4-hour Re-Exam in Computer Systems, B1-2021/22

Department of Computer Science, University of Copenhagen (DIKU)
Date: April 20, 2022

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 4-hour written exam in Computer Systems, B1-2021/22. Your submission will be graded as a whole on the 7-point grading scale, with internal censoring.

- You can answer in either Danish or English.
- 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. You can also hand a picture for the relevant question.

You can base you submission on either the pdf-file with form fields or the given LaTeX template. Please note that questions will only be given in the pdf-file. In case of differences between the two versions, the pdf-file is always the correct one.

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

Following this, 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 seek inspiration from other sources than course material, without proper citation.
- Copy-paste a text (even if it partly edited) without proper citation in your answer.
- **To use any resources from the internet during the exam.**
- **To communicate with anyone over the internet during the exam.**

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].

Errors and Ambiguities

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

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 sections: 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`

4-hour Re-Exam in Computer Systems, B1-2021/22
Department of Computer Science, University of Copenhagen
Date: April 20, 2022

Exam number:

1 Data Representation and Cache (about 12 %)

1.1 Data representation (about 4 %)

Question 1.1.1: How many bits are used to represent a ‘char‘, a ‘long‘ and a ‘float‘ in the normal 64-bit Intel architecture?

8, 64 and 32 bits respectively.

Question 1.1.2: Why is it not possible to represent $\frac{1}{10}$ (one tenth) in the IEEE floating point format? What is the consequence of this?

The fractional part of a floating point number is represented by a binary number, which 1/10 cannot be. This results in rounding of numbers and specifically that imprecision is added to currencies calculations. Thus, floating point numbers are not used directly for this.

Exam number:

1.2 Data Cache and Locality (about 8 %)

Consider the following C-like program that finds the number of the element in an a matrix (of size `N`×`M`) that is larger than a pivot. We assume some relevant high definition of `M` and `N`.

```c
long count_larger_than(long pivot, long matrix[N][M]) {
  int i, j;
  int cnt = 0;

  // For each element in array
  for (i = 0; i < M; i++) {
    for (j = 0; j < N; j++) {
      if (matrix[i][j] > pivot) {
        cnt++;
      }
    }
  }
  return cnt;
}
```

Question 1.2.1: Is it possible to improve the *temporal locality* of the program? If not, explain why this is the case. If it is possible, explain how and what the improvement gives.

Both i and j are used in a local context, so temporal locality is good.

Question 1.2.2: Is it possible to improve the *spacial locality* of the program? If not, explain why this is the case. If it is possible, explain how and what the improvement gives.

The program has very good spacial locality. We are reading the matrix by iterating over the columns in the inner loop.
The vector is also read in element order.

2 Operating Systems (about 32 %)

2.1 Multiple Choice Questions (about 6 %)

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

Question 2.1.1: Which of these functions *always* involve a system call (assuming no error occurs)?

[✓] **a)** `fork()`

[✓] **b)** `fopen()`

[ ] **c)** `malloc()`

[✓] **d)** `write()`

[ ] **e)** `fwrite()`

[✓] **f)** `fclose()`

`malloc()` needs not perform a system call when there is enough spare room in the heap. `fwrite()` does not issue a system call unless the buffer has to be flushed.

Question 2.1.2: Which of the following operating system components are *not* part of the kernel in a standard Unix system?

[✓] **a)** C compiler

[ ] **b)** Scheduler

[✓] **c)** Shell

[ ] **d)** Virtual memory manager

[ ] **e)** Device drivers

[✓] **f)** Debugger

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 and that no IO buffering occurs.

```
int main () {
  printf("1");
  if (fork() == 0) {
    printf("2");
  } else {
    pid_t pid; int status;
    if ((pid = waitpid(pid, NULL, 0)) > 0) {
      if (fork() == 0) {
        printf("3");
      } else {
        printf("4");
      }
    }
  }
  printf("5");
}
```

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

☒ **a)** 123455

☐ **b)** 125354

☒ **c)** 124355

☒ **d)** 125345

☒ **e)** 125435

☐ **f)** 152354

Exam number:

