4-hour Exam in Computer Systems, B1-2022/23

Department of Computer Science, University of Copenhagen (DIKU)
Date: January 25, 2023

1 Machine architecture (about 34 %)

1.1 Data Cache and Locality (about 8 %)

A byte-addressed machine with 32-bit addresses is equipped with a 16 kibibyte 4-way set-associative cache with a block size of 32 bytes. The cache uses "least-recently-used replacement policy".

Question 1.1.1: The address is separated into the following fragments when the cache is accessed

- block offset,
- cache tag, and
- set index.

What is bit-size of each and how are they ordered?

	31		0
Address fragment			
Bit-size			

Question 1.1.2: Calculate for the following addresses the value of block offset, cache tag, and set index.
Address: `0x76543210`

(Maximum 5 lines.)

Address: `0x01234567`

(Maximum 5 lines.)

Question 1.1.3: On the machine, the following stream of cache accesses are performed (read from top to bottom). Indicate for each of the address references:

- the set index,
- if the cache access is a miss or a hit, and
- the cache tags in LRU-order that are in the affected set after the access.

Addresses are given in hexa-decimal notation. Assume the cache is cold on entry.

Reference	Set index	Hit/Miss	State of Tags
0x0100004			
0x0100010			
0x0010008			
0x0110004			
0x0000008			
0x0300004			
0x0300010			
0x0110008			
0x0410004			
0x0A00008			
0x0110004			
0x0000008			
0x0300004			
0x0300010			

1.2 Microarchitecture and execution diagram (about 12 %)

Consider the following fragment of a program execution

```
1  .L3:    sb      a5,0(a1)
2          lbu     a5,1(a0)
3          addi    a0,a0,1
4          addi    a1,a1,1
5          bne     a5,zero,.L3
6  .L3:    sb      a5,0(a1)
7          lbu     a5,1(a0)
8          addi    a0,a0,1
9          addi    a1,a1,1
10         bne     a5,zero,.L3
```

All questions in this exercise refer to how this instruction sequence is executed on different micro-architectures.

Question 1.2.1: Give an execution diagram for a simple 5-stage pipeline with full forwarding as described in COD. Explain shortly any assumptions you may have to make.

	Code	Timing																			
	`.L3:`																				
1	`sb a5,0(a1)`																				
2	`lbu a5,1(a0)`																				
3	`addi a0,a0,1`																				
4	`addi a1,a1,1`																				
5	`bne a5,zero,.L3`																				
	`L3:`																				
6	`sb a5,0(a1)`																				
7	`lbu a5,1(a0)`																				
8	`addi a0,a0,1`																				
9	`addi a1,a1,1`																				
10	`bne a5,zero,.L3`																				

(Maximum 8 lines.)

Question 1.2.2: Give an execution diagram for a 2-way in-order superscalar with single cycle cache access as presented first in the section on super scalars in the online course notes. Explain shortly any assumptions you may have to make.

	Code	Timing																			
	`.L3:`																				
1	`sb a5,0(a1)`																				
2	`lbu a5,1(a0)`																				
3	`addi a0,a0,1`																				
4	`addi a1,a1,1`																				
5	`bne a5,zero,.L3`																				
	`L3:`																				
6	`sb a5,0(a1)`																				
7	`lbu a5,1(a0)`																				
8	`addi a0,a0,1`																				
9	`addi a1,a1,1`																				
10	`bne a5,zero,.L3`																				

(Maximum 8 lines.)

Question 1.2.3: Give an execution diagram for a 4-way out-of-order with realistic (3-stage pipelined) cache access as presented in the section on out-of-order microarchitecture in the online course notes. Explain shortly any assumptions you may have to make.

	Code	Timing																			
	`.L3:`																				
1	`sb a5,0(a1)`																				
2	`lbu a5,1(a0)`																				
3	`addi a0,a0,1`																				
4	`addi a1,a1,1`																				
5	`bne a5,zero,.L3`																				
	`L3:`																				
6	`sb a5,0(a1)`																				
7	`lbu a5,1(a0)`																				
8	`addi a0,a0,1`																				
9	`addi a1,a1,1`																				
10	`bne a5,zero,.L3`																				

(Maximum 8 lines.)

