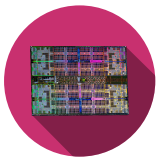


Alle Übungen CAr

Übungen Computerarchitektur

1 | Chip & Silikon Herstellung

1.1 Fabrikation

Wir möchten einen Chip herstellen, der eine Chipgröße von 15mm x 20mm hat. Die Chips werden auf Silizium-Wafern mit einem Durchmesser von 30cm hergestellt. Wir kaufen Ingots für 20′000CHF, von denen jeder in 100 Wafer zerlegt werden kann. Wir verwenden einen Prozess, der durchschnittlich $0,12$ Defekte pro cm^2 ergibt.

a) Wie hoch ist die Ausbeute bei der Herstellung?
b) Wie viele Chip haben auf einen Wafer platz?
c) Wie viele gute Chips erhalten wir pro Wafer?
d) Wie hoch sind die Kosten pro guten Chip?

fun/fabrication-01

1.2 Fabrikation

Wir möchten einen Chip mit einer Chipgröße von $150mm^2$ herstellen. Die Chips werden auf Silizium-Wafern mit einem Durchmesser von 20cm und einer Dicke von 5mm hergestellt. Wir kaufen Ingots für 30′000CHF, jeder hat eine Länge von 60cm. Wir verwenden einen Prozess, der durchschnittlich $0,2$ Defekte pro cm^2 ergibt, die Gehäuseausbeute (Packaging Yield) beträgt 80% und die Gehäusekosten liegen bei 0,06CHF pro Die.

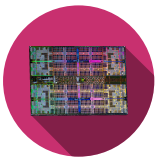
a) Wie viele Wafer pro Ingot?
b) Wie hoch sind die Kosten pro Wafer?
c) Wie hoch sind die Kosten pro cm^2 des Wafers?
d) Wie viele Dies können auf einen Wafer passen?
e) Wie viele gute Dies befinden sich auf einem Wafer?
f) Was kostet der fertige Chip?

fun/fabrication-02

1.3 Fabrikation

Ein Verfahren verwendet einen Wafer mit 20cm Durchmesser und einen $52,3mm^2$ großen Chip. Der Ingot kostet 6000CHF und kann in 30 Wafer zerlegt werden. Das Verfahren ergibt $0,03$ Fehler pro mm^2.

a) Wieviel kostet ein Wafer?



b) Wie hoch ist die Ausbeute?
c) Wie viele Chips passen auf einen Wafer?
d) Wie hoch sind die Kosten pro Chip?

fun/fabrication-03

2 | Moore'sches Gesetz & Denard-Skalierung

2.1 Dennard-Skalierung

Wenn sich die Anzahl der Transistoren pro Flächeneinheit (d.h. auf der gleichen Chipgröße) alle 24 Monate verdoppelt:
a) Um welchen Faktor nimmt die Länge der Seite eines Transistors, vorausgesetzt er ist quadratisch, jedes Jahr ab?
b) Ausgehend von einer Länge von $18,4$ nm (2014), was wäre die Größe eines Transistors im Jahr 2025? Wie verhält sich dies im Vergleich zum Wert von $6,75 \mu m$?

fun/dennardscaling-01

2.2 Die dynamische Leistungsaufnahme einer CMOS Schaltung ist:

- ☐ direkt proportional zur Frequenz
- ☐ umgekehrt proportional zur Frequenz
- ☐ direkt proportional zum Quadrat der Versorgungsspannung

fun/dennardscaling-02

3 | Stromverbrauch

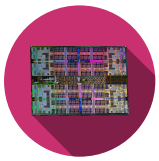
3.1 Lebensdauer des Handy-Akkus

Ein bestimmtes Mobiltelefon hat einen 8-Wattstunden-Akku und arbeitet mit 0,707 V. Nehmen wir an, dass das Mobiltelefon während des Betriebs mit 2 GHz arbeitet. Die Gesamtkapazität des Schaltkreises beträgt 10 nF, und der Aktivitätsfaktor ist 0,05. Wenn Sprache oder Daten aktiv sind (10% der Nutzungszeit), sendet es auch 3 W Leistung über seine Antenne aus. Wenn das Telefon nicht benutzt wird, sinkt die dynamische Leistung fast auf Null, da die Signalverarbeitung ausgeschaltet ist. Das Telefon zieht aber auch 100 mA Ruhestrom, egal ob es in Betrieb ist oder nicht.
Bestimmen Sie die Akkulaufzeit des Telefons:
a) wenn es nicht benutzt wird
b) wenn es ständig benutzt wird

fun/powerconsumption-01

1 | Prozessor Benchmark & Leistung

1.1 Welche der folgenden Aussagen sind richtig?



- [ ] Die wall-clock-time ist die insgesamt verstrichene Zeit, einschliesslich E/A, Betriebssystem-Overhead usw.
- [ ] Multi-threading verbessert den Durchsatz eines Prozesses
- [ ] Die CPU Zeit beinhaltet nicht die E/A-Zeit
- [ ] Multi-threading verbessert die Ausführungszeit eines Prozesses