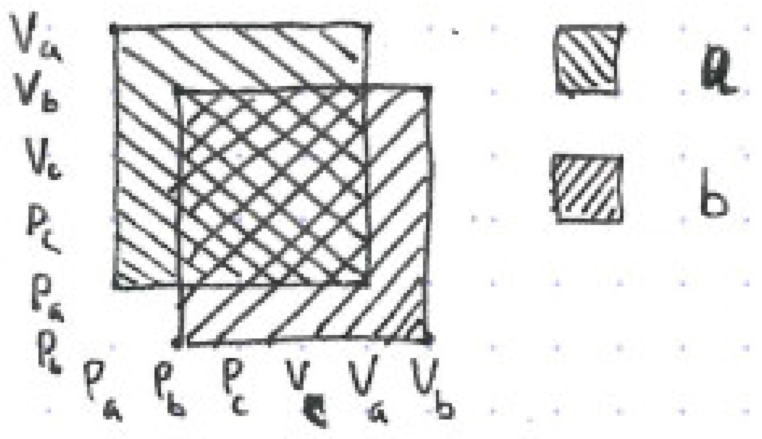
2.2 Short Questions (about 12 %)

Question 2.2.1: Consider two threads performing the following semaphore operations.

```
Initially: a = 1; b = 1; c = 1;

Thread 1:        Thread 2:
P(a);            P(b);
P(b);            P(a);
P(c);            P(c);
V(c);            V(c);
V(a);            V(b);
V(b);            V(a);
```

Draw a process graph with thread 1 along the horizontal axis and thread 2 along the vertical axis, show the forbidden regions, and argue whether this means deadlocks are possible or not. (You are welcome to attached the figure as an image in the handin.)



This graph shows the forbidden regions for semaphores a and b. An execution path can enter the corner where it cannot proceed either right or up, which implies a deadlock.

Question 2.2.2: Consider a system with the following properties:

- Memory is byte-addressed.
- Virtual addresses are 12 bits wide.
- Physical addresses are 11 bits wide.
- The page size is 32 bytes.
- The TLB is 4-way set associative with 4 sets and 16 total entries. Its initial contents are:

Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	1B	15	1	11	00	0	0A	01	1	04	00	1
1	0B	10	0	0F	10	1	1F	2F	0	00	00	0
2	12	34	1	1A	01	0	0D	00	0	00	00	1
3	0B	15	1	0B	02	0	1A	0A	1	1F	3F	1

- The page table contains 12 PTEs:

VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid
11	13	1	2F	01	1	02	3A	0	02	01	1
41	01	0	03	01	0	0B	0D	1	09	3F	1
00	00	1	2A	21	0	13	32	0	03	3F	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: `0x0`

1. Bits of virtual address

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

2. Address translation

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

3. Bits of phys. (if any)

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

Virtual address: `0x4a`

1. Bits of virtual address

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

2. Address translation

Parameter	Value
VPN	2
TLB index	2
TLB tag	0
TLB hit? (Y/N)	Y
Page fault? (Y/N)	N
PPN	0

3. Bits of phys. (if any)

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

Exam number: ___________

Virtual address: `0x2a`

1. Bits of virtual address

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

2. Address translation

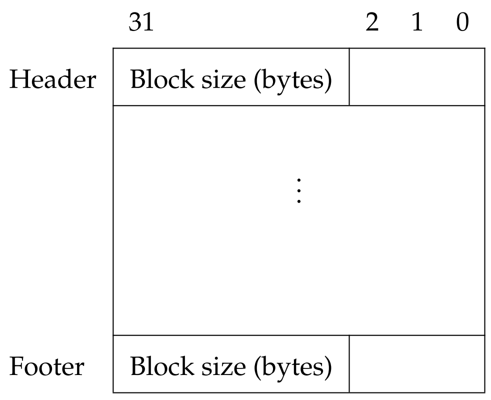
Parameter	Value
VPN	1
TLB index	1
TLB tag	0
TLB hit? (Y/N)	N
Page fault? (Y/N)	Y
PPN	

3. Bits of phys. (if any)

10	9	8	7	6	5	4	3	2	1	0

2.3 Long Questions (about 14 %)

Question 2.3.1: Consider an allocator that uses an implicit free list and immediate coalescing of neighbouring free blocks. 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.

