a) yield = $\frac{1}{(1+\frac{\text{def.rate}\cdot\text{die area}}{2})^2} = \frac{1}{(1+\frac{0,5\cdot 0,75}{2})^2} = 0,709 \rightarrow 71\%$

Exercise 1

Introduction

Consider a fabrication process with 25cm wafers (diameter) and a defect rate of 0.5 defects/cm^2.

$A = \pi r^2 = \pi \cdot 12,5^2 = 490,87 cm^2$

(a) Calculate the yield of the process, if the chip area is 0.75cm^2. How many functioning chips do you receive per wafer? $\lfloor \frac{490,87}{0,75} \rfloor = 654$ $\lfloor 654 \cdot 0,709 \rfloor = 463$ ✓

(b) Calculate the yield of the process, if the chip area is 1.5cm^2. How many functioning chips do you receive per wafer? $\frac{1}{(1+\frac{0,5\cdot 1,5}{2})^2} = 0,5289 \Rightarrow 53\%$ $\lfloor \frac{490,87}{1,5} \rfloor = 327$ $\lfloor 327 \cdot 0,5289 \rfloor = 172$ ✓

(c) Assume that the costs per wafer are 3000$: What is the minimum sale price of the chips of (a) and (b) to prevent financial losses? a) $\frac{3000\$}{654} = 6,48\$$ b) $\frac{3000\$}{172} = 17,45\$$

$A = \pi \cdot 17,5^2 = 962,11 cm^2$ yield same $\lfloor \frac{962,11}{0,75} \rfloor = 1282$ $\lfloor 1282 \cdot 0,709 \rfloor = 908$

(d) Assume that the wafer diameter can be increased by 10cm while having the same defect rate: Recalculate the yield and the number of functioning chips for the chip area given in (a).

(e) Calculate the maximum acceptable cost per wafer, given that the sale price for the chips should not be increased when using the bigger wafers. $\frac{x}{908} = 6,48 \Rightarrow 5883,84\$$

(f) Assume that for the parameters of (a), the fabrication process can be improved such that the resulting yield is 0.85. Calculate the defect rate. $\frac{1}{(1+\frac{x\cdot 0,75}{2})^2} = 0,85$

Note that the formulas presented in the lecture are approximated. In case you are interested in seeing more precise results and a graphical representation, you can check online calculators, e.g., https://caly-technologies.com/die-yield-calculator/

$1+\frac{x\cdot 0,75}{2} = \frac{1}{\sqrt{0,85}}$ $/-1\ /\cdot\frac{2}{0,75} = \frac{2}{\frac{3}{4}} = \frac{8}{3}$

$x = \left(\frac{1}{\sqrt{0,85}} - 1\right) \cdot \frac{8}{3} = 0,2257$ ✓

Exercise 2

Performance

A program uses 5% of its operations for floating point multiplications, 15% for floating point divisions and 30% for floating point additions. To speed up the execution of the program, the following suggestions are made:

$\frac{0,3T}{4} + 0,7T = 0,775$ ✓

(a) *Speed up the addition by factor 4.*

$\frac{0,05T}{8} + 0,95T = 0,95625$ ✓

(b) *Speed up the multiplication by factor 8.*

$\frac{0,3T}{1,5} + \frac{0,15T}{1,5} + 0,55T = 0,85$ ✓

(c) *Speed up the addition and division by factor 1.5, respectively.*

$\frac{0,05T}{2} + \frac{0,15}{2} + 0,8T = 0,9$ ✓

(d) *Speed up the multiplication and division by factor 2, respectively.*

Show the results for all options above.

(e) *Which option is the best one?*

a)

Exercise 3

Performance

Consider two processors P1 and P2 implementing the same instruction set. They have the following characteristics:

- P1: 2.2GHz clock rate; 1.9 average CPI
- P2: 1.3GHz clock rate; 1.1 average CPI

(a) Calculate the performance in terms of instructions per second for processors P1 and P2.

(b) Assume that processor P1 executes a benchmark program in 30 seconds. Calculate the corresponding number of cycles and the corresponding number of instructions.

(c) Assume that processor P2 executes the same benchmark program and requires the same number of instructions for its execution: Calculate the execution time.

Assume a third processor P3 implementing a different instruction set, which has a 3.5Ghz clock rate, an average CPI of 1.5 and executes $8 \cdot 10^{10}$ instructions for executing the benchmark program.
(d) Calculate the execution time.

(e) Calculate the MIPS rating of P1, P2 and P3.