per/benchmark-01

1.2 Was ist der Durchsatz (throughput)?

- [ ] Leistung pro Watt (die Anzahl der FLOPS pro Watt)
- [ ] Rate der Verarbeitungsarbeit (n Aufträge/Sekunde)
- [ ] die Zeit zwischen Beginn und Abschluss eines Ereignisses/Aufgabe/Programms (n Sekunden)
- [ ] der Prozentsatz der Zeit, in der ein System in Betrieb ist und läuft

per/benchmark-02

1.3 Was ist der SPEC?

- [ ] ist ein Benchmark-Katalog, der entwickelt wurde, um die Leistung auf der Grundlage der neuesten Java-Anwendungsfunktion zu messen
- [ ] ist ein Benchmark, der die Energie- und Leistungsmerkmale von Computern der Volume-Server-Klasse bewertet
- [ ] ist der weltweite Standard für die Messung der Grafikleistung auf der Grundlage professioneller Anwendungen
- [ ] ist eine Benchmark-Katalog, die Leistungsmessungen liefert, die zum Vergleich computerintensiver Arbeitslasten auf verschiedenen Computersystemen verwendet werden können.

per/benchmark-03

1.4 Was ist das Ziel der EEMBC-Benchmark?

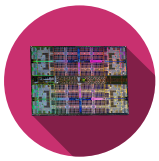
- [ ] zur Bewertung der Leistung von eingebetteten Mikroprozessoren
- [ ] zur Bewertung der Leistung von Ganzzahlberechnungen
- [ ] zur Messung der Energieeffizienz verschiedener Computersysteme
- [ ] zur Bewertung der Gleitkommaleistung

per/benchmark-04

1.5 Welche der folgenden Kennzahlen ist eine Energieeffizienzkennzahl?

- [ ] flops
- [ ] MIPS
- [ ] Leistung pro Watt
- [ ] Leistungsaufnahme

per/benchmark-05



1.6 Bei einem eingebetteten System sind sowohl der Stromverbrauch als auch die Leistung pro Watt wichtig.

- ☐ Wahr
- ☐ Falsch

per/benchmark-06

1.7 Prozessorleistung

Ein Programm besteht aus 5′000 Gleitkomma- und 25′000 Ganzzahlbefehlen. Prozessor A hat eine Taktrate von 2.0GHz. Fliesskommaanweisungen benötigen 7 Zyklen und Ganzzahlanweisungen 1 Zyklus.

a) Wie lange braucht dieser Prozessor, um das Programm auszuführen?
b) Was ist der durchschnittliche CPI für diesen Prozessor für das gegebene Programm?
c) Prozessor A führt Programm 2 aus, das aus 100′000 Gleitkomma- und 50′000 Ganzzahl-Befehlen besteht. Wie hoch ist der durchschnittliche CPI für dieses Programm?
d) Prozessor B hat einen durchschnittlichen CPI für Programm 2 von 3.5. Seine Taktrate beträgt 1.8 GHz. Wie viel Zeit benötigt er für die Ausführung des Programms?
e) Welcher Prozessor ist schneller und um wie viel schneller?

Prozessor _____ ist _____ mal schneller als Prozessor _____.

per/performance-01

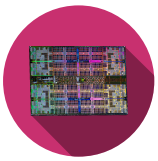
1.8 Prozessorleistung

Betrachten Sie die folgenden zwei Maschinenkonzepte mit ihren jeweiligen CPI's für verschiedene Befehlstypen. Computer A und Computer B haben den gleichen Befehlssatz:

Instruction Type	CPI_A	CPI_B	Compiler 1 Mix
Data Manipulation	1.5	1.0	25%
Arithmetic	1.0	1.5	30%
Shifting	1.0	1.2	10%
Branching	4.0	2.0	25%
Multiply	20	12	10%

a) Wie hoch ist die durchschnittliche CPI für jeden der Computer, die dieses Programm verwenden?
b) Computer A hat eine Clock-Zykluszeit von 0, 5ns. Computer B läuft mit 1, 8GHz. Schreiben Sie eine quantitative Aussage zum Vergleich der beiden Computer.
c) Wie hoch müsste die Taktrate des langsameren Computers sein, um die Leistung des schnelleren Computers zu erreichen?

per/performance-02



1.9 Prozessorleistung

Eine CPU läuft mit einer Basisfrequenz von 2GHz. Er führt ein Programm mit 5 Millionen Anweisungen mit der angegebenen Anweisungsmischung aus. newline Wie lange ist die Ausführungszeit des Programms?

Instruction	Frequency	$\text{CPI}_{\text{instr}}$
ALU	50%	3
Load	20%	5
Store	10%	4
Branch	20%	3

per/performance-03

1.10 Prozessorleistung

Eine CPU ist für eine optimale Leistung bei einem bestimmten Programm mit den folgenden Merkmalen ausgelegt. 25% aller Anweisungen sind Gleitkommaanweisungen mit einem durchschnittlichen CPI von 4.0, ausserdem enthält das Programm 2% FPSQR Anweisungen mit einem durchschnittlichen CPI von 20. Alle anderen Anweisungen haben einen durchschnittlichen CPI von 1.33. Es gibt zwei Konzeptalternativen:

1. Senkung des CPI von FPSQR -Anweisungen auf 2.0
2. Senkung des durchschnittlichen CPI aller Gleitkommaanweisungen auf 2.5

Welche Wahl ist die bessere?

per/performance-04

1.11 Prozessorleistung

Wir wollen einen neuen Computer kaufen. Darauf sollen hauptsächlich die Programme P_1 und P_2 laufen.
Welches Gewicht w_{p_1} und w_{p_2} müssen die Programme haben, damit:

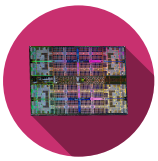
a) CPU A der beste Kauf ist?
b) CPU B der beste Kauf ist?
c) CPU C der beste Kauf ist?

