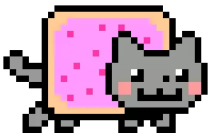
Computer Science 244

Semester Test 2 Group Memo's

With love from the CS244-2017 Class <3



Semester Test 2: 2017 **1**
Limerick Answers: 1

Semester Test 2: 2016 **2**
Short Questions 2
Long Questions 9

Semester Test 2: 2015 **18**
Short Questions 18
Calculations 19
Long Questions 22

Can you run me over while you're at it so I don't have to do this test? Hang in there, stranger <3 **27**

Semester Test 2: 2014 **28**
Short Questions 28
Calculations 29
Long Questions 29

Semester Test 2: 2013 **33**
Short Questions 33
Long Questions 33

Semester Test 2: 2012 **34**
Short Questions 34
Calculations 35
Long Questions 35

Semester Test 2: 2011 **36**
Short Questions 36
Calculations 36

Semester Test 2: 2017

The Group Answers

Limerick Answers:

There once was a man named Colt,
who kept three hens in a vault,
until one day, they all ran away,
"Oh no!" he cried, "Seg Fault!"

To the students of CS244
A seg fault is an error one can't ignore
Forget to free
And soon one will see
A message His minions adore!

I sit in H all day
Till even night goes away.
To Huffman encode
What and arduous load!
What more on this can I say?

Sometimes I wish I was more so,
An engineer, director or film - but, Oh!
A thought in my head,
Spoke what Willem had said
And down the drain the thoughts flow

There once was a lecturer named Willem
Whose standards were high as the ceiling
He gave us a test
Got 50 at best
And now the whole class wants to kill 'im

I came into class on day one
C, you must learn now my son
a pointer, a struct
Oh man, how I sucked
Then git push to cs.sun (cs dot sun)

8am is no time to learn
We were up all night our eyes burn!
No fun times were loading
All that happened was coding
Dear Rest, when will it be our turn?

Willem taught us sharks
To guide us through the dark
Gave us light
Filled with might
But where are our codegen marks

His Worshipfulness, Darth Bester:
The Galaxies' top case tester,
A distaste for MS,
But with code would impress
And could even destroy the Death Star.

Assembly! The young coder thought
A tool without which I won't be caught
Sadly, she drowned in nasm
Trying to escape a pointer chasm
And all the suffering was for naught

Please give me a score
Of more than 54
I tried my best
To pass this test
But I can't even write a limerick

Semester Test 2: 2016

The Group Answers

Short Questions

1.

(a) Watter een van die volgende Intel-instruksies het geen invloed op die stapelwyser nie? Which one of the following Intel instructions has no effect on the stack pointer? [1]

[A] `call` [B] `cmp` [C] `pop` [D] `ret`

B: cmp sets the condition codes in the machine status appropriately

(b) Watter een van die volgende kan *nie* in Mic-1 mikrokode geënkodeer word *nie*? Which one of the following *cannot* be encoded in Mic-1 microcode? [1]

[A] H = H – MDR [B] H = MDR – H [C] H = H + MDR [D] H = MDR + H

A: because the only possible source of a subtracted value is the H register

(c) Gestel 'n proporsionele allokasieskema word gebruik op 'n stelsel met 62 rame en twee prosesse, een van 10 blaaie, en een van 127. Hoeveel rame sou aan die groot proses toegeken word? Suppose a proportional allocation scheme is used on a system with 62 frames, and two processes, one of 10 pages, and one of 127 pages. How many frames would be allocated to the larger process? [1]

[A] 4 [B] 31 [C] 57 [D] 62

C: **m** = # of frames, $\mathbf{s}_i$ = size of process $\mathbf{p}_i$, $\mathbf{a}_i$ = allocation for $\mathbf{p}_i$

m = 62, $\mathbf{s}_1$ = 10, $\mathbf{s}_2$ = 127 $\mathbf{a}_i = \mathbf{s}_i / \Sigma\mathbf{s}_i * \mathbf{m}$

$\mathbf{a}_1$ = 10/137 * 62 ≈ **4** ; a_2 = 127/137 * 62 ≈ **57 = larger process**

(d) Wanneer 'n tweede-kans bladsyvervangingsalgoritme, met 'n verwysings- en 'n modifieerbis (in dié volgorde), gebruik word, watter een van die volgende bladsyklasse behoort eerste uitgeskop te word? When a second-chance page replacement algorithm, with a reference and a modify bit (in this order), is used, which one of the following page class should be evicted first? [1]

[A] (0,0) [B] (0,1) [C] (1,0) [D] (1,1)

A: Take ordered pair (reference, modify) @9.40 Dinobook

1. (0, 0) neither recently used nor modified – best page to replace
2. (0, 1) not recently used but modified – not quite as good, must write out before replacement
3. (1, 0) recently used but clean – probably will be used again soon
4. (1, 1) recently used and modified – probably will be used again soon and need to write out before replacement

2.

Bus Skew:
A problem where the signals on different lines of a bus travel at slightly different speeds, causing them to go out of time.

Volatile Memory:
Storage that only maintains its data while the device is powered.

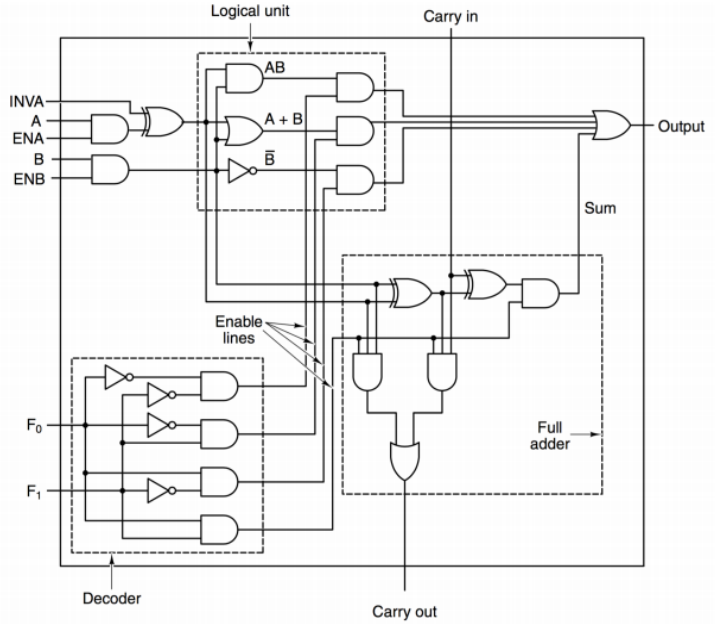
Associative Cache Memory:
A cache with n possible entries for each address is called an n-way set-associative cache. This is needed when our cache might try store different addresses in the same cache line due to a limited indexing system. 2-way or 4-way is usually best.

Register Renaming:
A form of pipelining that deals with data dependencies between instructions by renaming register operands to secret registers. This allows the OS to use ‘the same set’ of registers at once.

Relocatable Code:
Relocatable code resides at the address relative to the beginning of absolute code, when it is not known at compile time where a process is supposed to reside in memory. Thus, the code can run on any system and is guaranteed not to fight over memory/stack space, as the OS can assign the absolute code reference, from which relative references can be determined.

Copy-on-write:
A technique allowing child and parent processes to share the same pages, initially (with these pages marked as *COW*), so that a copy of the *COW* page is created **only** when either process writes to it.

Page (**computer** memory) ... A **page**, memory **page**, or virtual **page** is a fixed-length contiguous block of virtual memory, described by a single entry in the **page** table. It is the smallest unit of data for memory management in a virtual memory operating system.

3. 

4.

Func	F0	F1	ENA	ENB	INVA	INC
A + B	1	1	1	1	0	0
A + B + 1	1	1	1	1	0	1
B - A	1	1	1	1	1	1
-A	1	1	1	0	1	1

5.

4. Veronderstel 'n rekenaarmaatskappy vervaardig 'n greep-adresseerbare, 32-Mibis geheueskyfie vir gebruik in 'n 64-bis argitektuur. Antwoord die volgende vrae, en toon u bewerkings vir elkeen.

Suppose a computer company manufactures a byte-addressable, 32-Mibit memory chip for use in a 64-bit architecture. Answer the following questions, and show your calculations for each.