1.3 Assembler programming (about 14 %)

Consider the following program written in RICS-V assembler.

```
1  myfunc:
2          lbu     a5,0(a0)
3          beq     a5,zero,.L2
4  .L3:
5          sb      a5,0(a1)
6          lbu     a5,1(a0)
7          addi    a0,a0,1
8          addi    a1,a1,1
9          bne     a5,zero,.L3
10 .L2:
11         sb      zero,0(a1)
12         ret
```

Question 1.3.1: The code snippet is a function. Is this function calling other functions? Argue for your answer

(Maximum 5 lines.)

Question 1.3.2: Which registers hold the functions arguments (if any)? Argue for your answer.

(Maximum 5 lines.)

Question 1.3.3: The function contains a loop. Which instructions form the loop? Describe how you identified this.

(Maximum 10 lines.)

Question 1.3.4: Rewrite the above RISC-V assembler program to a C program. The resulting program must not have a goto-style and minor syntactical mistakes are acceptable.

(Maximum 20 lines.)

Question 1.3.5: Descripe shortly the functionality of the program.

(Maximum 8 lines.)

Question 1.3.6: What is a pseudo-instruction and give an example of a RISC-V pseudo-instruction?

(Maximum 5 lines.)

2 Operating Systems (about 33 %)

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 the following resources aquired during process execution are automatically freed by the kernel (or killed, deleted, etc. as appropriate) when a process terminates?

a) Child processes

b) Memory allocated with `malloc()`

c) Memory allocated with `mmap()`

d) Files on disk

e) File descriptors

f) Condition variables

(Maximum 5 lines.)

Question 2.1.2: Assuming valid arguments, which of the following C functions always do a system call?

a) `fopen()`

b) `fwrite()`

c) `fread()`

d) `fork()`

e) `read()`

f) `malloc()`

(Maximum 5 lines.)

Question 2.1.3:

Consider the C program below. For space reasons, we are not checking return codes for error, so assume that all functions return normally and that all IO is unbuffered.

```c
int main () {
  if (fork() == 0) {
    printf("1");
    if (fork() == 0) {
      printf("2");
    }
  } else {
    pid_t pid = fork();
    if (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) `1524355`

b) `5531245`

c) `1523545`

d) `3412555`

e) `5551234`

f) `1234555`

(Maximum 5 lines.)

2.2 Short Questions (about 12 %)

Question 2.2.1: Consider the following program. Assuming that `printf()` itself executes atomically and unbuffered, how many lines of output does it produce? Is there more than one answer (i.e. does it contain nondeterministic behaviour or race conditions)? Is the content of the lines also nondeterministic? If yes, in what way? If no, why not?

```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

int x = 1;