Program	CPU_A	CPU_B	CPU_C
Program P_1 (sec)	1	10	100
Program P_2 (sec)	100	10	1

per/performance-05

1.12 Prozessorleistung

Benutzen Sie das geometrische Mittel, um zu berechnen, welcher Computer der schnellste ist, wenn Sie die folgende Leistung von zwei Programmen auf drei CPU's betrachten:



a) CPU A ist der Schnellste!
b) CPU B ist der Schnellste!
c) CPU C ist der Schnellste!

Program	CPU_A	CPU_B	CPU_C
P_1 (sec)	40	15	20
P_2 (sec)	40	1000	150

per/performance-06

1.13 Prozessorleistung

Berechnen Sie den durchschnittlichen CPI für 5 Millionen Anweisungen mit den folgenden Befehlshäufigkeiten:

Instruction	Frequency	CPI_{instr}
ALU	40%	4
Load	30%	6
Store	5%	5
Branch	25%	4

Die Clockfrequenz des CPU beträgt 2 GHz

per/performance-07

1.14 Welches ist die beste Messgröße für einen Leistungsvergleich?

- ☐ arithmetisches Mittel
- ☐ geometrisches Mittel
- ☐ median
- ☐ maximale Leistung
- ☐ harmonisches Mittel

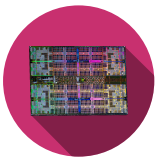
per/performance-08

1.15 Prozessorleistung

Berechnen Sie die Ausführungszeit in ms, unter der Annahme, dass der CPU mit den folgenden Befehlshäufigkeiten arbeitet:

Instruction	Frequency	CPI_{instr}
ALU	45%	5
Load	25%	6
Store	10%	5
Branch	20%	3

Für 2 Millionen Befehle und eine CPU Frequenz von 3 GHz.



per/performance-09

1.16 Amdahlsches Gesetz

Durch eine Verbesserung der Fliesskomma-Ausführungseinheit wurden 2x schnellere Fliesskomma-Befehle erzeugt. Im Durchschnitt sind 10% aller Befehle bei diesem Prozessor Fliesskomma-Befehle.

Wie hoch ist der Geschwindigkeitszuwachs insgesamt?

per/amdahls-law-01

1.17 Amdahlsches Gesetz

Wir wollen eine Gesamtbeschleunigung von 2 und können die Gleitkommaanweisungen um das Vierfache beschleunigen.

Wie hoch sollte der Anteil der Fliesskommaanweisungen sein?

per/amdahls-law-02

1.18 Amdahlsches Gesetz

Ein Program besteht aus 2 verschiedenen Elementen. Teil A hat eine Dauer von 15 und Teil B eine Dauer 5 Zeiteinheiten. Es gibt zwei Optimierungsvarianten:
1. Optimierung des A Teiles um das zweifache
2. Optimierung des B Teiles um das fünffache

Welche Optimierung ist vorteilhafter? Was sind die Implikationen?

per/amdahls-law-03

1 Implementierung

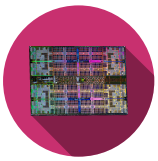
1.1 Was ist der Hauptunterschied zwischen einem harten und einem weichen Echtzeitsystem?

- ☐ in einem System mit harter Echtzeit müssen alle Fristen eingehalten werden, während in einem System mit weicher Echtzeit gelegentlich einige Fristen versäumt werden können.
- ☐ in einem Soft-Real-Time-System müssen alle Fristen eingehalten werden, während in einem Hard-Real-Time-System gelegentlich einige Fristen überschritten werden können.

imp/implementation-01

1.2 Was ist ein eingebettetes System?

- ☐ Jedes Rechnersystem ist ein eingebettetes System



- [ ] Ein Computersystem mit einer speziellen Funktion, oft mit zeitlichen Beschränkungen, ist ein eingebettetes System
- [ ] Ein Allzweckcomputer mit weniger als 1 GB RAM ist ein eingebettetes System
- [ ] Ein System, das einen ARM-Prozessor hat, ist ein eingebettetes System

imp/implementation-02

1.3 Schnellere Ausführungszeit bedeutet weniger Energie.

- [ ] Wahr
- [ ] Falsch

imp/implementation-03

1.4 Warum werden immer mehr SOC's anstelle von CPU's entwickelt?

- [ ] eingeschränkte Verfügbarkeit von Energie
- [ ] Beschleunigung häufig genutzter Funktionen
- [ ] beide der oben genannten Punkte

imp/implementation-04

1 Instruction-Set Architecture

1.1 Einfach C-Code zu RISC-V Assembler

Kompilieren Sie den folgenden C-Code in RISC-V Assembler.

