Queen Mary
University of London

Main Examination Period 2018-2019

ECS404U Computer Systems and Networks Duration: 2 hours 30 minutes

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.

Examiners: Dr A Khouzani and Prof E Robinson

Question 1

This question is about Computer Architecture

(a) Explain how a modern mobile phone is similar in architecture to a modern laptop. A good answer should cover the components of each, the way they are put together, and acknowledge where there are differences. **[5 marks — basic]**

> **Solution:** This should be a fairly standard piece of bookwork.
>
> Mobile phones and laptops have a fairly similar architecture with a broadly similar range of components put together in a similar way. They both include the same basic computational components (cpu and main memory) along with long-term storage (flash memory possibly supplemented by flash memory card on the phone, and either flash memory in the form of a SSD or a traditional magnetic hard disk on a laptop). These are all linked by a comms network provided by the motherboard. The mobile phone cpu often contains more, but less powerful cores, and smaller memories. In addition both phones and laptops have various IO devices. Both have screens, but the input devices differ (keyboard and trackpad versus touchscreen), and the phone has more in the way of network connections.
>
> 3 marks for identifying standard range of components and something about the functionality (2 if incomplete, 1 if largely correct but very deficient). 2 marks for giving coherent explanation.

(b) A laptop computer with a solid state disk (SSD) will typically run faster than an otherwise identical computer with a conventional magnetic hard disk (HDD). But the peak data transfer rates for the two forms of disk are broadly similar. Define the concepts of *bandwidth* and *latency* and use these to explain the observed patterns in performance. In addition to the definitions, a good answer will explain (briefly) the physical reasons why the SSD has better performance than the HDD.

[5 marks — medium]

> **Solution:** *Bandwidth* is the maximum rate of data transfer over a link, measured in bits per second. *Latency* is the time between request and (start of) response, measured in seconds. The fact that the peak transfer rates for an HDD and an SDD are similar means they have similar bandwidth. But they have very different latencies. The latency of an HDD is larger and less predictable because mechanical parts have to move into position (typically some hundredths of a second). The latency of an SDD is much lower, typically of the order of millionths of a second and is more or less uniform across the device. This makes the SDD faster in practice.
>
> 2 marks for correct definitions, 3 for a reasonable explanation.

(c) List two places where you might expect to find a hardware cache in a computer

Question continues on next page

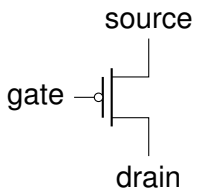
architecture. Taking one of these as an example, explain how that cache might be expected to improve the computer's performance. **[5 marks — medium]**

Solution: You'd expect to find hardware caches between any two core components of the memory hierarchy. This includes a cache on the cpu between registers and the comms component of the cpu, between the cpu and main memory, and for an HDD between main memory and the hard disk (often on the hard disk itself). Taking the cpu cache as an example, this stores data from main memory that might be needed soon. For example it stores the instruction sequence, enabling enhanced pipelining of instructions and enabling code for tight loops to be kept on the cpu without accessing the slower main memory or disk.

2 marks for giving reasonable locations, 3 for reasonable explanation making use of one of the examples.

(d) Draw the circuit diagram symbol for a *pmos transistor* and explain how it operates as a switch (labelling the parts you mention). **[5 marks — basic]**

Solution:



The source and drain are symmetric and act as connectors, the gate is the control of the switch. For pmos, when the potential at the gate is high there is no connection between source and drain. When the potential at the gate is low there is a connection.

3 marks for symbol and labelling, 2 for explanation.

(e) Figure 1 below depicts an *and gate*.

(i) Explain the sense in which this circuitry can be said to compute the logical function *and* ($\wedge$).

(ii) Explain in detail how the circuitry works to compute the correct answer for the operation True $\wedge$ False.

[5 marks — advanced]

Solution:

(i) The inputs are A and B and the output is at C. High potentials are used to represent True, and low potentials represent False. The circuitry computes *and* because the potential output at C corresponds to "A and B".

