Aufgabe 1: Binäre Darstellung

a) Sie müssen folgende Zahlen in einem Rechner speichern. Welche **minimalen** Bitbreiten sind dafür jeweils notwendig?

- Anzahl Studierende in diesem Kurs: 71
- Fläche Argentiniens in km^2: 2.780.400
- Minimale Entfernung der Erde von Pluto (in Kilometer): 4.275.000.000 Kilometer

b) Welche Datentypen würden Sie z.B. in Java für diese Zahlen verwenden?

c) In der Programmiersprache Java hat der Datentyp *long* 64 Bit (Zweierkomplement). Geben Sie den Wertebereich dafür möglichst kurz mit Hilfe von Zweierpotenzen an!

a) 1) $\lceil \log_2(71) \rceil = 7$ Bit $\quad (2^6 = 64,\ 2^7 = 128)$

2) $\lceil \log_2(2\,780\,400) \rceil = 22$ Bit $\quad (2^{21} \approx 2{,}1$ Mio$,\ 2^{22} \approx 4$ Mio$)$

3) $\lceil \log_2(4\,275\,000\,000) \rceil = 32$ Bit $\quad (2^{32} \approx 4{,}295 \cdot 10^9)$

b) 1) byte, short oder int

max = 127, 32768, $2^{31}-1$

Short eignet sich wahrscheinlich am besten, da keine vorlesung über 32000 studenten haben wird. 127 ist hingegen für VOs im Audimax zu klein

2) int

3) long (int ist mit 31 Bit für pos. Zahlen zu kurz.)

c) $-2^{63} \quad — \quad 2^{63}-1$

Aufgabe 2: Einheiten

a) Der Hauptspeicher eines Rechners ist 16 GiB groß. Wie groß ist der Speicher in Bytes?

b) Sie kaufen sich eine Festplatte mit einer Größe von 2 TB. Wie viele GiB an Daten kann diese Festplatte speichern?

c) Wie viele MiB an Daten könnte man theoretisch pro Sekunde über eine Netzwerkverbindung übertragen, die eine Übertragungsrate von 10 Gigabit pro Sekunde hat?

d) Angenommen, ein Prozessor braucht für das Addieren zweier ganzer Zahlen 4 ns. Wie viele solcher Additionen kann der Prozessor in einer Sekunde ausführen?

a) $16\,\text{GiB} = 16 \times 2^{30}\,\text{B} = 2^{34}\,\text{Byte}$

b) $\frac{2\,\text{TB}}{1\,\text{GiB}} = \frac{2 \times 10^{12}\,\text{B}}{2^{30}\,\text{B}} = \underline{1862{,}65}$

c) $\frac{10\,\text{Gbit/s}}{1\,\text{MiB/s}} = \frac{10 \times 10^{9}\,\text{bit}}{2^{20} \times 8\,\text{bit}} = \underline{1192{,}1}$

d) $\frac{1\,\text{s}}{4\,\text{ns}} = \frac{1}{4 \times 10^{-9}} = \frac{1}{4} \times 10^{9} = 0{,}25 \times 10^{9}$

$= \underline{250\,000\,000}$

Aufgabe 3: Zahlenumwandlungen

Führen Sie die folgenden Umwandlungen mit möglichst wenig Rechenschritten durch!

a) Wandeln Sie die Hexadezimalzahl $(B2E8.31A4)_{16}$ in eine Binärzahl um.

b) Wandeln Sie die Binärzahl $(10001101111.110100011001)_2$ in eine Oktalzahl um.

c) Wandeln Sie die Quaternärzahl $(213012.10321)_4$ in eine Hexadezimalzahl um.

a) $(B \mid 2 \mid E \mid 8 \mid 3 \mid 1 \mid A \mid 4)_{16}$

$= (1011\,0010\,1110\,1000.0011\,0001\,1010\,01(00))_2$

b) $(100\,0110\,1111.1101\,0001\,1001)_2$

2 1 5 7 . 6 4 3 1

$= (2757{,}6431)_8$

c) $(21, 30, 12 . 10, 32, 1(0))_4$

9 C 6 . 4 E 4

$= (9C6{,}4E4)_{16}$

Aufgabe 4: Rechnen im Binärsystem

Es sind die folgenden Binärzahlen gegeben:

$$A = (1011)_2\ ,\ B = (0110)_2\ ,\ C = (1101)_2\ ,\ D = (1110)_2\ ,\ E = (1110011.11)_2\ ,\ F = (11011110.1011)_2$$

Führen Sie mit diesen Zahlen die folgenden arithmetischen Operationen binär(!) durch.
Hinweis: Bei c) muss nur eine normale Addition durchgeführt werden. Die spezielle Behandlung von Gleitkommazahlen mit IEEE 754 wird auf dem nächsten Übungsblatt besprochen. Außerdem werden alle sechs Zahlen als positive Zahlen aufgefasst.

a) $A + B$

b) $A + C + D$

c) $E + F$

a)

$$\begin{array}{r} 1011 \\ +\ 0110 \\ \hline 10001 \end{array}$$

b)

$$\begin{array}{r} 1011 \\ +\ 1101 \\ +\ 1110 \\ \hline 100110 \end{array}$$

c)

$$\begin{array}{r} 1110011{,}11 \\ +\ 11011110{,}1011 \\ \hline 10101010{,}0111 \end{array}$$

Aufgabe 5: Zahlendarstellungen

Es sind folgende Zahlen gegeben:

$$A = (57BA)_{16}$$
$$B = (-460)_{10}$$
$$C = 0$$

Geben Sie die Zahlen A, B und C als 16 Bit lange Binärmuster in den nachfolgenden Zahlendarstellungen an:

a) Vorzeichenbitdarstellung

b) Einerkomplementdarstellung

c) Zweierkomplementdarstellung

Falls es in einer Zahlendarstellung für dieselbe Zahl unterschiedliche Darstellungen gibt, geben Sie alle an! Wandeln Sie zusätzlich alle Binärmuster in hexadezimale Notation um (z.B. 0101 1111 0010 1000 wird zu 5F28).

460	0
230	0
115	1
57	1
28	0
14	0
7	1
3	1
1	1

a) A: 0101 0111 1011 1010 = 57BA

B: 1 000 0001 1100 1100 = 81CC

C: 0000 0000 0000 0000 = 0000

1000 0000 0000 0000 = 8000

b) A: 0101 0111 1011 1010 = 57BA

B: 1111 1110 0011 0011 = FE33

C: 0000 0000 0000 0000 = 0000

1111 1111 1111 1111 = FFFF

c) A: 0101 0111 1011 1010 = 57BA

B: 1111 1110 0011 0100 = FE34

C: 0000 0000 0000 0000 = 0000

Aufgabe 1: Gleitkommazahlen

Lösen Sie die folgenden Aufgaben:

- Konvertieren Sie die Zahl -48.625 in das IEEE 754 Format (einfache Genauigkeit). Geben Sie die einzelnen Schritte an!
- Welche Zahl wird durch das Bitmuster 01000001101100011000000000000000 im IEEE 754 Format (einfache Genauigkeit) kodiert?
- In einem Hex-Editor werden die Hex-Ziffern 418D0000 angezeigt. Um welche Zahl handelt es sich, wenn das resultierende Bitmuster als eine IEEE 754 Zahl (einfache Genauigkeit) aufgefasst wird?

1)

$$(-48.625)_{10} = (-110000.101)_2$$

48	0
24	0
12	0
6	0
3	1
1	1

$$\begin{array}{r|l} \times .625 & \\ 1 & 25 \\ 0 & 5 \\ 1 & \end{array}$$

$$= (-1.10000101)_2 \times 2^5$$

$$= (-1)^1 \times (1 + (0.10000101)_2) \times 2^{132-127}$$

$$132 = 10000100$$

$$\underbrace{1}_{\text{Sign}}\ \underbrace{10000100}_{\text{exp}}\ \underbrace{10000101000000000000000}_{\text{Frac}}$$

2)

$$\underbrace{0}_{\text{Sign}}\underbrace{10000011}_{\text{exp}}\underbrace{01100011000000000000000}_{\text{Frac}}$$

$$(-1)^0 \cdot (1.01100011)_2 \cdot 2^{(10000011)_2 - 127}$$

$$= (1.01100011)_2 \cdot 2^{131-127} = (1,01100011)_2 \cdot 2^4$$

$$= (10110,0011)_2 = 22 + \frac{1}{8} + \frac{1}{16} = 22,1875$$

3)

$$(418D0000)_{16} = (\underbrace{0}_{\text{Sign}}\underbrace{100\ 0001}_{\text{exp}}\ \underbrace{1000\ 1101\ 0000\ 0000\ 0000\ 0000}_{\text{frac}})_2$$

$$(-1)^0 \cdot (1,00011101)_2 \cdot 2^{(10000011)_2 - 127}$$

$$= (1,00011101)_2 \cdot 2^{131-127} = (1,00011101)_2 \cdot 2^4$$

$$= (10001,101)_2 = 17,625$$

Aufgabe 2: Gleitkommazahlen

Sie haben folgendes Java Code-Fragment gegeben:

```
float f = 0.0f;
for(int i = 1; i < N; i++) {
        f += 0.1;
}
System.out.println(f);
```

Auf einer Testmaschine ergeben sich für verschiedene Werte von N folgende Zahlen als Ausgabe:
100958.24 (bei N = 1000000)
1087936.9 (bei N = 10000000)
2097152.0 (bei N = 100000000)
2097152.0 (bei N = 1000000000)

Warum gibt es keine exakten Ergebnisse bei allen Berechnungen (auch bei kleineren Werten für N)?
Warum verändert sich das Ergebnis nicht mehr für sehr große Werte von N?

1) Es gibt keine exakten Ergebnisse aufgrund von Rundungsfehlern da $(0.1)_{10}$ im Binärsystem eine periodische Darstellung hat: $(0.000110011\ldots)_2$
Es folgen also ähnliche Fehler wie 0.3333 + 0.3333 +0.3333 = 0.9999 anstatt von 1

2) Aufgrund der Gleitkommarepresentation ist es nicht möglich Zahlen zu addieren, welche um zuviele größenordnungen auseinanderliegen. Ist der Exponent so hoch, dass die letzte Stelle der Mantisse einen Stellenwert von über 0.1 repräsentiert, so ändert sich f bei der addition nicht mehr.