a)
```
a = b + c;
```

b)
```
a = b + c - d;
```

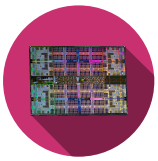
c)
```
a = b + 6;
```

d)
```
// int is a 32-bit signed word
int a = -372;
int b = a + 6;
```

e)
```
int a = 0xFEDC8765;
```

f)
```
int a = 0xFEDC8EAB;
```

isa/c-to-riscv-01



1.2 Algorithmik C-Code zu RISC-V Assembler

Kompilieren Sie den folgenden C-Code in RISC-V Assembler.

a)

```c
if (i == j){
  f = g + h;
}
f = f - i;
```

b)

```c
if (i == j){
  f = g + h;
}
else {
  f = f - i;
}
```

c)

```c
// add the numbers from 0 to 9
int sum = 0;
int i;

for (i=0; i!=10; i=i+1){
  sum = sum + i;
}
```

d)

```c
// add the powers of 2 from 1 to 100
int sum = 0;
int i;

for (i=1;i<101; i=i*2){
  sum = sum + i;
}
```

e)

```c
int array[5];
array[0] = array[0] * 2;
array[1] = array[1] * 2;
```

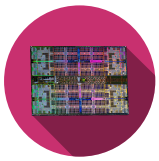
f)

```c
int array[1000];
int i;

for (i=0; i<100; i=i+1){
  array[i] = array[i] * 8;
}
```

g)

```c
char str[80] = "CAT";
int len = 0;

// compute length of string
while (str[len]) len++;
```

isa/c-to-riscv-02



1.3 Maschinencode zu RISC-V Assembler

Dekodieren Sie den folgenden Maschinencode in RISC-V-Assembler.

a) **`0x41FE 83B3`**
b) **`0xFDA4 8393`**

isa/machinecode-to-riscv-01

1.4 Logische Operationen mit Registern

Führen Sie den Assember-Code aus und geben Sie den Inhalt des Zielregisters **`rd`** an falls die Quellregister **`rs`** folgende Daten enthalten:

```
s1 = 0x46A1 F1B7
s2 = 0xFFFF 0000
```

a)
```
and s3, s1, s2
```

b)
```
or s4, s1, s2
```

c)
```
xor s5, s1, s2
```

isa/riscv-execution-01

1.5 Logische Operationen mit Werten

Führen Sie den Assember-Code aus und geben Sie den Inhalt des Zielregisters **`rd`** an falls die Quellregister **`rs`** folgende Daten enthalten:

```
t3 = 0x3A75 0D6F
```

a)
```
and s5, t3, -1484
```

b)
```
or s6, t3, -1484
```

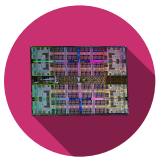
c)
```
xor s7, t3, -1484
```

isa/riscv-execution-02

1.6 Multiplikationen in RISC-V

Führen Sie den Assember-Code aus und geben Sie den Inhalt des Zielregisters **`rd`** an falls die Quellregister **`rs`** folgende Daten enthalten:

```

```



```
s1 = 0x4000 0000
s2 = 0x8000 0000
```

```
mulh s4, s1, s2
mul  s3, s1, s2
```

isa/riscv-execution-03

1.7 Division und Modulo

Führen Sie den Assember-Code aus und geben Sie den Inhalt des Zielregisters **`rd`** an falls die Quellregister **`rs`** folgende Daten enthalten:

```
s1 = 0x0000 0011
s2 = 0x0000 0003
```

```
div s3, s1, s2
rem s4, s1, s2
```

isa/riscv-execution-04

1.8 R-Typ zu Maschinencode

Kodieren Sie den folgenden RISC-V-Assembler in Maschinencode.

a)
```
add s2, s3, s4
```

b)
```
sub t0, t1, t2
```

c)
```
sll s7, t0, s1
```

d)
```
xor s8, s9, s10
```

e)
```
srai t1, t2, 29
```

isa/riscv-to-machinecode-01

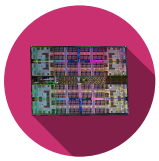
1.9 I-Typ zu Maschinencode

Kodieren Sie den folgenden RISC-V-Assembler in Maschinencode.

a)
```
addi s0, s1, 12
```

b)
```
addi s2, t1, -14
```

c)



```
lw t2, -6(s3)
```

d)
```
lh s1, 27(zero)
```

e)
```
lb s4, 0x1F(s4)
```

isa/riscv-to-machinecode-02

1.10 S-Typ zu Maschinencode

Kodieren Sie den folgenden RISC-V-Assembler in Maschinencode.

a)
```
sw t2, -6(s3)
```

b)
```
sh s4, 23(t0)
```

c)
```
sb t5, 0x2D(zero)
```

isa/riscv-to-machinecode-03

1.11 Realzeitsystem

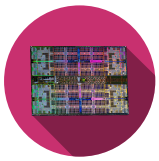
Was ist der Hauptunterschied zwischen einem „harten“ und einem „weichen“ Echtzeitsystem?