Given the contents of the heap shown on the left, show the new contents of the heap (in the right table) after a call to `free(0x001b010)` is executed. 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 `free`**
0x001b040	0x00000013	0x11
0x001b03c	0x001b611c	0x012c611c
0x001b038	0x001b512c	0x012c512c
0x001b034	0x00000013	0x11
0x001b030	0x0000002b	0x3a (or 0x2a)
0x001b02c	0x001b511c	0x012c511c
0x001b028	0x001b511c	0x012c511c
0x001b024	0x001b511c	0x012c511c
0x001b020	0x001b511c	0x012c511c
0x001b01c	0x001b511c	0x012c511c
0x001b018	0x001b711a	0x001b711a
0x001b014	0x001b601c	0x001b601c
0x001b010	0x001b601c	0x001b601c
0x001b00c	0x0000002b	0x2b (or 0x2a)
0x001b008	0x00000012	0x12 (or 0x13)
0x001b004	0x001b601c	0x001b601c
0x001b000	0x001b511c	0x001b511c
0x001affc	0x00000012	0x3a (or 0x13)

There is an error in the initial heap layout where the block starting at 0x001b00c mistakenly labels the preceding block as allocated (it is actually free). The answers in parens are for when the preceding block is assumed (and changed to be) allocated.

Question 2.3.2: When older operating systems were first ported to multi-processor computers, they tended to be implemented using a single so-called *big kernel lock* that would be held during system calls. This meant that only one process could be executing in kernel mode at once. Give a realistic example of a kind of multi-threaded program that would run well despite the big kernel lock, and one that would not run well. By "run well" we mean that the program is able to take advantage of multi-processor parallelism.

A program that mostly does computation in user space, without many system calls, would not be affected by the big kernel lock. An example of such a program is most kinds of simulations, or a compiler. A program whose threads issue many small system calls would be affected more heavily. For example, a web browser might not run well under a big kernel lock.

Exam number: ____________

3 Computer Networks (about 32 %)

3.1 Application Layer (about 8 %)

Application Layer
Transport Layer
Network Layer
Link Layer
Physical Layer

Question 3.1.1: Consider the five-layer internet stack shown above. What does it mean when we say that HTTP is an Application layer protocol? In this context, what is meant by Encapsulation and why is it needed?

The five layer model divides different systems into different categories, with any system within each category commiting to communicating the the above/below layers in a recognised format. Each layer only needs to implement its own functionality and can depend on the lower layers to implement more basic functionality. HTTP exists at the top layer, intended as the end point for networked applications such as browsers. Therefore it can send messages to other applications by writing messages in a recognised format such as TCP. The lower layers can then begin to route this message through the network. On the originating host each layer will treat the data from the above layer as the body of the message to be passed to the below layer. It will attach a header to this body desribing the message and how to pass it to subsequent layers. When the message arrives at the destination host these headers are one by one read and removed so that it can be correctly passed to the next layer. This process allows each layer to compartmentalise the relevent information so that the data for each layer can be removed without affecting the data to be passed to the next layer. This also means that implementations for each layer can be swapped out without affecting layers either above or below.

Exam number: ___________

Question 3.1.2: You have started a new job and have been tasked by your employer to create and host a new webpage for clients, so that they can better understand the business. Your manager has come to you with two options for how this could be done and has asked you to recommend one. You could either host it on a CDN such as Akamai, or use a peer-to-peer network such as BitTorrent. Discuss the differences between the two options, and what features of each would factor into your recommendation.

A CDN would be beneficial for hosting web-content compared to P2P because of the following reasons:
Availability and Fault Tolerance: CDN ensures that the availability of the content is not dependent on the number of active clients, their bandwidth limitations, P2P configuration settings but rather on the load distribution on the server side in the CDN. This also includes fault tolerant guarantees.
Level of Control: Usage of a CDN would provide a high-level of control to the service to configure the locations of the CDN services, load-balancing of available servers which is not possible to do using a P2P service. Configuration based on geographic locations, distance to clients.
Caching Implications: For content that changes often, CDNs can utilize efficient caching strategies to minimize the cost of propagation of changes. Managing this in a P2P service would require more sophisticated version management and is often impractical to implement without client intervention.
Maintenance Overhead: Usage of CDN allows the hosting service to make all possible optimizations at different layers of the networks stacks (DNS load balancing, usage of transport layer protocols, usage of network layer multi-casting protocols) which is not possible to do by only utilizing application-level P2P service.
Economic Implications: (Extra) Usage of a P2P service would put the burden of network transfer costs on the the client for distributing the web-pages of the company.

