8-hour Take-Home Re-Exam in Computer Systems

Department of Computer Science, University of Copenhagen (DIKU)
Date: April 21, 2021

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 exam set for the 8 hour written take-home re-exam in Computer Systems (CompSys), B1+2-2020/21. This document consists of 25 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.

I addition to this document you are given two templates to write your answer. One is formatted as a LaTeX document, while the other is a pdf-file with included form fields. You can use either of these to make your answer.

- You can answer in either Danish or English.
- Remember to write your exam number (and only your exam number) on the hand-in.
- Do not exceed the line limits indicated in the answer-templates (see below).
- Only hand-in one pdf-file with your answers on eksamen.ku.dk. Note: you can only hand-in once.

Expected usage of time and space

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

Furthermore, all questions in the answer-templates includes limitation to the number of lines for answers in a standard page and font formatting. These limitations are more than enough to include a complete and satisfactory answer of the question; thus, full answers of the question does not necessarily use all available space.
These limitations are strict, meaning we reserve the right to *not* evaluate any part of the answer exceeding the given limits.

Exam Policy

This is an *individual*, open-book exam. Thus, you are not allowed to discuss the exam set with anyone else, until the exam is over (be aware of co-students that can have gotten extra time). It is expressly forbidden:

Exam number: ____________

- To discuss or share any part of this exam, including, but not limited to, partial solutions, with anyone else.
- To help or receive help from others.
- To seek inspiration from other sources, including the Internet, without proper citation in your answer.
- Copy-paste a text (even if it partly edited) without proper citation in your answer.
- To post any questions or answers related to the exam on *any fora*, before the exam is over, including the course discussion forum on Absalon and Discord.

Breaches of this policy will be handled in accordance with the standard disciplinary procedures. Possible consequences range from your work being considered `void`, to *expulsion from the university*[1].

You may use the course book, notes and any documents printed or stored on your computer and other device. If you use any other material, you must cite this in your answer.

Errors and Ambiguities

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

If you are in doubt, you can contact `m.kirkedal@di.ku.dk` for clarification. But you will not get answers to clarification of course curriculum.

If there during the exam are corrections or clarifications to any question, these will be posted as an announcement on Absalon course page. Be sure that you receive notifications from this.

IMPORTANT
It is important to consider the context and expectations of the exam sets. Each question is designed to cover one of more learning goals from the course. The total exam set will cover all or (more likely) a subset of all the learning goals; this will vary from year to year.
The course has many more learning goals than are realistic to cover during a 4-hour exam. And even though we limit the tested learning goals, it will still be too much.
Therefore, it is not expected that you give full and correct answer to all questions. Instead you can focus on parts in which you can show what you have learned.
It is, however, important to note that your effort will be graded as a whole. Thus, showing your knowledge in a wide range of topics is better that specialising in a specific topic. This is specially true when considering the three overall topics: Arc, OS, and CN.

Specifically, for this exam set it was need to solve:
* about 45 % to pass
* about 60 % to get a mid-range grade
* about 80 % to get a top grade
* no one solved all parts correctly!
NB! we adjust the exam set from year to year, so the above is not written in stone and can change slightly for your exam.

[1] `https://uddannelseskvalitet.ku.dk/docs/overordnede-dokumenter/Ordensregler-010914.pdf`

8-hour Take-Home Re-Exam in Computer Systems
Department of Computer Science, University of Copenhagen
Date: April 21, 2021

Exam number: ____________

1 Machine architecture (about 33 %)

1.1 Assembler programming (about 13 %)

Consider the following program written in X86prime-assembler.

```
.LFB8:
    subq $8, %rsp
    movq %r11, (%rsp)
    movq $1, %edx
.L14:
    cble %rdi, %rdx, .L22
    leaq (%rsi, %rdx, 8), %r11
    movq (%r11), %rcx
    leaq -1(%rdx), %rax
.L15:
    leaq (%rsi, %rax, 8), %r11
    movq (%r11), %r8
    cbge %rcx, %r8, .L16
    leaq 8(%rsi, %rax, 8), %r11
    movq %r8, (%r11)
    subq $1, %rax
    cbne $-1, %rax, .L15
.L16:
    leaq 8(%rsi, %rax, 8), %r11
    movq %rcx, (%r11)
    addq $1, %rdx
    jmp .L14
.L22:
    movq (%rsp), %r11
    addq $8, %rsp
    ret %r11
```