(a) Hoeveel adresse is beskikbaar op elke skyfie? How many addresses are available on each chip? [1]

(b) Hoeveel skyfies is nodig om 'n woord te voltooi? How many chips are necessary to complete a word? [1]

(c) Wat is minimum geheuegrootte vir dié opstelling? What is the minimum memory size for this setup? [1]

Not Sure Possible (uneducated) guess based on p. 178-180: 32 Mb / 64 b = 32 * 2^{20} / 64 = 2^{19} addresses

Not Sure Possible (uneducated) guess based on p. 178-180: 64 chips for a 64-bit architecture

Another guess:

A) 32 MiBit chip that is byte addressable => 32 xe 1024 / 8 = 4 MiBit addresses

B) Word size on 64 bit architecture = 64 bit = 8 bytes. 512 Bytes available in 4 MiBit, therefore 1/512 chips per word. - Not sure here at all…

C) 512 Bytes

6.

Page Replacement Algorithm slides method:

5. (a) Beskou die ry van bladsyverwysings hieronder. Gegee dat daar drie rame is—en onthou, hulle almal is aanvanklik leeg—*hoeveel bladsyfoute* vind plaas onder die gespesifiseerde vervangingstrategieë? Toon u "bewerkings" in die kassies voorsien.

Consider the sequence of page references below. Assuming three frames—and remembering that all frames are initially empty—*how many page faults* occur for the specified replacement algorithms? Show your "calculations" in the boxes provided.

3, 4, 1, 5, 3, 4, 2, 3, 4, 1, 5, 2

Langsgelede gebruik / Least-recently-used (LRU)

3	4	1	5	3	4	2	3	4	1	5	2

Eerste-in, eerste-uit / First-in, first-out (FIFO)

3	4	1	5	3	4	2	3	4	1	5	2

Dino book method:

N	3	4	1	5	3	4	2	3	4	1	5	2
F1	3	3	3	5	5	5	2	2	2	1	1	1
F2		4	4	4	3	3	3	3	3	3	5	5
F3			1	1	1	4	4	4	4	4	4	2
Fault?	1	1	1	1	1	1	1	0	0	1	1	1
												10

N	3	4	1	5	3	4	2	3	4	1	5	2
F1	3	3	3	5	5	5	2	2	2	2	2	2
F2		4	4	4	3	3	3	3	3	1	1	1
F3			1	1	1	4	4	4	4	4	5	5
Fault?	1	1	1	1	1	1	1	0	0	1	1	0
												9

Page Replacement Alg

(b) Herhaal die berekening vir die vorige vraag, maar gebruik nou vier rame. Definieer en illustreer daarná **Belady se anomalie** aan die hand van u berekeninge.

Repeat the calculations of the previous question, but now use four frames. Then, define and illustrate **Belady's anomaly** with reference to your calculations.

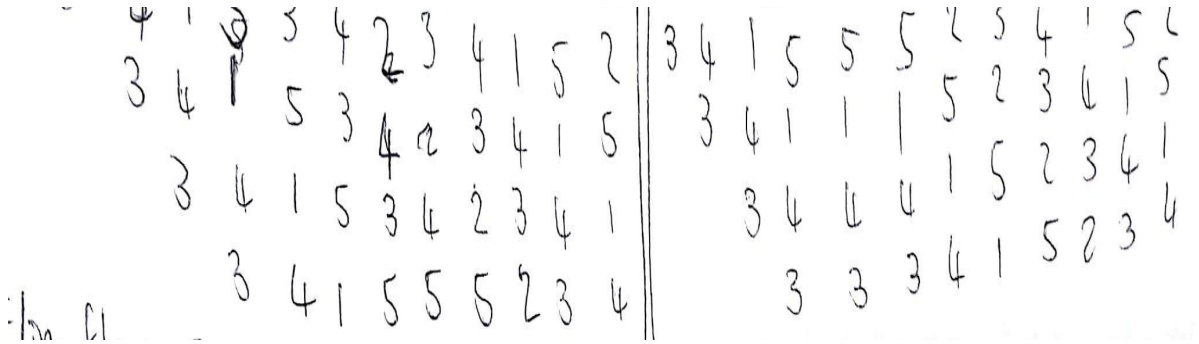
Langsgelede gebruik / Least-recently-used (LRU)

3	4	1	5	3	4	2	3	4	1	5	2

Eerste-in, eerste-uit / First-in, first-out (FIFO)

3	4	1	5	3	4	2	3	4	1	5	2

orithm slides method:



If we do these calculations, we'll see that the FIFO implementation has **more** page faults once we've introduced another frame. This is Belady's Anomaly, that adding more frames does not necessarily mean fewer page faults, and can actually lead to more page faults. LRU is a stack algorithm that doesn't have Belady's anomaly.

Long Questions

(a) Verduidelik kortliks die verskil tussen 'n **flip-flop** en 'n **wipkring**.

Briefly explain the difference between a **flip-flop** and a **latch**. [2]

6. a) **Flip-flop:** Grabs a value during the rising or falling edge of the clock signal

Latch: A 1-bit memory that stores previous input values

Flip-flop is *edge-triggered*, while a latch is *level-triggered*.

Latch -> changes output when input is changed

Flip-Flop -> changes output when control signal goes from high to low, or low to high (change of state)

(b) Verduidelik hoe die Mic-1 se N- en Z-bisse bedraad is, en hoe hulle betrokke is by vertakking op mikroinstruksievlak.

Explain how the Mic-1's N and Z bits are wired, and how they are involved with branching on the microinstruction level. [3]

b) N and Z bits are received from the ALU and are wired in such a way as to be saved in a pair of one bit flip-flops. This helps with branching (When an algorithm makes a choice to do one of two things) as saving the ALU status flags in N and Z makes the correct values available and stable for the MPC computation.

[The N bit is used to symbolise whether the bytecode is negative (N is 1) or non-negative (N is 0)

The Z bit is used to decide if the bytecode is zero (if Z is 1) or non-zero (Z is 0)

The Z value is XOR'ed for any compare because if the 2 values are the same the XOR will return 0 which then branches to the condition required whereas if the values are different XOR will return 1 and thus Z is 0 and so it branches to the false condition]

(c) Bespreek kortliks die ontwerp van die stelselbus van 'n tipiese Intel-stelsel. Skenk spesifiek aandag aan die spoed van die gekoppelde komponente.

Briefly discuss the design of the system bus of a typical Intel system. Pay particular attention to the speed of the connected components. [3]

c) Modern personal computers generally have a special-purpose bus between the CPU and memory (These are often of a short length to increase speed) and (at least) one other bus for the I/O devices(Which are often slower than the special-purpose buses above). A minimal system, with one memory bus and one I/O bus, is illustrated below.

In brief, Two buses involved, Internal and external. The internal bus connects the CPU to the ALU for fast data transfers(This bus is short and narrow, to increase speed). The external

bus connects the CPU to memory and I/O devices(Often wider and thus slower buses than internal buses).

(d) Noem en bespreek kortliks *twee* strategieë om die uitvoerspoed van die datapad te vermeerder. Name and briefly discuss *two* strategies for increasing the speed of execution of the data path. [3]

d) 1) Reduce the number of clock cycles needed to execute an instruction.
2) Simplify the organization so that the clock cycle can be shorter.
3) Overlap the execution of instructions.

