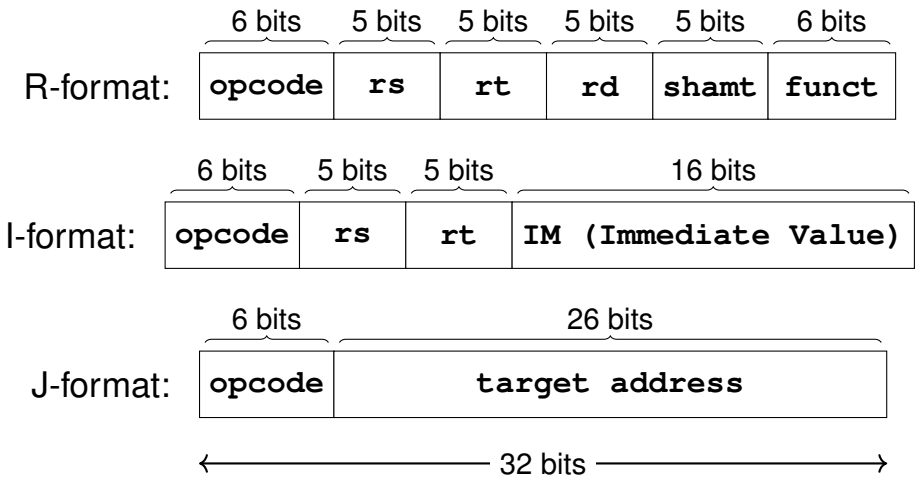

Figure 5: The 3 different formats for 32-bit MIPS instructions (Question 3(c)-ii).

Reg. No.	Reg. Name	purpose (usage)
\$0	**`$zero`**	always holds 32 bits of zeros (no overwrite)
\$1	\$at	Reserved to be used by the assembler
\$2 · · · \$3	**`$v0`**, **`$v1`**	Return values from subroutines
\$4 · · · \$7	**`$a0`** · · · **`$a3`**	Arguments to subroutines
\$8 · · · \$15	**`$t0`** · · · **`$t7`**	General purpose (temporary registers)
\$16 · · · \$23	**`$s0`** · · · **`$s7`**	General purpose (save registers)
⋮		

Table 1: The naming convention for some of the 32-bit MIPS registers (Question 3(c)-ii).

> **Solution:** The instruction is of Immediate type. So based on the Immediate format, the 5 bits after the first 6 bits represent **`rs`**, which in this case is

Question continues on next page

01010, i.e., register `$10`, which from the table, represents `$t2`. The next 5 bits represent **rt**, which for us, it is **01001**, i.e., register `$9`, which is `$t1`. The next 16 bits represent the immediate value: **0000000000100011**, which is number 35 in decimal. So the overall instruction is:
`ADDI $t1 $t2 0x0023`.
or equivalently
`ADDI $t1 $t2 35`.
It adds number 35 to the content of register `$t2` and puts the results in register `$t1`. It does so whether or not an arithmetic overflow takes place and doesn't raise any flags.

Marking scheme: 6 points: -1 point for each mistake (including if the order of t1 and t2 is wrong) capped from below at 0.

[8 marks — medium]

Question continues on next page

(d) A programme in 32-bit MIPS Assembler is shown in the following:

```
1  .text
2  MAIN:
3      li $v0, 5    # set v0 to "5" to select the "read integer" syscall
4      syscall      # invoking the syscall to read the integer (to $v0)
5      addu $s0, $v0, $zero      # equivalent to 'move $s0, $v0'
6  START:
7      li $s1, 0
8  LOOP:
9      beq $s0, $zero, DONE
10     andi $t0, $s0, 1
11     addu $s1, $s1, $t0
12     srl $s0, $s0, 1
13     j LOOP
14 DONE:
15     addu $a0, $s1, $zero      # equivalent to 'move $a0, $s1'
16     li $v0, 1        # set v0 to "1" to select the "print interger" syscall
17     syscall          # invoking the syscall to print the integer (from $a0)
18
19     li $v0, 10    # set v0 to "10" to select the "exit" syscall
20     syscall      # invoking the syscall to exit.
```

As a reminder, the MIPS instruction `srl` is `shift right logical`. In particular, it has the format `srl rd, rt, sa`, which means the contents of the register **`rt`** are shifted right by `sa` many bits ('shift amount'), inserting zeros into the emptied bits in the left, and the resulting bit-sequence is placed in register **`rd`**. You should be quite familiar with the rest of the instructions.

