# Eksamen 2023

Af Jesper Bertelsen, AU-ID: au689481, Studentnr: 202204617

Indholdsfortegnelse

[Eksamen 2023 1](#_Toc138068540)

[Theoretical questions 1](#_Toc138068541)

[1. Define temporal and spatial locality. What is a Stride-1 access pattern, and what type of locality does it exploit? (4 points) 1](#_Toc138068542)

[2. Briefly describe how fractional numbers are represented in binary. Can you represent 515.7 with 4 bits for the decimals? If not, what are the two closest numbers to it you can represent with 4 decimal bits? (4 points) 2](#_Toc138068543)

[3. How do you calculate the number of bits needed to represent a positive integer? Represent the following integers (in hexadecimal) in binary: 0x50, 0x33, 0x341 Calculate: 0x50 & 0x33, 0x341 ^ 0x33 (4 points). 2](#_Toc138068544)

[4. What is “big endian” and “little endian” byte ordering and what’s the difference? (3 points) 3](#_Toc138068545)

[5. What are the six general instruction processing stages in a CPU studied in class? Briefly describe them. (4 points) 3](#_Toc138068546)

[6. Briefly describe how the stack is used when a function calls another function, which returns data to the caller. (4 points) 3](#_Toc138068547)

[7. What do the following two operations do? What is their difference? Which one would correspond to the C code: \*( p + 4) = var? (4 points) 3](#_Toc138068548)

[8. What is a process graph? What is a critical section and unsafe region in a process graph? What is a “safe trajectory” on the process graph? (4 points) - Mangler 4](#_Toc138068549)

[Besvarelsen mangler 4](#_Toc138068550)

[9. What problem is solved by the Translation Lookaside Buffer (TLB)? (3 points) 4](#_Toc138068551)

[Practical questions ( 11 points each ). 4](#_Toc138068552)

[1. Assume a pipeline with 4 stages: 4](#_Toc138068553)

[2. (5, error?) Describe the operation of the following HCL code function, i.e., what does it achieve. As in class, assume that capitalized letters are words (e.g., integers of n bits). Otherwise, they represent booleans. 6](#_Toc138068554)

[3. Given the following C code and resulting assembly code: 6](#_Toc138068555)

[6](#_Toc138068556)

[4. Consider a memory system that consists of three cache layers L1, L2, and L3 cache above the main memory. The L1 hit rate is 80%, the L2 hit rate is 90% and the L3 cache is 97%. The access time for L1, L2 and L3 is 1 cycle, 10 cycles, and 30 cycles, respectively. Assume all content is in memory and that accessing the memory has a cost of 300 cycles. -Mangler 8](#_Toc138068557)

[5. Draw the process graph from the C code below. Give an example of a feasible and an infeasible ordering. 8](#_Toc138068558)

[6. Consider a directly mapped cache with the following properties: 8](#_Toc138068559)

[Main memory size: 64 bytes Cache block size: 8 bytes S = 2 sets 8](#_Toc138068560)

[E = 1 block/set 8](#_Toc138068561)

## Theoretical questions

### Define temporal and spatial locality. What is a Stride-1 access pattern, and what type of locality does it exploit? (4 points)

Locality is reusing addresses and thereby minimizing the jumps from the previous memory location to the next one.

When taking advantage of Temporal locality, a memory reference, that is previously used, is saved for future references.

When taking advantage of Spatial locality, a memory reference near the previously used memory address is likely to be used.

A stride 1 access pattern comes from spatial locality and is when each of a vector’s addresses, say a collection of addresses, is used in a sequential way, one by one.

### Briefly describe how fractional numbers are represented in binary. Can you represent 515.7 with 4 bits for the decimals? If not, what are the two closest numbers to it you can represent with 4 decimal bits? (4 points)

Fractional numbers is common to integers with the one exceptions for the decimals.

So 515 is represented as byte of an integer. The decimals next to the dot is then calculated in a different way.

Where the bits next to the dot holds the value: , where *n* is the bits location next to the d. for the binary representation.

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

Which is its closest fractional value from 4 bits.

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

By calculating it this way, it’s not possible to represent 0.7 exactly, one might be able to do it with the IEE(something more) way of doing it, but one might not be able to.

I am not that familiar with the other way, it’s something with fractions and exponential to represent it, but I think that it for standards uses 31 bites.