Question continues on next page

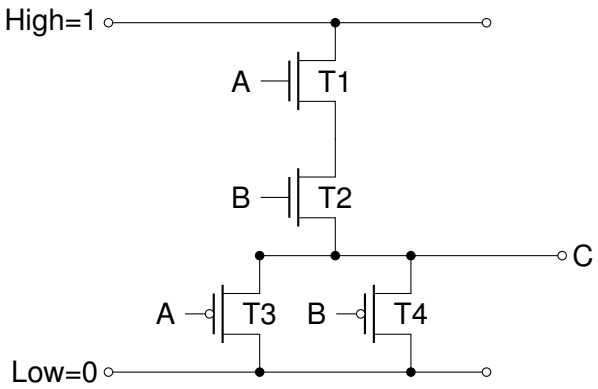


Figure 1: and gate

(ii) This is an example. To compute True $\wedge$ False, A would be high (1) representing True, and B would be low (0) representing False. Since B is 0 and T2 is nmos, there would be no connection at T2, and hence no connection from High to C. But since T4 is pmos, and B is 0, there would be a connection at T4. This means C is connected to Low and hence the output is 0. That corresponds to False, which is indeed True $\wedge$ False as required.

(i) 2 marks

(ii) 3 marks: 2 if explanation has minor deficiencies, 1 if major.

Question 2

This question is about forms of digital representation.

(a) This part is about the binary representation of *unsigned* integers.

(i) The following bit sequences represent unsigned integers in binary form. Translate them to standard decimal giving your reasoning:

- `11110101`
- `00001001`

(ii) Using the standard *binary long addition* algorithm, compute the *sum* of `11110101` and `00001001` as unsigned binary integers. Your answer should also be an unsigned binary integer.

(iii) Using the standard *binary long multiplication* algorithm, compute the *product* of `11110101` and `00001001` as unsigned binary integers. Your answer should also be an unsigned binary integer. **[7 marks — basic]**

Solution:

(i) `11110101` represents $1 * 2^7 + 1 * 2^6 + 1 * 2^5 + 1 * 2^4 + 1 * 2^2 + 1 * 2^0 = 128 + 64 + 32 + 16 + 4 + 1 = 245$

`00001001` represents $1 * 2^4 + 1 * 2^0 = 8 + 1 = 9$

(ii)

```
 11110101
 00001001  +
-----------
        1  carries
 11111110  answer
-----------
```

(iii)

```
     11110101
     00001001  *
---------------
     11110101
  11110101000
---------------
 111111         carries
 100010011101   answer
---------------
```

(i) 2: 1 if process clear but minor errors

(ii) 2: 1 if process clear but minor errors

(iii) 3: 2 if process clear but minor errors, 1 if significant errors

(b) This part is about the binary representation of *signed* integers.

The bit sequences `11110101` and `00001001` are now viewed as representations of *signed* integers using *8-bit two's complement*.

(i) In two's complement, how can we tell whether a bit sequence represents a positive or a negative number?

(ii) What numbers are represented by `11110101` and `00001001`?

Question continues on next page

(iii) *Using binary methods only*, calculate the sum and product of `11110101` and `00001001` as 8-bit two's complement numbers. Marks will mainly be given for the method(s) being used, which should be clearly explained. You need not repeat any calculations carried out previously. **[8 marks — medium]**

Solution:

(i) If the binary begins with 1 then then number is negative. If it begins with 0 it is positive. -

(ii) `11110101` represents $-1 * 2^7 + 1 * 2^6 + 1 * 2^5 + 1 * 2^4 + 1 * 2^2 + 1 * 2^0 = -128 + 64 + 32 + 16 + 4 + 1 = -128 + 117 = -11$
`00001001` still represents $1 * 2^4 + 1 * 2^0 = 8 + 1 = 9$

(iii) We can use the same methods but take only the rightmost eight bits. So the sum is represented by 11111110 and the product by 10011101.

(i) 1 if clear

(ii) 2: 1 if process clear but minor errors

(iii) 5: answers correct and process clear, subtract 2 if not understanding can reuse previous work: 1 for minor errors in computation and 2 for major.

(c) The 64-bit IEEE floating point representation uses 1 bit for the *sign*, 11 bits for the *exponent* and the remaining 52 bits for the trailing *significand*.

(i) Using a decimal floating point example, explain the meanings of the terms *sign*, *exponent* and *significand*.

(ii) Using the example 11000000 00001100 00000000 ... explain which bit sequences give the sign, the exponent and the significand and how they are put together to make a floating point number. You do not need to calculate the decimal representation of the number. What is sought is an explanation of how the representation works in this example.

[5 marks — advanced]

Solution:

(i) Consider the example $-1.23 * 10^{-1} = -0.123$. The *sign* is whether the number is overall positive or negative ($-$ in this instance), the *exponent* is the power of 10, (-1 in this instance) and the *significand* is the number between 1 and 10 that gives the actual digits (1.23 in this instance).