Exam number:

Question 1.1.1: Describe the details of the program. This includes (but is not limited to):

- Identify the instructions that implements the call convention; makes it possible to call the piece of code as a function.
- Identify the control structures of the program (e.g. loops, conditionals, and function calls).
- Identify registers which must have specific values at call time, for the function to have a specific purpose.
- Specify for each used register, the C type that best reflects the use of the register. (If a register is used in more than one way, specify this.)

The first 2 and last 3 instructions implement the calling convention, handling the return address and adjusting the stack pointer.
The function has two nested loops. Outer loop starts at L14 and ends above L22, inner loop starts at L5 and ends above L16.
The function uses two registers without initialization, so they must be parameters: `%rdi` and `%rsi`.
Types for all registers in use (except for the calling convention):
`%edx / %rdx` - signed long (index into an array)
`%rsi` - pointer to signed long (the array indexed)
`%r11` - pointer to signed long (pointer to element in array)
`%rcx` - signed long (value loaded from array)
`%rax` - signed long (another index into array)
`%r8` - signed long (another value loaded from array)

Question 1.1.2: Rewrite the above X86prime-assembler program to a C program. The resulting program must not have a goto-style and minor syntactical mistakes are acceptable.

```
void sort(long num_elem, long array[]) {
  for (long i = 1; i < num_elem; ++i) {
    long x = array[i];
    long j = i - 1;
    while (j >= 0 && array[j] > x) {
      array[j + 1] = array[j];
      --j;
    }
    array[j + 1] = x;
  }
}
```

Question 1.1.3: Descripe shortly the functionality of the program.

This is insertion sort. Parameters are an array to sort and the size of that array. Conceptually the array is split in two parts, a sorted part followed by the unsorted part. Initially the sorted part is empty, and the unsorted part fills the entire array. The array is sorted gradually by building the sorted part in the beginning of the array. One element at a time is taken from the unsorted part and inserted at the proper place within the sorted part, until the entire array is sorted.

Exam number: ____________

1.2 Performance / Pipeline (about 10 %)

Consider the following execution graph (afviklingsplot) that illustrates the execution of a sequence of instructions on a 5-step pipeline machine, similar to the one described here: https://x86prime.github.io/afviklingsplot/pipeline/. The sequence comprises two iterations of a loop, so it's not an error that the instructions are repeated.

```
.L15:
    leaq (%rsi, %rax, 8), %r11    FDXMW
    movq (%r11), %r8               FDXMW
    cbge %rcx, %r8, .L16            FDDXMW           (1)
    leaq 8(%rsi, %rax, 8), %r11      FFDXMW
    movq %r8, (%r11)                   FDXMW
    subq $1, %rax                       FDXMW
    cbne $-1, %rax, .L15                 FDXMW
.L15
    leaq (%rsi, %rax, 8), %r11              FDXMW        (2)
    movq (%r11), %r8                         FDXMW
    cbge %rcx, %r8, .L16                      FDDXMW       (1)
    leaq 8(%rsi, %rax, 8), %r11                FFDXMW
    movq %r8, (%r11)                             FDXMW
    subq $1, %rax                                 FDXMW
    cbne $-1, %rax, .L15                           FDXMW
```

Question 1.2.1: The third and tenth instruction (marked by (1)) stalls in the D-stage. Give an explanation of why this stall is needed.

This stall is needed to wait for arrival data in %r8 which is fetched from the data cache by the previous instruction.

Question 1.2.2: The eighth instruction (marked by (2)) is fetched 2 cycles later than what would be expected, if it was just following directly after the seventh instruction. So obviously it is not. Give an explanation of why this delay occurs.

