Wichtig: Das ist eine rekonstruierte Klausur, bei Fragen NICHT an den Professor wenden!

(Klausur nicht vollständig)

Technische Informatik - Sommersemester 2025

1. Bitpattern

Welchen Wert stellt das folgende Bitmuster einer 8-Bit-Ganzzahl mit Vorzeichen im Zweierkomplement dar (das höchstwertige Bit steht links)?
1111 1011

2. Assembly

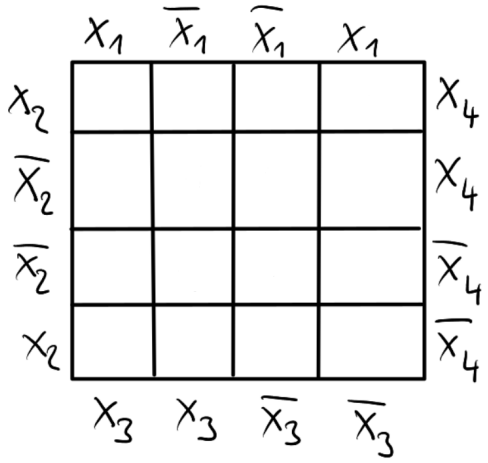
Was berechnet die Funktion tutwas? Angeben in mathematischer Formel.

```
.syntax unified
global tutwas

tutwas:
push {r3, r4, r5, lr}
mov r4, 40
movs r0, #0

loop:
cmp r4, #4
beq end

ldr r3, [r1]
ldr r5, [r2]
subs r3, r3, r5
muls r3, r3, r3
adds r0, r0, r3

adds r1, #4
adds r2, #4

subs r4, r4, #1
b loop
end:
pop{r3, r4, r5, pc}
```

3. KV-Diagramm

Wie viele Hazards hat diese Formel? Bitte deutlich markieren.

$$(x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \neg x_2 \wedge x_4) \vee (\neg x_1 \wedge \neg x_2 \wedge \neg x_3)$$



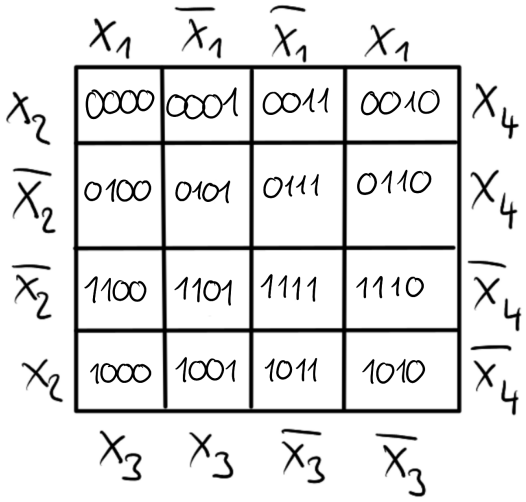
4. Gray Code

4.1 Erstellen

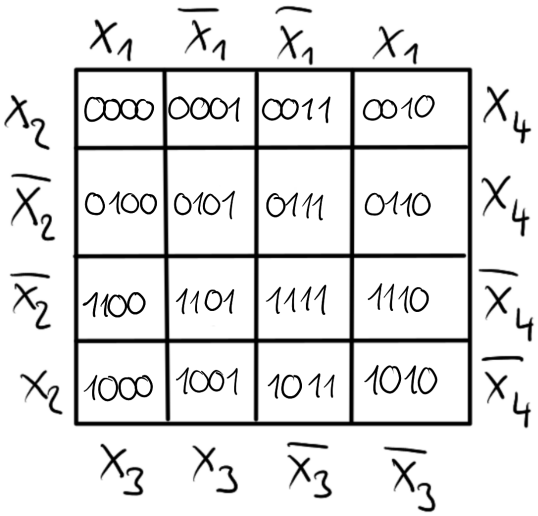
Generieren Sie einen 4-Bit Gray-Code. Schreiben Sie (ca. zwei Sätze) wie Sie dabei vorgehen.

4.2 Gray Code - KV

Zeichenen Sie einen 4-Bit Gray-Code in das KV Diagramm ein:



Zeichnen Sie erneut eine Möglichkeit des 4-Bit Gray-Codes in das KV-Diagramm ein:



5. Binomische Formel

Im Folgenden nehmen wir für die Daten Fließkommazahlen einfacher Genauigkeit an (32 Bit = 4 Byte). Wir betrachten die binomische Formel für Vektoren:

$$(\vec{a}+\vec{b})\cdot(\vec{a}+\vec{b}) = \vec{a}\cdot\vec{a} + 2\,\vec{a}\cdot\vec{b} + \vec{b}\cdot\vec{b}$$