(ii) In this case the sign is 1, so we have a negative number. The bits giving the exponent are 10000000000. As an unsigned integer this represents $2^{10} = 1024$, but because of the way exponents are represented (as the number above -1023, this actually represents 1. The bits giving the significand are

Question continues on next page

1100 00000000 ..., these represent the numbers after a point and the number before is always 1. So this represents 1.11. Putting these together we have $-1.11 * 2^1 = -3.5$. In order to gain full marks students would not be expected to provide all this detail.

(i) 2: correct definitions linked to reasonable example, 1 if minor errors.

(ii) 3: correct analysis of bit pattern and allocation to sign, exponent and significand, with some indication of how a number is constructed from these (full detail not expected). 2: if defiiciencies in bit pattern analysis or failure to acknowledge complexities of representation. 1: if more serious deficiencies.

(d) In UTF-8 the character 'A' is represented in Hex by 41 and the character 'Ω' is represented in Hex by *CEA*9.

(i) What are the actual bit sequences that represent 'A' and 'Ω'?

(ii) Explain the relationship between ASCII and UTF-8, using 'A' and 'Ω' as examples to make your points. **[5 marks — advanced]**

Solution:

(i) A = 41 = 01000001, Ω = *CEA*9 = 11001110 10101001

(ii) UTF-8 is a representation system covering the entire unicode character set (so many symbols from many languages). ASCII is a much older system only covering 128 characters, with the printable ones being standard US characters, 'A' for example. ASCII is included as the first 128 characters of UTF-8 and the binary representations are unchanged (so 'A' is the same in UTF-8 and ASCII). All other characters have at least two bytes. An example of that is Ω which is a two-byte character beginning with the bit sequence 110 (not 0 as all the ASCII characters do).

(i) 2: correct decoding with 1 if minor errors

(ii) 3: reasonable account of representation with following features: uses examples properly, establishes link between UTF-8 and ASCII, gives some idea of role of each (1 point each).

Question 3

This question is about Assembly Language and Instruction Set Architecture.

(a) Processors are often described as 32-bit or 64-bit. Explain what these terms mean, making clear the difference between them.

[4 marks — medium]

Solution: Primarily, this is the width of each CPU register. This in turn affects a wide range of architectural differences. For instance, it determines the length of integers, the length of memory addresses, the width of the memory and data buses, and so on. So a 32-bit processor has 32-bit wide CPU registers, the length of each integer is 32 bits, each memory address is 32 bits long, the data and memory buses are 32 bits wide. In a 64-bit architecture, these are 64 bits.

3 points if the width of the CPU registers is mentioned. 2 points for each of the other ones (the length of integers, the length of memory addresses, the width of the memory and data buses), capped at 4 in total.

(b) Briefly describe the idea of "pipelining" in computer architecture.

[4 marks — basic]

Solution: In general, when a set of processes need to take place in a series of stages, instead of waiting for one process to finish all the stages (which leaves all but the active stage idle), we can "pipe" the processes: when a process finishes the first stage and moves on to the second stage, the next process can enter the first stage, and so on. This makes sure that no stage is idle and speeds up the average finish time. In the context of computer architecture, the stages are the different stages in a CPU: e.g. fetching an instruction from memory, decoding and preparing the inputs for the ALU, the arithmetic or logical operation by the ALU, writing the results to the memory, writing the results into a register.

4 points. Partial points possible. It is OK if the answer is specific to MIPS.

(c) Provide one advantage and one disadvantage of a "Reduced Instruction Set Computer (RISC)" (like MIPS) as compared to a "Complex Instruction Set Computer (CISC)" (like the x86-family).

[4 marks — basic]

Solution: A RISC instruction set has fewer mor ebasic instructions and enables a simpler processor design that is often capable of executing those instructions faster than on a CISC architecture. But it means that common instruction sequences have to be implemented as a sequence of basic instructions instead of as a single (potentially faster) instruction.

Question continues on next page

2 points each advantage and disadvantage.