The first two are obvious, but reducing the number of instruction cycles necessary for fetching instructions requires more than just an additional circuit to increment the PC. In order to speed up the instruction fetching to any significant degree, the third technique—overlapping the execution of instructions—must be exploited. Separating out the circuitry for fetching the instructions—the 8-bit memory port, and the MBR and PC registers—is most effective if the unit is made functionally independent of the main data path. In this way, it can fetch the next opcode or operand on its own, perhaps even performing asynchronously with respect to the rest of the CPU and fetching one or more instructions ahead.
cache

(e) **LNS-afhanklikhede** word ook dikwels "ware afhanklikhede" genoem. Motiveer dié karakterisering, met verwysing na ander instruksie-afhanklikhede. **RAW dependencies** are also often called "true dependencies". Motivate this characterisation, with reference to other instruction dependencies. [3]

e) **RAW dependencies** are when instructions need to wait for a result from a previous microstep to be calculated (= *stalling*), and only then read the result. They are **true dependencies**, because they are truly dependent on the result of previous microsteps, unlike other dependencies, which only need to wait for another microstep to complete an operation on a register (reading or writing, i.e. WAR or WAW), before it can overwrite that register. (pp. 297, 318)

(f) Motiveer en bespreek die gebruik van kasgeheue met betrekking tot die **lokaliteitsbeginsels**. Motivate and discuss the use of cache memory with respect to the **principles of locality**. [3]

f) *Use of cache memory w.r.t. the Locality Principle:*

In brief,

Locality Principle = memory references made in a short time interval tend to only use a small fraction of total memory

Spatial locality = locations in memory with numerically similar addresses to recently accessed memory are likely to be accessed in the near future
Temporal locality = when recently accessed memory locations are likely to be accessed again

In *long*,
Cache memory significantly improves the performance of a machine in most cases as it reduces the time that the CPU must wait between requesting data/operands for an instruction, and receiving it. By reducing this time, we cut out the CPU stalling that occurs when it is waiting on more data for processing.

Locality caching helps this process by pre-fetching data in the same proximity as recently-fetched data (this is chosen by pre-requesting data in similar address space as the recently accessed memory location). This ensures that data which, by logic should be required soon, is available in an incredibly fast cache for the CPU to load directly.

Temporal locality assumes that data recently accessed by the CPU might be needed very soon by the following instruction. It thus keeps recently-accessed/written data in cache, so that should the data be needed again - it will be available in a faster time as opposed to re-reading it from main memory.

Both of these principles decrease the time that the CPU must wait by ‘guessing’ data which will be required in the near future. It therefore increases the speed at which the CPU can operate (not the clock speed, but reduces waiting), and thus is important for our cache system.

(g) Gee ’n kort voorbeeld van hoe **spekulatiewe uitvoering** katastrofiese gevolge kan hê indien die stelselargitektuur nie gepaste beskerming deur hardeware voorsien nie.	Give a brief example of how **speculative execution** can have catastrophic consequences if the system architecture does not provide suitable hardware protections.	[2]

g) Speculative execution is when one executes code before it is known whether the result is desired or not. It is essential that they do not cause irrevocable results after being executed, in case the code was not needed - to avoid overwriting registers before this is known, rename the destination registers.

(a) Bespreek kortliks die verskil tussen **segmentering** en **paginering**. Briefly discuss the difference between **segmentation** and **paging**. [3]

7. a) Briefly discuss the difference between segmentation and paging

Consideration	Paging	Segmentation
Need the programmer be aware of it?	No	Yes
How many linear addresses spaces are there?	1	Many
Can virtual address space exceed memory size?	Yes	Yes
Can variable-sized tables be handled easily?	No	Yes
Why was the technique invented?	To simulate large memories	To provide multiple address spaces

Figure 6-9. Comparison of paging and segmentation.

Segmentation: (involves programmer) – Memory is allocated in various sizes (segments) depending on the need for address space by the process. These segments may be individually protected/shared between processes.

Paging: (transparent to programmer) – Memory is divided into small partitions, all of equal size, referred to as "page frames." When a process is loaded it gets divided into pages which are then loaded into frames.

(b) Verduidelik wat die verskil tussen **dinamiese binding** en **dinamiese laaiing** is. Explain what the difference between **dynamic linking** and **dynamic loading** is. [3]

b) Dynamic linking vs Dynamic loading

Dynamic loading is when a routine is not loaded until it is called. Dynamic linking is postponed until execution time, where a small piece of code called a stub is used to indicate how to locate the appropriate library routine. (Static linking is when the system libraries are treated like any other module and combined by the loader into the binary program image).

(c) Verduidelik kortliks wat **versparteling** is. Briefly explain what **thrashing** is. [3]

c) Briefly explain what thrashing is.

If the number of frames allocated to a low-priority process falls below the minimum number required by the computer architecture, we must suspend that process's execution. We should then page out its remaining pages, freeing all its allocated frames. This provision introduces a swap-in, swap-out level of intermediate CPU scheduling.

In fact, look at any process that does not have "enough" frames. If the process does not have the number of frames it needs to support pages in active use, it will quickly page-fault. At this point, it must replace some page. However, since all its pages are in active use, it must replace a page that will be needed again right away. Consequently, it quickly faults again, and again, and again, replacing pages that it must bring back in immediately.
This high paging activity is called **thrashing**. A process is thrashing if it is spending more time paging than executing.

Why does thrashing occur?
Σ size of locality > total memory size
Limit effects by using local or priority page replacement

(d) Omskryf kortliks die bedryf van 'n **omskakeling-sydelingse buffer.**	Briefly describe the operation of a **translation look-aside buffer.**	[3]

d) To speed up the mapping of memory, special tables, called TLBs (Translation Lookaside Buffers), are provided to compare the current virtual address being referenced to those referenced in the recent past. Two such tables are provided for mapping instruction and data addresses.
FROM WIKIPEDIA
A **translation lookaside buffer** (**TLB**) is a memory cache that is used to reduce the time taken to access a user memory location.[1][2] It is a part of the chip's memory-management unit (MMU). The TLB stores the recent translations of virtual memory to physical memory and can be called an address-translation cache. A TLB may reside between the CPU and the CPU cache, between CPU cache and the main memory or between the different levels of the multi-level cache. The majority of desktop, laptop, and server processors include one or more TLBs in the memory management hardware, and it is nearly always present in any processor that utilizes paged or segmented virtual memory.

(e) Verduidelik waarom Unix-lêerstelsels nie vaste skakels na gidse toelaat nie.	Explain why Unix file systems do not allow hard links to directories.	[3]

e) *Explain why Unix file systems do not allow hard links to directories*

To avoid endless loops - he said that in class
Here's an explanation I found online, not 100% sure of how correct everything is:

This is just a bad idea, as there is no way to tell the difference between a hard link and an original name (as they both have **exactly** the same inode index which points to the same place in the HDD).
Allowing hard links to directories would break the directed acyclic graph structure of the file system, possibly creating directory loops and dangling directory subtrees, which would make fsck and any other file tree walkers error prone.
A hard link is just an extra directory entry pointing to that inode.
If you were allowed to do this for directories, two different directories in different points in the filesystem could point to the same thing. In fact, a subdirectory could point back to its grandparent, creating a loop.
Why is this loop a concern? Because when you are traversing, there is no way to detect you are looping (without keeping track of inode numbers as you traverse). Imagine you are writing the du command, which needs to recurse through subdirs to find out about disk usage. How would du know when it hit a loop? It is error prone and a lot of bookkeeping that du would have to do, just to pull off this simple task. R.I.P.

Summarized:
Hard links would break the directed acyclic graph structure of the file system that could possibly lead to directory loops. This is an issue because , when traversing directories, there would be no way to detect whether you are looping.
655

(f) Bespreek, vergelyk en evalueer die allokasiestrategieë verskaf deur FAT aan die een kant, en Linux inodes aan die ander. U analise moet verwysings na lêergrootte, toegangstyd en fragmentering insluit.	Discuss, compare, and evaluate the allocation strategies provided by FAT on the one hand, and Linux inodes on the other. Your analysis must include references to file size, access time, and fragmentation.	[6]

f) https://www.youtube.com/watch?v=V2Gxqv3bJCk
https://www.youtube.com/watch?v=tMVj22EWg6A
According to the Dino book
FAT (File Allocation Table) variation
This simple but efficient method of disk-space allocation was used by the MS-DOS operating system. A section of disk at the beginning of each volume is set aside to contain the table. The table has one entry for each disk block and is indexed by block number. The FAT i s used in much the same way as a linked list. The directory entry contains the block number of the first block of the file. The table entry indexed by that block number contains the block

number of the next block in the file. This chain continues until it reaches the last block, which has a special end-of-file value as the table entry.
An unused block is indicated by a table value of 0. Allocating a new block to a file is a simple matter of finding the first 0-valued table entry and replacing the previous end-of-file value with the address of the new block. The 0 is then replaced with the end-of-file value.
The FAT allocation scheme can result in a significant number of disk head seeks, unless the FAT is cached. The disk head must move to the start of the volume to read the FAT and find the location of the block in question, then move to the location of the block itself. In the worst case, both moves occur for each of the blocks. A benefit is that random-access time is improved, because the disk head can find the location of any block by reading the information in the FAT
(p557 DINO)
(**inodes in UNIX**)
inode — a data structure for storing file system metadata — with pointers to its data. Within the inode is the checksum of each block of data. If there is a problem with the data, the checksum will be incorrect, and the file system will know about it. If the data are mirrored, and there is a block with a correct checksum and one with an incorrect checksum, ZFS (Zettabyte File System) will automatically update the bad block with the good one. Similarly, the directory entry that points to the inode has a checksum for the inode. Any problem in the inode is detected when the directory is accessed. This checksumming takes places throughout all ZFS structures, providing a much higher level of consistency, error detection, and error correction than is found in RAID disk sets or standard file systems. The extra overhead that is created by the checksum calculation and extra block read-modify-write cycles is not noticeable because the overall performance of ZFS is very fast. (p493 DINO)