Question 3.1.3: In the context of CDNs, what is meant by the 'Enter Deep' and 'Bring Home' design philosophies? Explain the differences between them and some of the reasons for each to be used.

Content Distribution Networks are clusters of resources used to provide massive amounts of data to a large userbase, and is very commonly used in video streaming services. They are a way of locating data nearer to users than by centrally storing it. DNS requests for the data will direct client requests to the closest (in theory) CDN and retreive the data from there. Two broad approaches to where to place these CDNs and how many to use exists. First is Enter Deep, in which case as many as possible small(ish) clusters are built in many different locations. The idea here is to place them as physically close as possible to as many different users as possible. This keeps response times to a minium. However, this approach can be both expensive due to the large amount of hardware needed, and also difficult to main. Maintaining this much hardware can also present its own difficulties. In contrast, Bring Home CDNs use few fewer clusters that are (relatively) cheaper and easier to maintain, but can have longer response times. In both contexts, maintainence can refer to physically ensuring the hardware is still in a runnable condition, but also to the data within each cluster. This is as many CDNs will not be large enough to store the entire catalogue for larger sites such as youttube or netflix. Therefore a cache-like system can be used where only popular or recently viewed shows are stored in many locations and others are retreived more remotely when necessary.

Exam number:

3.2 Transport Layer (about 8 %)

Question 3.2.1: Within the context of Reliable Data Transfer, Explain what *Go-Back-N* is. How does it differ from *Selective Repeat*?

In reliable data transfer we need to track each packet to check that has been delivered. This is usually done through a counter that is attached to each sent packet. In GBN packets are aknowledged in order, so if a packet that with a higher sequence number is recieved only the most recent contiuos packet is acknowledged. Any out of order packets are discarded, whilst in order ones are passed to the application layer. This ensures packets are recieved in order without lots of buffering/caching. The sender can only send a given window size of packets, with the window consisting of unacknowledged packets. If once a packet is acknowledged in order, the window moves along. Once each packet is sent a timer is started and if it finishes before the acknowledgment is recievd it and all subsequent packets are sent again. Selective repeat differsn in this regard as each packet is tracked individually and only negatively acknowledged or timed out packets are resent. selective repeat also differs in that it allows packets to be acknowledged out of order and so utilises caching of out of order packets in the reciever.

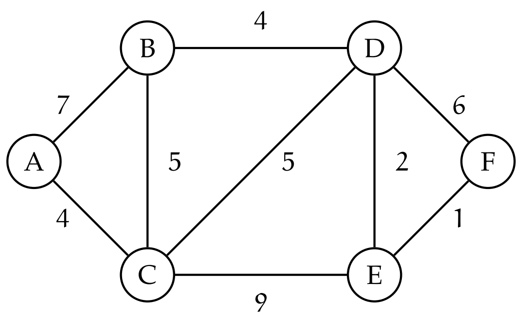
Question 3.2.2: TCP implements *Congestion Control* and *Flow Control*. These terms are often used interchangeably as they have similar results. What result is this and why are both needed?

As a reliable service TCP minimised packet loss. There are two points of packet loss, either in the Network or the reciever. Both can be averted by reducing the rate at which packets are sent, and so both congestion control and flow control have the effect of throttling the sender. Flow control tries to not overwhelm the reciever by only sending as many messages as it can recieve. This is done by requesting that the reciever communicates how much space it has to buffer messages, and then ensuring the sender does not send more messages than that. This is communicated by the reciever appending a window size to all messages back to the sender. Congestion control also throttles the sender by starting to only send a few messages, then ramping up the speed of commucation until packet loss occurs. This is taken as a sign of congestion and so the rate of sending will be halved, only to gradually increse again. By this method we can try to use as much of the network as possible whilst being approximately fair to other users. Without both of these systems packets would be lost more often, networks would become congested, and communications would be slowed.