### How do you calculate the number of bits needed to represent a positive integer? Represent the following integers (in hexadecimal) in binary: 0x50, 0x33, 0x341 Calculate: 0x50 & 0x33, 0x341 ^ 0x33 (4 points).

To check how many bits required to represent a positive integer, one could do the following test.

Where *k* is the amounts of bits used to represent an unsigned integer of its value, and -1 to take its sign bit into account, making it an integer.

Doing the & operation, the output of each bit of the compared values are put into an AND gate.

The ^ operation is an xor. This is done to all the bits of both values again.

### What is “big endian” and “little endian” byte ordering and what’s the difference? (3 points)

Big endian and little endian is ways of interpreting byte order.

Little endian reads the Least Significant Byte as the first byte from the left, were as the Big endian has the opposite convention, reading the MSB as the first byte from the left.

### What are the six general instruction processing stages in a CPU studied in class? Briefly describe them. (4 points)

The six general stages of instruction for a CPU is

1. Fetching
2. Decode
3. Execute
4. Memory
5. PC update
6. Write back

### Briefly describe how the stack is used when a function calls another function, which returns data to the caller. (4 points)

When a function calls a functions, then it jumps to a given location.   
In assemply the jumplocation could’ve been called

And when calling it, depended on which condition is needed to be fullfilled before the jump, a jump instruction is chosen.

When returning a value the value is put into which is used to store values like these.

In assembly it might look like this.

### What do the following two operations do? What is their difference? Which one would correspond to the C code: \*( p + 4) = var? (4 points)

So in C a pointer of a value + an Offset of 4 is writen to be the var.

In assembly that is

So it corresponds to the last instruction.

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

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

### What is a process graph? What is a critical section and unsafe region in a process graph? What is a “safe trajectory” on the process graph? (4 points) - Mangler

### Besvarelsen mangler

### What problem is solved by the Translation Lookaside Buffer (TLB)? (3 points)

The translation lookaside buffer translates virtual page numbers into physical page numbers, and is a part of the MMU to speed up the process of having to fetch and await response from the cache, so it stores crucial addresses, that has been fetched from the cache.

## Practical questions ( 11 points each ).

### Assume a pipeline with 4 stages:

Stage 1 logic takes 100 ps   
Stage 2 logic takes 40 ps

Stage 3 logic takes 25 ps

Stage 4 logic takes 25 ps

Registers in each stage add a delay of 20ps.

Answer the following:

* Compute the throughput for the pipeline in GIPS.

As it is a non uniform pipeline, the instructions is as fast as its slowest part.

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

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

With 4 instructions, an a total time of 270ps the GIPS is calculated:

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

Which is er

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

* Compute the delay for an instruction.

The delay for one instruction is 20ps times the amount of times, that the register is used, which is 4.

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

If only taken account to the register delay, then the

If the delay waiting for instructions is taken into account, then the

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

* If you transformed it into a 3-stage pipeline, what is the best choice of stages to merge? What is the resulting throughput and delay?

The best states to merge would be the stages requiring the least amount of time to perform, as these adds the most delay.

Then it would be:

Stage 1: 100ps

Stage 2: 40ps

Stage 3: 50ps

The delay would then be:

The GIPS would then be:

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

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

And the total delay will then be

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

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

* If you transformed it into a 2-stage pipeline, what is the best choice of stages to
* merge? What is the resulting throughput and delay?

The two best merges of a 2 stage pipeline for this example would be the stage 2 & 3.

Stage 1: 100ps

Stage 2: 90ps

Then the throughput would be.

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

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

And the total delay will then be.

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

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

### (5, error?) Describe the operation of the following HCL code function, i.e., what does it achieve. As in class, assume that capitalized letters are words (e.g., integers of n bits). Otherwise, they represent booleans.

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

Automatisk genereret beskrivelse

The HCL makes an array of possible values. The values are assigned if they met the condition.

Cnd 1. If A is greater than or equal to B AND A is less than or equal to C then assign A as its value.

Cnd 2. If B is greater than or equal to A AND C is less than or equal to A, then assign A as its value.

Cnd 3. If B is greater than or equal to A AND B is less than or equal to C, then assign B as its value.

Cnd 4. If B is less than or equal to A AND B is greater than or equal to C, then assign B as its value.

Cnd 5. If 1 == True, which it is, assign C.

With this an array is made.

