# Exam questions

Forberedelse af Jesper Bertelsen

Indholdsfortegnelse

[Exam questions 1](#_Toc137804635)

[Section 1. Basics, representing & manipulating information. 3](#_Toc137804636)

[Section 1.1. List all 5 basic logic gates and give the condition when the output is 1. 3](#_Toc137804637)

[Section 1.2. How do you calculate the number of bits needed to represent a positive integer? 3](#_Toc137804638)

[Section 1.3. How are negative integers represented in binary? Show it with 3 bits in the table below! Generally, how is the representable range different between signed and unsigned for a given length? 3](#_Toc137804639)

[Section 1.4. What is overflow and which operations are vulnerable to it? 4](#_Toc137804640)

[Section 1.5. Briefly describe the ADD operation for two signed integers! 4](#_Toc137804641)

[Section 1.6. Briefly describe the ADD operation for a signed and an unsigned integer! 4](#_Toc137804642)

[Section 1.7. How do compilers speed up multiplications? 4](#_Toc137804643)

[Section 1.8. What is the “word size” of a computer? What are the typical values? 4](#_Toc137804644)

[Section 1.9. What is “big endian” and “little endian” byte ordering and what’s the difference? 4](#_Toc137804645)

[Section 1.10. Briefly describe how fractional numbers are represented in binary. What is the decimal value of 1010.001? 4](#_Toc137804646)

[Section 1.11. Briefly describe how the IEEE Floating Point standard represents fractional numbers! 4](#_Toc137804647)

[Section 1.12. Why is it impossible for any finite size floating point standard to represent arbitrary fractional values precisely? 4](#_Toc137804648)

[Section 1.13. As we go further from zero, how does the distribution of representable numbers change when using the IEEE floating point standard? Why? 4](#_Toc137804649)

[Section 1.14. What are the rounding modes of IEEE floating point numbers? Give the rounded value of 2.5 with each mode! 4](#_Toc137804650)

[Section 2. Machine level programming 4](#_Toc137804651)

[Section 2.1. What is the difference between Instruction Set Architecture and Microarchitecture? Give an example of each. 4](#_Toc137804652)

[Section 2.2. What is the difference between machine code and assembly? 4](#_Toc137804653)

[Section 2.3. Briefly describe what steps are necessary to turn C source code to an executable program. 4](#_Toc137804654)

[Section 2.4. What is the main difference between the -O1, -O2, -O3 and -Os optimization flags of gcc? What data types and operations are available in assembly? 4](#_Toc137804655)

[Section 2.5. What is the role of number 8 in the following memory addressing mode and what is it generally used for? movq 8(%rbp),%rdx 4](#_Toc137804656)

[Section 2.6. How does the compiler compile a switch expression and why is it better than a sequence of if-else? What are limitations of using switch, from the programmer’s perspective? 4](#_Toc137804657)

[Section 2.7. What is the stack, and where does the stack pointer %rsp point to? What do the push and pop instructions do? 4](#_Toc137804658)

[Section 2.8. Briefly describe how the stack is used when a function calls another function, which returns data to the caller. 4](#_Toc137804659)

[Section 2.9. Why does it take a little more time to call a function with 9 parameters than with 5? 4](#_Toc137804660)

[Section 2.10. Briefly describe what a stack frame is and list at least 3 components of a Linux stack frame! 4](#_Toc137804661)

[Section 2.11. How are C arrays laid out in the memory? How many bytes are allocated when you write int arr[13];? Why is arr[i] equivalent to \*arr+i? 4](#_Toc137804662)

[Section 2.12. What does it mean that C stores multidimensional arrays in a row-major order? 4](#_Toc137804663)

[Section 2.13. How is a C structure represented in memory? What are the best practices when declaring a structure, and why? What is alignment and how does it change the memory allocated for a structure? 4](#_Toc137804664)

[Section 2.14. What are XMM registers and what are they used for? 4](#_Toc137804665)

[Section 2.15. Briefly describe the Single Instruction Multiple Data (SIMD) technique and what it’s good for! What kind of registers are used with SIMD instructions? 5](#_Toc137804666)

[Section 2.16. What is an exceptional control flow? Briefly describe it and give at least 3 examples. 5](#_Toc137804667)

[Section 2.17. What is the difference between synchronous and asynchronous exceptions? Give at least two examples of both. 5](#_Toc137804668)

[Section 2.18. What is a process, and what logical abstractions does it provide? Why are these abstractions important? How does the operating system identify the running processes? 5](#_Toc137804669)

[Section 2.19. Briefly describe how a single core CPU runs multiple processes concurrently. Explain the concept of context switching, and the important design decisions about it. 5](#_Toc137804670)