- ☐ In einem Hard-Real-Time-System müssen alle Fristen eingehalten werden, während in einem Soft-Real-Time-System gelegentlich einige Fristen überschritten werden können.
- ☐ In einem Soft-Real-Time-System müssen alle Fristen eingehalten werden, während in einem Hard-Real-Time-System gelegentlich einige Fristen überschritten werden können.
- ☐ Eine Spielkonsole muss unter einem „harten“ Echtzeitsystem laufen, da es sonst nicht möglich ist, ein Spiel laufen zu lassen.
- ☐ Der Spritzkopf eines Druckers muss unter einem „harten“ Echtzeitsystem laufen, da die gedruckte Seite sonst fehlerhaft sein kann.
- ☐ Die Geschwindigkeitsmesssonde eines Flugzeugs muss unter einem „harten“ Echtzeitsystem laufen, da sich sonst der Autopilot abschalten kann.
- ☐ Ein „weiches“ Echtzeitsystem schneller ist als ein „hartes“ Echtzeitsystem.
- ☐ Ein „hartes“ Echtzeitsystem schneller ist als ein „weiches“ Echtzeitsystem.
- ☐ Windows 10 ist ein OS, das in der Lage ist, in „harter“ Echtzeit zu arbeiten.
- ☐ Ubuntu Desktop ist ein OS, das in der Lage ist, in „harter“ Echtzeit zu arbeiten.
- ☐ RTLinux ist ein OS, das in der Lage ist, in „harter“ Echtzeit zu arbeiten.

isa/riscv-to-machinecode-04

1.12 U-Typ zu Maschinencode

Kodieren Sie den folgenden RISC-V-Assembler in Maschinencode.



```
lui s5, 0x8CDEF
```

isa/riscv-to-machinecode-05

1.13 J-Typ zu Maschinencode

Kodieren Sie den RISC-V-Assembler-Befehl für den **ersten** Sprungbefehl im Maschinencode. Das Programm lautet wie folgt:

```
0x0000540C        jal ra,func1 # <--
0x00005410        add s1, s2, s3
...
0x001ABC04 func1: add s4, s5, s8
...
```

isa/riscv-to-machinecode-06

2 | Laborergänzung

Um Ihnen zu helfen können Sie gerne der RISC-V-Interpreter auf https://course.hevs.io/car/riscv-interpreter/ sowie Ripes verwenden.

Achten Sie sich auf die Typen der Variablen!

- Der Typ **int** wird als vorzeichenbehaftete 32-Bit-Größe betrachtet.
- Der Typ **unsigned int** wird als unsignierter 32-Bit-Typ betrachtet.
- Wenn eine Zahl dahinter steht (z. B. **int16_t**), bedeutet dies, dass die Variable x-Bit lang ist (hier 16). Wenn ein **u** vorangestellt ist, ist er unsigniert.

uint8_t ist also ein vorzeichenloses Byte, während **int8_t** ein vorzeichenbehaftetes Byte ist.

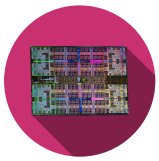
2.1 Grundrechenarten

a)

```
int b = 1;
int c = 2;
a = b + c;

int b = -1;
int c = 2;
a = b + c;

int b = -12;
int c = 2023;
a = b + c;
```

b)

```
int b = 2;
int c = 3;
int e = -1;
int f = -78;
int g = 2023;
int h = -12;
a = b - c;
d = (e + f) - (g + h);
```



isa/lab-basic-calc

2.2 Speicherzugriff

```
uint16_t a = mem[3];
mem[4] = a;

int16_t a = mem[3];
mem[4] = a;
```

isa/lab-memory

2.3 Grundlegende Algorithmen

1. Übertrage den 8Bit Wert des Speichers an Addresse 0x0000'1000 seriell Bit für Bit im LSB des Speichers in der Addresse 0x0000'1001. Die restlichen Bits der Speicheraddresse 0x0000'1001 müssen ‚0' betragen. Berechnen sie die BaudRate in $\frac{\text{Instructions}}{\text{Bit}}$ für die gesamte Übertragung?

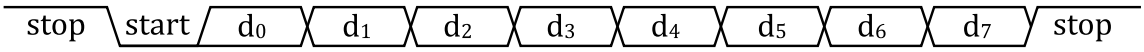


Abbildung 7: UART Serielle Übertragung

2. Multipliziere zwei 4-bit Zahlen zusammen benutzte hierzu zusätzlich einen der Befehle **bne**, **bge**. Der Algorithmus funktioniert folgendermassen: Eine Multiplikation ist das gleiche wie die x-fache Addition der gleichen Zahl. Zum Beispiel: $2 * 9 = 9 + 9 = 18$.

isa/lab-basic-algos

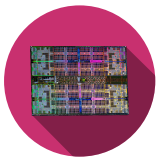
2.4 Branching

2.4.1 If / else

```
int a = 1, b = 2, c;

if(a == b) {
  c = 0;
} else if(a > b) {
  c = 1;
} else {
  c = 2;
}
```

2.4.2 Switch case

```
int a;

switch(mem[2]) {
  case 0:
    a = 17;
    break;
  case 3:
    a = 33;
    break;
  case 8:
  case 12:
    a = 10;
    break;
  default:
    a = 99;
}
```



2.4.3 While / Do While

```
// A
int a = 10;

do{a = a - 1;}
while(a != 0);

// B
int a = 10;

do{a = a - 1;}
while(a >= 0);

// C
unsigned int a = 10;

do{a = a - 1;}
while(a >= 0);
```