void* thread(void* arg) {
  x = 2;
  printf("%d\n", x);
  return NULL;
}

int main() {
  printf("%d\n", x);
  pthread_t tid;
  pthread_create(&tid, NULL, thread, NULL);
  fork();
  printf("%d\n", x);
  exit(0);
}
```

(Maximum 10 lines.)

Question 2.2.2: Consider a system with the following properties:

- Memory is byte-addressed.
- Virtual addresses are 15 bits wide.
- Physical addresses are 13 bits wide.
- The page size is 128 bytes.
- The TLB is 3-way set associative with 4 sets and 16 total entries. Its initial contents are:

Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	0F	10	0	11	15	1	1F	2E	1
1	0A	11	1	11	15	0	07	12	1
2	13	33	1	00	00	0	00	00	1
3	14	21	1	00	12	0	10	0A	1

- The page table has the following contents:

VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid
00	00	1	21	10	0	3F	23	0	12	34	1
01	33	1	0C	0D	0	08	17	1	13	15	1
A0	21	0	FA	00	1	A2	32	0	03	43	0

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.*

Virtual address: `0xef2`

1. Bits of virtual address

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

2. Address translation

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

3. Bits of phys. (if any)

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

Virtual address: `0x8f`

1. Bits of virtual address

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

2. Address translation

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

3. Bits of phys. (if any)

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

Virtual address: `0x1af`

1. Bits of virtual address

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

2. Address translation

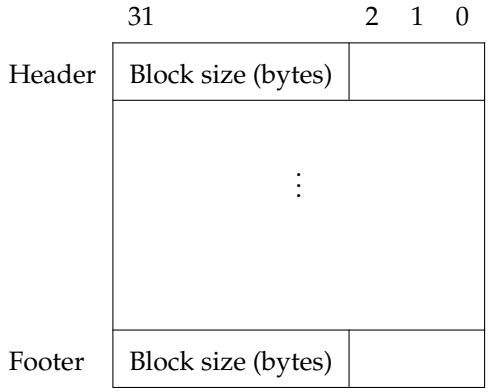
Parameter	Value
VPN	
TLB index	
TLB tag	
TLB hit? (Y/N)	
Page fault? (Y/N)	
PPN	

3. Bits of phys. (if any)

12	11	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 minimum block size is 8. 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 partial contents of the heap shown on the left, show the new contents of the heap after a call to `malloc(8)` is executed that returns `0xd1c008` (in the middle column), followed by a call `free(0xd1c028)` (rightmost column).

- All numbers are hexadecimal, and so should your answers be.
- Note that the address grows from bottom up.
- Some parts of the heap may lie outside the area shown.
- Assume that the allocator uses immediate coalescing, that is, adjacent free blocks are merged immediately each time a block is freed.
- Perform the minimum number of memory changes required.

Address	Original value	After `malloc`	After `free`
00d1c04c	00000100	00000100	00000100
00d1c048	00000020	00000020	00000020
00d1c044	00000042		
00d1c040	00000021		
00d1c03c	0000001d		
00d1c038	00000018	00000018	00000018
00d1c034	00000000	00000000	00000000
00d1c030	00000000	00000000	00000000
00d1c02c	00000000	00000000	00000000
00d1c028	0000001d	0000001d	0000001d
00d1c024	00000021		
00d1c020	00000022		
00d1c01c	00000000	00000000	00000000
00d1c018	00000000	00000000	00000000
00d1c014	00000000		
00d1c010	00000000		
00d1c00c	00000000	00000000	00000000
00d1c008	00000000	00000000	00000000
00d1c004	00000022		
00d1c000	0000000b		
00d1bffc	0000000b		