The FAT contains an entry for every file stored on the volume that contains the address of the file's starting cluster. Each cluster contains a pointer to the next cluster in the file, or an end-of-file indicator at (0xFFFF)

	FAT	INODES

Size	File size is no longer fixed. We do not need to know the size of the file at the beginning FAT32 file size support tops out at 4GB and volume size tops out at 2TB.	Easy to grow a file - Change a value in the block bitmap and add a pointer to the inode UNIX INODES is limited by design to 16EB (Exabytes). One Exabyte is the equivalent of one billion Gigabytes
Access time	A benefit is that random-access time is improved, because the disk head can find the location of any block by reading the information in the FAT The FAT allocation scheme can result in a significant number of disk head seeks, unless the FAT is cached (DIS)	
Fragmentation	external fragmentation: free space you know about but cant use it (DIS) Internal fragmentation(DIS)	No external fragmentation(ADD) Internal fragmentation(DIS) Internal fragmentation still possible but less so than FAT(DIS) .

Semester Test 2: 2015

The Group Answers

Short Questions

Question 1:

(a) Hoeveel bisse wyd is segmentregisters op Intel-argitektuur? / How many bits wide are segment registers on Intel architecture? [1]

A. 8 B. 16 C. 32 D. 64

B) 16 bits

(b) Watter een van die volgende Intel-instruksies het geen invloed op die stapelwyser nie? / Which one of the following Intel instructions has no effect on the stack pointer? [1]

A. `call` B. `push` C. `ret` D. `test`

D) test (like cmp, test only affects the flags not the stack)

(c) Wat is die grootste bladsygrootte op die Intel IA-32 argitektuur? / What is the largest page size on Intel IA-32 architecture? [1]

A. 4 KiB B. 4 MiB C. 4 GiB D. 4 TiB

Intel IA-32 architecture allows page sizes of either 4 KiB or 4 MiB, thus the largest page size is clearly 4 MiB. According to DINO book it is either 4KB or 4MB but i'm guessing it is just a difference in notation.
4Mib according to wikipedia?

(d) Watter Mic-1 register hou altyd die boonste stapelwaarde aan? / Which Mic-1 register always keeps the stack top value? [1]

A. CPP B. LV C. SP D. TOS

D) TOS (pg 274) (SP is the pointer to the value, TOS contains the value)

A) Only H can be on the right hand side of a subtract (pg 269)

(e) Watter een van die volgende kan *nie* in Mic-1 mikrokode geënkodeer word *nie*? / Which one of the following *cannot* be encoded in Mic-1 microcode? [1]

A. H = H – MDR B. H = MDR – H C. H = H + MDR D. H = MDR + H

(f) Gestel 'n proporsionele allokasieskema word gebruik op 'n stelsel met 62 rame en twee prosesse, een van 10 blaaie, en een van 127. Hoeveel rame sou aan die groot proses toegeken word? / Suppose a proportional allocation scheme is used on a system with 62 frames, and two processes, one of 10 pages, and one of 127 pages. How many frames would be allocated to the larger process? [1]

A. 4 B. 31 C. 57 D. 62

C) 57, (as calculated in 2016)

(g) Wanneer 'n tweede-kans bladsyvervangingsalgoritme, met 'n verwysings- en 'n modifieerbis (in dié volgorde), gebruik word, watter een van die volgende bladsyklasse behoort eerste uitgeskop te word?
A. $(0,0)$ B. $(0,1)$ C. $(1,0)$ D. $(1,1)$

When a second-chance page replacement algorithm, with a reference and a modify bit (in this order), is used, which one of the following page class should be evicted first? [1]

A) The page is unmodified and has not be needed recently

2 a) Bus skew:

See 2016 above

b) Unified cache:

When instructions and data use the same cache.

c) Sign extension:

Done by duplicating the MBR sign bit (leftmost bit) into the upper 24 bit positions of the B bus.

d) Internal fragmentation with respect to memory allocation:

when space is wasted internally to some page during memory allocation.

e) External fragmentation with respect to memory allocation

External fragmentation is the various free spaced holes that are generated in either memory or disk space.

Calculations

3. Gegee vyf geheuepartisies van grootte (1) 100 K, (2) 500 K, (3) 200 K, (4) 300 K, (5) 600 K, (in orde), hoe sou elkeen van die eerste-passing en beste-passing algoritmes die plasing van prosesse van 212 K, 417 K, 112 K en 426 K (in dié volgorde) behartig? *Neem aan dat die partisielys as 'n sirkulêre lys aangehou word.* Dit wil sê, die soektog begin telkens by die partisie waaruit daar tydens die vorige iterasie geheue toegeken is.

Given five memory partitions of size (1) 100 K, (2) 500 K, (3) 200 K, (4) 300 K, (5) 600 K, (in order), how would each of the first-fit and best-fit algorithms place processes of 212 K, 417 K, 112 K, and 426 K (in this order)? *Assume the partition list is kept as a circular list.* That is to say, the search is started every time at the partition from which memory was allocated during the previous iteration. [2]

Process size	Partition number for:		
	first-fit	best-fit	worst-fit(extra)
212 K	(2)	(4)	(5)
417 K	(5)	(2)	(2)
112 K	(3)	(3)	(4)

426 K	no space	(5)	no space

4. (a) Beskou die ry van bladsyverwysings hieronder. Gegee dat daar drie rame is—en onthou, hulle almal is aanvanklik leeg—*hoeveel bladsyfoute* vind plaas onder die gespesifiseerde vervangingstrategieë? Toon u "bewerkings" in die kassies voorsien.

Consider the sequence of page references below. Assuming three frames—and remembering that all frames are initially empty—*how many page faults* occur for the specified replacement algorithms? Show your "calculations" in the boxes provided. [4]

4, 3, 2, 1, 4, 3, 5, 4, 3, 2, 1, 5

Least-recently-used (LRU)

N	4	3	2	1	4	3	5	4	3	2	1	5
F1	4*	4	4	1*	1	1	5*	5	5	2*	2	2
F2		3*	3	3	4*	4	4	4	4	4	1*	1
F3			2*	2	2	3*	3	3	3	3	3	5*
Fault	1	1	1	1	1	1	1	0	0	1	1	1
Total	10											

First-in,first-out (FIFO)

N	4	3	2	1	4	3	5	4	3	2	1	5
F1	4*	4	4	1*	1	1	5*	5	5	5	5	5
F2		3*	3	3	4*	4	4	4	4	2*	2	2
F3			2*	2	2	3*	3	3	3	3	1*	1
Fault	1	1	1	1	1	1	1	0	0	1	1	0
Total	9											

(b) Herhaal die berekening vir die vorige vraag, maar gebruik nou vier rame. Definieer en illustreer daarná **Belady se anomalie** aan die hand van u berekeninge.