(d) A short piece of 32-MIPS Assembly code is shown in Figure 2. Trace the execution of this code by giving the values in registers `$t0` and `$t1` as you step through the execution. That is, write down the values of both registers `$t0` and `$t1` (as a pair) each time either one of them changes as you step through the programme until the programme finishes execution. You may assume the execution is started at line 1 with 0 in all registers, and proceeds until the program halts. **[6 marks — basic]**

```
1  addi    $t0, $zero, 4
2  addi    $t1, $zero, 3
3
4  LOOP:
5  beq     $t0, $zero, DONE    #branch on equals
6  add     $t1, $t1, $t1
7  addi    $t0, $t0, -1
8  j  LOOP                     # jump
9
10 DONE:
11 addi $v0, $zero, 10         # set v0 to "10" to select the exit syscall
12 syscall                     # invoking the syscall to exit.
```

Figure 2: 32-MIPS Assembly code

Solution:

`$t0`	4	4	3	3	2	2	1	1	0
`$t1`	3	6	6	12	12	24	24	48	48

6 points for all correct entries. -2 for each wrong entry for `$t1` from the trace: 3, 6, 12, 24, 48, bottomed at 0.

(e) Write down the equivalent MIPS assembly for the java code in Figure 3. Assume that the values of variables `a` and `b` are already loaded in registers `$t0` and `$t1` respectively, i.e., `$t0` holds `a` and `$t1` holds `b`. You do not need to initialise the variables, or insert the exit system call, so something like lines 4-10 of Figure 2 will do.

```
while (a < b) {
  b = b - 10;
  a = a + 2;
}
```

Figure 3: Java code

Question continues on next page

[7 marks — advanced]

Solution:

```
addi    $t0, $zero, 4
addi    $t1, $zero, 3

LOOP:
slt     $t2, $t0, $t1
beq     $t2, $zero, DONE
addi    $t1, $t1, -10
addi    $t0, $t0, 2
j   LOOP

DONE:
addi $v0, $zero, 10         # set v0 to "10" to select exit syscall
syscall                     # invoking the syscall to exit.
```

7 points for a correct code. (-1) point for each pseudo-instruction. (-1) point if the code implements `while (a <= b)` instead but is correct otherwise. If the code is incorrect otherwise, the score should be capped at 4, and partial points on the severity of the mistake.

Question 4

This question is about Computer Networks.

(a) This part is about TCP (Transmission Control Protocol) and UDP (User Datagram Protocol): two of the main transport layer protocols in the internet.

For each of the items in the following list, determine whether it is present in the header of the datagram of the protocol. That is, for each of the following, state whether it is part of a TCP header, part of the UDP header, both, or neither. Your answers must include a short explanation explaining the role played by the port or sequence numbers, and why that means they are, or are not present in the header.

- (source and destination) port numbers
- datagram sequence number

Solution:

- (source and destination) port numbers: present in both TCP and UDP headers, this is because once the datagram reaches its destination, the destination needs to know which application this belongs to, and the application on the destination needs to know which application from the source this belongs to.
- datagram sequence number: only present in TCP headers, this is because only TCP guarantees reliable in-sequence delivery of datagrams which requires identifying datagrams through their sequence number, but UDP would not have any use for it.

2 marks for correct identifications and 2 points for correct explanations: 8 points in total.

[8 marks — advanced]

(b) This part is also about TCP and UDP.

Which of the two protocols, TCP and UDP, is more suitable for communication of live audio and video in a voice-over-IP application (like Skype)? Justify your answer.

[7 marks — basic]

Solution:

UDP is the more suitable choice, because the perceived quality of service (quality of experience) is very sensitive to latency and buffering, but not as much sensitive to occasional reception of out of order or erroneous packets.

3 points for correct identification and 4 points for clear explanation as to why.

(c) When data frames are routed from a client computer to a remote server over many hops, using TCP/IP, determine which one of the following stays the same throughout

Question continues on next page

the trip and which one changes. Provide a brief reasoning for each answer.

- destination port number
- destination MAC address

[5 marks — advanced]

Solution:

- destination port number: this is fixed: because this is related to identifying the application layer program running on the destination, nothing to do with the route.
- destination MAC address: this changes: at each "hop", the destination MAC address is the MAC address of the next immediate device on the next immediate shared medium (e.g. the router on a link).

1 point for correct identification and 2 points for clear explanation for each, capped at 5 in total.

(d) Explain how TCP can find out whether the communicated datagrams are corrupted (due to noise on the communication channel).

[5 marks — medium]

Solution:

This is through use of "checksum": these are extra bits that are added to the datagram so that they act like a integrity check against accidental errors: if an error takes place, then the checksum will not be valid. A simple example will be if the "sum" of all bytes in the datagram are added up, and the number is added to the header.

3 points by mentioning checksum, 2 more points for any correct explanation.

End of questions