### Et billede, der indeholder tekst, kvittering, Font/skrifttype, hvid Automatisk genereret beskrivelseGiven the following C code and resulting assembly code:

### Et billede, der indeholder tekst, kvittering, Font/skrifttype, hvid Automatisk genereret beskrivelse

Figure 1: C function

Figure 2: Assembly of the function

Et billede, der indeholder tekst, kvittering, Font/skrifttype, hvid

Automatisk genereret beskrivelseHow would the C code change if the assembly code changes to the following:

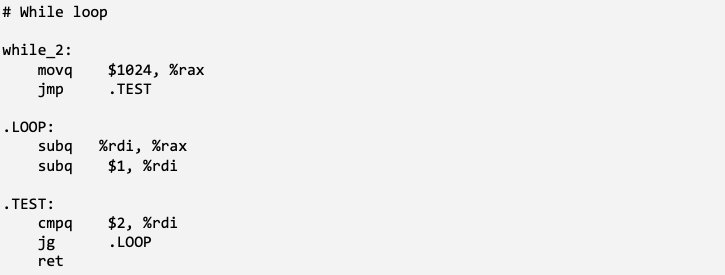


Figure 4: New assembly code

Figure 3: Assembly of the function

Let’s compare:

Instead of the function while\_1, a function while 2 will be called.

Instead of the constant 1, then a constant of 1024 will be put into %rax.

In the loop, instead of multiplying the result by it self times n,

now %rax is %rax - n.

The last part of the loop is the same.

In .Test, it now compares n with 2 instead, but keeps the same jump condition.

In C the new func would be.

}

}

### Consider a memory system that consists of three cache layers L1, L2, and L3 cache above the main memory. The L1 hit rate is 80%, the L2 hit rate is 90% and the L3 cache is 97%. The access time for L1, L2 and L3 is 1 cycle, 10 cycles, and 30 cycles, respectively. Assume all content is in memory and that accessing the memory has a cost of 300 cycles. -Mangler

* What is the average access time of a memory address, in CPU cycles? Use the values from the table below to determine the cache hit rates.
* Based on your calculations give the general formula of access time for this 3-level cache configuration.

Besvarelsen Mangler

### Draw the process graph from the C code below. Give an example of a feasible and an infeasible ordering.



### Consider a directly mapped cache with the following properties:

### Main memory size: 64 bytes Cache block size: 8 bytes S = 2 sets

### E = 1 block/set

* Describe the cache structure and the bits of each address used for selecting the set the tag.

Tag = 1 byte

Valid = 1 byte

Block size = 8 Bytes

With two sets, that equals to 10 Bytes \* 2 used from main memory.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| V | Tag | Cache block size | | | | | | | |
| V1 | Tag1 | B1\_0 | B1\_1 | … | … |  |  |  | B1\_7 |
| V2 | Tag2 | B2\_0 | B2\_1 | … | … |  |  |  | B2\_7 |

To access these, the bits used as addresses are:

(V1)(Tag1) 000 or (V1)(Tag1) 001

Where the highest block address value is 111 = 7.

Depended on the value of the sets tags and value, the bit of the address corresponding to each of these, needs to be the same value for it to be a HIT!

* The code reads bytes through a one dimensional char array of 32 elements (i.e., 32 bytes). The byte array starts in address 0x000. It reads in a stride-one pattern.
* Which address reads cause a cache miss? Calculate the cache miss ratio

The main memory can store up to 64/10 = 6 sets of cache, leaving the last 4 bytes unavailable in this case.

The first M[0; 15] of memory is being used by the two sets.

The tags and values needs to be identical to the caches values.

So it’s:

V1 Tag1 00(0000 - 0111)

V2 Tag2 00(1000 - 1111)

If the block address part doesn’t start with 00, a miss will occur, as these blocks aren’t occupied yet.

If the block address part of 0000 -> 0111 doesn’t have tags and valids similar to the first sets tag and valid, then a different miss will occur. Permission is denied.

The same goes for block address part 1000 -> 1111 with set 2s valid and tag.

The miss ratio for the blocks to be empty is

The chance of missing the combination of the tag and validation is .

As these are two of the 6 combinations, the miss ratio is calculated to be:

The chances of a miss of for the validation and tag is

Making the miss ratio

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

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

This was an assumption and might be wrong.

* Consider the same array as before, but you use a random-access pattern as follows: 0, 8, 7, 23, 2, 17, 1, 31, 20, 3, 5, 6, 4, 21, 11, 27. Calculate the new cache miss ratio. How does it compare to the stride-one pattern?