[Section 2.20. What states can a process be in, and how does it transition between them? Why does the C fork() call return twice, and how do we know which branch we are in? 5](#_Toc137804671)

[Section 2.21. What are the similarities and differences between the parent and child process after using fork()? 5](#_Toc137804672)

[Section 2.22. What is a process graph? What is a zombie process, why is it bad and how is it handled? 5](#_Toc137804673)

[Section 2.23. What is the fundamental issue with a server implemented as an iterative process? How do we solve it? 5](#_Toc137804674)

[Section 2.24. List and briefly describe 3 major problems that arise with concurrent programming What are the 3 ways of writing concurrent programs? Briefly describe each! 5](#_Toc137804675)

[Section 2.25. Give at least 2 advantages and 2 disadvantages of a process-based concurrent server implementation! 5](#_Toc137804676)

[Section 2.26. Give at least 2 advantages and 2 disadvantages of an event-based concurrent server implementation! 5](#_Toc137804677)

[Section 2.27. Give at least 2 advantages and 2 disadvantages of a thread-based concurrent server implementation! 5](#_Toc137804678)

[Section 2.28. What is a thread and how is it different than a process? 5](#_Toc137804679)

[Section 2.29. When do we call two threads concurrent? When are they actually running at the same time? 5](#_Toc137804680)

[Section 2.30. Briefly describe the process graph and trajectories on it. 5](#_Toc137804681)

[Section 2.31. What is a critical section and unsafe region in a process graph? What is a “safe trajectory” on the process graph? 5](#_Toc137804682)

[Section 2.32. What is a semaphore? What problem does it solve and how? What is the difference between a mutex and a semaphore? Briefly describe both. What is a thread safe function? List and briefly describe the 4 classes of thread-unsafe functions! Briefly describe Amdahl’s law and give a numeric example of it! 5](#_Toc137804683)

[Section 3. Central processing unit 6](#_Toc137804684)

[Section 3.1. What is the Arithmetic/Logic unit? 6](#_Toc137804685)

[Section 3.2. What is the purpose of the hardware control language? 6](#_Toc137804686)

[Section 3.3. What is a logic gate and what are the standard logic gate types? 6](#_Toc137804687)

[Section 3.4. What is instruction encoding? 6](#_Toc137804688)

[Section 3.5. What is a register? 6](#_Toc137804689)

[Section 3.6. What is the purpose of the Jump instruction? 6](#_Toc137804690)

[Section 3.7. What are the six general instruction processing stages? Briefly describe them. 6](#_Toc137804691)

[Section 3.8. What is the benefit of pipelining? 6](#_Toc137804692)

[Section 3.9. What is the main limitation of pipelining? 6](#_Toc137804693)

[Section 3.10. What are the two types of hazards in pipelining and why do they happen? 6](#_Toc137804694)

[Section 4. Memory management 6](#_Toc137804695)

[Section 4.1. What is the difference between volatile and nonvolatile memory? 6](#_Toc137804696)

[Section 4.2. Why does it take a long time to read or write a value in the main memory? 6](#_Toc137804697)

[Section 4.3. List 5 components from a disk drive and briefly describe them. 6](#_Toc137804698)

[Section 4.4. What’s the easiest way for a HDD manufacturer to increase the capacity of a drive? 6](#_Toc137804699)

[Section 4.5. When a HDD needs to read an address, what takes up most of the time and what is free? 6](#_Toc137804700)

[Section 4.6. What is the advantage of logical disk blocks? 6](#_Toc137804701)

[Section 4.7. Why does the CPU not wait for a disk read? How does it get the data instead? 6](#_Toc137804702)

[Section 4.8. Name a few advantages and disadvantages of SSD over HDD. What can we do about the disadvantages? 6](#_Toc137804703)

[Section 4.9. Define temporal and spatial locality. How do they help with slow memory and disk access times? 6](#_Toc137804704)

[Section 4.10. What is a Stride-1 access pattern, and what type of locality does it exploit? 6](#_Toc137804705)

[Section 4.11. Briefly describe the memory hierarchy (2-3 sentences, focus on the most important properties) 6](#_Toc137804706)

[Section 4.12. What is a data cache in general? What is a cache miss and what happens when it occurs? 6](#_Toc137804707)

[Section 4.13. What is the difference between the 3 cache miss types? 6](#_Toc137804708)

[Section 4.14. Approximately how many clock cycles does it take to access data from the following memories: registers, CPU caches, main memory, disk, network storage. 6](#_Toc137804709)

[Section 4.15. How do we calculate the cache size? 6](#_Toc137804710)