2.4.4 For

```
int a = 0, i;

for(i = 4; i > mem[0]; i = i - 1) {
  a = a + i;
}
```

isa/lab-branch

2.5 Functions

a)

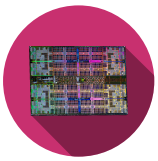
```
int a = 1, b;
b = doubleIt(a);
b = doubleItOpti(a);

...

// Non-optimized version
// Let's assume a is saved in s0
int doubleIt(int myvar) {
  int a = myvar; // we WANT a in s0 !
  a = a * 2;
  return a;
}

// Optimized version
// Choose your registers freely
// Try having the less possible
//   instructions
int doubleItOpti(int myvar) {
  int a = myvar; //
  a = a * 2;
  return a;
}
```

b)

```
int a=1, b=2, c=3,
 d=4, e=5, f=6, g=7,
 h=8, i=9, j=10, res;
res = sum(a,b,c,d,e,f,
  g,h,i,j);

...

int sum(int v1, int v2,
  int v3, int v4, int v5,
  int v6, int v7, int v8,
  int v9, int v10){
  int c;
  c = v1 + v2 + v3 + v4 +
    v5 + v6 + v7 + v8 +
    + v9 + v10;
  return c;
}
```

isa/lab-fcts

2.6 Advanced Algorithmus



2.6.1 Modulo

Der Modulo % ist eine Operation, die auf zwei positive ganze Zahlen angewendet wird und ist nichts anderes als der Rest der Division. Zum Beispiel ergibt 5 geteilt durch 3 ergibt 1 (man kann 3 in 5 einmal hineinlegen), **Rest 2**.
Das Modulo einer Zahl durch 0 ist nicht definiert.
Die Definition für vorzeichenbehaftete Zahlen weicht je nach Sprache ab. Wir behandeln hier nur die unsigned Version.
Modulo ist vielseitig einsetzbar und ermöglicht es, Werte zu begrenzen, Informationen zu extrahieren, eine X- und Y-Position aus einem X*Y-Wert zu berechnen in einem Array mit bekannter Grösse ...

- Geben Sie einen Code an, mit dem diese Operation für jede positive ganze Zahl unter Verwendung des Sets RV32IM durchgeführt werden kann.
- Wie kann das Gleiche in RV32I implementiert werden? Beschreiben Sie das/die Konzept(e).

Die Aufgabe des Compilers ist es, den Code so gut wie möglich zu optimieren. Wenn die erkannte Operation ein Modulo ist, bei dem eine Konstante eine Potenz von 2 ist (z. B. x % 2, y % 8 ...), ist eine Variante möglich, die keine Division enthält.

- Geben Sie diese Variante an.

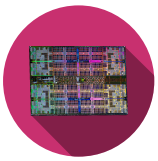
Das Konzept des Modulo für reelle Zahlen kam mit der Entwicklung der Rechenleistung auf und das Ergebnis weicht auch je nach Sprache ab. In C ist eine spezielle Funktion der std-Bibliotheken erforderlich, **fmod()**. In Python ist diese Operation nativ. In jedem Fall sind sie ressourcenintensiver und erfordern die Behandlung einiger Sonderfälle (NaN, infinity).

2.6.2 °F -> °C

Wir möchten eine Funktion erstellen, die Fahrenheitgrade in Celsius umwandeln kann, da die Fahrenheitswerte zwischen 32 und 1000 schwanken, ist eine Genauigkeit auf das Grad ausreichend.
Die Formel ist einfach: $C = (F - 32) * \frac{5}{9}$.
Es wäre praktisch wenn eine FPU zur Verfügung stünde, aber es wird nur den RV32I-Basisbefehlssatz unterstützt.
Um dieses Problem zu umgehen, können Sie einige Tricks anwenden, der Algorithmus lautet wiefolgt:

A. Berechnen Sie $C = F - 32$
B. Mit 5 multiplizieren
C. Durch 9 teilen
   - $\frac{1}{9}$ kann durch eine spezielle Binäre Darstellung gespeichert werden

```
+-----+-----+-----+-----+-----+-----+------------+------------+
| b31 | b30 | b29 | b28 | b27 | ... |         b1 |         b0 |
+-----+-----+-----+-----+-----+-----+------------+------------+
| 2^0 | 2^-1| 2^-2| 2^-3| 2^-4| ... |      2^-30 |      2^-31 |
+-----+-----+-----+-----+-----+-----+------------+------------+
| 1   | 1/2 | 1/4 | 1/8 | 1/16| ... |1/1737418240|1/2147483648|
+-----+-----+-----+-----+-----+-----+------------+------------+
```



ins diesem Fall ist dies $0000\_1110\_0011\_1000\_1110\_0011\_1000\_1110_2$.
- Die Konstante kann vorberechnet werden und beträgt $\frac{2^n}{9} + 1$. Je grösser n ist, desto höher ist die Genauigkeit. Die Anzahl Bits definieren die maximale Grösse von n. Die $+1$ ist eine Aufrundung für die verlorene Genauigkeit.
- Nehmen wir $n = 16$. Unsere magische Zahl lautet daher $\text{magic} = \frac{2^n}{9} + 1 = \frac{65536}{9} + 1 = 7282$.
- Wir müssen den Wert mit dieser magischen Zahl multiplizieren