Aufgabe 3: MIPS Grundlagen

a) Sie haben folgende Sequenz von MIPS-Befehlen gegeben. Wie lautet ein entsprechender Ausdruck in einer Hochsprache (z.B. Java)? Sie dürfen beliebige Variablenbezeichnungen verwenden!

```
add $t0, $s1, $s2
sub $t1, $s3, $t0
sub $s0, $s4, $t1
```

b) Sie haben folgenden Ausdruck in einer Hochsprache gegeben:

```
f = g + (h - 70)
```

Geben Sie für diesen Ausdruck eine entsprechende Sequenz von MIPS-Befehlen an. Gehen Sie davon aus, dass die Werte für f, g und h in den Registern $s0, $s1, $s2 gespeichert werden!

a) s0 = s4 - (s3 -(s1 + s2))

b)

```
addi $t0, $s2, -70
add $s0, $s1, $t0
```

Aufgabe 4: MIPS Speicherzugriffe

Sie haben folgende Sequenz von MIPS-Befehlen gegeben:

```
lw  $t0, 12($s2)
lw  $t1, 0($s2)
sub $t0, $t0, $t1
sw  $t0, 8($s3)
```

Gehen Sie davon aus, dass die Basisadressen der Arrays A und B in den Registern $s3 und $s2 gespeichert werden. Beantworten Sie bitte folgende Fragen:

- Welcher Ausdruck in einer Hochsprache (z.B. Java) wird dadurch beschrieben?
- Wie viele Bytes müssen A und B im Speicher zumindest belegen, damit der obige Code auf korrekte Positionen in den Arrays zugreift.

a) A[2] = B[3] - B[0]

b) A benötigt mind 12 Byte, B benötigt 16 Byte

Aufgabe 5: Instruktionsformate

a) Sie haben folgendes Bitmuster gegeben:

0010 0001 0001 0001 0000 0000 0111 0110

Welcher MIPS-Assemblerbefehl entspricht diesem Bitmuster?

b) Sie haben folgende Befehle gegeben:

```
ori $t1, $s7, 128
add $t8, $s7, $zero
sw $t1, 32($t2)
```

Geben Sie für jeden Befehl den Typ und die Werte der einzelnen Felder (als dezimale Zahlen) im entsprechenden Instruktionsformat für diesen Typ an!

a) 0010 0001 0001 0001 | 0000 0000 0111 0110

opcode | rs | rt | immediate

$(08)_{16}$ | 8 | 17 | $(0076)_{16} = (118)_{10}$

$\Downarrow$ | $t0 | $s7

addi => I

addi $s1, $t0, 118

b)

ori: format: I
opcode: 13
rs: $s7 = 23
rt: $t1 = 9
immediate: 128

add: format: R
opcode: 0
rs: $s7 = 23
rt: $zero = 0
rd: $t8 = 24
shamt: 0
funct: 32

sw: format: I
opcode: 43
rs: $t2 = 10
rt: $t1 = 9
immediate: 32

Aufgabe 1: Logische Operationen

Gehen Sie für diese Aufgabe von folgenden Registerinhalten aus:

```
$s0 = 0xBADC0DED
$s1 = 0xDEADC0DE
```

Sie haben folgende MIPS-Sequenz gegeben:

```
or  $t0, $s0, $s1
and $t1, $s0, $s1
sub $s2, $t0, $t1
```

Welche Werte beinhalten die Register $t0, $t1, und $s2 nach der Ausführung der obigen Sequenz und wie ergeben sich diese Werte?

1) $t0 = $s0 | $s1

```
    0x BADC0DED
OR  0x DEADC0DE
 =  0xFEFDCDFF
```

A: 1010 10
B: 1011 11
C: 1100 12
D: 1101 13
E: 1110 14
F: 1111 15

$t1 = $s0 | $s1

```
    0x BADC0DED
AND 0x DEADC0DE
 =  0x9A8C00CC
```

$s2 = $t0 - $t1

```
    0x FEFDCDFF
 -  0x 9A8C00CC
    0x 6471CD33
```

Aufgabe 2: Logische Operationen

Sie haben folgende Zuweisung in Java gegeben:

```
y = -(x >>> 31) | (-x >>> 31);
```

Dabei sind y und x Integer-Variablen. x muss vor der Zuweisung mit einem Wert initialisiert worden sein.

- Geben Sie eine Abfolge von MIPS-Befehlen an, die diese Zuweisung umsetzt. Gehen Sie dabei davon aus, dass sich der Wert für x im Register $s0 befindet. Beachten Sie auch, dass ganze Zahlen im Zweierkomplement dargestellt werden.
- Welche Operation wird durch diese Anweisung realisiert?
- Ändert sich das Ergebnis, wenn wir in Java folgende Zuweisung implementieren?

  ```
  y = (x >> 31) | (-x >>> 31);
  ```

  Welchen Befehl müsste man dann für den linken Teil der ODER-Verknüpfung in MIPS benutzen?

1)
```
add $t0, $s0, $zero
srl $t0, $t0, 31
sub $t0, $zero, $t0
sub $t1, $zero, $s0
srl $t1, $t1, 31
or $s1, $t0, $t1
```

2) sgn(x) := -1 wenn x < 0
+1 wenn x > 0
0 wenn x = 0

3) Nein, das ergebnis bleibt gleich, da:

(x >> 31) == -(x >>> 31)

beides ist -1, wenn x negativ ist und ansonsten 0

MIPS: sra (shift right arithmetic) statt srl (logical)

Aufgabe 3: Verzweigung

Sie haben folgendes Code-Fragment gegeben:

```
        slt  $t2, $zero, $t0
        bne  $t2, $zero, ELSE
        j DONE
ELSE:   addi $t2, $t2, 3
DONE:   addi $t2, $t2, 1
```

Gehen Sie davon aus, dass das Register $t0 am Anfang den Wert 0x80000000 enthält.

a) Welchen Wert enthält $t2 nach dem Ausführen des folgenden Code-Fragments? Wie entsteht dieser Wert?

b) Welchen Wert enthält $t2, wenn das Register $t0 am Anfang den Wert 0x7FFFFFFF enthält? Wie entsteht dieser Wert?

c) Wie kann man das Code-Fragment kürzer (weniger Zeilen) schreiben und trotzdem das gleiche Ergebnis erzielen?

```
a) slt $t2, $zero, $t0:
       $t2 = ($zero < $t0)?1:0
           = (0 < INT_MIN)?1:0 = 0

   bne $t2, $zero, ELSE:
       if($t2 != $zero) GOTO ELSE
       if(0 != 0) GOTO ELSE
       if(false)

   j DONE: GOTO DONE

   addi $t2, $t2, 1:
        $t2 = $t2 + 1 = 0 + 1 = 1

b) slt $t2, $zero, $t0:
       $t2 = ($zero < $t0)?1:0
           = (0 < INT_MAX)?1:0 = 1

   bne $t2, $zero, ELSE:
       if($t2 != $zero) GOTO ELSE
       if(1 != 0) GOTO ELSE
       if(true) GOTO ELSE

   addi $t2, $t2, 3:
        $t2 = $t2 + 3 = 1 + 3 = 4

   addi $t2, $t2, 1:
        $t2 = $t2 + 1 = 4 + 1 = 5

c)     slt  $t1, $zero, $t0
       addi $t2, $zero, 1
       beq  $t1, $zero, DONE
       addi $t2, $t2, 4
DONE:
```

Aufgabe 4: Schleife

Sie haben folgendes Code-Fragment gegeben:

```
loop:   bge  $s1, $s0, end
        add  $s3, $s3, $s2
        addi $s1, $s1, 1
        add  $s2, $s2, $s1
        j    loop
end:    ...
```

Gehen Sie davon aus, dass am Anfang die Register $s0, $s1, $s2 und $s3 jeweils die Werte 10, 0, 0 und 0 enthalten. Beantworten Sie bitte folgende Fragen:

- Welchen Wert enthält $s3 nach dem Ausführen des Code-Fragments?
- Welcher Code in einer Hochsprache (z.B. Java oder C) entspricht ungefähr diesem Code-Fragment? Sie dürfen wieder beliebige Bezeichner verwenden.

1)

```
s0 = 10, s1 = s2 = s3 = 0
while(!(s1 >= s0)){
    s3 += s2
    s1 += 1
    s2 += s1
}
```

Alternativ

```
s1 = s2 = s3 = 0
while(s1 < 10){
    s3 += s2
    s1 += 1
    s2 += s1
}
```

2)

durchlauf	s3	s1	s2
0	0	0	0
1	0	1	1
2	1	2	3
3	4	3	6
4	10	4	10
5	20	5	15
6	35	6	21
7	56	7	28
8	84	8	36
9	120	9	45
10	165	10	55
ABBRUCH			

Aufgabe 5: Unterprogramm

Sie haben folgendes Unterprogramm gegeben:

```
func:   add $v0, $a0, $zero
        blt $a0, $a1, endif
        add $v0, $a1, $zero
endif:  jr $ra
```

Beantworten Sie bitte folgende Fragen:

- Wie viele Parameter werden laut Konvention übergeben?
- Wo wird der Rückgabewert abgelegt?
- Was macht dieses Unterprogramm?
- Wird hier der Stack benutzt?
- Welcher Wert wird in $ra abgelegt, wenn das Unterprogramm aufgerufen wird? Sie müssen hier nur allgemein antworten und keinen konkreten Wert angeben.

1) 2 Parameter: $a0 und $a1

2) Rückgabewert in $v0

3) $v0 = min($a0, $a1)

4) Nein, dies ist nicht notwendig, da keine register aus $v und $a verwendet werden und auch kein unterprogram aufgerugen wird.