[Section 4.16. How does the cache controller interpret a memory address? 6](#_Toc137804711)

[Section 4.17. What is the “tag” in a cache line? 6](#_Toc137804712)

[Section 4.18. How do we read data from a direct mapped cache (step by step)? At which step do we know for sure if we have a cache hit or miss? 6](#_Toc137804713)

[Section 4.19. What is the “memory mountain”? What is the main difference between physical and virtual memory addressing? What are the main advantages of virtual memory? Why is virtual memory considered a cache? What is a Page Table and a Page Fault? How are Page Faults handled? What is Memory Thrashing and why do poorly written programs cause it? Which one is larger, virtual or physical memory (in number of addressable bytes)? 7](#_Toc137804714)

[Section 4.20. What problem is solved by the Translation Lookaside Buffer (TLB)? What is the motivation for using multi-level page tables? 7](#_Toc137804715)

[Section 4.21. When using multi-level page tables, how does the memory manager interpret the virtual address? Illustrate your answer with a drawing! 7](#_Toc137804716)

[Section 4.22. In what cases do we get a Segmentation Fault? What is memory mapping and what is the swap file used for? 7](#_Toc137804717)

[Section 4.23. What is the heap and what functions can a C programmer use to interact with it and what do they do exactly? 7](#_Toc137804718)

[Section 4.24. How does *malloc* work in general, and what constraints must be satisfied by its implementations (allocators)? 7](#_Toc137804719)

[Section 4.25. What rules must be followed by a C programmer when using *malloc*? What happens when the rules are violated? 7](#_Toc137804720)

[Section 4.26. Why is it challenging to make a good allocator (e.g. what goals does it have and why are they conflicting)? 7](#_Toc137804721)

[Section 4.27. What is internal and external fragmentation in memory allocation? How does *free()* know how many bytes to deallocate when it receives a pointer? 7](#_Toc137804722)

## Section 1. Basics, representing & manipulating information.

### List all 5 basic logic gates and give the condition when the output is 1.

The 5 basic gates goes like this:

And, or, nor, nand & xor.

For the output to be 1, the following conditions is met.

And: x & y is 1

Or: x or y is 1

Nor: x & y is not 1,

Nand: x or y is not 1

Xor: is 1 & y is not 1 or reverse

### How do you calculate the number of bits needed to represent a positive integer?

A positive integer uses as many bits as a negative integer if they are both signed.

If so, then the MSB will the sign bit, 0 for positive, 1 for negative.

represent the maximum amounts of outputs available with k amounts of bits. Taking into account that 0 is also a value, the maximum number will then be .

Ex.

============================

as a maximum value.

============================

### Et billede, der indeholder tekst, skærmbillede, nummer/tal, Font/skrifttype Automatisk genereret beskrivelseHow are negative integers represented in binary? Show it with 3 bits in the table below! Generally, how is the representable range different between signed and unsigned for a given length?

Let’s look at the example showing signed an unsigned:

For 4 bits, looking at 0 -> 7, the signed values are the same.

There from an onwards, the signed differs from the unsigned.

Were the unsigned needs 3 bits to represent 7, the signed needs 4 bits, making the unsigned able to represent 15 with 4 bits, where the signed represent .   
Let’s fill out the table.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit pattern | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| Unsigned value | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Signed value | 0 | 1 | 2 | 3 | - 4 | - 3 | - 2 | - 1 |

Figure 1: Unsigned / signed representation

### What is overflow and which operations are vulnerable to it?

Overflow is when the amounts of bits acquired to represent a value is less than the amounts of bits needed to represent a given value. This can happen from operations.

With add operations of two max values, with let’s say 4 bits each unsigned, they each represent, 15 ( 1111 ). Adding those two together leads to a carry to the 5th bit.

If only 4 bits are acquired, this will result in overflow.

Adding isn’t as vulnerable to overflow, as this operation can only lead to an overflow of +1 bit.

When multiplying, the above example could have looked like.

We see that the amounts of bits in times the amounts of bits in y equals the amount of bits in . Then an overflow of +4 bits occurred in this example.

### Briefly describe the ADD operation for two signed integers!

Adding two signed integers is nearly the same as with adding two unsigned. If the *nth* bit of both numbers is equal to 1, then the next more significant bit gets a carry, let’s say the th bit.

With addition of signed integers, if the addition of the sum of the two results in having 1 more bit than what the two added numbers separately had, then the sign bit becomes the new MSB.

Ex.

### Briefly describe the ADD operation for a signed and an unsigned integer!

Kan ikke huske. Men vil tænke at man caster den som en positiv signed int og så laver addition af 2 signed integers.