(i) Suppose when prompted at the console, **we input number 11** (in decimal). Provide the trace of the values in registers **`$s0`** and **`$s1`** (as a pair) when this code is executed until it finishes. That is, write down the new values in both **`$s0`** and **`$s1`** each time either one of them changes as you step through the programme until the programme finishes execution. You can choose to present the values either in decimal, hex, or binary, as long as you clearly specify which.

Solution:

`$s0`	?	11	11	11	5	5	2	1	1	0
`$s1`	?	?	0	1	1	2	2	2	3	3

Marking scheme: 5 points for all correct entries. -1 point for each wrong entry capped at 0 from below. If only one round of the programme is provided, then capped at 1/5. The states representing unknown (?) are not necessary for the correct answer.

(ii) Express in a simple sentence what does this programme effectively do? (Your answer should be a sentence starting like: "this programme takes an integer value

from the user and computes . . . ")

Solution: Counts the number of 1's in the binary representation of the number.

Marking scheme: 3 points. This can also be expressed (though not so straigtforwardly) in decimal way as well: counts the number of whole power of 2 that fits in the number.

[8 marks — medium]

Question 4

Computer Networks

(a) Suppose Alice, a QMUL student, lives in Croydon, south of London. She uses her laptop to connect to internet through wifi, both at home and on our campus. For each of the following entries, determine if it changes based on whether she is connected to internet at home or on campus (your answers should be supported by a brief explanation):

(i) The MAC address of her laptop;

(ii) The public IP address of her laptop;

(iii) The MAC address of the wifi access point that connects her to internet;

(iv) The TCP port number of her web browser.

[9 marks — basic]

Solution:

(i) doesn't change, because it is the same device.

(ii) changes, because this is what is assigned by your access provider, which is different home vs campus;

(iii) changes, as the devices are different;

(iv) changes, as is assigned randomly each time by the OS.

Marking scheme: 1 point per each correct identification and 1 point per each correct reasoning, capped at 9 in total.

(b) 'IP header' refers to the information present at the beginning of an IP packet. Name at least 4 entries in the 'IP header', and for each of them, briefly describe what it is used for (i.e., what role does it play).

Hint 1: The IP (Internet Protocol) is right above the Link layer and right below the Transport layer in the TCP/IP protocol stack.

Hint 2: If it matters at all, you can consider either IPv4 or IPv6, but there are common

Question continues on next page

entries present in both of them, based on the services that the IP layer needs to provide.

[8 marks — medium]

Solution: the main ones discussed during the class are:

- destination IP address: so that the routers know where to route the packet to;
- source IP address: to tell the final recipient where the original source of a packet is;
- TTL – time to live, which is 'hop limit' in IPv6, which prevents packets spending too much time hopping around, e.g. getting trapped in a circle of hops!
- Protocol: to tell what is the protocol associated with its data (its payload), so that it knows which Transport layer programme to had the payload over to, e.g. is TCP, UDP, ICMP, etc.

Other entries include:

- Total Length (indicating the total length of the packet, header and data),
- Version (to tell whether this is an IPv4 or IPv6 IP packet)
- ...

Marking scheme: 2 points per each entry: 1 for name and 1 point for its role, capped at 8 points in total.

(c) This part is about DNS name servers. Recall: DNS stands for Domain Name System.

(i) What service does a DNS name server provide in connection with internet communication?

Solution: DNS server's main role is to simply map (translate) human readable domain names to actual IP addresses.

Marking scheme: 4 points for a clear description. 2 points if the other way around is mentioned.

(ii) Describe a way in which the use of DNS makes the internet vulnerable to attack. How can an attacker use the existence of DNS servers to compromise the integrity of internet traffic?

Solution: Relying on an external entity to provide this translation introduces reliability and security issues. In terms of reliability, e.g., if the DNS server's availability is attacked, say through a DDoS attack, then the clients cannot access the internet, even though the rest of the network is perfectly fine (single

Question continues on next page

point of failure). Also if the integrity of the DNS server or its responses are compromised, then the clients will be directed to malicious servers or man in the middles without noticing it.

Marking scheme: 4 points for any correct entry. Partial points generously possible here.

[8 marks — advanced]

End of questions