5) $ra beinhaltet die Rücksprungadresse. (Die Adresse der Programmzeile nach Aufruf des Unterprograms)

Aufgabe 1: Schleifen

Sie haben folgendes Code-Fragment in einer Hochsprache gegeben:

```
int sum = 0;
for(int i = 200; i > 0; i = i/4)
    sum = sum + i;
```

Geben Sie eine entsprechende Sequenz von MIPS-Befehlen an, die das Code-Fragment realisiert. Beachten Sie dabei:

- Der Wert der Variable i wird im Register $s0 gespeichert.
- Der Wert der Variable *sum* wird im Register $s1 gespeichert.

```
        li $s1, 0
        li $s0, 200
loop:   blez $s0, done
        add $s1, $s1, $s0
        sra $s0, $s0, 2
        j loop
done:
```

Aufgabe 2: Programm verstehen

Sie haben folgendes Programm gegeben:

```
        .data

size:   .word   10
arr:    .word   10, -5, 8, 0, 6, 87, -68, 97, 25, -32

        .text

main:   lw $s1, size
        ori $s2, $zero, 0
        ori $s3, $zero, 0
        ori $s4, $zero, 0
        ori $s5, $zero, 0
        la $s6, arr

loop:   bge $s5, $s1, done
        lw $s7, 0($s6)
        addu $s2, $s2, $s7
        blez $s7, lez
        addu $s3, $s3, $s7
lez:    bgez $s7, inc
        addu $s4, $s4, $s7
inc:    addi $s5, $s5, 1
        addi $s6, $s6, 4
        j loop

done:   li $v0, 10
        syscall
```

Beschreiben Sie, welche Funktionalität durch das Programm realisiert wird. Gehen Sie dabei auf folgende Punkte ein:

- Welche Pesudobefehle werden in diesem Programm verwendet?
- Wie viele Schleifen finden Sie in diesem Programm?
- Welche drei Werte werden in diesem Programm berechnet und in den Registern $s1, $s2 und $s3 abgelegt? Wie werden diese Werte berechnet?

```
s1 = 10
arr = [10, -5, ..., -32]
s2, s3, s4, s5 = 0
s6 = ref(arr[0])

while(s5 < s1){
    s7 = arr[s6/4]
    s2 += s7
    if(s7 > 0) s3 += s7
    if(s7 < 0) s4 += s7
    s5++;
    s6 += 4;
}
```

```
arr = [10, -5, ..., -32]
sum, sumpos, sumneg = 0

for(int i = 0; i < arr.length; i++){
    e = arr[i]
    sum += e
    if(e > 0) sumpos += e
    if(e < 0) sumneg += e
}
```

Pseudobefehle: `la, bge`

1 Schleife, 2 Ifs (sprung nach unten)

Programm berechnet summe der Arrayelemente sowie die Summe aller positiven und aller negativen elemente

Aufgabe 3: MIPS Theorie

Beantworten Sie bitte folgende Fragen zu MIPS:

- Wie groß ist der Adressbereich in Bytes, der in MIPS bei einem bedingten Sprung abgedeckt werden kann?
- Wie groß ist der Adressbereich in Bytes, der in MIPS bei einem unbedingten Sprung abgedeckt werden kann?
- Welcher MIPS-Instruktion entspricht das Maschinenwort 0x00000000? Welches Instruktionsformat wird hier benutzt und welchen Nutzen könnte diese Instruktion haben?

1) Bedingter Sprung: PC-relative addressing:

16 Bit eingabe wird mit 4 multipliziert und dann zur aktuellen adresse hinzugezählt.

$2^{16} \cdot 4$ Byte $= 2^{18}$ B $= 2^{8}$ KiB $= 256$ KiB

2) unbedingter Sprung: Pseudodirekte Adressierung

26 Bit eingabe, wird mit 4 multipliziert und mit den ersten 4 bits des befehlszählers verknüpft

$2^{26} \cdot 4$ Byte $= 2^{28}$ B $= 2^{8}$ MiB $= 256$ MiB

3)

opcode	rs	rt	rd	shamt	funct
0	0	0	0	0	0

opcode 0, funct 0: sll

sll $zero, $zero, 0

Dieser Befehl erfüllt nicht wirklich einen zweck, da er 0 um 0 bit shifted und das Ergibnis (welches auch 0 ist) versucht in das $zero register zu schreiben, welches aber readonly ist.

Praktischer Nutzen: ein CPU zyklus ohne funktion, um bsp in einer schleife eine gewisse zeit abzuwarten.

Aufgabe 4: Programm übersetzen

Beantworten Sie bitte kurz folgende Fragen:

- Welche Aufgabe erfüllt ein Compiler?
- Welche Aufgabe erfüllt ein Assembler?
- Welche Aufgabe erfüllt ein Linker?
- Wozu haben wir Betriebssystemaufrufe (system calls)?

Compiler: Übersetzen eines Programms von einer (Hoch-)Sprache in Assembler Code

Assembler: Übersetzten eines Assembler Programmes in ein Modul in Maschinencode, welches allerdings noch nicht ausführbar ist, da librarys fehlen und noch keine absoluten referenzen existieren

Linker: Verbinden des Moduls mit Bibliotheken und output von Ausführbarem Maschienencode

System Calls: Zugriff auf funktinen des Betriebsystems wie z.b. Input/Output von Text, Beenden des Programms, etc.

Aufgabe 5: RISC und CISC

Für diese Fragen müssen Sie zunächst ein paar Seiten aus folgendem Buch lesen:

`Roland Hellmann, Rechnerarchitektur : Einführung in den Aufbau moderner Computer, 2. Auflage, De Gruyter Oldenbourg, 2016.`

Sie können innerhalb der TU (oder mit VPN von außen) über den TU CatalogPlus (catalogplus.tuwien.ac.at/) auf die Online-Version des Buches zugreifen. Lesen Sie in Teil 1 (Grundlagen) die Seiten 43 - 47. Beantworten Sie dann bitte folgende Fragen:

- Was versteht man unter der semantischen Lücke zwischen Hochsprachen und der Maschinensprache eines Prozessors?
- Welche Vor- bzw. Nachteile haben CISC-Architekturen?
- Was versteht man unter dem Paretoprinzip bei Computerprogrammen?
- Welche Vor- bzw. Nachteile haben RISC-Architekturen?

1) Unter Semantische Lücke versteht man die unterschiedliche Leistungsfähigkeit von Befehlen in Hochsprache und Maschinensprache. Während bei CISC Rechnern ein Hochsprachenbefehl oft in wenige Maschinenbefehle übersetzt wird, kann es sein, dass man bei RISC rechnern wesentlich mehr Maschinenbefehle benötigt.

2) CISC Architekuren

Vorteile:

einfachere compiler und einfacherere optimierung
bessere leistung

Nachteile:

Befehle werden großteils nicht genutzt
Maschinenbefehle brauchen unterschiedlich lange
komplexerer prozessoraufbau

3) Unter dem Paretoprinzip versteht man die Tatsache, dass 80% eines Programmes aus nur 20% der Maschinenbefehle besteht, die restlichen 80% der Maschinenbefehlee werden nur in 20% des Programmes genutzt. (Allgemeine 80-20 Regel)

4) RISC Architekur

Vorteile:

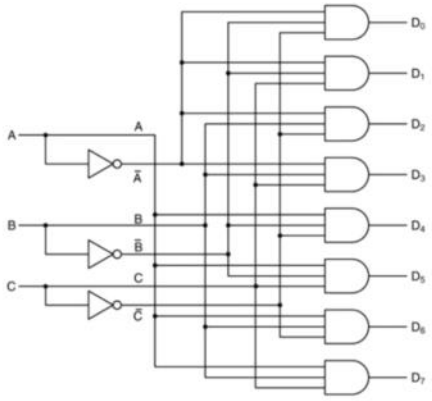
Mehr Platz im Prozessor für Cache, register, etc.
Jeder befehl braucht einen takt

Nachteile:

Große Semantische Lücke, die vom Compiler bewältigt werden muss
Programme werden länger, da mehr Befehle benötigt werden

Aufgabe 1: Schaltnetz interpretieren

Sie haben folgendes Schaltnetz gegeben:



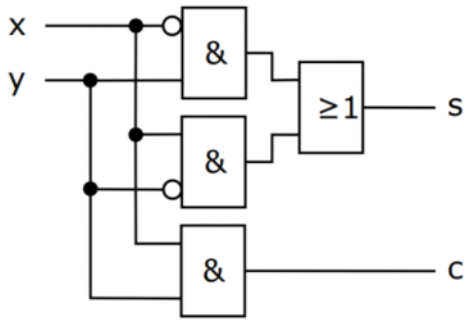
Beschreiben Sie diese Schaltung:

- Welche Gatter werden in dieser Schaltung verwendet?
- Welche Notation für Gatter wird in dieser Schaltung verwendet?
- Welche Aufgaben erfüllen $D_0, D_1, ..., D_7$ und A, B, C?
- Welche Funktionalität wird durch diese Schaltung realisiert?
  Hinweis: Vergleichen Sie dieses Schaltnetz mit den in der Vorlesung besprochenen Schaltnetzen.

1) NOT und AND Gatter
2) US Norm
3) A,B,C Eingänge
   D0, D1, …, D7 Ausgänge
4) Decoder

Aufgabe 2: Schaltnetz interpretieren

Sie haben folgendes Schaltnetz gegeben:



Beschreiben Sie diese Schaltung:

- Welche Gatter werden in dieser Schaltung verwendet?
- Erstellen Sie die Wahrheitstabelle für diese Schaltung!
- Was wird durch diese Schaltung realisiert?
- Könnte man die gleiche Schaltung mit weniger Gatter realisieren? Wenn ja, warum könnte auch die obige Schaltung sinnvoll sein?

1) AND und OR Gatter (und NOT für die Inputs)
2)

x	y	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

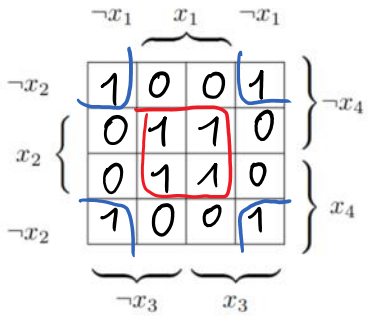
3) S = XOR
C = AND
Kann auch als Addition von z zwei 1 Bit zahlen mit Übertrag aufgefasst werden.

4) Schaltung wäre auch durch ein XOR und ein AND zu realisieren. Obige Schaltung kann sinnvoll sein, wenn man nur AND, OR und NOT zur Verfügung hat.