This delay occurs because the eigth instruction is the target of a conditional branch. That branch does not resolve until it reaches the X stage, so the F stage of the target instruction cannot be earlier than that.

Exam number:

Question 1.2.3: For this question, imagine that we have "stretched" the pipeline to 7 stages to allow more time for cache access. Instead of the stages F-D-X-M-W we have F-f-D-X-M-m-W, where the original F-stage is split into F-f and the original M-stage is split into M-m.

Redo the execution graph of the above instruction sequence for the new pipeline. Give an explanation of any additional stalls caused by the longer pipeline.
(Note, that there might be too few columns in the table. In that case wrap around and start from the beginning again.)

```
.L15:
    leaq (%rsi, %rax, 8), %r11    FfDXMmW
    movq (%r11), %r8               FfDXMmW
    cbge %rcx, %r8, .L16            FfDDDXMmW           (1)
    leaq 8(%rsi, %rax, 8), %r11      FfffDXMmW
    movq %r8, (%r11)                  FFFfDXMmW
    subq $1, %rax                        FfDXMmW
    cbne $-1, %rax, .L15                  FfDXMmW
.L15
    leaq (%rsi, %rax, 8), %r11                FfDXMmW        (2)
    movq (%r11), %r8                           FfDXMmW
    cbge %rcx, %r8, .L16                        FfDDDXMmW       (1)
    leaq 8(%rsi, %rax, 8), %r11                  FfffDXMmW
    movq %r8, (%r11)                              FFFfDXMmW
    subq $1, %rax                                    FfDXMmW
    cbne $-1, %rax, .L15                              FfDXMmW
```

There are 2 kinds of additional stalls:
(1) we now have to wait an additional cycle for data to be read from cache
(2) since the X stage of a branch now occurs one cycle later due to the longer pipeline, the F-stage of the target instructions is delayed by one additional cycle.

Exam number:

1.3 Data Cache (about 10 %)

In the following we examine a 2-way set-associative data cache with a total of 32 cache-blocks of 16 bytes each.

Question 1.3.1: Describe how an address is partitioned into the three components (tag, index, byte-offset) used when accessing such a cache.

The byte offset must be able to address 16 bytes, so it takes up the lowest 4 bits (0-3). The index must be able to select one of 16 sets, so it take up the next 4 bits (4-7). The tag is the rest of the address (8-whatever the address size)

Exam number:

Question 1.3.2: Determine tag and index for each of the addresses in the following list:

`0x010`, `0x011`, `0x02F`, `0x112`, `0x213`, `0x314`, `0x11C`, `0x715`, `0x012`, `0x11F`, `0x220`, `0x020`.

Address	**tag**	**index**
0x010	0x0	0x1
0x011	0x0	0x1
0x02F	0x0	0x2
0x112	0x1	0x1
0x213	0x2	0x1
0x314	0x3	0x1
0x11C	0x1	0x1
0x715	0x7	0x1
0x012	0x0	0x1
0x11F	0x1	0x1
0x220	0x2	0x2
0x020	0x0	0x2

Exam number:

Question 1.3.3: Write down a simulation of how the cache state changes as it is accessed according to the above sequence. Assume LRU replacement.

Show the cache content by showing the tags for each valid entry for each index. Ignore invalid entries.

example:

```
2:{3,5} 3:{1}
```

means the caches has 3 valid entries, at index 2 (in set nr 2) we have tags 3 and 5, and at index 3 we have the tag 1. keep the tags within each set sorted by time of access, so the newest accessed tag is first. That way you know how to find the least recently used tag for replacement.

