4-hour Written Re-Exam in Computer Systems

Department of Computer Science, University of Copenhagen (DIKU)
Date: April 17, 2018

Preamble

Solution

Disclaimer: The reference solutions in the following range from exact results to sketch solutions; this is entirely on purpose. Some of the assignments are very open, which reflects to our solutions being just one of many. Of course we try to give a good solution and even solutions that are much more detailed than what we expect you to give. On the other hand, you would expect to give more detailed solutions that our sketches. Given the broad spectrum of solutions, it is not possible to directly infer any grade level from this document. All solutions have been written in the in-lined space with this colour.

This is the text is an excerpt of the exam set for the 4 hour written re-exam in Computer Systems (CompSys), B1+2-2017/18. This document consists of 10 pages excluding this preamble; make sure you have them all. Read the rest of this preamble carefully. Your submission will be graded as a whole, on the 7-point grading scale, with external censorship.

- You can answer in either Danish or English.
- Remember to write your exam number on all pages.
- You do not have to hand-in this preamble.

Expected usage of time and space

The set is divided into sub-parts that each are given a rough guiding estimate of the time needed. However, your exact usage of time can differ depending on prior knowledge and skill.

Furthermore, all questions includes formatted space (lines, figures, tables, etc.) for in-line answers. Please use these as much as possible. The available spaces are intended to be large enough to include a satisfactory answer of the question; thus, full answers of the question does not necessarily use all available space.
If you find yourself in a position where you need more space or have to redo (partly) an answer to a question, continue on the backside of a paper or write on a separate sheet of paper. Ensure that the question number is included and that you in the in-lined answer space refers to it; e.g. write "*The [rest of this] answer is written on backside of/in appended page XX.*"

For the true/false and multiple-choice questions with one right answer give only one clearly marked answer. If more answers are given, it will be interpreted as incorrectly answered. Thus, if you change your answer, make sure that this shows clearly.

Exam Policy

This is an *individual*, open-book exam. You may use the course book, notes and any documents printed or stored on your computer, but you may not search the Internet or communicate with others to answer the exam.

Exam number: _____

Errors and Ambiguities

In the event of errors or ambiguities in the exam text, you are expected to state your assumptions as to the intended meaning in your answer. Some ambiguities may be intentional.

4-hour Written Re-Exam in Computer Systems
Department of Computer Science, University of Copenhagen
Date: April 17, 2018

Exam number: _____

1 Machine architecture (80 minutes)

1.1 True/False Questions (8 minutes)

For each statement, answer True or False. (Put one "X" in each.)	True	False
a) Within Boolean arithmetic then ~(A & B) = (~A)^(~B).		X
b) The largest `unsigned char` has the value 256.		X
c) The lowest `signed char` has the value -128.	X	
d) Assume `x` and `y` are signed natural values (e.g. `long`), then the C expression `(x < y) == ((-x) > (-y))` is always evaluated to true.		X
e) In the Linux call model, the return address of a procedure call is located in a special purpose register.		X

a) is false, though is looks like DeMorgan's law. Write truth table or consider the size of the domains.
d) is false as the arithmetic negation of the smallest (negative) value of a two's complement number does not have positive representative. Thus for this number then `x == -x`.
e) is false as the return address is located on the stack.

1.2 Multiple Choice Questions (8 minutes)

In each of the following questions choose one answer.

Multiple Choice Questions, 1.2.1: In a pipelined architecture, resolving correctness of a predicted jump is performed in the:

☐ **a)** Fetch phase (`F`),

☐ **b)** decode phase (`D`),

☒ **c)** execute phase (`X`), or

☐ **d)** memory phase (`M`).

Multiple Choice Questions, 1.2.2: The 6-bit two's complement number `100101` represents the value

☐ **a)** 37,

☐ **b)** 27,

☐ **c)** 17,

☐ **d)** -17,

☒ **e)** -27, or

☐ **f)** -37.

Exam number: _____

1.3 Data Cache (16 minutes)

Given a byte-addressed machine with 16-bit addresses. The machine is equipped with a 4-way set associative data cache of 8 kilobytes. Cache have a block size of 16 bytes.

Data Cache, 1.3.1: For each bit in the table below, indicate which bits of the address would be used for

- block offset (denote it with `O`),
- cache tag (denote it with `T`), and
- set index (denote it with `S`).

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

Data Cache, 1.3.2: Consider the stream of storage references in hexadecimal format:

`0xA010` , `0xF020` , `0xFF20` , `0xFF0C` , `0x0028` , `0xF0A4` , `0xF034` .

Assuming that the cache initially is cold followed by referencing the above stream. Given that the cache uses LRU replacement, what is the effect of a following reference to address: `0x0024`.

Block Offset:	`0x4`
Set Index:	`0x02`
Cache tag:	`0x00`
Cache hit or miss:	hit
In case of cache miss, which cache tag is evicted:	

Data Cache, 1.3.3: Briefly argument for your answer regarding cache hit/miss and address eviction and show the content of the set before and after the reference.