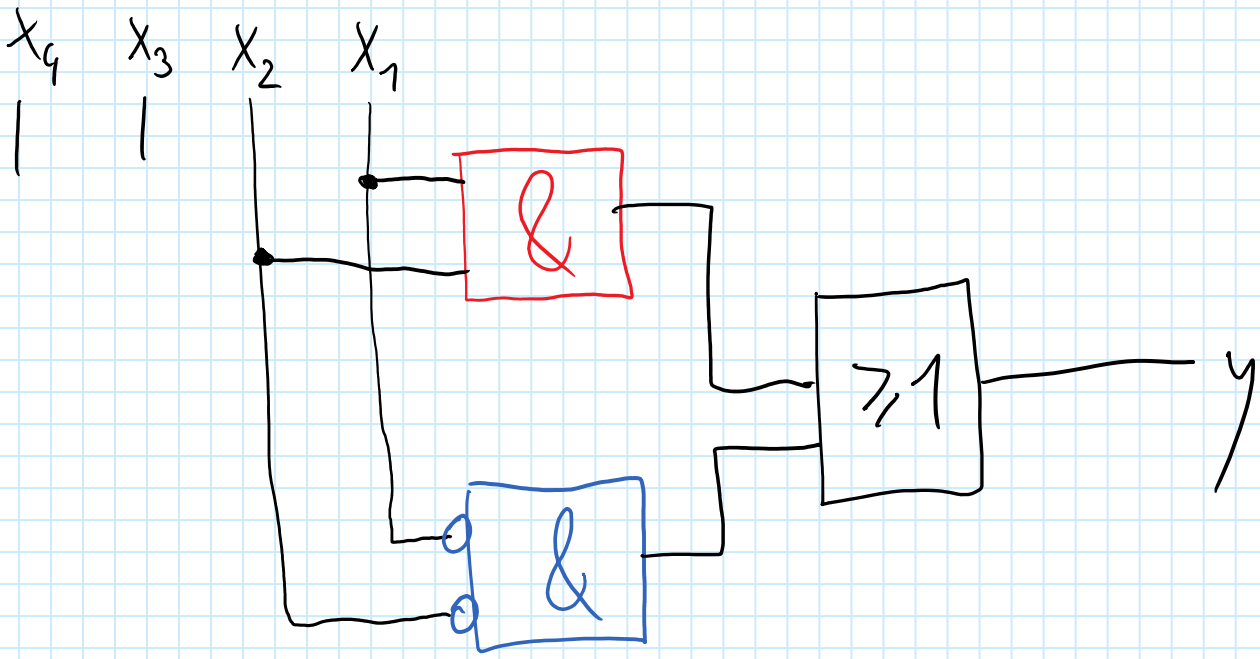
Aufgabe 3: Minimierung

Sie haben eine Wahrheitstabelle mit vier Eingängen x_4, x_3, x_2 und x_1 und einem Ausgang y gegeben. Wie lautet das KV-Diagramm dafür? Geben Sie eine minimale Schaltung in disjunktiver Form (Normalform) dafür an!

x_4	x_3	x_2	x_1	y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



$x_1 \wedge x_2$ $\neg x_1 \wedge \neg x_2$



Aufgabe 4: Addition, Subtraktion

Sie haben einen Rechner gegeben, der Zahlen mit 8 Bit im Zweierkomplement darstellt. Gehen Sie für die folgenden Berechnungen von den drei dezimalen Zahlen A = 112, B = -72, C = 85 aus.

a) Geben Sie für jede Zahl das entsprechende Bitmuster im Speicher an!

b) Geben Sie für jede der folgenden Berechnungen den Rechengang und das Ergebnis in binärer Darstellung an. Geben Sie bei jedem Ergebnis auch an, ob es korrekt ist, oder ein Überlauf aufgetreten ist.

- A + B
- A – C
- A + C

112	0
56	0
28	0
14	0
7	1
3	1
1	1

72	0
36	0
18	0
9	1
4	0
2	0
1	1

85	1
42	0
21	1
10	0
5	1
2	0
1	1

$A = (112)_{10} = (1110000)_2$ $B = -(72)_{10} = -(1001000)_2$ $C = (85)_{10} = (1010101)_2$

A: 0111 0000 B: 10111000 C: 01010101

$$\begin{array}{rl} A & 0111\ 0000 \\ +B & 1011\ 1000 \\ \hline & 0010\ 1000 \end{array}$$

$= 40$

kein Overflow

$$\begin{array}{rl} A & 0111\ 0000 \\ -C & 01010101 \\ \hline & 00011011 \end{array}$$

$= 16+8+2+1 = 27$

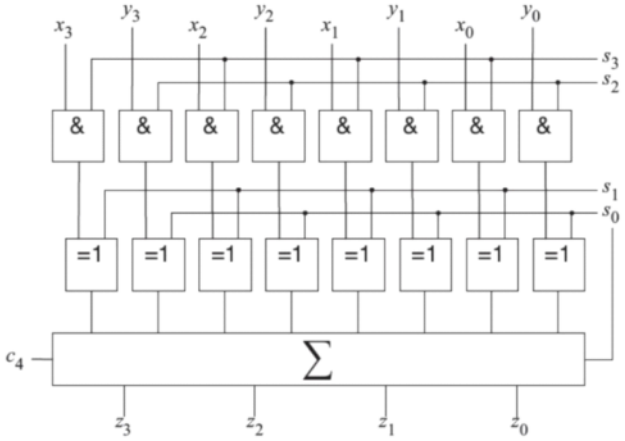
kein Overflow

$$\begin{array}{rl} A & 0111\ 0000 \\ +C & 01010101 \\ \hline & 11000101 \end{array}$$

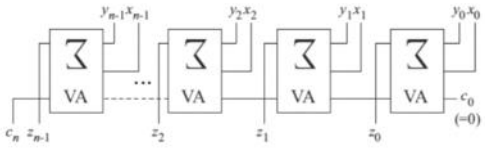
$= -(0111011)_2 = -32-16-8-2-1 = -99$ Overflow $(+)+(+)=(-)$

Aufgabe 5: ALU

Sie haben folgendes Schaltnetz (spezielle ALU) gegeben:



Dieses Schaltnetz verarbeitet als Eingabe zwei Zweierkomplementzahlen $x = x_3x_2x_1x_0$ und $y = y_3y_2y_1y_0$ mit jeweils vier Bit. Abhängig von den Steuersignalen s_0, s_1, s_2 und s_3 wird durch den 4-Bit Addierer (Block mit Summenzeichen) die Zweierkomplementzahl $z = z_3z_2z_1z_0$ berechnet.
Beispiel: Für $s_0 = 0$, $s_1 = 0$, $s_2 = 1$ und $s_3 = 1$ berechnet dieses Schaltnetz $x + y$.
Der interne Aufbau für den hier verwendeten n-Bit Addierer sieht folgendermaßen aus:



Hinweis: c_0 wird bei der speziellen ALU durch s_0 bestimmt und ist nicht immer 0!
Welche Berechnungen werden durch folgende Kombinationen der Steuersignale realisiert bzw. welche Ergebnisse erhält man dann?

- $s_0 = 1$, $s_1 = 0$, $s_2 = 1$ und $s_3 = 1$
- $s_0 = 0$, $s_1 = 1$, $s_2 = 1$ und $s_3 = 1$

S2, S3: Setzen y bzw auf x auf 0, falls 0, ansonsten y bzw x
S0, S1: Falls 1 invertiert y bzw x, ansonsten nicht, S0 ist zusätzlich das CarryInto der ersten Alu (daher, wenn S0==1 => y <- -y (Zweierkomplement, invertieren +1))

1) x-y

2) (-x-1)+y = y-x-1 (carry bit fehlt, daher wird nur einserkomplement von x gebildet)

Aufgabe 1: Multiplikation

Zeigen Sie, wie die beiden vorzeichenlosen dezimalen Zahlen 12 und 6 miteinander binär multipliziert werden. Gehen Sie dabei von der beschriebenen Implementierung auf Folie 30 (Rechnerarithmetik) aus und erstellen Sie eine Tabelle wie auf Folie 31 (Rechnerarithmetik). Die gegebenen Zahlen werden mit vier Bit kodiert.

$(12)_{10} = (1100)_2$ $\quad$ $(6)_{10} = (0110)_2$

6 x 12

Iteration		Schritt	Multiplikator	Multiplikand	Produkt
0	Anfangswerte		0110	0000 1100	0000 0000
1	1a:	0 ⇒ keine Op			0000 0000
	2:	schiebe Multiplikand nach links		0001 1000	
	3:	schiebe Multiplikator nach rechts	0011		
2	1a:	1 ⇒ Prod += Multiplikand			0001 1000
	2:	schiebe Multiplikand nach links		0011 0000	
	3:	schiebe Multiplikator nach rechts	0001		
3	1:	1 ⇒ Prod += Multiplikand			0100 1000
	2:	schiebe Multiplikand nach links		0110 0000	
	3:	schiebe Multiplikator nach rechts	0000		
4	1:	0 ⇒ keine Op			0100 1000
	2:	schiebe Multiplikand nach links		1100 0000	
	3:	schiebe Multiplikator nach rechts	0000		

$(0100\ 1000)_2 = (64+8)_{10} = (72)_{10} \overset{\checkmark}{=} (12 \times 6)_{10}$

Aufgabe 2: Takt, Taktzyklen

Sie haben folgende historische Daten von CPU-Taktfrequenzen gegeben:

- Motorola 68000 (1979): 8 MHz
- AMD Am5x86 (1995): 150 MHz
- Intel Core i7 3630QM (2012): 3.2 GHz

a) Berechnen Sie für jeden einzelnen Prozessor die Länge des Taktintervalls in Nanosekunden!

b) Eine Leistungsmessung ergibt, dass auf dem Motorola 68000-Prozessor für ein bestimmtes Programm durchschnittlich 2 Millionen Befehle pro Sekunde ausgeführt werden können. Wie lange dauert dann ein Befehl im Durchschnitt (in Taktzyklen)?

c) Auf dem Intel Prozessor benötigt ein Programm 3.5 Sekunden. Angenommen, ein Befehl benötigt im Durchschnitt 4 Taktzyklen. Wie viele Befehle werden bei diesem Programm ausgeführt?

a) $\frac{1}{8\,MHz} = \frac{1}{8} \cdot 10^{-6}\,s = 0{,}125 \cdot 10^{-6}\,s = 125 \cdot 10^{-9}\,s = \underline{125\,ns}$

$\frac{1}{150\,MHz} = \frac{1}{1{,}5} \cdot 10^{-2} \cdot 10^{-6}\,s = \frac{2}{3} \cdot 10^{-8}\,s = 0{,}\dot{6} \cdot 10^{-8} = 6{,}\dot{6} \cdot 10^{-9} = \underline{6{,}66\,ns}$