### How do compilers speed up multiplications?

Multiplication is resource heavy so a way to avoid multiplying when doing multiplication is to use left shifting bits and then adding shifts together.

Say you want 4 x 4 which is 16. Instead of multiplying you left shift two bits.

Say if we were two multiply 4 with 6, then it would result in 24.

But we can add shifts together:

So .

### What is the “word size” of a computer? What are the typical values?

Jeg er lidt usikker på hvad det præcis betyder, men det bruges i forbindelse med at snakke om hvor mange bytes eller bits der er påkrævet.

### What is “big endian” and “little endian” byte ordering and what’s the difference?

Big endian and little endian is two ways of interpretting bytes. I can’t say which has which way of interpretting them, but one of them interprets the Most Significant Byte as the one on the left and one of them interprets the MSB as the one from the right. It’s from back when computers were a new thing so one decided to do it their way and the other in the opposite way. I think it’s between Linux and Windows, but I am not sure of that.

### Briefly describe how fractional numbers are represented in binary. What is the decimal value of 1010.001?

The bits to the left of a fractional number you read as with non fractional numbers.

The values to the right of a fractional number you read as , where *n* is the bit’s order from the right of the dot where the first number to the left of the dot is .

You then add that number to the decimal representation of the binary value.

The decimal value of then is:

===================================

===================================

### Briefly describe how the IEEE Floating Point standard represents fractional numbers!

### Why is it impossible for any finite size floating point standard to represent arbitrary fractional values precisely?

### As we go further from zero, how does the distribution of representable numbers change when using the IEEE floating point standard? Why?

### What are the rounding modes of IEEE floating point numbers? Give the rounded value of 2.5 with each mode!

## Section 2. Machine level programming

#### What is the difference between Instruction Set Architecture and Microarchitecture? Give an example of each.

#### What is the difference between machine code and assembly?

Machine code uses bits of instructions to tell the computer what to do. The assembly converts these instructions into symbolic readable text.

#### Briefly describe what steps are necessary to turn C source code to an executable program.

To turn C source code into an executable program, a compiler must compile the C code to machine level code, that the computer understands.

#### What is the main difference between the -O1, -O2, -O3 and -Os optimization flags of gcc? What data types and operations are available in assembly?

(arg) fortæller om addressen, pointeren.

D(R) fortæller om en adresse forskudt med D bytes.

#### What is the role of number 8 in the following memory addressing mode and what is it generally used for? movq 8(%rbp),%rdx

Mov is a move command and q stands for Quad word size, where one Word size is 2 bytes, making movq move a word size of 8 bytes.

In this example it follows the instruction movq D(r), destination, where R is the register location and D is the offset from it. The (R) points to the register.

The instruction then moves the address of %rbp + I guess positive 8 bytes, and puts the address into %rbx.

Et billede, der indeholder tekst, skærmbillede, menu, dokument

Automatisk genereret beskrivelse

#### How does the compiler compile a switch expression and why is it better than a sequence of if-else? What are limitations of using switch, from the programmer’s perspective?

Let’s have a look at both switch and if else conditions.

The switch statement makes an array of locations to jump to. An index then keeps count of which address to fetch from its jump list array.

Then a comparison is met, checking if the index is not in the array, and if the condition is not met, then program goes to the jump location at its index.

It seems to me, that the if statement checks whether the conditions is met.

What I interpret as the difference, is that the if statement looks for conditions to be met and the switch only checks for when the conditions is not met.

If statements makes jumps, when the condition is met. Switch jumps to the default, when the condition is met otherwise it jumps to address of its index.

The switch case doesn’t have to check for the index to be equal to its destination, it just jumps to it’s destination.

I think that this one, is hard to explain.

Et billede, der indeholder tekst, skærmbillede, menu, Font/skrifttype

Automatisk genereret beskrivelse

#### 

#### What is the stack, and where does the stack pointer %rsp point to? What do the push and pop instructions do?

The holds the address of the “Top element of a stack.

Popping take

#### Briefly describe how the stack is used when a function calls another function, which returns data to the caller.

#### Why does it take a little more time to call a function with 9 parameters than with 5?

#### Briefly describe what a stack frame is and list at least 3 components of a Linux stack frame!

#### How are C arrays laid out in the memory? How many bytes are allocated when you write int arr[13];? Why is arr[i] equivalent to \*arr+i?

#### What does it mean that C stores multidimensional arrays in a row-major order?

#### How is a C structure represented in memory? What are the best practices when declaring a structure, and why? What is alignment and how does it change the memory allocated for a structure?