The address is already in the cache as the block containing it was loaded with `0x0028`. The cache is 4-way and the only other following block loaded into the same set `0x01` is `0xF034`.

Exam number: _____

1.4 Assembler programming (30 minutes)

Consider the following program written in X86-assembler.

```
1  program:
2    movq  %rdi, %rax
3    movl  $0, %edx
4    jmp L2
5  L3:
6    addq  (%rax), %rdx
7    addq  $8, %rax
8  L2:
9    cmpq  %rsi, %rdx
10   jl  L3
11   subq  %rdi, %rax
12   sarq  $2, %rax
13   ret
```

Assembler programming, 1.4.1: Rewrite the above X86-assembler program to a C program and explain the functionality of the program. The resulting program should not have a goto-style and minor syntactical mistakes are acceptable.

The program take two arguments (`%rdi` and `%rsi`) and calculates twice the number of elements in an unbounded array (*a) that is a required to reach max.

```
long program(long *arr, long max) {
  long *arr_ptr = arr; // Located in %rdi
  long sum = 0; // Located in %rdx
  while (sum < max) {
    sum += *arr_ptr;
    arr_ptr++;
  }
  return (arr_ptr - arr) * 2;
}
```

That the return values was twice the size was unintended and due to a copy-paste error; the instruction should have right-shifted by 3. Any answer that does not include this is also accepted and this part should show understanding of pointer arithmetic.

Exam number: _____

Assembler programming, 1.4.2: Argument for your choice of statements and expression. Specify which part of the program are not directly translated and argue why.

We see on the register types that values are `long`. First argument (`%rdi` which is used in `%rax`) is a pointer to an array; this is seen by the indexing in line 6. The second (`%rsi`) is a number. `%rdx` is overwritten by 0 (in `%edx`) and therefore not an input.

The program structure is a standard while loop as following translation from BOH. The comparison of `%rsi` to `%rdx` with the following jump-less-than is the condition of `sum` < lstinlinemax.

The array is iterated using pointer arithmetic which is seen in line 7. Line 11 finds the number of elements from the initial array pointer, which in line 12 is divided by 4 using a right-shift. (Note, this was intended to be 8 (shift 3) to account for the length a `long`; thus the return value is twice the length.)

Assembler programming, 1.4.3: Briefly describe the semantic difference between logical and arithmetical shifts.

Right shifts are identical. A logical left shift always adds a 0 as the most significant bit. Arithmetical left shifts duplicated the most significant bit to ensure the sign of a two's complement number.

Exam number: _____

2 Operating Systems (80 minutes)

2.1 True/False Questions (8 minutes)

For each statement, answer True or False. (Put one "X" in each.)	True	False
a) `fopen()` is not a system call.	X	
b) There is never more physical memory than virtual memory.		X
c) System calls are implemented via signals.		X
d) Virtual memory requires a disk.		X
e) System calls run in user mode.		X
f) Condition variables cannot be efficiently implemented solely with mutexes.	X	

2.2 Multiple Choice Questions (12 minutes)

In each of the following questions, you may put one or more answers.

Multiple Choice Questions, 2.2.1: Which of the following operations are guaranteed to execute atomically?

☐ **a)** `pthread_cond_signal()`

☒ **b)** `pthread_mutex_lock()`

☐ **c)** `x++` (when `x` is `int`)

☐ **d)** `memcpy(&x, &y, sizeof(x))`

☒ **e)** `pthread_cond_wait()`

☐ **f)** `exit(0)`

Multiple Choice Questions, 2.2.2: Consider a demand-paged system with the following time-measured utilisations:

CPU utilisation	50%
Paging disk	0.7%
Other I/O devices	75%

Which of the following would likely improve CPU utilisation?

☒ **a)** Install a faster CPU.

☐ **b)** Install a bigger paging disk.

☐ **c)** Install a faster paging disk.

☐ **d)** Install more main memory.

☒ **e)** Increase the degree of multiprogramming.

Exam number: _____

2.3 Long Questions (36 minutes)

Long Questions, 2.3.1: Which of the following programming techniques and data structures are "good" for a demand-paged environment, and which are "bad" (performance-wise)? Explain your answers.

- Stack
- Hash table
- Sequential search of array
- Sequential search of linked list
- Binary search of array
- Vector operations (such as vector addition or computing dot products)

A stack is efficient because operations exhibit good *locality*—we are always operating on addresses near each other, which minimises page faults. A hash table can be inefficient, because a good hash algorithm will ensure that accesses are evenly distributed among buckets. A sequential search of an array is efficient, because of good locality. Sequentially searching of a linked list is likely inefficient, because logically neighboring nodes can be arbitrarily distant in memory. Binary search of an array is likewise also inefficient (from a memory access point of view), again because of large jumps in addresses. Vector operations tend to be efficient, because they involve sequentially traversing arrays.