$\frac{1}{3{,}2\,GHz} = \frac{1}{3{,}2} \cdot 10^{-9}\,s = 0{,}3125 \cdot 10^{-9}\,s = 312{,}5 \cdot 10^{-12}\,s = \underline{312{,}5\,ps}$

b) $\frac{8\,MHz}{2 \cdot 10^6\,Bef/s} = \frac{8}{2} \cdot \frac{10^6\,Zyklen}{10^6\,Bef} = \underline{4\ Taktzyklen/Befehl}$

c) $3{,}5\,s \cdot \frac{3{,}2\,GHz}{4} = \frac{3{,}5 \cdot 3{,}2}{4} \cdot 10^9 \cdot \frac{s}{s} = \underline{2{,}8 \cdot 10^9}$

Aufgabe 3: Prozessor

Sie haben folgende MIPS-Befehle gegeben:

- and $rd, $rs, $rt
- sw $rt, Offset($rs)

Schreiben Sie für jeden davon die logischen Werte der folgenden Steuersignale an und erklären Sie den Sinn dieser Steuersignale:

- RegWrite
- RegDst
- ALUSrc
- MemWrite
- MemtoReg
- ALUOp (2 Bits)

	and	sw	
RegWrite	1	0	Schreiben in Register: Ja (1), Nein (0)
RegDst	1	X	Zielregister für Schreibzugriff: rt (0), rd (1)
ALUSrc	0	1	Input für ALU: Reg1+Reg2 (0), Reg1+Imm (1)
MemWrite	0	1	Schreibzugriff auf Speicher: Ja (1), Nein (0)
MemtoReg	0	X	Quelle für Schreiben in Register: ALU (0), Speicher (1)
ALUOp	10	00	Operationscode für ALU Steuerung: add (00), sub (01), Funktion aus Funktionscode (10)

Aufgabe 4: Prozessor

Sie haben folgende MIPS-Befehle gegeben:

- add $rd, $rs, $rt
- beq $rs, $rt, label

Welche der folgenden Komponenten des Datenpfads des MIPS-Prozessors werden von den genannten Befehlen benötigt:

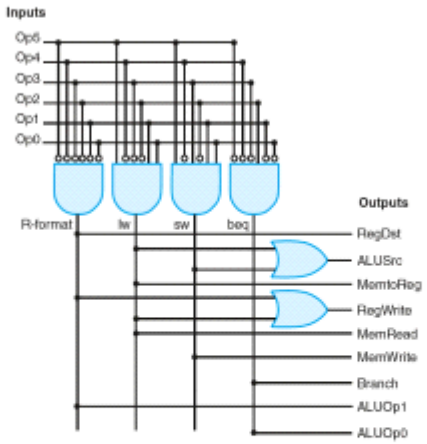
- Instruction memory
- Data memory
- Registers
- ALU
- Sign-extend
- ALU control

Erklären Sie, warum eine Komponente für die Ausführung eines Befehls benötigt wird bzw. nicht benötigt wird!

	add	beq	
Instruction memory	Ja	Ja	Das Instruction Memory wird bei jedem Befehl gebraucht, da dieser von dort geladen wird.
Data memory	Nein	Nein	In beiden Fällen müssen werdenLese- oder Schreiboperationen am Data Memory durchgeführt.
Registers	Ja, r/w	Ja, readonly	
ALU	Ja, HauptALU wird zum addieren verwendet	Ja, HauptALU subtrahiert um zu vergleichen, Zweite ALU berechnet die Zieladresse	
Sign-extend	Nein	Ja	Sign-extend wird benötigt um 16 Bit immidiates auf 32 Bit werte zu extenden. (Z.b. die relative Sprunngadresse bei beq)
ALU control	Ja	Ja	Immer wenn die HauptALU benötigt wird, benötigt man auch die ALU controll, welche aus dem Befehlt die notwendigen Steuersignale für die ALU generiert.

Aufgabe 5: Prozessor

Sie haben folgendes Schaltnetz gegeben:



Beschreiben Sie die Arbeitsweise dieses Schaltnetzes:

a) • Bei welchem Inputs liefern die einzelnen UND-Gatter jeweils eine logische 1?

b) • Welchen Sinn haben die beiden ODER-Gatter?

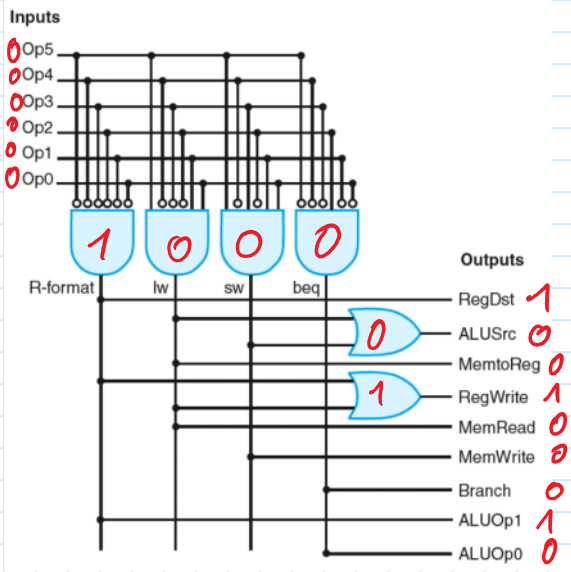
c) • Zeigen Sie für den Befehl sub, wie die einzelnen Outputs entstehen.

d) • Welche Komponente im MIPS-Datenpfad wird durch dieses Schaltnetz realisiert?

a) R-format: 000000 (Op0 … Op5)
lw: 110001
sw: 110101
beq: 001000

b) ALUSrc und RegWrite müssen bei zwei verschiedenen Befehlen jeweils 1 sein

c) sub: Opcode 0



d) Hauptsteuereinheit

Opcode ↦ Steuersignalen

Aufgabe 1: Datenpfad - Wiederholung

Gehen Sie für diese Aufgabe von einem einfachen MIPS-Datenpfad (z.B. Folie 26 (Prozessor)) aus. Der Prozessor holt folgenden Befehl aus dem Speicher von der Adresse 0x00400076:

10101100101000100000000001111000

Gehen Sie weiters von folgenden Registerbelegungen aus (Registernummer/Belegung als Dezimalwert):

1/-1000, 2/200, 3/123, 4/30000, 5/100, 6/30

Hinweis: Hier sind mehrere Register angegeben. Wählen Sie jene Register aus, die sie für den Befehl benötigen.

Beantworten Sie folgende Fragen!

a) Welchen Output liefert die Komponente Sign-extend für dieses Befehlswort?

b) Wie lauten die Inputs für die Komponente ALU-control und welcher Input wird von der ALU-control benutzt?

c) Wie lauten die Inputs für die ALU und die zwei weiteren Addierer (Add-Komponenten)?

32 8 21 0 0 7 8

10101100101000100000000001111000

OPCode rs rt immidiate

43 5 2 0x0078

SW

I-Format

SW $2, 0x0078($5)

a) 0x0078 → 0x0000 0078

b) 00, ALUcontroll hat ALUOp und func als input

Bei 00 wird jedoch die func ignoriert (weil I-Format) und immer addiert

c) ALU: $5 = $(100)_{10}$, 0x0078

Instruction add: 0x00400076, 4

(Instruction add2: 0x0040007A, 0x0000 0078 << 2 = 0x0000 01E0)

Aufgabe 2: Pipelining - Grundlagen

Gehen Sie von einem MIPS Datenpfad aus und nehmen Sie folgende Verzögerungen in den einzelnen Stufen an: IF = 200 ps, ID = 150 ps, EX = 250 ps, MEM = 300 ps, WB = 100 ps

Beantworten Sie bitte folgende Fragen:

a) Wie groß ist die Taktzykluszeit für einen Prozessor ohne Pipelining?

b) Wie groß ist die Taktzykluszeit für einen Prozessor mit Pipelining?

c) Wie lange benötigt ein lw-Befehl bei einem Prozessor ohne Pipelining?

d) Wie lange benötigt ein lw-Befehl bei einem Prozessor mit Pipelining?

e) Wie lange benötigt ein lw-Befehl bei einem Prozessor mit Pipelining, wenn die MEM-Phase nur mehr 200 ps benötigt?

a) $200+150+250+300+100 = 1000\,ps = 1\,ns$

b) 300 ps

c) 1000 ps

d) $5 \cdot 300\,ps = 1500\,ps$

e) $5 \cdot 250\,ps = 1250\,ps$

Ohne Pipelining: Taktzyklus solange wie Summe

1 Befehl = 1 Taktzyklus

Mit Pipelining: Taktzyklus so lange wie längste Operation

1 Befehl = 5 Taktzyklen

Aufgabe 3: Pipelining - Abhängigkeiten

Sie haben folgende Befehlssequenz gegeben:

```
or  $1 , $2 , $3
or  $2 , $1 , $4
lw  $4 , 0($1)
or  $5 , $3 , $2
```

a) Wo gibt es hier Abhängigkeiten?

b) Welche Arten von Konflikten können hier auftreten, wenn diese Befehlsfolge in einer MIPS-Pipeline ohne Forwarding ausgeführt wird? Wie kann man diese Konflikte auflösen?

c) Können Konflikte auftreten, wenn diese Befehlsfolge in einer MIPS-Pipeline mit Forwarding ausgeführt wird? Wenn ja, wie kann man diese Konflikte auflösen?

d) Wie viele Taktzyklen benötigt man bei einer MIPS-Pipeline ohne bzw. mit Forwarding für die Abarbeitung der ganzen Sequenz?