Repeat the calculations of the previous question, but now use four frames. Then, define and illustrate **Belady's anomaly** with reference to your calculations. [5]

Least-recently-used (LRU)

N	4	3	2	1	4	3	5	4	3	2	1	5
F1	4*	4	4	4	4	4	4	4	4	4	4	5*
F2		3*	3	3	3	3	3	3	3	3	3	3
F3			2*	2	2	2	5*	5	5	5	1*	1
F4				1*	1	1	1	1	1	2*	2	2
Fault	1	1	1	1	0	0	1	0	0	1	1	1
Total	8											

First-in,first-out (FIFO)

N	4	3	2	1	4	3	5	4	3	2	1	5
F1	4*	4	4	4	4	4	5*	5	5	5	1*	1
F2		3*	3	3	3	3	3	4*	4	4	4	5*
F3			2*	2	2	2	2	2	3*	3	3	3
F4				1*	1	1	1	1	1	2*	2	2
Fault												
Total	10											

This is Belady's Anomaly, that addings does not necessarily mean fewer page faults.

Long Questions

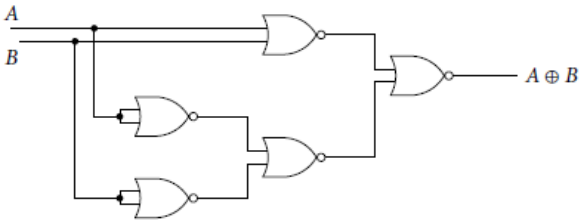
5. (a) Verduidelik kortliks die verskil tussen 'n **flip-flop** en 'n **wipkring**.

Briefly explain the difference between a **flip-flop** and a **latch**. [2]

(b) Gee die waarheidstabel vir die eksklusiewe-OF operator, en teken dan sy stroombaandiagram deur slegs NOF-hekke te gebruik. Toon alle tussenstappe.

Give the truth table for the exclusive-OR operation, and then draw its circuit diagram, using only NOR gates. Show all intermediary steps. [2]

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



$$
\begin{aligned}
A \oplus B &= (A'B' + AB)' \\
&= \big((A'B')'' + (AB)''\big)' \\
&= \big((A'' + B'')' + (A' + B')'\big)' \\
&= \big((A + B)' + (A' + B')'\big)'.
\end{aligned}
$$

(c) Voltooi die volgende tabel, waarin u aandui hoe om die toevoerwaardes A en B, die elektriese grond V_{Gnd} en die ekstern-gereguleerde stroombaanlading V_{CC} te koppel sodat 'n twee-toevoer multiplekser die funksies $\overline{A}$, AB en $A + B$, onderskeidelik, bereken. Die kontrolelyn C kies tussen D_0 en D_1 op die standaardmanier.

Complete the following table, in which you show how to wire up the input values A and B, the electrical ground V_{Gnd}, and the externally regulated circuit voltage V_{CC} so that a two-input multiplexer computes the functions $\overline{A}$, AB, and $A + B$, respectively. The control line C chooses between D_0 and D_1 in the standard way. [3]

Inputs	**Not A**	**A + B**	**AB**
D0	V_{cc}	B	V_{gnd}
D1	V_{gnd}	V_{cc}	B
C	A	A	A

$V_{cc} = 1$; $V_{gnd} = 0$.

(d) Noem en bespreek kortliks *twee* strategieë om die uitvoerspoed van die datapad te vermeerder.

Name and briefly discuss *two* strategies for increasing the speed of execution of the data path. [2]

1. *Reduce* the number of clock cycles needed to execute an instruction
2. *Simplify* organization so that the clock cycle can be shorter
3. *Overlap* execution of instructions

(e) **LNS-afhanklikhede** word ook dikwels "ware afhanklikhede" genoem. Motiveer dié karakterisering, met verwysing na ander instruksie-afhanklikhede.

RAW dependencies are also often called "true dependencies". Motivate this characterisation, with reference to other instruction dependencies. [3]

(f) U word aangewys as konsultant vir 'n span wat 'n nuwe programmeertaal en meegaande virtuele masjien moet ontwerp; laasgenoemde sal ook as hardeware geïmplementeer word. Die taal is LL(1), sal 'n rekursief-dalende vertaler hê en moet voorwaardelike vertakking vir twee-komplement heelgetalle oor die relasionale operators =, ≠, <, ⩽, > en ⩾ ondersteun. Die masjienontwerp het reeds instruksies vir bisgewyse EN, OF, NIE en XOF. Bied nou 'n ontwerp aan waarin u *so min as moontlik* ISA-vlak instruksies byvoeg om dié vereistes te vervul. Vir volpunte moet u ontwerp gedetailleerde idees oor die betrokke digitale logika en afbeelding van ISA-instruksies na mikro-operasie instruksiesekwensies insluit. *Oorweeg alle kwessies geïmpliseer deur die vereistes.*

You are appointed as consultant to a team that must design a new programming language and associated virtual machine; the latter will also be implemented as hardware. The language is LL(1), will have a recursive-descent compiler, and must support conditional branching for two's complement integers over the relational operators =, ≠, <, ⩽, >, and ⩾. The machine design already contains instructions for bitwise AND, OR, `not`, and `xor`. Now, present a design in which you add *as few as possible* ISA-level instructions to accomplish these requirements. For fullmarks, your design has to include detailed ideas on the digital circuits involved and the mapping of ISA instructions to sequences of micro-operations. *Consider all issues implied by the requirements.* [5]

6. (a) Bespreek kortliks die verskil tussen **segmentering** en **paginering**.

Briefly discuss the difference between **segmentation** and **paging**. [4]

'Segmentation is allocating memory space per process, whereas paging is dividing memory into blocks that can be saved to and loaded from virtual memory when needed by the CPU. **Paging** does not distinguish, protect procedures and data separately. Paging does not facilitate sharing of procedures unlike segmentation. Paging is transparent to the programmer but has to be aware of the memory allocated to a process in segmentation (else seg. fault).

(b) Verduidelik volledig wat met **aanvraagpaginering** bedoel word. Hoe verskil dit van of stem dit ooreen met die gebruik van kasgeheue?

Explain fully what is meant by **demand paging**. How is it the same as or different from the use of cache memory? [4]

Basically when a page is brought into primary memory from the disk only when it is requested, not in advance.

A program can be run on a machine using virtual memory, the page table must just be set to indicate that each virtual page is in secondary memory. When the CPU tries to fetch the first instruction, it gets a page fault, so the first instruction is loaded into memory. Following instructions will only be brought in if there is a page fault (ie, when needed).

(c) Verduidelik kortliks wat **versparteling** is. Briefly explan what **thrashing** is. [2]

Constant page faults resulting in nothing productive being done

If a process doesn't have enough pages, the page fault rate is very high. This leads to the process swapping pages in and out of main memory. This equals low CPU utilization.

(d) Omskryf kortliks die bedryf van 'n **omskakeling-sydelingse buffer**. Briefly describe the operation of a **translation look-aside buffer**. [3]

A hardware table that quickly maps virtual page numbers onto physical-page-frame numbers. The TLB holds only the 128 most recently used virtual page numbers in each category.
Each TLB entry holds a virtual page number and the corresponding physical-page-frame number.

(e) Watter strategie word deur tipiese UNIX-lêerstelsels gebruik om te verseker dat 'n indeksblok 'n groot omvang van lêergroottes en vinnige toegang toelaat, maar self nie te veel ruimte in beslag neem nie? Which strategy is used by a typical UNIX file system to ensure that an index block allows a large range of file sizes and rapid access, but does not occupy too much space itself? [3]

An Inode has only 12 direct blocks, each with a pointer to a data block. Additionally, an Inode has a pointer to an indirect block, a double indirect block and a triple indirect block. Each indirect block contains another 12 pointers to data blocks, and a double indirect block contains 12 pointers to indirect blocks. This allows for large files to be accommodated for, while small files only need to use the 12 direct blocks saving space.