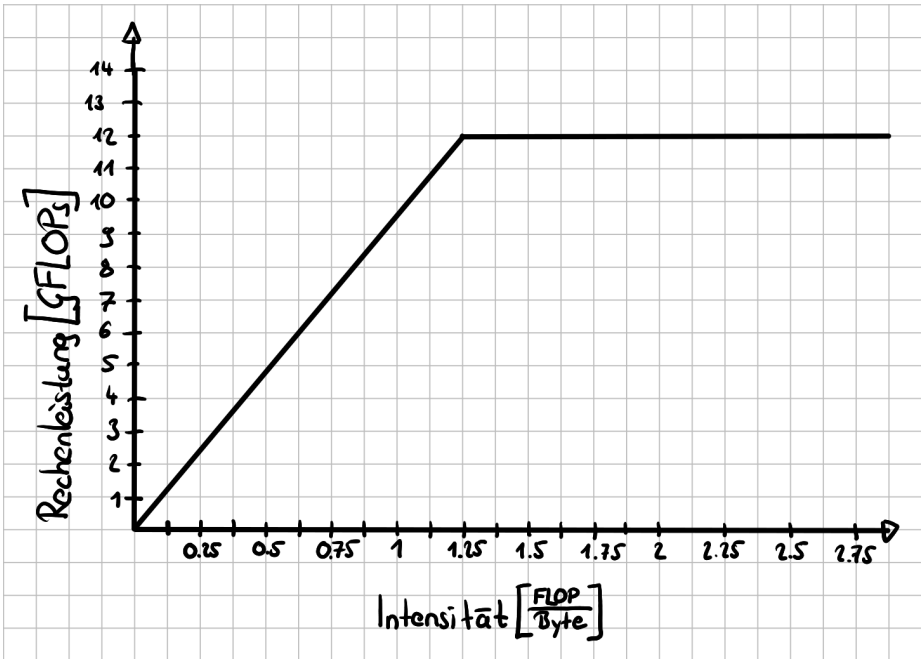
1. Ausdruck:

$$(\vec{a}+\vec{b})\cdot(\vec{a}+\vec{b}) = \frac{3}{8}$$

2. Ausdruck:

$$\vec{a}\cdot\vec{a} + 2\,\vec{a}\cdot\vec{b} + \vec{b}\cdot\vec{b} = \frac{7}{8}$$

Zeichnen Sie beide Ausdrücke in das Roofline-Diagramm ein, wobei Sie annehmen, dass beide nur durch die Speicherbandbreite limitiert sind und ansonsten optimal implementiert wurden.
Erwarten Sie, dass sich die Laufzeit (also die Zeit in Sekunden, bis der Ausdruck berechnet ist) der beiden Ausdrücke unterscheidet? Wenn ja, wie? Wenn nein, warum nicht?



Roofline-Modell

6. Caches

Moderne Caches sind üblicherweise nicht als Direct-Mapped Caches ausgeführt, sondern als N-Way Set-Associative Caches.

Warum verwendet man N-Way Caches und nicht einfach einen größeren Direct-Mapped Cache?

- ☐ Die Zugriffszeiten sind bei N-Way Caches oft schneller, da ein Datum mehrfach im Cache vorhanden sein kann.
- ☐ Da einem Datum im N-Way Cache mehrere Cache-Blöcke zugeordnet sind, ist die Wahrscheinlichkeit von Cache-Konflikten geringer.
- ☐ Da im N-Way Cache mehrere Daten mit demselben Tag gleichzeitig existieren können, kann die Verweildauer eines Datums im Cache länger sein, wodurch sich die effektive Zugriffsgeschwindigkeit erhöht.
- ☐ Da im N-Way Cache mehrere Cache-Blöcke dieselben Daten enthalten können, kann der Cache die Daten komprimieren.
- ☐ In einem viermal größeren Direct-Mapped Cache müssten die Tags viermal länger sein, was die Suche im Cache verlangsamen würde.
- ☐ N-Way Caches speichern jedes Datum mehrfach, was eine höhere Ausfallsicherheit bietet.

7. PIO Code

```
from machine import Pin
from time import sleep
import rp2

@rp2.asm_pio(set_init=
(rp2.PIO.OUT_LOW,
rp2.PIO.OUT_LOW))
def jmpdemo():
   set(pins, 0)
   label("start")
   set(x, 0)
   pull(noblock)
   mov(x, osr)
   jmp(not_x, "start")
   mov(pc, x)
   set(pins, 0)
   jmp("start")
   set(pins, 1)
   jmp("start")
   set(pins, 2)
   jmp("start")
   set(pins, 3)
   jmp("start")

p1 = Pin(15, Pin.OUT)
p2 = Pin(16, Pin.OUT)

sm = rp2.StateMachine(
0, jmpdemo,
freq=2000, set_base=p1)
sm.active(1)

ofs = jmpdemo[1]
print("offset", ofs)

print('b')
sm.put(0xC + ofs)
sleep(1)
print("0")
sm.put(0x6 + ofs)
sleep(3)
print("5")
sm.put(0x8 + ofs)
sleep(1)
```