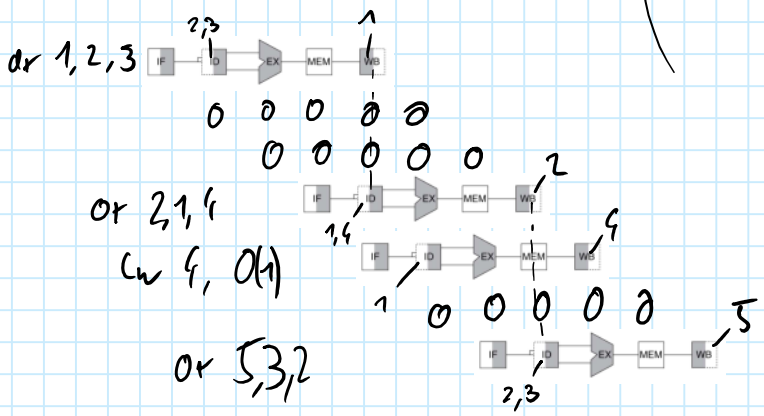
a) Das 2. or benötigt das ergebnis des ersten
lw benötigt ebenso das ergebnis des ersten or
Das letze or benötigt das ergebnis des 2. or

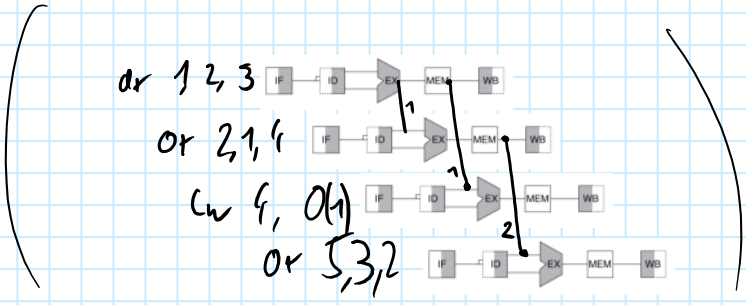
b) Es entstehen Datenkonflikte.
Zwischen einer Abhägigkeit müssen mindestens 2 Taktzyklen vergehen
Man kann daher zwei bubbles (nop befehle) zwischen den ersten ors einfügen, sowie ein nop vor dem letzten or. (lw übernimmt hier einen der beiden Taktzyklen

c) Forwarding von EX/MEM nach ID/MEM kann aufjedenfall den ersten konflikt der direkthintereinander ausgeführten or befehle auflösen.
Bei lw und dem dritten or müssten die Daten allerdings erst von MEM/EX geforwarded werden. (Also über zwei phasen statt eine hinweg)
Möchte man dies aus welchem Grund auch immer vermeiden, kann man die Befehle einfach tauschen (sie haben keine Abhängigkeit voneinander). Jetzt bekommt das 3. or direkt das Ergebnis vom 2. geforwardet und lw wird erst nach zwei anderen Zyklen nach seiner Abhängigkeit aufgerufen und erhält damit die korrekten Werte aus den Registern.

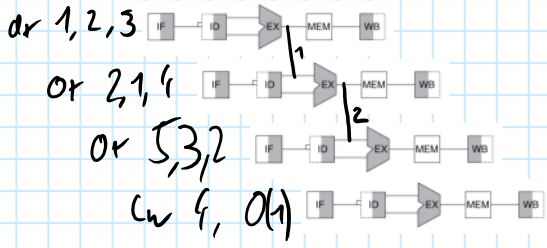
d) 11 Zyklen ohne Forwarding
8 Zyklen mit forwarding

b)





c)



Aufgabe 4: Pipelining - Bubbles

Sie haben folgende Befehlssequenz gegeben:

```
lw $2, 8($3)
add $1, $2, $3
or $1, $1, $2
add $4, $6, $3
```

Gehen Sie von einem MIPS-Prozessor mit Pipelining und Forwarding aus. Beantworten Sie bitte folgende Fragen:

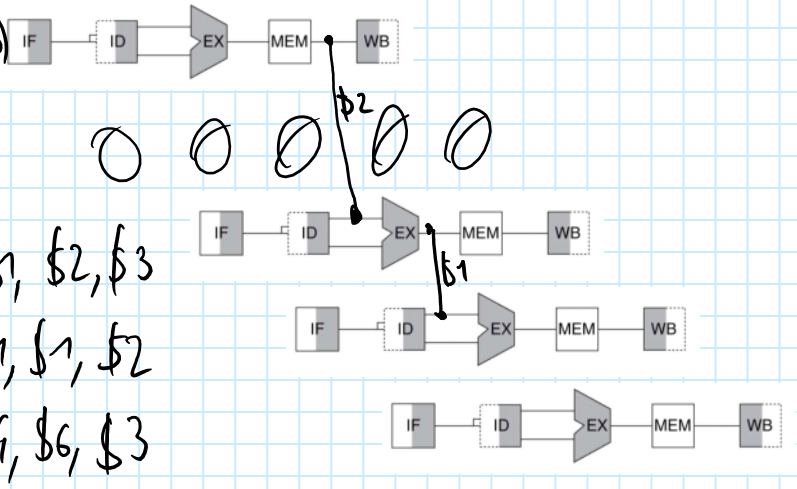
a) Wo müssen Sie Leertakte (Bubbles) einführen, damit der Code korrekt verarbeitet werden kann. Gehen Sie in diesem Fall wie zum Beispiel auf Folie 41 (Prozessor) vor.

b) Ordnen Sie die Befehle so um, dass Sie keine Leertakte mehr benötigen!

c) Können Sie die Befehle beliebig umordnen? Erklären Sie Ihre Antwort!

a)

lw $2, 8($3)



add $1, $2, $3

or $1, $1, $2

add $4, $6, $3

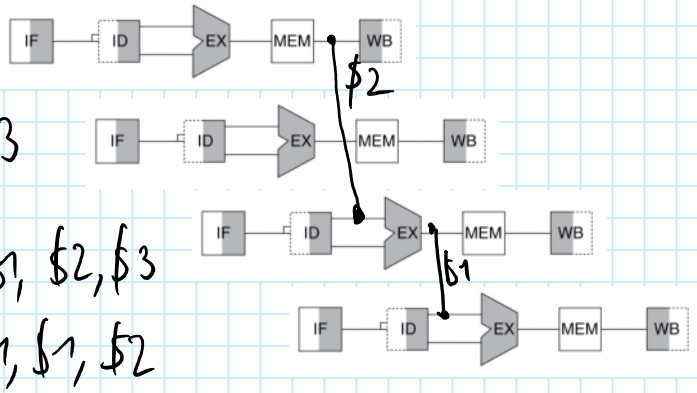
b)

lw $2, 8($3)

add $4, $6, $3

add $1, $2, $3

or $1, $1, $2



c) Umordnen ist nur dann möglich, wenn alle Abhängigkeiten in derselben reihenfolge (write vor read) bleiben

Aufgabe 5: Pipelining - Steuerkonflikte

Sie haben folgendes MIPS-Fragment gegeben:

```
        1 sub $t1, $t5, $t3
Label1  2 sw $t1, 0($t2)
        3 add $t2, $t2, $t3
        4 add $t1, $t3, $t3
        5 bne $t2, $t4, Label1
        6 sub $t5, $t5, $t1
        7 sw $t5, 0($t2)
```

Hinweise:

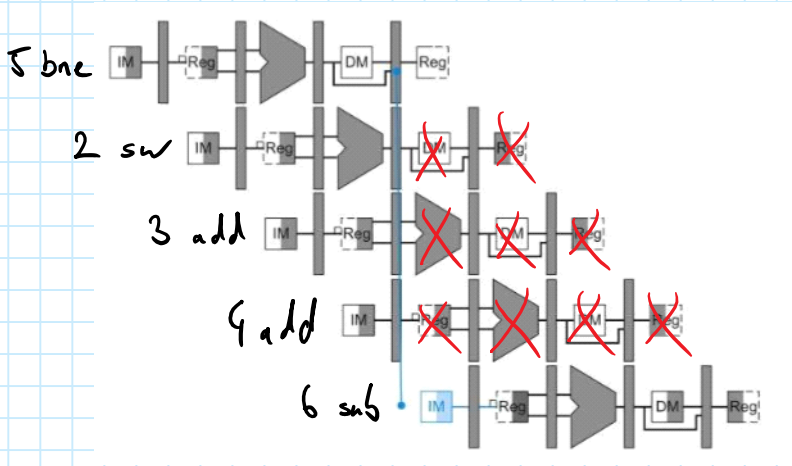
- Der Branch-Befehl entscheidet in der MEM-Stufe, ob gesprungen wird.
- Bei der Ausführung wird immer angenommen, dass der Sprung ausgeführt wird.

Beantworten Sie bitte folgende Fragen:

- Wo kommt es hier zu Steuerkonflikten und wie wirken sich diese aus, wenn dieses Fragment auf einem Prozessor mit Pipelining und Forwarding ausgeführt wird?
- Muss man auch noch zusätzliche Leertakte an anderen Stellen einführen?
- Es wird angenommen, dass ein Sprung immer ausgeführt wird. Welchen Nachteil könnte diese Strategie haben?

1) Es gibt einen Steurkonflikt beim bne-Befehl, da erst nach 3 Zwischentakten feststeht, ob gesprungen wird oder nicht. Der Konflikt tritt nur einmal am Ende der Schleife auf, da immer angenommen wird, dass gesprungen wird und nur im Fall, dass es nicht passiert die vorhergesagte Entscheidung falsch ist.

In diesem Fall müssen die Befehle aus dem falschen Zweig, welche bereits mit der ausführung begonnen haben zu NOP Befehelen umgewandelt werden, damit es keine ungewollten auswirkungen auf register oder speicher gibt und der korrekte nächste befehl aus dem Instruction memory geholt werden.



2) Nein, alle Datenkonflikte (siehe farbliche markierung im Code) lassen sich durch forwarding beheben, da kein Load-Use-Konflikt auftritt.

3) Würde man z.b. eine Kopfgesteuere Schleife statt einer Fußgesteuerten verwenden so würden die 3 Bubbles in jedem Durchlauf generiert und nur im letzten Durchlauf würde die Optimierung durch diese annahme eine Wirkung zeigen.

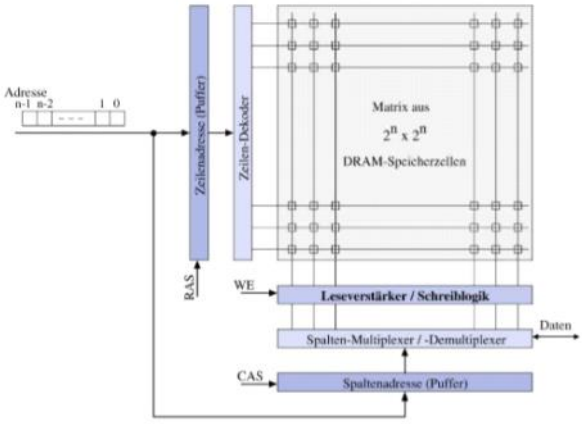
```
loop: beq $1, $2, done
      # do something
      j loop
done: # do something else
```

Aufgabe 1: RAM Speicher

Gehen Sie von einer DRAM-Speicherorganisation wie auf Folie 14 (Speicherhierarchie) aus.
Ein DRAM Speicher ist 512 MiB groß.