(f) Kan die tipiese UNIX-lêerstelsel as 'n algemene grafiekgids beskou word? Bespreek kortliks, met verwysing na skakels. Can the typical UNIX file system be viewed as a general graph directory? Discuss briefly, with reference to links. [2]

No, hard links to directories are typically not allowed on a Unix file system resulting in an acyclic tree, not a general graph

7. Verskaf die masjienkode vir die Intel-saamsteltaalinstruksies hieronder. Skryf u antwoorde in die kassies voorsien en pas elke instruksiekomponent by sy beskrywing. Indien 'n betrokke instruksie nie 'n sekere komponent het nie, laat daardie kassie oop. Gebruik heksadesimale notasie in u antwoorde *en neem aan dat die verwerker in 32-bis modus bedryf word wanneer die instruksies uitgevoer gaan word.*

(a) `xor dx, 0x00`

(b) `setnz bl`

(c) `xor [esi+edi*2+0x2357], edx`

Provide the machine code for the Intel assembly language instructions below. Write your answers in the boxes provided, matching each instruction component with its description. If a particular instruction does not have a certain component, leave that box open. Use hexadecimal notation in your answers, *and assume that the processor operates in 32-bit mode when the instructions are to be executed.* [10]

	(a)	(b)	(c)
Prefixes	0x66		
Opcode	0x83	0x0f 0x95	0x31
ModR/M	0xf2	0xc3	0x94
SIB			0x7e
Displacement			0x57 0x23 0x00 0x00
Immediate	0x00		

8. (a) Die volgende diagram illustreer vier blokke geheue asook die waardes wat by die onderskeie posisies gestoor word. Die adresse word met inkremente van 4 grepe gegee. Alle waardes is in heksadesimaal en u mag greep-ordening ignoreer. Die diagram gee ook die waardes wat tans in die SVE-registers is.

The next diagram illustrates four blocks of memory and the values that are currently residing in them. The addresses are given in increments of 4 bytes. All values are given in hexadecimal, and you may ignore byte ordering. The diagram also gives the values currently in the CPU registers. [6]

0x20000010	0x00050820
0x2000000c	0x00000208
0x20000008	0x0005082c
0x20000004	0x1234abcd
0x20000000	0x00015184
0x1ffffffc	0x00050820
0x1ffffff8	0x0000a134
0x0000a140	0x36980000
0x0000a13c	0x00ae308c
0x0000a138	0x00000180
0x0000a134	0x44332211

0x0000020c	0x20000000
0x00000208	0x0000a138
0x00000204	0x20000008
0x00000200	0x00050820
0x00050838	0x00002000
0x00050834	0x0000000c
0x00050830	0x00000040
0x0005082c	0x37648608
0x00050828	0x001ee04f
0x00050824	0x00050828
0x00050820	0x37648788

eax	0x00050820
ebx	0x00000180
ecx	0x37648788
edx	0x0000020c
edi	0x1ffffffc
esi	0x0000a138
ebp	0x00050830
esp	0x00050830

Die onderstaande saamsteltaal-instruksies word nou in volgorde uitgevoer. Bepaal wat die inhoud van die onderskeie registers is sodra die instruksies almal uitgevoer het en skryf u antwoorde in die ooreenstemmende blokkies hieronder.

The assembly language instructions shown below are now executed in order. Determine the contents of the registers after this sequence of instructions has completed execution, and write your answers in the boxes provided below.

```
pop  eax
lea  edi, [ebx+eax*2]
mov  esp, [edi+4]
pop  ecx
mov  edx, [ecx-4]
```

eax:	0x00000040	ebx:	0x00000180	ecx:	0x0005082c
edx:	0x001ee04f	edi:	0x00000200	esp:	0x2000000c

(b) Voltooi die `inst2hexstr`-funksie hieronder deur die ontbrekende instruksies op die stippellyne in te vul. Die funksie ontvang 'n onbetekende 32-bis heelgetal en moet dié omskakel na 'n ASCII-string wat die heksadesimale voorstelling, links met nulle ingevul, van die getal gee. Let op die volgende: (1) Die lokale `int`-veranderlike `i` word met 'n register vervang word. (2) U mag aanneem dat ruimte vir die string `s` deur die roeper geallokkeer en groot genoeg vir die resultaat is. (3) Die karakterskikking `hexchars` word onderaan die kode gedefinieer. (4) `ROTATE_LEFT(n, m)` is 'n C-makro wat die `m` mees beduidende bisse in *n* na die *m* mins beduidende posisies roteer. (5) U moet presies agt instruksies neerskryf, een en slegs een per stippellyn.

Complete the `int2hexstr` function below by writing the missing instructions on the dotted lines. The function receives an unsigned 32-bit integer and must convert this to an ASCII string that gives the hexadecimal representation, zero-padded on the left, of the integer. Note the following: (1) The `int` variable `i` has been replaced by a register. (2) You may assume that space for the string `s` has been allocated by the caller and that the space is large enough for the result. (3) The character array `hexchars` is defined at the bottom of the code. (4) `ROTATE_LEFT(n, m)` is a C macro that rotates the `m` most significant bits of *n* into the *m* least significant positions. (5) You must write down exactly eight instructions, one and only one per dotted line. [8]

9. a) Tell the story—or at least, one of the stories—of this module as a fairy tale, classic or modern. Of course, you'll need heroes and villains. Whether good triumphs over evil is up to you. (And don't write what you think I think you should think. Write what you really think.)

```
        %define juliet [capulet]
        %define rosaline [capulet + 4]
        %define romeo [montague]
        sub     romeo, rosaline
        add     romeo, juliet
        cmp     romeo, juliet
        jne     .inter_family_conflict
.inter_family_conflict:
        sub     romeo, juliet
        mov     eaxile, romeo
        mov     sadness, juliet
        inc     sadness
```

pop juliet
ret romeo
pop romeo
mov tears, audience

b) You drive alone in your car on a remote and deserted road. Suddenly, you see an engineer and a socio-informatician, walking in front of you in the road. Which do you run over first, and why?

I would run over the socio-informatician, then run them over again to see if the error is reproducible.

I would run over the engineer to put him out of his misery.

Can you run me over while you're at it so I don't have to do this test?

Hang in there, stranger <3

Semester Test 2: 2014

The Group Answers

Short Questions

(My attempt)

1 a) Bus skew: The small difference that arises from different propagation delays of different bus lines.

b) Edge-triggered digital circuit: a circuit which becomes active at the negative or positive edge of the clock signal i.e. when the signal changes from low to high or vice versa.

c) Sign extension: conversion of an 8-bit MBR to a 32-bit word by duplicating the MBR sign bit into the upper 24 bit positions of the B bus, while preserving its sign and value.

d) Associative cache memory: a cache with multiple possible entries for each address.

e) Write-through (cache): immediately updating a data word in main memory and cache once it has changed.

f) Write allocation (cache): when data is loaded into the cache on a write miss, then written into memory.

g) Internal fragmentation: when space is wasted internal to some page during memory allocation.

h) External fragmentation: when memory is wasted as segments are loaded and removed due to ‘holes’ forming in the memory i.e. empty chunks.

i) Relocatable code: code generated of which the address is not known at compile time, and final binding is delayed until load time.

j) Thrashing: when a process does not have enough frames to support the pages in active use, and consequently spends more time paging than executing, causing delays in execution.

k) Sequential access: information in a file is processed in order, one record after the other.

l) Absolute path name: a path which starts at the root and follows a path down to the specified file.

2 a) 16-bits

b) A state with both outputs equal to 0 is inconsistent, because it forces both gates to have two 0s as input.

c) ?

d) If two different logical paths use the output of a flip-flop one (path) interprets as a 0 & the other as a 1, which puts the machine in an inconsistent state.

e) Copy-on-write is a technique to create processes quickly in which parent and child processes initially share pages. When either process writes to the shared page, a copy is made onto a free page (requested from a pool of free pages), and mapped to the address space of that process.

f) There are four microinstruction registers because the fields on each micro-operation are not active at the same time, i.e. there are four instructions that have control of the data path at any instance.

Calculations

?

Long Questions

5 a) Synchronous buses are driven by a clock (crystal oscillator) while asynchronous buses are not. Synchronous buses are easy to work with due to their discrete time intervals. However, this may cause unnecessary latency if transfers are completed before the next cycle. This latency may also nullify any performance improvements in newer technologies if they are connected to the bus, since it's only as fast as its slowest component. Asynchronous buses provide an advantage in timing since full handshakes are timing independent. However, they are not as common as synchronous buses since they are not as easy to build and there isn't as much investment in them as in synchronous buses.

5 b) Static RAMs have an internal structure similar to that of a basic D flip-flop. They retain their contents as long as the power is on. Static RAMs are very fast and are popularly used as cache memory.

Dynamic RAMs comprise an array of cells, each containing one transistor and a tiny capacitor. This allows for a very high density (bits/chip) i.e. a large capacity. DRAMs require complex interfacing to refresh the electric charge which leaks out of the capacitors. DRAMs are almost always used for main memory, however are slow compared to static RAMs.