(f) Given the results calculated so far: Show that frequency/clock rate is not a good performance metric.

a) $MIPS = \frac{\text{Instr. count}}{\text{exec. time} \cdot 10^6} = \frac{\text{Clock rate}}{CPI \cdot 10^6}$

P1: $\left\lfloor \frac{2{,}2 \cdot 10^9}{1{,}9 \cdot 10^6} \right\rfloor = 1157$ MIPS P2: $\frac{1{,}3 \cdot 10^9}{1{,}1 \cdot 10^6} = 1181$ MIPS

b) #cycles: $2{,}2 \cdot 10^9 \cdot 30 = 66 \cdot 10^9$

#instr.: $1157 \cdot 10^6 \cdot 30 = 34{,}71 \cdot 10^9$

c) $1181 \cdot 10^6 \cdot x = 34{,}71 \cdot 10^9 \Rightarrow x = \underline{29{,}39\,s}$

d) $\frac{3{,}5 \cdot 10^9}{1{,}5} = 2{,}3\dot{} \cdot 10^9$ IPS $\Rightarrow$ IPS $= \frac{\#\text{instr.}}{\text{exec. time}} \Rightarrow$ exec time $= \frac{8 \cdot 10^{10}}{2{,}3\dot{} \cdot 10^9} = \underline{34{,}29\,s}$

e) P1: 1157, P2: 1181, P3: 2333

f) P3 größte MIPS aber längste exec. time

Exercise 4

Instructions/Processor/Pipelining

(a) Consider the following RISC-V assembly code (32-bit RISC-V version). It was written for a 5-stage RISC-V pipeline, where forwarding and handling of control hazards are implemented. Describe in one sentence as precisely as possible which functionality it implements.

```
1  <func>
2    lw      a5,4(a0)
3    lw      a4,8(a0)
4    nop
5    add     a5,a5,a4
6    lw      a4,0(a0)
7    nop
8    add     a5,a5,a4
9    lw      a0,12(a0)
10   nop
11   add     a0,a5,a0
12   srli    a0,a0,0x2
13   jalr    zero,0(ra)
```

(b) Explain why the "nop" instructions at (4), (7) and (10) are required.

(c) Rewrite the code of (a) for a 5-stage RISC-V pipeline that neither supports forwarding nor hazard detection. Try to keep the performance as high as possible. Explain your changes.

(d) Optimize the code of (a) with respect to the code size. Consider a 5-stage RISC-V pipeline, where forwarding and handling of control hazards are implemented.

Exercise 5

Instructions/Processor/Pipelining

Consider the following RISC-V assembly code (32-bit RISC-V version).

```
1  <func>:
2           addi a5,zero,0
3           nop
4           nop
5           nop
6  label1:  bne  a5,a2,label2
7           nop
8           nop
9           nop
10          jalr zero,0(ra)
11          nop
12          nop
13          nop
14 label2:  addi a3,zero,0
15          nop
16          nop
17          nop
18          addi a4,zero,0
19          nop
20          nop
21          nop
22 label3:  slli a6,a4,0x2
23          nop
24          nop
25          nop
26          add  a6,a0,a6
27          nop
28          nop
29          nop
30          lw   a6,0(a6)
31          nop
32          nop
33          nop
34          add  a3,a3,a6
35          nop
36          nop
37          nop
38          addi a4,a4,1
39          nop
40          nop
41          nop
42          bge  a5,a4,label3
43          nop
44          nop
45          nop
46          slli a4,a5,0x2
47          nop
48          nop
49          nop
50          add  a4,a1,a4
51          nop
52          nop
53          nop
54          sw   a3,0(a4)
55          nop
56          nop
57          nop
58          addi a5,a5,1
59          nop
60          nop
61          nop
62          jal  zero,label1
63          nop
64          nop
65          nop
```

(a) Describe in one sentence as precisely as possible which functionality it implements. Hint: ***a0*** *holds the base address of an array containing the input,* ***a1*** *holds the base address of an array for the output,* ***a2*** *contains the number of array entries.*

(b) The code above was written for a basic 5-stage RISC-V pipeline without forwarding and hazard detection. Optimize the code for an enhanced version of the pipeline, where forwarding and handling of control hazards are implemented. You may re-arrange and remove instructions, but you are not allowed to add or modify instructions. Explain your optimizations and justify potentially remaining "nop" instructions.

(c) Describe which further improvements can be made by rewriting the code above completely. Write the corresponding assembly code.