Write one line for each access. Show the cache content before and after each access, the accessed address as (index,tag) pair, and whether the access is a hit or miss.

example:

```
2:{3,5} 3:{1}    (1,456)    Miss    1:{456}, 2:{3,5}, 3:{1}
```

```
                     (1,0)   Miss   1:{0}
1:{0}                (1,0)   Hit    1:{0}
1:{0}                (2,0)   Miss   1:{0},2:{0}
1:{0},2:{0}          (1,1)   Miss   1:{1,0},2:{0}
1:{1,0},2:{0}        (1,2)   Miss   1:{2,1},2:{0}
1:{2,1},2:{0}        (1,3)   Miss   1:{3,2},2:{0}
1:{3,2},2:{0}        (1,1)   Miss   1:{1,3},2:{0}
1:{1,3},2:{0}        (1,7)   Miss   1:{7,1},2:{0}
1:{7,1},2:{0}        (1,0)   Miss   1:{0,7},2:{0}
1:{0,7},2:{0}        (1,1)   Miss   1:{1,0},2:{0}
1:{1,0},2:{0}        (2,2)   Miss   1:{1,0},2:{2,0}
1:{1,0},2:{2,0}      (2,0)   Hit    1:{1,0},2:{0,2}
```

Exam number: ____________

2 Operating Systems (about 33 %)

2.1 Multiple Choice Questions (about 6 %)

In each of the following questions, you may put "X" in one or more answers. Use the lines to argue for your choices.

Question 2.1.1: Which of the following operations always involves transferring control to the kernel?

- ☒ **a)** System call
- ☐ **b)** Reading from memory
- ☐ **c)** Writing to memory
- ☒ **d)** Opening a file
- ☒ **e)** Integer division by zero
- ☐ **f)** Floating-point division by zero

Question 2.1.2: Which of the following components of a Unix system run in kernel mode?

- ☒ **a)** The virtual memory manager
- ☐ **b)** `malloc()`
- ☐ **c)** The shell
- ☒ **d)** File system implementation
- ☒ **e)** Process scheduler
- ☐ **f)** The C compiler

Exam number:

Question 2.1.3: Consider the C program below. For space reasons, we are not checking error return codes, so assume that all functions return normally.

```c
int main () {
  if (fork() == 0) {
    if (fork() != 0) {
        pid_t pid; int status;
        if ((pid = wait(&status)) > 0) {
            printf("z");
        }
    } else {
        printf("x");
    }
  }
  else {
    printf("y");
    exit(0);
  }
  printf("v");
  return 0;
}
```

Which of the following strings is a possible output of the program? Place any number of marks.

☐ **a)** xyvvz

☐ **b)** zvxvy

☒ **c)** yxvzv

☒ **d)** xyvzv

☒ **e)** xvzvy

Exam number:

2.2 Short Questions (about 12 %)

Question 2.2.1: For this question, assume that `counter++` and `counter-` execute *atomically* but otherwise have their usual semantics (they don't normally in C, but there exist special instructions that do). Is the following then a correct implementation of semaphores? Would it be correct without the atomicity assumption? Explain your answer.

```
int counter = 1;

void P() {
  while (counter-- <= 0) {
    counter++;
  }
}

void V() {
  counter++;
}
```

The invariant for semaphores is that after a call to `P` or `V`, the counter is non-negative. This implementation maintains that by undoing the decrement if the old value was less than or equal to zero (implying that the decrement made it negative). If the increment/decrement operators are not atomic, updates to the shared variable `counter` might be lost if the read-modify-write process is interrupted.

Question 2.2.2: Consider a system with the following properties:

- Memory is byte-addressed.
- Virtual addresses are 14 bits wide.
- Physical addresses are 13 bits wide.
- The page size is 16 bytes.
- The TLB is 4-way set associative with 4 sets and 16 total entries. Its contents are:

Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	0E	11	1	01	00	1	09	00	0	11	00	0
1	23	12	1	00	12	0	19	0A	0	2F	3F	0
2	21	10	0	15	15	1	2F	2F	1	04	15	1
3	3F	10	1	1A	00	0	19	00	1	31	15	1

- The page table contains 12 PTEs:

VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid
22	33	1	01	02	0	41	01	1	15	01	1
01	33	1	0C	0D	0	08	17	1	13	15	1
A0	21	0	FA	00	1	A2	32	0	03	43	1