- Wie viele Zeilen hat dieser Speicher?
- Wie groß ist eine Adresse?

  Hinweis: Hier geht es nur um eine Adresse für eine einzelne Speicherzelle. Diese Speicherzelle wird nur ein Bit beinhalten. Es können aber z.B. 8 solche Speicher parallel abgefragt werden und damit wieder ein Byte ausgelesen werden.
- Wie viele Bits werden für die Spaltenadressierung verwendet?
- Nehmen Sie an, dass eine Speicherzelle alle 50 ms einen Refresh benötigt. Der periodische Refresh wird immer auf eine ganze Zeile ausgeführt. Wie viel Zeit (in Nanosekunden) darf maximal zwischen dem Refresh von zwei aufeinanderfolgenden Zeilen liegen?



1) $512\ MiB = 2^9\ MiB = 2^9 \cdot 2^{20}\ B = 2^{29} \cdot 2^3\ Bit$
$= 2^{32}\ Bit$

$2^{16} \times 2^{16} \Rightarrow \underline{2^{16}\ Zeilen}$

2) $\underline{32\ Bit}$ Adressen

3) $\underline{16\ Bit}$ für Spaltenadresse

4) $\frac{50\ ms}{2^{16}\ Zeilen} = \frac{50}{1024 \cdot 64}\ ms = \underline{763\ ns}$

$\hookrightarrow \frac{\cancel{50}}{1024 \cdot \cancel{64}}\ ms \leq \frac{1}{1024}\ ms \leq \frac{1}{1000}\ ms \leq 1\ \mu s$

Aufgabe 2: Flash-Speicher

Für diese Fragen müssen Sie zunächst ein paar Seiten aus folgendem Buch lesen:

`Roland Hellmann, Rechnerarchitektur : Einführung in den Aufbau moderner Computer, 2. Auflage, De Gruyter Oldenbourg, 2016.`

Sie können innerhalb der TU (oder mit VPN von außen) über den TU CatalogPlus (catalog-plus.tuwien.ac.at/) auf die Online-Version des Buches zugreifen. Lesen Sie in Teil 5 (Speicher und Peripherie) die Seiten 282 - 283. Beantworten Sie dann bitte folgende Fragen:

- Welche Arten von Flash-Speichern gibt es?
- Wie unterscheiden sich SLC- und MLC-Speicher voneinander?
- Was versteht man unter dem COW-Verfahren und warum wird dieses Verfahren angewendet?
- Was versteht man unter Wear-Leveling und warum wird es angewendet? Welche Nachteile ergeben sich daraus?

1) NOR und NAND Speicher, sowie SLC und MLC

NOR Speicher sind robust, schreibzugriff ist allerdings nur blockweise möglich, außerdem sind NOR-Speicherzellen eher teurer

NAND Speicher ist optimiert auf sequenzielle Zugriffe

2)	SLC: 1 Speicherzelle speichert 1 Bit
	MLC: 1 Speicherzelle kann unterschiedliche Spannungswerte speichern, daher mehrere Bit (meist 2 Bit = 4 Werte)

3) Copy on Write bezeichnet ein Verfahren, wobei bei einem Schreibzugriff der ganze Block, welcher beschrieben werden sollte, an eine neue Stelle kopiert wird und die änderung nur in der kopie gespeichert wird. Dies ist notwendig, da ein flashspeicher nur blockweise beschrieben werden kann, nachdem ein block gelöscht wurde. Möchte man also nur ein bit ändern, muss man den ganzen block zwischenspeichern, löschen und mit der änderung neubeschreiben. Durch COW erspart man sich das löschen und zwischenspeichern, indem man die änderung gleich an einer anderen Stelle schreibt. Hier muss nur noch gelöscht werden, wenn kein platz im speicher mehr frei ist.

4) Da Flash Speicherzellen nur einen begrenzte Lebensdauer in Schreibzylklen haben, zählt bei Wear Leveling der Flash Controller mit, wie oft in einen Block geschrieben wird, um die Daten so zu belegen, dass die Schreibzugriffe idealerweise gleichverteilt sind. Wird bei einem Block das limit erreicht, so wird dieser nicht mehr verwendet und stattdessen ein ersatzblock verwendet
Ein nachteil daraus ist einer der Datensicherheit: Da die verbrauchten blöcke nie wieder beschrieben werden können, könnten eventuelle vertrauliche daten von unbefugten noch ausgelesen werden. (Hierfür müsste man lediglich den Controller austauschen/reprogrammieren)

Aufgabe 3: Caching - Zusatzinformationen

Gehen Sie von einem direkt abgebildeten Cache mit einer Größe von 512 KiB und 32 Byte großen Blöcken aus. Es werden 32 Bit große Adressen verwendet. Beantworten Sie folgende Fragen:

a) Wie groß ist der **gesamte** Speicherbedarf (also Daten + alle zusätzlich benötigten Verwaltungsbits) für den Cache, wenn Durchschreibetechnik verwendet wird?

b) Wie verändert sich der gesamte Speicherbedarf, wenn der Cache die Rückschreibetechnik bei Schreiboperationen verwendet? Welche Vor- und Nachteile ergeben sich bei der Rückschreibetechnik?

c) Ändert sich der gesamte Speicherbedarf für den oben genannten Cache mit Durchschreibetechnik, wenn der Cache vierfach satzassoziativ realisiert wird? Wenn ja, wie?

a) 32 Byte Blöcke: $\Rightarrow$ 5 Bit Offset

$$\frac{512\ \text{KiB}}{32\ \text{B}} = \frac{2^9}{2^5} \cdot 2^{10} = 2^{14}\ \text{Blöcke} \Rightarrow 14\ \text{Bit Index}$$

$$\Rightarrow 32 - 14 - 5 = 11\ \text{Bit Tag}$$

1 Block: 11 Bit Tag
1 Bit Valid
32B = 256 Bit Daten

$$268\ \text{Bit/Block} \times 2^{14} = 268 \times 2^{11}\ \text{Byte} = \underline{536\ \text{KiB}}$$

b) 1 Block: 11 Bit Tag
1 Bit Valid
1 Bit Dirty
256 Bit Daten

$$269\ \text{Bit/Block} \times 2^{14} = 269 \cdot 2\ \text{KiB} = \underline{538\ \text{KiB}}$$

Schnellere Schreibzugriffe
Langsamere Zugriffe, wenn block ausgetauscht werden muss
Speicher sind nicht immer syncron

c)

Nein, der Speicher bleibt gleich. Er wird nur umsortiert, sodass nur mehr ein viertel der Adressen notwendig sind, jede Adresse allerdings 4 Blöcke (inkl Tags und Valid Bits) paralell zurückgibt.

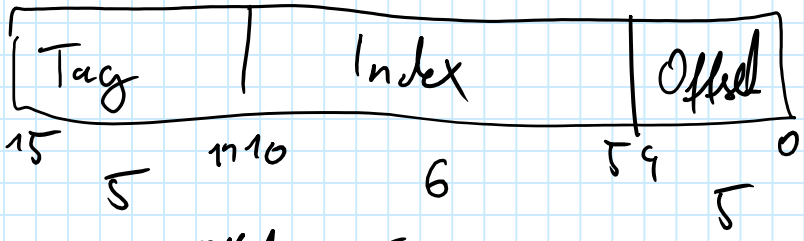
Aufgabe 4: Caching - Zugriffe

Für einen direkt abgebildeten Cache-Entwurf mit 16-Bit-Adressen werden die folgenden Bits einer Adresse für den Cache-Zugriff verwendet: Tag (Bit 15-11), Index (Bit 10-5), Offset (Bit 4-0). Beantworten Sie folgende Fragen:

a) Wie groß ist ein Cache-Block in Bytes? Wie viele Blöcke hat der Cache?

b) Es treten Zugriffe auf folgende Blöcke (nicht Byteadressen!) auf:
1, 1, 2, 18, 4, 83, 1, 142, 132, 4, 132, 2
Geben Sie für jeden Zugriff an, ob es sich um einen Hit oder um einen Miss handelt!

c) Wie hoch ist die Trefferrate für die obige Sequenz?



a) Blockgröße: $2^{\text{Offset}} = 2^5 = 32$ Byte

#Blöcke: $2^{\text{Index}} = 2^6 = 64$ Blöcke

b)

Block	Index	
1	1	Miss
1	1	Hit
2	2	Miss
18	18	Miss
4	4	Miss
83	19	Miss
1	1	Hit
142	14	Miss
132	4	Miss
4	4	Miss
132	4	Miss
2	2	Hit

c) $= \frac{3}{12} = \frac{1}{4} = \underline{0{,}25}$

Aufgabe 5: Caching - Ersetzungsalgorithmen

Gehen Sie von einem vollassoziativen Cache mit der Größe von 128 Byte und 32 Byte großen Blöcken aus. Als Ersetzungsalgorithmus wird LRU verwendet. Die Cache-Blöcke werden der Reihe nach befüllt. Beantworten Sie folgende Fragen:

Hinweis: Die Zugriffe sind hier auf Byteadressen gegeben. Der Cache arbeitet aber immer mit Blöcken (in diesem Fall 32 Byte groß) und auch der Hauptspeicher wird als eine Sequenz solcher Blöcke gesehen, die dann als Ganzes in den Cache transferiert werden.

a) Sie haben folgende Sequenz von Zugriffen (Byteadressen mit 16 Bits) auf den Hauptspeicher gegeben: 0x0010, 0x0020, 0x0040, 0x0022, 0x0080, 0x0068, 0x0015, 0x0023, 0x0094, 0x0100
Wie hoch ist die Trefferrate für diese Sequenz?

b) Sie haben folgende Sequenz von Zugriffen (Byteadressen mit 16 Bits) auf den Hauptspeicher gegeben: 0x0000, 0x0020, 0x0040, 0x0060, 0x0080, 0x0000, 0x0020, 0x0040, 0x060, 0x0080
Wie groß ist die Trefferrate für diese Sequenz?