5 c) ???

5 d) (pg 283)
Reduce the number of clock cycles needed to execute an instruction: (The number of clock cycles needed to execute a set of operations is called the path length) Sometimes the path length can be shortened by adding specialized hardware.
Simplify the organization so that the clock cycle can be shorter: ???

Overlap the execution of instructions: Separating out the circuitry for fetching the instructions is most effective if the unit is made functionally independent of the main data path. In this way, it can fetch the next opcode or operand on its own, perhaps even performing asynchronously with respect to the rest of the CPU and fetching one or more instructions ahead.

5 e) RAW dependency. R4 is dependent on the result of the previous instruction.

5 f) WAR dependency. The system may attempt to read and write to/from R1 simultaneously. This may be prevented by replicating hardware in that segment of the pipeline.

6 a) Paging and segmentation are both memory management schemes. Paging is a technique in which memory is divided into fixed-size logical blocks called frames, allowing a process to be stored in a non-contiguous manner. Page size is determined by hardware. In contrast, segments are of variable sizes determined by the user, and may be shared/protected between processes. Another difference is that segmentation may cause external fragmentation, whereas paging may result in internal fragmentation.

6 b) A translation lookaside buffer increases the speed of virtual memory operations by mapping recently used virtual page numbers onto physical-page-frame numbers. In this way, the TLB acts as a dedicated cache for virtual memory and its corresponding physical memory address lookup. If the TLB doesn't contain the required virtual address entry, a TLB miss occurs and the processor has to walk through the page table, which takes considerably more time.

7 a) To explain this like a C program:

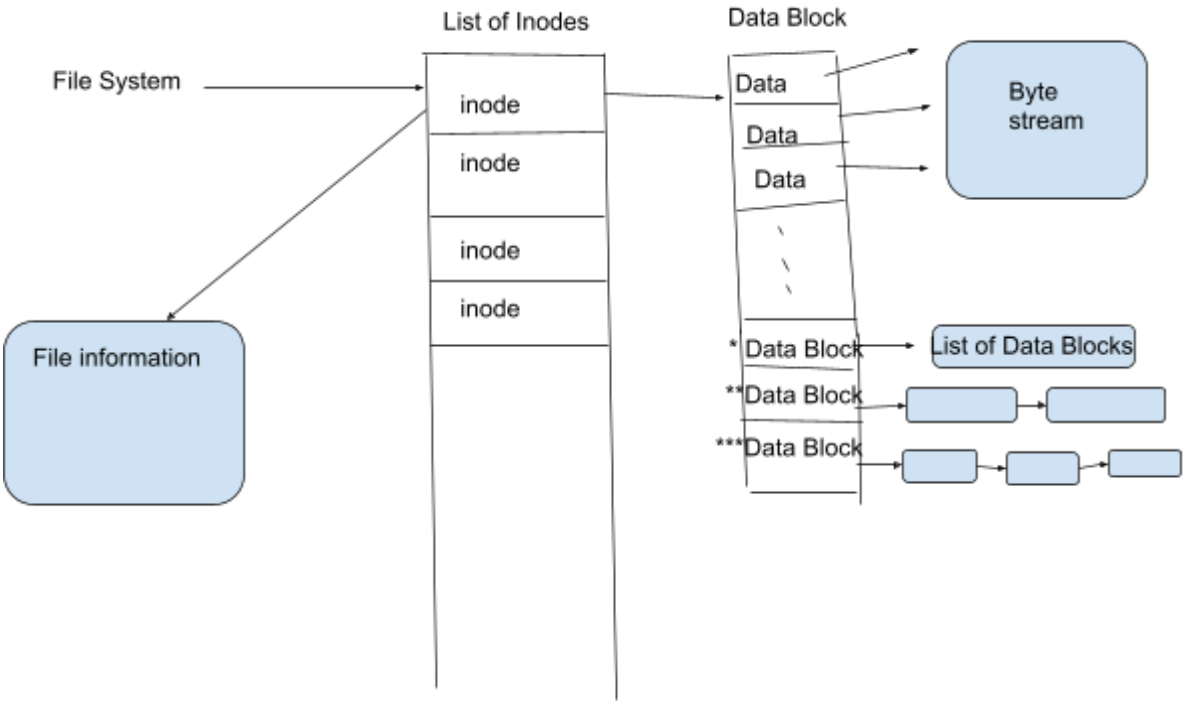
```
/* basic data for a file seen as a bit stream */
struct data {
      char *data;
}

struct inode {
      /* each inode contains 15 pointers to data blocks */
      *data_block data_ptr;
      int permissions;
      char *file_type;
      int file_size;
      /* and other parameters */
}

/* First 12 pointers directly to data blocks */
/* 13th pointer to an indirect block, containing pointers to other
blocks */
/* 14th pointer points to a doubly-indirect block, a block
containing 128 addresses of singly indirect blocks */
/* the 15th pointer points to a triply indirect block (which
contains pointers to doubly indirect blocks, etc.) */
```

```
struct data_block {
      *data direct_data;
      *data_block 13_indirect_data;
      **data_block 14_indirect_data;
      ***data_block 15_indirect_data;
}
```

File system graphically described using “C-like” logic:



7 b) (DISCLAIMER: Directly from Stack Overflow)

EXPLAINED USING BRUCE LEE
(ALL YOU NEED TO KNOW. TY, EMER)

https://medium.com/meatandmachines/explaining-the-difference-between-hard-links-symbolic-links-using-bruce-lee-32828832e8d3

Underneath the file system files are represented by inodes (or is it multiple inodes not sure)
A file in the file system is basically a link to an inode.

A hard link then just creates another file with a link to the same underlying inode.

When you delete a file it removes one link to the underlying inode. The inode is only deleted (or deletable/over-writable) when all links to the inode have been deleted.

A symbolic link is a link to another name in the file system.

Once a hard link has been made the link is to the inode. deleting renaming or moving the original file will not affect the hard link as it links to the underlying inode. Any changes to the data on the inode is reflected in all files that refer to that inode.

Note: Hard links are only valid within the same File System. Symbolic links can span file systems as they are simply the name of another file.

Semester Test 2: 2013

The Group Answers

Short Questions

1. a) False: It is synchronous
   b)
   c)

2. a) Instructions and data use the same cache
   b) Immediately updating the required entry in the main memory rather than waiting until later to replace the whole cache line - simpler and more reliable, but also more memory traffic.
   c) Bringing memory into cache on a write miss, rather than just writing the data out directly to memory.
   d) **Internal fragmentation:** when space is wasted internal to some page during memory allocation.
   e) **External fragmentation:** when memory is wasted as segments are loaded and removed due to 'holes' forming in the memory i.e. empty chunks.
   f) **Relocatable Code:** Relocatable code resides at the address relative to the beginning of absolute code, when it is not known at compile time where a process is supposed to reside in memory. Thus, the code can run on any system and is guaranteed not to fight over memory/stack space, as the OS can assign the absolute code reference, from which relative references can be determined.
   g) Adding more frames does not necessarily mean fewer page faults, and can actually lead to more page faults. Counterintuitive, thus an anomaly.
   h) When a process does not have enough frames to support the pages in active use, and consequently spends more time paging than executing, causing delays in execution.

Long Questions

3. *God be with you (Unfortunately this is definitely in the test, according to the breakdown) Crying is permitted*

4.

Semester Test 2: 2012

The Group Answers

Short Questions

1. (a) When is a clocked SR latch in an inconsistent state? [1]

   (b) How can the Z-bit of the Mic-1's Arithmetic Logic Unit (ALU) be wired up so that it is 1 when the result of the ALU is zero, and 0 otherwise? [1]

   (c) Why does the pipelined Mic-4 architecture need four Memory Instruction Registers (MIRs)? [1]

   (d) How wide are the segment registers of the Intel IA-32 architecture? [1]

   a) S = R = 1
   b) Not sure how to wire but ALU result should be "nanded" with zero. The result of this is the z-bit.
   c) ?
   d) 16 bits

2. Define each of the following concepts in one sentence. [7]

   (a) An **edge triggered** digital circuit.