D. und danach durch 2^n dividieren. Im Fall $n = 16 \rightarrow \frac{1}{65536}$

Um die Arbeit zu vereinfachen, werden mehrere Annahmen getroffen:
- Die magische Zahl und die Temperatur in Fahrenheit sind immer positiv.
- Die Grösse der grössten Multiplikation beträgt:

$$\begin{aligned} \text{nbBits}_{\text{max}_{\text{fahrenheit}}} + \text{nbBits}_{\text{mult5}} + \text{nbBits}_{\text{magicNumber}} &= \\ 10(\text{max. 1000-32}) + 3 + (n - \text{nbBits}_{\text{div9}} + 1) &= \\ 10 + 3 + (16 - 4 + 1) &= \\ = 26 \text{ bits} & \end{aligned} \tag{1}$$

  - Sie überschreitet nie 32 Bit für $n < 23$.

Testen und optimieren Sie die Funktion:
- Schreiben Sie den entsprechenden Code.
- Testen Sie mit verschiedenen Werten von Fahrenheit.
- Testen mit mehreren Werten für n (16, 18, 20). *Vergessen Sie nicht die magische Zahl neu zu berechnen.*
- Testen Sie die Funktion mit $n = [22, 23]$, für $°F = [100, 400, 1000]$.

isa/lab-adv-algos

1 Architektur

1.1 Stack-Architektur

Auswertung des Ausdrucks $\frac{a+bc}{a+dc-e}$ unter Verwendung eines stack-basierten Prozessors.

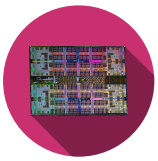
a) Schreibe Pseudo-Code der Berechnung.
b) Wie viele direkte und indirekte Speicherreferenzen sind bei einer unendlichen Stackgrösse ?
c) Wie viele direkte und indirekte Speicherreferenzen sind bei einer Stackgrösse von 2 erforderlich?

arc/stack-01

1.2 Stack-Architektur

Auswertung des Ausdrucks $\frac{(a+b)^2}{\pi} * (a + b + c)$ unter Verwendung eines stack-basierten Prozessors.

a) Schreibe Pseudo-Code der Berechnung.
b) Wie viele direkte und indirekte Speicherreferenzen sind bei einer Stackgrösse von 4 erforderlich?



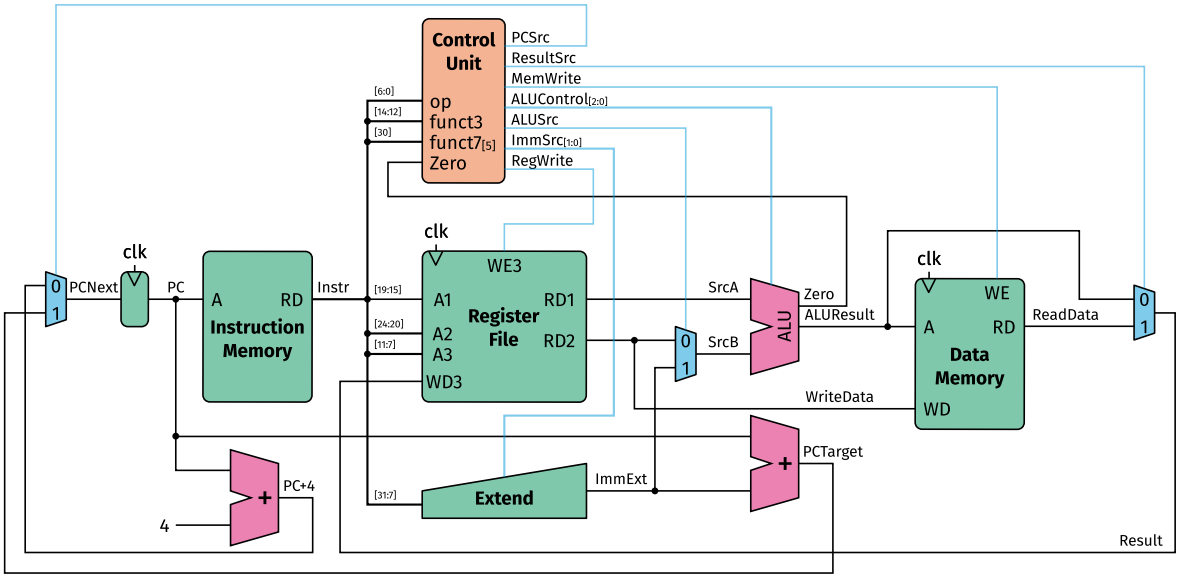
c) Wie viele direkte und indirekte Speicherreferenzen sind bei einer Stackgrösse von 3 erforderlich?
d) Wie viele direkte und indirekte Speicherreferenzen sind bei einer Staclgrösse von 2 erforderlich?