Exercise 6

Instructions/Processor/Pipelining

(a) Consider the following RISC-V assembly code (32-bit RISC-V version). Complete the missing machine code.

```
<func>:
   101f4:  00052503  lw   a0,0(a0)
   101f8:  ________  lw   a5,0(a1)
   101fc:  ________  bltu a0,a5,10208
   10200:  ________  sub  a0,a0,a5
   10204:  ________  bgeu a0,a5,10200
   10208:  ________  jalr zero,0(ra)
```

(b) Describe in one sentence as precisely as possible which functionality the code shown in (a) implements.

(c) Consider the following RISC-V machine code (32-bit RISC-V version). Complete the missing assembly instructions.

```
<func>:
    101f4:  00052503  lw   a0,0(a0)
    101f8:  0005a783  ____________
    101fc:  00f57463  ____________
    10200:  00008067  ____________
    10204:  40f50533  ____________
    10208:  ff5ff06f  ____________
```

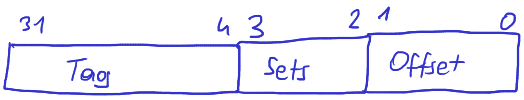
(d) Describe in one sentence as precisely as possible which functionality the code shown in (c) implements.

Exercise 7

Memory

You are given a processor system with a cache having the following design parameters:

- four-way set-associative
- LRU
- block size: 1 word ⇒ 32 Bit = 4 Byte ⇒ 2 Bit Offset
- number of sets: 4 ⇒ 2 Bit



block size ↓ weil 1 word = 4 Byte ↓ Block addr.: 192 : 4 — um Set & Offset bits zu entfernen

Set: block addr. % # sets

Tag: ⌊192 : 2^4⌋ =

(a) For the following byte addresses, write down (a) the block address, (b) the set, (c) the tag, (d) if the access results in a hit or a miss and (e) the tag of the evicted entry.

Byte address	Block address	Set	Tag	Hit/Miss	Evicted
192_{10}	48	0	12	Miss	
0_{10}	0	0	0	Miss	
480_{10}	120	0	30	Miss	
448_{10}	112	0	28	Miss	
192_{10}	48	0	12	HIT	
80_{10}	20	0	5	Miss	0
488_{10}	122	2	30	Miss	
480_{10}	120	0	30	HIT	
256_{10}	64	0	16	Miss	28
60_{10}	15	3	3	Miss	
8_{10}	2	2	0	Miss	
448_{10}	112	0	28	Miss	12
8_{10}	2	2	0	HIT	
72_{10}	18	2	4	Miss	
344_{10}	86	2	21	Miss	
168_{10}	42	2	10	Miss	30

(b) Show the cache state after the last access.

Set	Valid	Tag	Valid	Tag	Valid	Tag	Valid	Tag
0	1	28	1	5	1	30	1	16
1	0	/	0	/	0	/	0	/
2								
3								

Exercise 8

Memory

(a) Assume a system with a 2-way set associative cache using byte addressing. The partitioning of the main memory address looks as follows:

4Bit 3Bit => 2 · 4Byte words

Calculate the block size, the number of blocks, the number of sets, the capacity (only data) and the overall size of the cache.

(b) Given the system shown in (a): Give five alternative cache designs with the same capacity (only data) while keeping the same block size. Additionally, show the partitioning of the main memory address, respectively.

2-Way mit 4 Bit => $2 \cdot 2^4$ Blocks = 32 Blocks ✓

Blocksize: 8 Byte = 64 bit

Tag: 25 Bit

Size = #blocks (blocksize + Tag + 1) (Valid bit)

= 32 (64 + 25 + 1) = 2880 bit = 360 byte ✓

Capacity ist ohne Tag, also : 32 · 64 = 2048 bit = 256 byte ✓

Exercise 9

capacity = #blocks · blocksize

Memory

(a) Assume that the accesses to memory addresses shown in the tables below are given. For those accesses compare different cache designs (by filling the following tables). The cache is initially empty, byte addressing is used and the replacement strategy is LRU.

→ Offset von 3 Bit

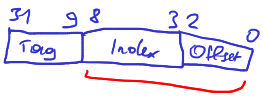
(I) A direct-mapped cache with a capacity (data) of 512 bytes and a block size of 8 bytes.

Block addr : Byte add. : blocksize 8

Set: Bl.add % #Sets 64

Tag: ⌊Byte add : 2^9⌋

Capacity : 2^3 = 512 : 8 = 64 ⇒ 64 Blocks ⇒ Index 6 Bits



$2^9 = 512$

Byte address	Block address	Set	Tag	Hit/Miss	Evicted
11332_{10}	1416	8	22	Miss	
11344_{10}	1418	10	22	Miss	
10818_{10}	1352	8	21	Miss	22
11840_{10}	1480	8	23	Miss	21
11328_{10}	1416	8	22	Miss	23
11856_{10}	1482	10	23	Miss	22

(II) A 2-way set-associative cache with a capacity (data) of 512 bytes and a block size of 8 bytes.