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	7,A	4,A	∞	∞	∞
1	A,C	7,A	4,A	9,C	13,C	∞
2	A,C,B	7,A	4,A	9,C	13,C	∞
3	A,C,B,D	7,A	4,A	9,C	11,D	15,D
4	A,C,B,D,E	7,A	4,A	9,C	11,D	12,E
5	A,C,B,D,E,F	7,A	4,A	9,C	11,D	12,E

Forwarding table on node A:

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

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

Exam number:

3.4 Network Security (about 8 %)

Question 3.4.1: Describe the properties that are needed to make a cryptographic hash function (compared to normal hash functions).

Shares same propperties as regular hash functions, always computes fixed length output. Must be computationally infeasible to derive input from the output without some key. Must be improbable that any two messages would have the same output. Must be infeasible to brute force through inputs to get output.

Question 3.4.2: What does it mean that the symmetric key is a session-key? Where would this be used and why is this important?

That it is a session-key, means that it is only used for a specific session. A new connection should use a new randomly generated key. Key reuse can result in connection being vulnerable to replay attacks or brute-force if it limits the key space.

4 Machine architecture (about 24 %)

4.1 Assembler programming (about 14 %)

Consider the following program written in X86prime-assembler.

```
.START:
    subq $8, %rsp
    movq %r11, (%rsp)
    movq $1, %r8
    jmp .L2
.L1:
    leaq 8(%rdi, %rax, 8), %r11
    movq %rcx, (%r11)
    addq $1, %r8
.L2:
    cbl %rsi, %r8, .L4
    leaq (%rdi, %r8, 8), %r11
    movq (%r11), %rcx
    leaq -1(%r8), %rax
.L3:
    cbg $0, %rax, .L1
    leaq (%rdi, %rax, 8), %r11
    movq (%r11), %rdx
    cbge %rcx, %rdx, .L1
    leaq 8(%rdi, %rax, 8), %r11
    movq %rdx, (%r11)
    subq $1, %rax
    jmp .L3
.L4:
    movq (%rsp), %r11
    addq $8, %rsp
    ret %r11
```

Exam number:

Question 4.1.1: Identify the control structures of the program (e.g. loops, conditionals, and function calls).

Vi har en ydre løkke fra L26 -> L29 og inden i en indre løkke fra L25 -> L29.
Endvidere er der et betinget exit (break) fra den indre løkke. Alternativt kan man opfatte det som om den indre løkke har en kompliceret betingelse. Det bliver tydeligere nedenfor prolog = kode for at opsætte stakramme til funktionen
epilog = kode for at nedtage stakramme for funktionen

Se side **??** for detaljer.

Question 4.1.2: Specify which registers are used for pointers. How can you see this? What are the operations?

%rsp, %rdi, %r11

Question 4.1.3: Which registers contains functions arguments and what are their type? Describe how you identified this.

%rdi, %rsi

Exam number:

Question 4.1.4: 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;
  }
}
```

Se side 23 for detaljer.

Question 4.1.5: Descripe shortly the functionality of the program.

Funktionen er insertionsort.

Den leverede tabel sorteres gradvist. Den ydre løkke gennemløber alle elementerne et for et og udvider den sorterede del af tabellen. Den indre løkke placerer hvert element det rigtige sted blandt de sorterede elementer.

Exam number:

4.2 Pipeline and instruction level parallelism (about 10 %)

Consider the following fragment of a program

```
Loop:    movq  (%r12),%r15
         movq  %r15,(%r13)
         addq  $8,%r12
         addq  $8,%r13
         cbne  $0,%r15,Loop
```

The following execution graph (afviklingsplot) illustrates the execution sequence of two iterations on a simple 5-step pipeline machine, where the instructions have to wait for all operants to be ready in the D-step:

```
Loop:    movq  (%r12),%r15     FDXMW
         movq  %r15,(%r13)      FDDXMW
         addq  $8,%r12           FFDXMW
         addq  $8,%r13             FDXMW
         cbne  $0,%r15,Loop         FDXMW
Loop:    movq  (%r12),%r15           FDXMW
         movq  %r15,(%r13)            FDDXMW
         addq  $8,%r12                  FDXMW
         addq  $8,%r13                   FDXMW
         cbne  $0,%r15,Loop               FDXMW
```

Unless explicitly stated, the following questions assumes that all jumps are correctly predicted and all access to the memory results in a cache hit.

Question 4.2.1:

It is possible to make a change to the pipeline such that values that are written to the memory should now be ready before the X-step instead of the D-step as shown above. This can affect instruction 2 and 7 in the above execution graph.

How much would this change reduce the execution of the above loop?

Ja, der er ikke brug for en stall i `movq %r15,(%r13)`.

Udførelsestiden er således reduceret med (6-5)/6 = 1/6.

Question 4.2.2: Make an execution diagram, showing the execution of above code on a two-way superscalar machine, as presented in the course note. (See on the title "Eksempel: Superskalar".) Explain shortly any assumptions you may have to make.

	Code	Timing																	
	`Loop:`																		
1	`movq (%r12),%r15`																		
2	`movq %r15,(%r13)`																		
3	`addq $8,%r12`																		
4	`addq $8,%r13`																		
5	`cbne $0,%r15,Loop`																		
	`Loop:`																		
6	`movq (%r12),%r15`																		
7	`movq %r15,(%r13)`																		
8	`addq $8,%r12`																		
9	`addq $8,%r13`																		
10	`cbne $0,%r15,Loop`																		

Der accepteres to forskellige svar her. Det skyldes beskrivelsen af superskalare processorer som findes i noten om afviklingsplot, og så den beskrivelse der er givet i forelæsninger og på slides og i vejledning over discord i løbet af kurset kan give anledning til forskellige fortolkninger.
Noten om afviklingsplot angiver at tilgang til lageret kræver to clock-perioder i 'M'-trinnet. Dette er egentlig en fejl, sådan at forstå, at selvom en sådan maskine kunne bygges, så vil det ikke være det typiske valg. I virkelighedens maskiner er der kun en clock-periode i 'M' trinnet - hvilket var hvad undervisningen i efteråret i øvrigt antog.
Se 25 for plots.
Begge løsninger og forståelse af ændringer til en superskalar maskine giver kredit.

Exam number: ________

Below you see the execution graph for the code on a more realistic modern 3-way out-of-order machine, where access to the primary cache is extended over 3 clock-periods and where values that are written to memory should only be ready in the X-step:

```
Loop:   movq (%r12),%r15     F----QAM--C
        movq %r15,(%r13)     F----Q-AM-VC
        addq $8,%r12         F----QX----C
        addq $8,%r13          F----QX---C
        cbne $0,%r15,Loop     F----Q---B-C
Loop:   movq (%r12),%r15       F----QAM--C
        movq %r15,(%r13)       F----Q-AM-VC
        addq $8,%r12           F----QX----C
        addq $8,%r13            F----QX---C
        cbne $0,%r15,Loop       F----Q---B-C
```

(The machine presented in the note has a 2-cycle fetch redirection delay when predicting a taken branch. As you can see above, the machine used here has only a single cycle delay. This is not an error)

Question 4.2.3: How does this machine compare in clock-periods per iteration to the simple 5-step pipeline machine presented in the beginning of the section?

Her bruges 2 clock perioder per iteration mod 6 i den allerførste del af opgaven.

Question 4.2.4: Assume that 10% of jumps is predicted wrongly, while the rest (90%) is predicted correctly.

How many clock-periods does every iteration of the loop takes on average? Explain how you found your answer.

Her er det nødvendigt at fastlægge omkostningen for et gennemløb med en fejlforudsigelse. Efter en fejlforudsigelse kan den efterfølgende F fase først ske efter B-fasen for hop instruktionen, da det er på det tidspunkt fejlforudsigelsen detekteres.