arc/stack-02

2 Single-Cycle RISC-V

2.1 Single-Cycle-Prozessorbetrieb

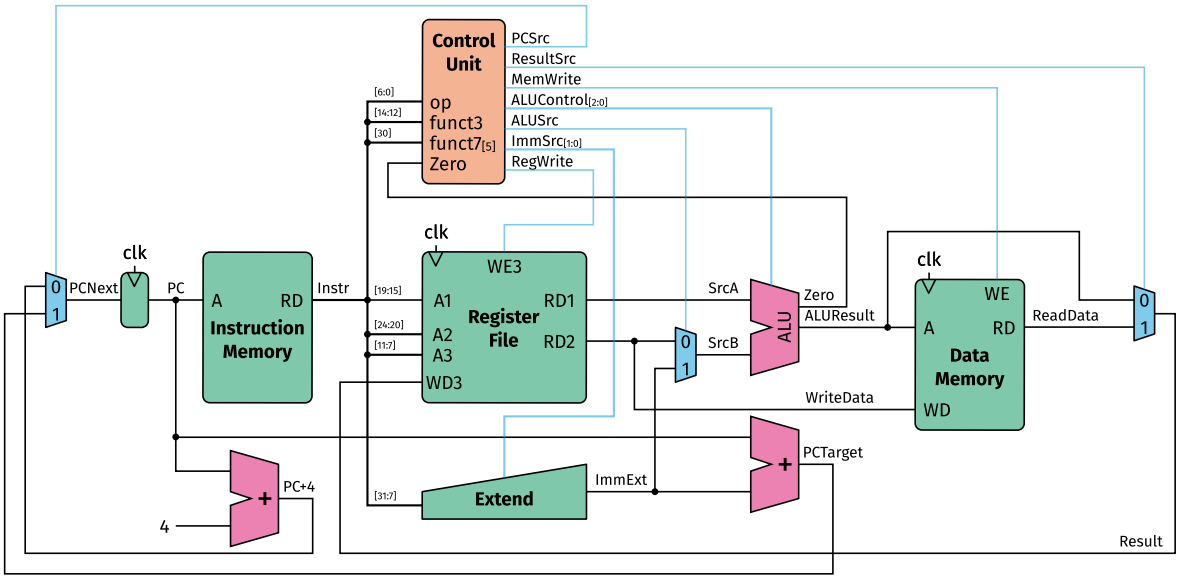
Bestimmen Sie die Werte der Steuersignale und die Teile des Datenpfads, die bei der Ausführung einer **and**-Anweisung verwendet werden. Zeichnen Sie direkt auf dem Bild das Innenleben des Prozessors.



arc/scr-01

2.2 Einzelzyklus mit Anweisung `jal` verlängern

Zeigen Sie, wie der gegebene RISC-V-Einzyklusprozessor so geändert werden kann, dass er den Sprung- und Verknüpfungsbefehl **`jal`** unterstützt. **`jal`** schreibt **`PC+4`** nach **`rd`** und ändert den PC auf die Sprungzieladresse **`PC+imm`**.



arc/scr-02

2.3 Einzelzyklus-Prozessorleistung

Ein mit einem 7-nm-CMOS-Fertigungsprozess hergestellter Einzyklus Prozessor weist die folgenden Zeitmerkmale auf.

Element	Parameter	Delay(ps)
Register clk-to-Q	t_{pcq}	40
Register Setup	t_{setup}	50
Multiplexer	t_{mux}	30
AND-OR Gate	$t_{\text{AND\_OR}}$	20
ALU	t_{ALU}	120
Decoder (Control Unit)	t_{dec}	25
Extend Unit	t_{ext}	35
Memory Read	t_{mem}	200
Register File Read	t_{RFread}	100
Register File Setup	t_{RFSetup}	60

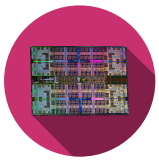
Das Programm für den SPECINT2000-Benchmark enthält 100 Milliarden Anweisungen. Berechnen Sie die Berechnungszeit des Benchmarks für diesen Einzyklus Prozessor.

arc/scr-03

3 Multi-Cycle RISC-V

3.1 Mehrzyklus-Prozessorleistung

Das Programm für den SPECINT2000-Benchmark besteht aus ungefähr 25% loads, 10% stores, 11% branches, 2% jumps und 52% R- oder I-Typ ALU-Befehlen.



Ermitteln Sie den durchschnittlichen CPI für diesen Benchmark für den von uns entwickelten Multi-Cycle-Prozessor.

arc/mcr-01

3.2 Multi-Zyklus-Prozessorleistung

Ein mit einem 7-nm-CMOS-Fertigungsprozess hergestellter Multi-Zyklus Prozessor weist die folgenden Zeitmerkmale auf.

Element	Parameter	Delay(ps)
Register clk-to-Q	t_{pc}	40
Register Setup	t_{setup}	50
Multiplexer	t_{mux}	30
AND-OR Gate	$t_{\text{AND\_OR}}$	20
ALU	t_{ALU}	120
Decoder (Control Unit)	t_{dec}	25
Extend Unit	t_{ext}	35
Memory Read	t_{mem}	200
Register File Read	t_{RFread}	100
Register File Setup	t_{RFSetup}	60

Das Programm für den SPECINT2000-Benchmark enthält 100 Milliarden Anweisungen. Nutzen sie die CPI_{avg} der vorherigen Aufgabe.
Berechnen Sie die Berechnungszeit des Benchmarks für diesen Multi-Zyklus Prozessor.

arc/mcr-02