Note that all addresses are given in hexadecimal. In the following questions, you are asked, for various virtual addresses, to show the translation from virtual to physical addresses in the memory system just described. *Hint: there is one TLB hit, one page table hit, and one page fault (not necessarily in that order). This should help you double-check your work.*

Exam number:

Virtual address: `0x120`

1. Bits of virtual address

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

2. Address translation

Parameter	Value
VPN	12
TLB index	2
TLB tag	4
TLB hit? (Y/N)	Y
Page fault? (Y/N)	N
PPN	15

3. Bits of phys. (if any)

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

Virtual address: `0x130`

1. Bits of virtual address

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

2. Address translation

Parameter	Value
VPN	13
TLB index	3
TLB tag	4
TLB hit? (Y/N)	N
Page fault? (Y/N)	N
PPN	15

3. Bits of phys. (if any)

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

Exam number:

Virtual address: `0x140`

1. Bits of virtual address

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

2. Address translation

Parameter	Value
VPN	14
TLB index	0
TLB tag	5
TLB hit? (Y/N)	N
Page fault? (Y/N)	Y
PPN	

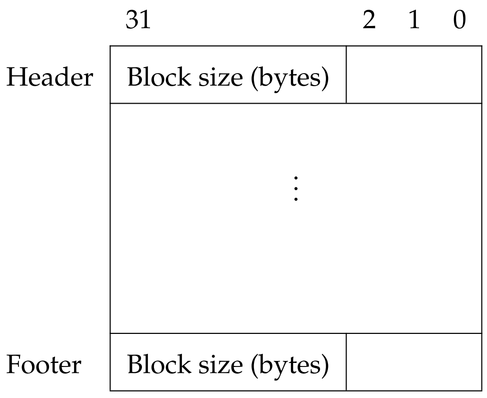
3. Bits of phys. (if any)

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

Also answer the following: Describe shortly how you calculated your answers.

2.3 Long Questions (about 15 %)

Question 2.3.1: Consider an allocator that uses an implicit free list. The layout of each allocated and free memory block is as follows, with one 32-bit word per row:



Each memory block, either allocated or free, has a size that is a multiple of eight bytes, rounding up allocations if necessary. Thus, only the 29 higher order bits in the header and footer are needed to record block size, which includes the header and footer. The usage of the remaining 3 lower order bits is as follows:

- `bit 0` indicates the use of the current block: 1 for allocated, 0 for free.
- `bit 1` indicates the use of the previous adjacent block: 1 for allocated, 0 for free.
- `bit 2` is unused and is always set to be 0.

Important: We must *never* create blocks with zero payload (i.e. we must *never* create blocks with size 8).

Given the heap shown on the left, show the new heap contents after *consecutive* calls to

1. `malloc(8)`. Assume that the return value is `0x400b010` and that the allocator minimises internal fragmentation.
2. `realloc(0x400b010, 20)`. Assume the the return value is `0x400b010`, meaning that the existing allocation is resized.

Your answers should be given as hex values. Note that the address grows from bottom up. Assume that the allocator uses immediate coalescing, that is, adjacent free blocks are merged immediately each time a block is freed.

Exam number:

Address	**Original value**	**After `malloc`**	**After `realloc`**
0x400b02c	0x000000f0	0xf0	0xf2
0x400b028	0x00000022	0x12	0x23
0x400b024	0x000000f1	0xf1	0xf3
0x400b020	0x00000018	0x18	0x18
0x400b01c	0x00000012	0x12	0x12
0x400b018	0x00000011	0x13	0x11
0x400b014	0x400b511c	0x400b511c	0x400b511c
0x400b010	0x400b601c	0x400b601c	0x400b601c
0x400b00c	0x00000022	0x13	0x23
0x400b008	0x00000011	0x11	0x11
0x400b004	0x400b601c	0x400b601c	0x400b601c
0x400b000	0x400b511c	0x400b511c	0x400b511c
0x400affc	0x00000011	0x11	0x11

Also answer the following: Based on the view of the heap shown in the able above, could a call `malloc(100)` succeed, and what would it return?