Unterschied ist dass es halb so viele Sets gibt

⇒

Tag ist dann also um das doppelte größer als vorhin

Byte address	Block address	Set	Tag	Hit/Miss	Evicted
11332_{10}					
11344_{10}					
10818_{10}					
11840_{10}					
11328_{10}					
11856_{10}					

(b) Find two different better alternative cache designs instead of the ones presented in part (a), which achieve a better hit rate for the given accesses. In general, valid solutions have to vary different cache design parameters, respectively, and can only change one design parameter at a time compared to the configuration in (a).I or (a).II. Explain why (or show that) your solution achieves a better hit rate.

Exercise 10

Memory

(a) In the lecture it was shown that oftentimes a better hit rate can be achieved when increasing cache associativity. Is this always the case? If yes, explain why. If no, show a counterexample.

(b) Assume that you are given a system and you are asked to find out the cache design parameters. The only information given is the following:

- Byte addressing is used
- Block size: 4, 8, 16, 32 or 64Byte
- Associativity: 1-, 2-, 4-, or 8-way set-associative
- Capacity (data): 2048Byte or 4096Byte
- Replacement: LRU

The only way to find out the actual design parameters is to give the system a sequence of accesses (with an initially empty cache) and observing the hit rate of the cache after finishing the complete sequence. Explain your solution.
(I) Propose a sequence of accesses for finding out the block size of the cache.
(II) Propose a sequence of accesses for finding out the associativity of the cache.
(III) Propose a sequence of accesses for finding out the capacity of the cache.

Exercise 11

Memory

Assume a system with Virtual Memory (VM). The Virtual Address width is 14 bit and the page size is 256Byte. The Physical Address (PA) width is 12bit. A 2-way set-associative TLB with overall 4 blocks/entries and a block size of 1 page table entry is implemented, which uses LRU replacement.

(a) Illustrate the detailed subdivision of the Virtual Address (also considering the TLB) and show the translation to (and the subdivision of) the Physical Address.

(b) The following virtual (byte) addresses are accessed: 0x628, 0x308, 0x9FC, 0x1A0
Given (A) the page table and (B) the TLB below: Fill the corresponding tables based on the information given.

*For the **Page Table**:*

- *Note down the final state of the page table after all accesses are completed.*
- *If a page must be brought from disk, assume it is brought to the next highest page number.*

Page Table (V: Valid; PP#: Physical page number)

	V	PP#/Disk
0	1	10
1	1	8
2	0	Disk
3	0	Disk
4	1	5
5	1	9
6	1	7
7	0	Disk
8	0	Disk
9	1	6
10	0	Disk

V	PP#/Disk
1	11

das 1. freie wenn man raufgeht

Exercise 11 continued on next page...

*For the **TLB**:*

- *Note down if the access results in a TLB hit (Yes/No) and if it causes a page fault (Yes/No).*
- *Show the state of the TLB after each access. In case of multiple invalid entries in a set, evict the leftmost entry.*

TLB (V: Valid; PP#: Physical page number; LAT: Last Access Time (higher number means more recent access))

Initial State

Set	V	Tag	PP#	LAT	V	Tag	PP#	LAT
0	1	2	5	1	1	0	10	0
1	1	4	6	0	0			

0110

Accessed VM address: 0x628 → TLB Hit? ✗ | Page Fault? ✗

Set	V	Tag	PP#	LAT	V	Tag	PP#	LAT
0	1	2	5	0	1	3	7	1
1	1	4	6	0	0			

finded man in Page Table in Nummer 6

0011

Accessed VM address: 0x308 → TLB Hit? ✗ | Page Fault? ✓

Set	V	Tag	PP#	LAT	V	Tag	PP#	LAT
0	1	2	5	0	1	3	7	1
1	1	4	6	0	1	1	11	1

1001

Accessed VM address: 0x9FC → TLB Hit? ✓ | Page Fault? ✗

Set	V	Tag	PP#	LAT	V	Tag	PP#	LAT
0	1	2	5	0	1	3	7	1
1	1	4	6	1	1	1	11	0

0001

Accessed VM address: 0x1A0 → TLB Hit? ✗ | Page Fault? ✗

Set	V	Tag	PP#	LAT	V	Tag	PP#	LAT
0	1	2	5	0	1	3	7	1
1	1	4	6	1	1	0	8	1