```
Loop: movq (%r12),%r15 F----QAM--C
      movq %r15,(%r13) F----Q-AM-VC
      addq $8,%r12 F----QX----C
      addq $8,%r13 F----QX---C
      cbne $0,%r15,Loop F----Q---B-C
Loop: movq (%r12),%r15 F----QAM--C
                    ....
```

Man kan se at hver 10nde gennemløb (som ses ovenfor) tager 11 clock perioder. De resterende 9 gennemløb tager stadig 2 cykler.
Den gennemsnitlige tid for et gennemløb bliver da (11+9*2)/10 = 2.9 clock perioder.

Vedr. 4.1.4
Vi starter med at omskrive til pseudo-C med goto og registernavne:

```
    prolog.
    r8 = 1;
    goto L24
L26:
    rdi[rax] = rcx
    ++rax
L24:
    if rsi < r8 goto L29
    rcx = rdi[r8]
    rax = r8 - 1
L25:
    if 0 > rax goto L26
    rdx = rdi[rax]
    if rcx > rdx goto L26
    rdi[rax] = rdx
    --rax
    goto L25
L29:
    epilog.
```

Den ydre løkke starter ved L24 (kan vi se fordi der øverst hoppes til L24) Vi flytter nu blokken fra L26 til nederst i den ydre løkke, således at L24 ligger øverst - det gør strukturen mere indlysende og får løkkebetingelsen op i toppen:

```
    prolog.
    r8 = 1;
L24:
    if rsi < r8 goto L29
    rcx = rdi[r8]
    rax = r8 - 1
L25:
    if 0 > rax goto L26
    rdx = rdi[rax]
    if rcx > rdx goto L26
    rdi[rax] = rdx
    --rax
    goto L25
L26:
    rdi[rax] = rcx
    ++rax
    goto L24
L29:
    epilog.
```

Fortsætter på næste side...

Vi kan nu omskrive den ydre løkke:

```
    prolog.
    r8 = 1;
    while (rsi >= r8) {
        rcx = rdi[r8]
        rax = r8 - 1
L25:
        if 0 > rax goto L26
        rdx = rdi[rax]
        if rcx > rdx goto L26
        rdi[rax] = rdx
        --rax
        goto L25
L26:
        rdi[rax] = rcx
        ++rax
    }
    epilog.
```

Så den indre:

```
    prolog.
    r8 = 1;
    while (rsi >= r8) {
        rcx = rdi[r8]
        rax = r8 - 1
        while (0 <= rax) {
            rdx = rdi[rax]
            if (rcx > rdx) break;
            rdi[rax] = rdx
            --rax
        }
        rdi[rax] = rcx
        ++rax
    }
    epilog.
```

Og endeligt i rigtig C:

```
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;
  }
}
```

Det er også fint at give en løsning der stadig bruger “break” i stedet for at omskrive til en mere kompliceret betingelse for den indre løkke.
Bemærk at læsningen fra array[j] i den indre løkke sker to gange i C kildeteksten, men af compileren er blevet optimeret til en enkelt tilgang i maskinkoden.

Vedr. 4.2.2

Hvis man følger noten om afviklingsplot får man:

```
Loop:   movq (%r12),%r15    FDXMMW
        movq %r15,(%r13)    FDXXXMM
        addq $8,%r12         FDXXW
        addq $8,%r13         FDDDXW
        cbne $0,%r15,Loop     FDDX
Loop:   movq (%r12),%r15       FFDXMMW
        movq %r15,(%r13)       FFDXXXMM
        addq $8,%r12             FDXXW
        addq $8,%r13             FDDDXW
        cbne $0,%r15,Loop         FDDX
```

Hvis man derimod følger den øvrige undervisning får man:

```
Loop:   movq (%r12),%r15    FDXMW
        movq %r15,(%r13)    FDXXM
        addq $8,%r12         FDXW
        addq $8,%r13         FDDXW
        cbne $0,%r15,Loop     FDX
Loop:   movq (%r12),%r15       FDXMW
        movq %r15,(%r13)       FDXXM
        addq $8,%r12            FDXW
        addq $8,%r13            FDDXW
        cbne $0,%r15,Loop        FDX
```

Vi antager i begge tilfælde at et korrekt forudsagt taget hop blot forsinker F med 1 clock.