Yes. The word at `0x400b02c` is a header for a free block of 240 bytes, which is enough to satisfy the request. `malloc(100)` would return `0x400b030`

Exam number: ____________

Question 2.3.2: How might TLB interactions make context-switching between different processes be less efficient than context-switching between threads within the same process?

Threads within the same process operate within the same virtual memory space, which means they exhibit more TLB locality. In contrast, processes have distinct memory spaces, so (in principle) the TLB must be vacated on a context switch. In practice, TLB entries are usually tagged with address space identifiers, so the TLB need not be completely emptied, but different processes might still compete for TLB space, while threads within the same process could shared TLB entries for shared data.

Exam number:

3 Computer Networks (about 34 %)

3.1 Application layer (about 8 %)

Question 3.1.1: Consider the following HTTP request to get the CompSys course page.

```
GET compSys/ HTTP/1.1
Host: absalon.ku.dk
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
```

Explain the usage and need of the different header fields.

Interesting parts that should be described are:
* Host and path
* Version
* Keep-alive
* Cache handling

Question 3.1.2: Assume, that you get the following reply from the server.

```
HTTP/2.0 200 OK
Date: Wed, 21 Apr 2021 16:15:13 GMT
Server: Apache/2.4.6 (Red Hat Enterprise Linux) OpenSSL/1.0.2k-fips mod_fcgid/2.3.9
Last-Modified: Thu, 15 Apr 2010 12:34:45 GMT
ETag: "122-48445b8360340"
Accept-Ranges: bytes
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
<html><head><title>My Page</title></head><body>
Coming soon...
</body></html>
```

Which parts of this response correctly matches the request and which parts are wrong? How can these be corrected?

The following are errors: Wrong HTTP version. The server cannot upgrade (only downgrade) the version that the client asks for.

No Content-Length

No extra line change before content

Exam number:

Question 3.1.3: Assume that you are connecting the KU-server from your study exchange at MIT in USA and that KU has a Content Delivery Service agreement with Akamai Technologies such that all KU web-content is located at a nearby server.

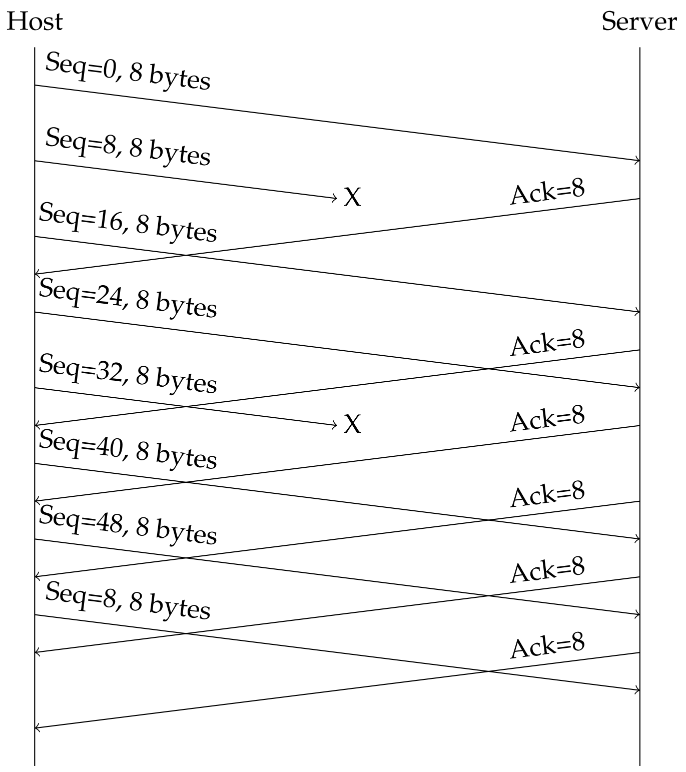
Which changes (if any) would there be to the HTTP request and answer? Argue for your answer.