#### What are XMM registers and what are they used for?

#### Briefly describe the Single Instruction Multiple Data (SIMD) technique and what it’s good for! What kind of registers are used with SIMD instructions?

#### What is an exceptional control flow? Briefly describe it and give at least 3 examples.

#### What is the difference between synchronous and asynchronous exceptions? Give at least two examples of both.

#### What is a process, and what logical abstractions does it provide? Why are these abstractions important? How does the operating system identify the running processes?

#### Briefly describe how a single core CPU runs multiple processes concurrently. Explain the concept of context switching, and the important design decisions about it.

#### What states can a process be in, and how does it transition between them? Why does the C fork() call return twice, and how do we know which branch we are in?

#### What are the similarities and differences between the parent and child process after using fork()?

#### What is a process graph? What is a zombie process, why is it bad and how is it handled?

#### What is the fundamental issue with a server implemented as an iterative process? How do we solve it?

#### List and briefly describe 3 major problems that arise with concurrent programming What are the 3 ways of writing concurrent programs? Briefly describe each!

#### Give at least 2 advantages and 2 disadvantages of a process-based concurrent server implementation!

#### Give at least 2 advantages and 2 disadvantages of an event-based concurrent server implementation!

#### Give at least 2 advantages and 2 disadvantages of a thread-based concurrent server implementation!

#### What is a thread and how is it different than a process?

#### When do we call two threads concurrent? When are they actually running at the same time?

#### Briefly describe the process graph and trajectories on it.

#### What is a critical section and unsafe region in a process graph? What is a “safe trajectory” on the process graph?

#### What is a semaphore? What problem does it solve and how? What is the difference between a mutex and a semaphore? Briefly describe both. What is a thread safe function? List and briefly describe the 4 classes of thread-unsafe functions! Briefly describe Amdahl’s law and give a numeric example of it!

## Section 3. Central processing unit

##### What is the Arithmetic/Logic unit?

##### What is the purpose of the hardware control language?

##### What is a logic gate and what are the standard logic gate types?

##### What is instruction encoding?

##### What is a register?

##### What is the purpose of the Jump instruction?

##### What are the six general instruction processing stages? Briefly describe them.

##### What is the benefit of pipelining?

##### What is the main limitation of pipelining?

##### What are the two types of hazards in pipelining and why do they happen?

## Section 4. Memory management

###### What is the difference between volatile and nonvolatile memory?

###### Why does it take a long time to read or write a value in the main memory?

###### List 5 components from a disk drive and briefly describe them.

###### What’s the easiest way for a HDD manufacturer to increase the capacity of a drive?

###### When a HDD needs to read an address, what takes up most of the time and what is free?

###### What is the advantage of logical disk blocks?

###### Why does the CPU not wait for a disk read? How does it get the data instead?

###### Name a few advantages and disadvantages of SSD over HDD. What can we do about the disadvantages?

###### Define temporal and spatial locality. How do they help with slow memory and disk access times?

###### What is a Stride-1 access pattern, and what type of locality does it exploit?

###### Briefly describe the memory hierarchy (2-3 sentences, focus on the most important properties)

###### What is a data cache in general? What is a cache miss and what happens when it occurs?

###### What is the difference between the 3 cache miss types?

###### Approximately how many clock cycles does it take to access data from the following memories: registers, CPU caches, main memory, disk, network storage.

###### How do we calculate the cache size?

###### How does the cache controller interpret a memory address?

###### What is the “tag” in a cache line?

###### How do we read data from a direct mapped cache (step by step)? At which step do we know for sure if we have a cache hit or miss?

###### What is the “memory mountain”? What is the main difference between physical and virtual memory addressing? What are the main advantages of virtual memory? Why is virtual memory considered a cache? What is a Page Table and a Page Fault? How are Page Faults handled? What is Memory Thrashing and why do poorly written programs cause it? Which one is larger, virtual or physical memory (in number of addressable bytes)?

###### What problem is solved by the Translation Lookaside Buffer (TLB)? What is the motivation for using multi-level page tables?

###### When using multi-level page tables, how does the memory manager interpret the virtual address? Illustrate your answer with a drawing!

###### In what cases do we get a Segmentation Fault? What is memory mapping and what is the swap file used for?

###### What is the heap and what functions can a C programmer use to interact with it and what do they do exactly?

###### How does *malloc* work in general, and what constraints must be satisfied by its implementations (allocators)?

###### What rules must be followed by a C programmer when using *malloc*? What happens when the rules are violated?

###### Why is it challenging to make a good allocator (e.g. what goals does it have and why are they conflicting)?

###### What is internal and external fragmentation in memory allocation? How does *free()* know how many bytes to deallocate when it receives a pointer?