Exam number: _____

3 Computer Networks (80 minutes)

3.1 True/False Questions (8 minutes)

For each statement, answer True or False. (Put one "X" in each.)	True	False
a) Implementation of link layer protocols span both hardware (network controllers) and software (operating systems).	X	
b) Peer-to-peer architectures exhibit better scalability because adding peers results in an increase in cumulative bandwidth available for all communicating parties.	X	
c) For a TCP connection, the receive window can never become zero.		X
d) Convergence time of OSPF protocol is independent of the number of edges in a network.	X	

3.2 Multiple Choice Questions (15 minutes)

In each of the following questions choose one answer.

Multiple Choice Questions, 3.2.1: Consider a two dimensional even parity scheme for error detection. Using this scheme compute the parity bits of the 8-bit ASCII[1] representation of "REEXAM" where each byte of the word forms a row for the two dimensional parity scheme. The resultant parity bits (row followed by column parity bits) are

☒ **a)** 111100 00000110

☐ **b)** 111101 00000010

☐ **c)** 100101 00001110

☐ **d)** None of the above

Multiple Choice Questions, 3.2.2: The broadcast address of the network 117.18.31.54/18 is

☐ **a)** 117.18.31.255

☒ **b)** 117.18.63.255

☐ **c)** 117.18.127.255

☐ **d)** None of the above

[1] ASCII codes of A-Z lie contiguously between decimal numbers 65-90.

Exam number: _____

3.3 Short Questions (24 minutes)

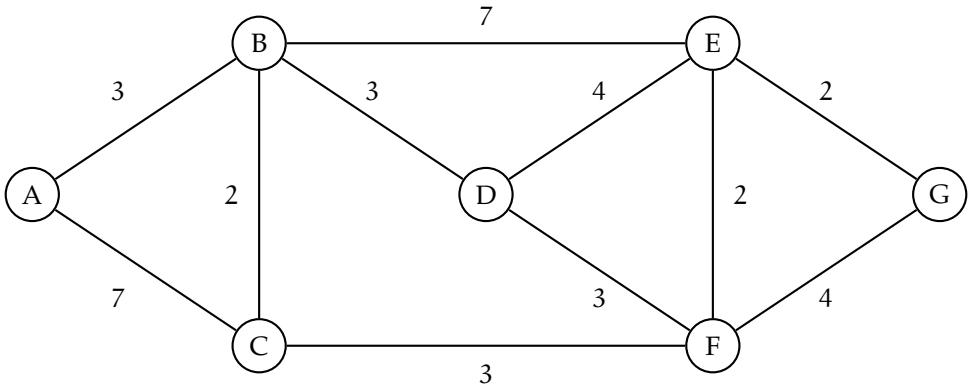
Short Questions, 3.3.1: Why is an ARP query sent within a broadcast frame while the ARP response is sent within a frame having a specific destination address?

An ARP query is sent to resolve an IP address to a MAC address by the link layer. Since the destination MAC address is unknown, a link layer frame consisting of the ARP query is broadcast using the special MAC broadcast address. Since an ARP response is sent on receipt of an ARP query which contains the MAC address of the sender, the link layer frame containing the ARP response can be sent to the specific MAC address of the ARP query sender.

Exam number: _____

3.4 Network Routing (18 minutes)

Consider the network topology outlined in the graph below



Network Routing, 3.4.1: Apply the link state routing algorithm and compute the forwarding tables on nodes A and D. (Note: Remember to show the steps of the algorithm.)

The computation of link state routing algorithm on node A:

Step	N′	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)	D(G),p(G)
0	A	3,A	7,A	∞	∞	∞	∞
1	AB		5,B	6,B	10,B		
2	ABC					8,C	
3	ABCD				10,D/B	8,C	
4	ABCDF				10,D/B/F		12,F
5	ABCDFE						12,F/E
6	ABCDFEG						

In the above computation, if there are multiple paths with same cost they have been shown with / to demarcate them being acceptable values.

The computation of link state algorithm on node D:

Step	N′	D(A),p(A)	D(B),p(B)	D(C),p(C)	D(E),p(E)	D(F),p(F)	D(G),p(G)
0	D	∞	3,D	∞	4,D	3,D	∞
1	DB	6,B		5,B	4,D		
2	DBF			5,B	4,D		7,F
3	DBFE					3,D	6,E
4	DBFEC	6,B					
5	DBFECA						
6	DBFECAG						

In the above computation because of equal cost paths existing the choice of nodes in N′ can be inverted in step 1 and 2 and in steps 5 and 6.

Exam number: _____

Forwarding table on node A follows:

Destination node	Edge
all nodes	(A,B)

Forwarding table on node D follows:

Destination node	Edge
A	(D,B)
B	(D,B)
C	(D,B)
E	(D,E)
F	(D,F)
G	(D,E)

Network Routing, 3.4.2: List the problems that are overcome using hierarchical routing.

Some of the problems that are solved by hierarchical routing are:

1. Scalability issues in message communication overheads with growing network size.

2. Scalability issues in routing algorithm computing overheads with growing network size.

3. Scalability issues in storage and retrieval overheads of routing information with growing network size.

2. Administrative autonomy which allows each autonomous network to operate and administer itself while still being able to inter-operate with other networks.