Question 2.3.2: Outline a scheme for how two processes running on two physically distinct computers, connected over a network, can use virtual memory to transparently share a region of memory, such that writes from process A into that region will be visible when process B reads that region, and the other way around. You are not asked to explain this in terms of concrete Unix system calls (or even whether it is possible using current Unix systems). Also do not focus on the exact form of the network communication. Instead, explain what the page tables should contain, how the kernels running on the two machines may handle page faults, and what high-level information is exchanged over the network. What kind of performance characteristics does your design have—ie. are there workloads that would be very slow?

(Maximum 12 lines.)

3 Computer Networks (about 33 %)

3.1 Application Layer (about 7 %)

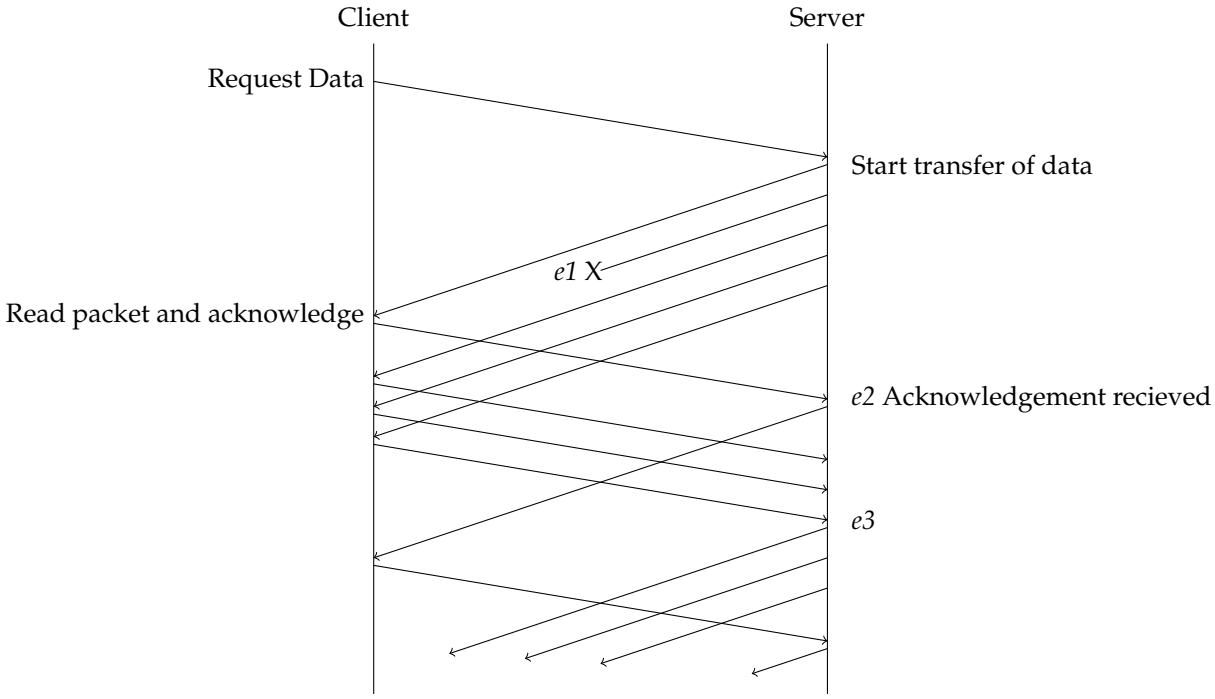
Question 3.1.1: Within the subject of networking, what is meant by the concept of a protocol? Describe what the purpose of protocols are, and what they enable us to do within a network. How does this relate to the concept of layering?

(Maximum 16 lines.)

Question 3.1.2: What services can the UDP protocol provide? How does the UDP protocol enable these services?

(Maximum 12 lines.)

3.2 Transport Layer (about 12 %)



The above diagram shows a client requesting some data from the server. The server responds by sending numerous packets, each of which are acknowledged by the client in turn. At *e1*, the second packet to be sent by the server is lost.

Question 3.2.1: How large is the sending window in server shown above? Briefly justify your answer.

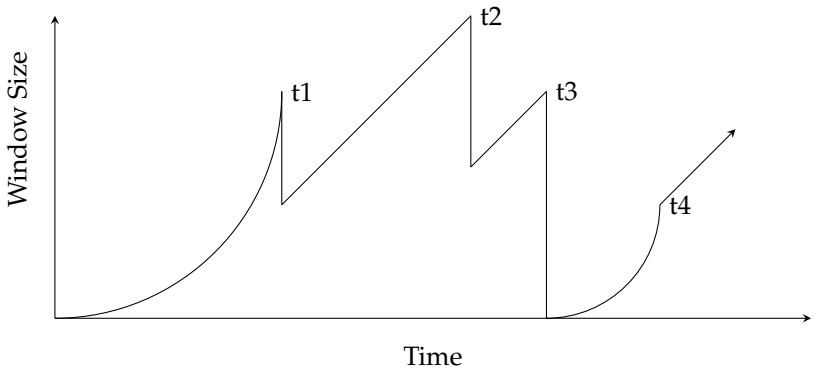
(Maximum 5 lines.)

Question 3.2.2: At *e2* an acknowledgement is received by the server, and a further packet is sent. However, it is not until *e3* that further packets are sent. From this, what protocol can you deduce is being used to fast retransmit lost packets by the server? Briefly justify your answer.

(Maximum 5 lines.)

Question 3.2.3: If another protocol for re-transmission had been selected how would the system behave differently? You may to wish to discuss the contents of messages, performance of the system, or any other factor you feel is relevant.

(Maximum 14 lines.)



Question 3.2.4: The diagram above shows the size of a TCP Transmission Window as it differs over time. What events occur at *t1*, *t2*, *t3*, and *t4*.

(Maximum 10 lines.)

3.3 Network Layer (about 7 %)

Question 3.3.1: Two methods of addressing on a network as IP addresses and MAC addresses. Of these, IP addresses are said to be hierarchical, and MAC addresses are not. Why is this and what is the reason for the difference?

(Maximum 15 lines.)

Question 3.3.2: The two most common routing algorithms are the Link State Algorithm and the Distance Vector Algorithm. Where would each be used, and what advantages does each have over the other?

(Maximum 15 lines.)

3.4 Network Security (about 7 %)

Question 3.4.1: In the context of a Network Security, a Rainbow Table is a precomputed list of hashes. How would an attacker use one, and how would the use of salts help address them?

(Maximum 15 lines.)

Question 3.4.2: Congratulations, you have been hired as the newest member of a small tech company. The company operates a server that is accessible from the public internet, so that employees can access their work wherever they are in the world. Your new boss explains that the server is completely secure as it is protected by a firewall, and therefore no other security steps have been taken. What are the three most significant security issues you would expect to encounter in such a system?

(Maximum 15 lines.)