   (b) **Bus skew.**

   (c) **Write back** with respect to cache memory.

   (d) **Write allocate** with respect to cache memory.

   (e) **Internal fragmentation** with respect to memory allocation.

   (f) **External fragmentation** with respect to memory allocation.

   (g) **Static linking** of object modules.

   a) Only activated when clock is in transition phase.
   b) A problem where the signals on different lines of a bus travel at different speeds.
   c) When a cache line is modified but memory is not and eventually the dirty line has to be written back to memory, potentially after many writes have been made to it.
   d) If the probability is high that a word just written will be written again soon, the word should be loaded into the cache on write misses.
   e) When a program does not fill an integral number of pages exactly and there is wasted space in memory.
   f) After a system has been running for a while, memory is divided into "chunks" with some containing segments and others containing holes.
   g) Object modules or libraries are included in the binary executable statically at compile-time (this is what he was complaining the IT guys did with an early version of Inetkey)

Question 3 is about Boolean Algebra, therefore not in the test (I think)

Calculations

4. (a) Convert the number −13 to IEEE single-precision floating-point format. Give the result as eight hexadecimal digits. Show your calculations. [2]

   (b) Convert the following IEEE single-precision floating-point numbers from hexadecimal to decimal. Show your calculations. [2]

Long Questions

Semester Test 2: 2011

The Group Answers

Short Questions

1. a) D: 32-bit general-purpose register
   b) C
   c) E
   d) A
   e) D: (dino p. 383 & 384)

2. a) i) logical
      ii) linear
      iii) physical

Calculations

~~3. Old work~~

4. a) No. The size of the referred-to memory region is ambiguous. Should this instruction move the value that ax points to, into the single byte at address eax, or the 32-bit representation into the 4 bytes starting at address eax? Since either is a valid possible interpretation, the assembler must be explicitly directed as to which is correct, using *byte*, *word*, or *dword*.

   b) *Pretending this question doesn't exist :S :S*

Long Questions

5. a) The TLB is a solution to an extreme number of virtual addresses needed for a normal page table. It maps the virtual page numbers onto the physical frame numbers, but only holds the most recently used virtual page numbers. A virtual address is now presented and looked up in the TLB, and if a match is found it is a hit; with no match, it is a TLB miss, and the OS has the job of handling this (note: it is not a page fault since the page may be in memory).

   b) *First-fit:*
   2 (now it has left 288K), **5** (now 183K), **5** (now 71K), now there's no space for 426K.

Best-fit:
4 (now it has left 88K), **2** (now 83K), **3** (now 88K), **5**

c) The system will check out some internally held table (kept with the process) to determine if the reference was valid. Only if it was, will we bring in the new page. We find a free frame, read the page into this frame, then update the internal table and page table to show this. Now, we restart the instruction.

d) Page fault (page not in frames yet)
Currently requested page
i) LRU:

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3
		1	2	3	4	2	1	5	6	6	1	2	3	7	6	3	3	1	2
			1	1	3	4	2	1	5	5	6	1	2	2	7	6	6	6	1

ii) FIFO:

1	**2**	**3**	**4**	4	4	**5**	**6**	**2**	**1**	1	**3**	**7**	**6**	6	**2**	**1**	1	**3**	3
	1	2	3	3	3	4	5	6	2	**2**	1	3	7	7	6	2	**2**	1	1
		1	2	**2**	2	3	4	5	6	6	2	1	3	**3**	7	6	6	2	2
			1	1	**1**	2	3	4	5	5	6	2	1	1	3	7	7	6	**6**

6. a) A file descriptor is an entry in the file-system table. A pointer to this entry is used by all file operations. When a process closes the file, the entry is removed, and when all processes close the file, its updated data is copied back onto the disk.

b) In the UNIX File System, inodes are pre-allocated on a disk. This ensures good performance, since every file will be kept close to its inode block and thus when looking for the file, seek time will be reduced.

7. a) Calling conventions are protocols that are generally followed, that allow programmers to share their code and develop libraries that many programs can use. In *cdecl*, we need to push parameters from the right to the left before a call. The caller has to preserve the eax, ecx, edx; the callee has to preserve the ebx, edi, esi, ebp. We use the eax to store our return values in. We also need to move the ebp into the esp before the rest of the code, and executing the code move it back.

~~b) *Ctrl-F "signed integer" does not bring up much in the relevant chapters*~~

~~c) *Same as above*~~

d) Call-by-value: a copy of the original parameter is created and this copy is passed to the function being called. Any updates inside the function will not affect the original value that was sent.

Call-by-reference: the actual address of the variable is passed to the called function, and any updates inside will directly modify the original because we are modifying the content at the memory location given.

8. ~~a) Old work~~
~~b) Old work~~
~~c) Old work~~

Multiplexers

At the digital logic level, a multiplexer is a circuit with 2 n data inputs, one data output, and n control inputs that select one of the data inputs. The selected data input is ‘‘gated’’ (i.e., sent) to the output. Figure 3-11 is a schematic diagram for an eight-input multiplexer. The three control lines, A, B, and C, encode a 3-bit number that specifies which of the eight input lines is gated to the OR gate and thence to the output. No matter what value is on the control lines, seven of the AND gates will always output 0; the other one may output either 0 or 1, depending on the value of the selected input line. Each AND gate is enabled by a different combination of the control inputs.

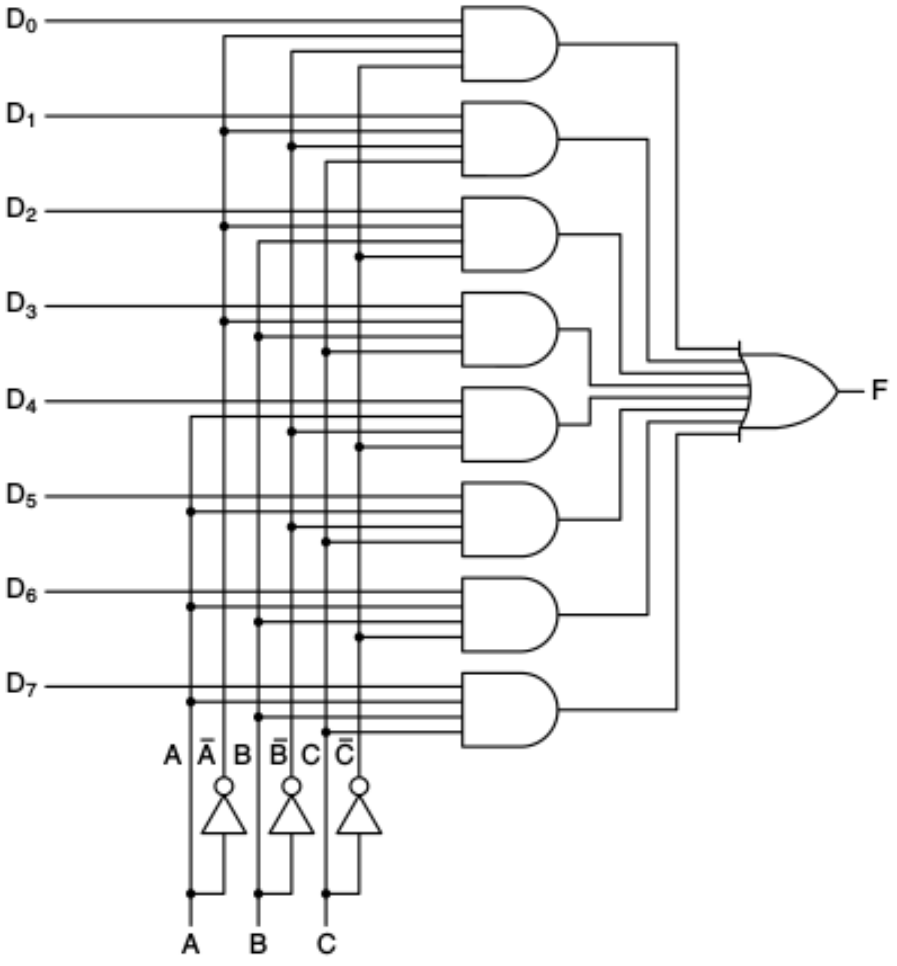


Figure 3-11. An eight-input multiplexer circuit.