The request does not require any changes. As a users you are not expected to know this in advance. However, the requests asks for no caching, which can significantly reduce the effect of the CDN.
The response will be different. You are likely to get a `301 Moved Permanently` back with the location of the Akamai server, that you should contact. This will come based in the server you are asked to contact through the geographical location in DNS. Then your browser will send a new request to this server for the new content.

Exam number:

3.2 Reliable Data Transfer (about 10 %)

Consider the following TCP communication diagram.



Question 3.2.1: What is happening with the TCP communication. Explain which TCP mechanisms that are in effect. Agrue for your answer.

Fast re-transmit. After the 3rd duplicate Ack=8, the Host resends the missing frame with Seq=8.

Question 3.2.2: What should the next message that the server sends contain. Argue for your answer.

Ack = 32. There is a later lost frame with seq=32. The server should send ack equal to the frame that it expects should be the next, which is the lost one.

Question 3.2.3: What should the next message that the client sends contain. Argue for your answer.

Seq=56. After fast re-transmit, the host expects that all other send frames have arrived without a problem and it should thus continue sending.

Question 3.2.4: Describe how TCP slow start works, in which situation it is used and what effect it has.

Slow start is used during a TCP connection, when a connection is completely dropped. Not just a single package-loss.

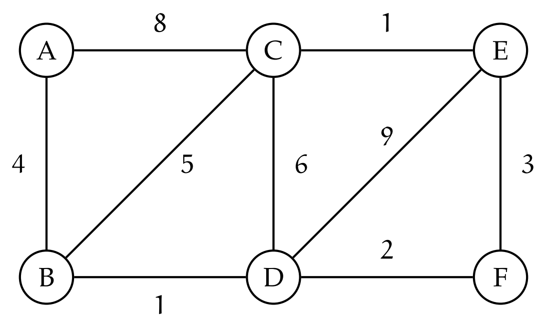
Slow start increases the number of send frames by doubling the window size every time the send window is successful. This happens until the window size more than half of the window size before the connection dropped.

Slow start makes sure that the network is not flooded just after a connection drop (thus slow), but after initial success it quickly (exponentially) ramps-up until half the window size.

Exam number: ____________

3.3 Network Layer (about 8 %)

Question 3.3.1: Consider the network topology outlined in the graph below.



Apply the link state routing algorithm and compute the forwarding table on node **A** by filling out the following tables.

Steps of the algorithm:

Step	N′	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	4, A	8,A	∞	∞	∞
1	A,B	4, A	8,A	5,B	∞	∞
2	A,B,D	4, A	8,A	5,B	14, D	7, D
3	A,B,D,F	4, A	8,A	5,B	10, F	7, D
4	A,B,D,F,C	4, A	8,A	5,B	9, C	7, D
5	A,B,D,F,C,E	4, A	8,A	5,B	9, C	7, D

Forwarding table on node A:

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

The order is strictly defined above by choosing the node with the least cost.

Question 3.3.2: In which situations are the link state algorithm typically used? What are the advantages of using the link state algorithm over the distance vector algorithm?

In networks that tend to be very static and where optimality is important. That is mainly in backbone routers.

Exam number:

3.4 Network Security (about 8 %)

Question 3.4.1: Bob wants to send the message "`Hi guys, do you want meet for a beer tonight?`" so several of his friends. He knows that with the current Covid-19 restrictions, this is not allowed, so the message should be send confidentially.

How can Bob use the IT security mechanisms from the lecture book to achieve confidentiality? Explain your details.

The best is to use asymmetric encryption. This however requires that all his friends starts by sharing a public key with Bob. This can either be used to share the message directly (encrypted with the public key) or staring a shared-secret from which a key for symmetric encryption can be derived.

Question 3.4.2: Explain what properties that are required for a hash function to be called a cryptographic hash function. Why are these properties important?

The following three are central to security: * Infeasible to generate a message that yields a given hash value

* Infeasible to find two different messages with the same hash value

* A small change to a message should change the hash value so extensively