32 Byte → 5 Bit Offset

$\frac{128 \text{ Byte}}{32 \text{ Byte}} = 4$ Blöcke

a)

Adresse	Adresse	Block					
0x0010	**0000 0000 000**1 0000	0	Miss	**d[0]**			
0x0020	**0000 0000 001**0 0000	1	Miss	d[0]	**d[1]**		
0x0040	**0000 0000 010**0 0000	2	Miss	d[0]	d[1]	**d[2]**	
0x0022	**0000 0000 001**0 0010	1	Hit	d[0]	**d[1]**	d[2]	
0x0080	**0000 0000 100**0 0000	4	Miss	d[0]	d[1]	d[2]	**d[4]**
0x0068	**0000 0000 011**0 1000	3	Miss	**d[3]**	d[1]	d[2]	d[4]
0x0015	**0000 0000 000**1 0101	0	Miss	d[3]	d[1]	**d[0]**	d[4]
0x0023	**0000 0000 001**0 0011	1	Hit	d[3]	**d[1]**	d[0]	d[4]
0x0094	**0000 0000 100**1 0100	4	Hit	d[3]	d[1]	d[0]	**d[4]**
0x0100	**0000 0001 000**0 0000	8	Miss	**d[8]**	d[1]	d[0]	d[03]

$\frac{3}{10} = 0{,}3$ Trefferrate

b) Sie haben folgende Sequenz von Zugriffen (Byteadressen mit 16 Bits) auf den Hauptspeicher gegeben:
0x0000, 0x0020, 0x0040, 0x0060, 0x0080, 0x0000, 0x0020, 0x0040, 0x060, 0x0080
Wie groß ist die Trefferrate für diese Sequenz?

b)

Adresse	Adresse	Block					
0x0000	**0000 0000 000**0 0000	0	Miss	**d[0]**			
0x0020	**0000 0000 001**0 0000	1	Miss	d[0]	**d[1]**		
0x0040	**0000 0000 010**0 0000	2	Miss	d[0]	d[1]	**d[2]**	
0x0060	**0000 0000 011**0 0000	3	Miss	d[0]	d[1]	d[2]	**d[3]**
0x0080	**0000 0000 100**0 0000	4	Miss	**d[4]**	d[1]	d[2]	d[3]
0x0000	**0000 0000 000**0 0000	0	Miss	d[4]	**d[0]**	d[2]	d[3]
0x0020	**0000 0000 001**0 0000	1	Miss	d[4]	d[0]	**d[1]**	d[3]
0x0040	**0000 0000 010**0 0000	2	Miss	d[4]	d[0]	d[1]	**d[2]**
0x0060	**0000 0000 011**0 0000	3	Miss	**d[3]**	d[0]	d[1]	d[2]
0x0080	**0000 0000 100**0 0000	4	Miss	d[3]	**d[4]**	d[1]	d[2]

$\frac{0}{20} = 0$ Trefferrate

Aufgabe 1: Virtueller Speicher - Allgemein

Sie haben ein System mit virtuellem Speicher gegeben, bei dem insgesamt 2^{40} Bytes adressiert werden können. Auf ihrem System stehen 4 GiB physischer Hauptspeicher zur Verfügung. Der Festplattenspeicher ist in ausreichendem Maße vorhanden, d.h. Sie brauchen bei den Berechnungen von keinen Einschränkungen ausgehen. Virtuelle und physische Seiten sind 16 KiB groß. Beantworten Sie folgende Fragen:

a) Wie viele Bits hat eine physische Adresse?

b) Wie viele virtuelle Seiten gibt es in diesem System?

c) Wie viele physische Seiten gibt es in diesem System?

d) Wie viele Bits werden für die Nummer einer virtuellen Seite verwendet?

e) Wie viele Einträge hat die Seitentabelle?

f) Zusätzlich zur Seitennummer enthält ein Seitentabelleneintrag noch ein Valid-Bit und ein Dirty-Bit. Wie groß ist daher die Seitentabelle in Bytes? Gehen Sie davon aus, dass ein Eintrag immer eine ganze Zahl von Bytes belegt.

a) $4\,GiB = 2^2 \cdot 2^{30}\,B = 2^{32}\,B \Rightarrow$ 32-Bit Adresse

b) $1\ \text{Seite} = 16\,KiB = 2^4 \cdot 10^{10}\,B = 2^{14}\,B$

$$\frac{2^{40}\,B}{2^{14}\,B/\text{Seite}} = \frac{2^{40}}{2^{14}}\ \text{Seiten} = 2^{40-14}\ \text{Seiten} = 2^{26}\ \text{Seiten}$$

c) $$\frac{2^{32}\,B}{2^{14}\,B/\text{Seite}} = 2^{32-14}\ \text{Seiten} = 2^{18}\ \text{Seiten}$$

d) 26 Bit

e) 2^{26}

f) $\cancel{26}$ 18 Bit Seitennummer → Nur physische Seiten

\+ 2 Bit Valid/Dirty

$\cancel{28}$ Bit $\leq$ 4 Byte $\times\ 2^{26} = 2^{28}$ Byte

20 Bit $\leq$ 3 Byte $\times\ 2^{26} = 3 \cdot 2^{26}$ Byte

Aufgabe 2: Virtueller Speicher - Theorie

Für diese Fragen müssen Sie zunächst ein paar Seiten aus folgendem Buch lesen:

`Roland Hellmann, Rechnerarchitektur : Einführung in den Aufbau moderner Computer, 2. Auflage, De Gruyter Oldenbourg, 2016.`

Sie können innerhalb der TU (oder mit VPN von außen) über den TU CatalogPlus (catalogplus.tuwien.ac.at/) auf die Online-Version des Buches zugreifen. Lesen Sie in Teil 5 (Speicher und Peripherie) die Seiten 285 - 289. Beantworten Sie dann bitte folgende Fragen:

- Was versteht man unter einem Programm im Zusammenhang mit der Speicherverwaltung?
- Was versteht man unter einem Prozess?
- Was bedeutet Multitasking?
- Was versteht man unter einem Kontextwechsel?
- Welche Aufgaben hat die MMU?
- Was versteht man unter einer Speicherzugriffsverletzung?
- Was versteht man unter einem TLB?

1) Program: Abfolge an Anweisungen, kann mehrmals (auch gleichzeitig) instanziert werden.
2) Prozess: Instanz eines Programms (also eine Ausführung des Programms vom Prozessor)
3) Multitasking: Scheinbar gleichzeitiges Ausführen von Prozessen durch oft hin und herwechseln.
4) Kontextwechsel: Wechsel zwischen zwei Prozessen, wobei der Prozesskontext (insbesondere die Register) des einen Prozesses abgeschpeichert und des anderen Prozess geladen wird.
5) Memory Management Unit: Übersetzung von logischen Adressen in Physikalische adressen und bereitstellen von ausgelagertem Speicher im Hauptspeicher.
6) Wenn ein Programm versucht auf eine logische Adersse außerhalb seines Adressraumes zuzugreifen, kann diese nicht Übersetzt werden und es entsteht eine Speicherzugriffverletzung. (Und das Programm stürzt meist komplett ab.)
7) Translation Look Aside Buffer: Eine Art Cache für Adressumsetzungen

Aufgabe 3: Architekturen - Theorie

Beantworten Sie bitte folgende Fragen:

- Beschreiben Sie kurz die Grundkomponenten eines Von-Neumann-Rechners.
- In der Vorlesung wurde auch eine andere Architektur (Harvard-Architektur) genannt. Wie unterscheidet sich diese von der Von-Neumann-Architektur?
- Wo findet man in heutigen Rechnern beide Architekturansätze? Erklären Sie kurz Ihre Antwort!

1) Von-Neumann: ein Prozessor bestehend aus Steuerwerk und ALU ist mit einem Busystem mit Arbeitsspeicher und IO verbunden. Der Arbeitsspeicher beinhaltet hierbei sowohl Daten als auch Programme.

2) In der Harvard-Achietektur werden separate Speicher für Programme und Daten verwendet. Anstatt eines Bussystem ist hier die Controllunit direkt mit ALU, Instruction Memory, Data Memory und IO verbunden.

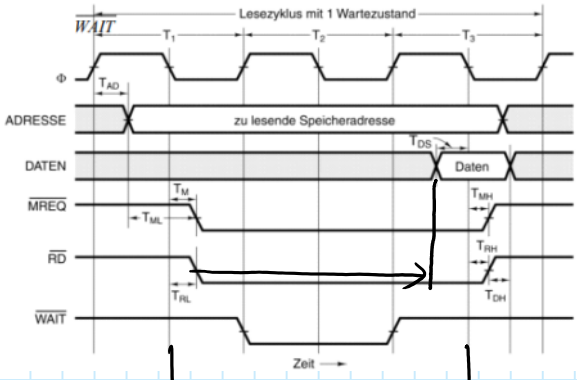
3) Heutige Prozessoren sind von außen betrachtet ein Von-Neumann-Prozessor, welcher Daten und Programme im selben Arbeitsspeicher ablegt. Im Prozessor selber ist die unterste Cache ebene jedoch nach Daten und Instruktionene getrennt, was einer Harvard-Architektur entspricht.

Aufgabe 4: Synchroner Bus

Gehen Sie von dem synchronen Bus auf Folie 19 (Ein-/Ausgabe) aus.

- Wie lange dauert ein Taktzyklus?
- Wie lange dauert das Zeitintervall von einer steigenden zu einer fallenden Flanke?
  Hinweis: In diesem Fall werden die Referenzpunkte in der Mitte der Flanke benutzt.
- Wie schnell muss ein Speicher seine Daten zur Verfügung stellen?
  Hinweis: Achten Sie darauf, ab wann ein Speicher anfangen kann, die Daten aus den Speicherzellen zu holen.
- Gehen Sie davon aus, dass wir einen Speicherchip zur Verfügung haben, der die Daten innerhalb von 10 ns zur Verfügung stellen kann. Warum könnte in diesem Fall ein asynchroner Bus von Vorteil sein?

Symbol	Parameter	Min [in ns]	Max [in ns]
T_{AD}	Adressausgabeverzögerung		4
T_{ML}	Adresse ist vor $\overline{MREQ}$ stabil	2	
T_M	$\overline{MREQ}$-Verzögerung von der fallenden Flanke von Φ in T_1		3
T_{RL}	RD-Verzögerung von der fallenden Flanke von Φ in T_1		3
T_{DS}	Dateneinrichtezeit vor der fallenden Flanke von Φ	2	
T_{MH}	$\overline{MREQ}$-Verzögerung von der fallenden Flanke von Φ in T_3		3
T_{RH}	$\overline{RD}$-Verzögerung von der fallenden Flanke von Φ in T_3		3
T_{DH}	Datenhaltezeit nach der Deaktivierung von $\overline{RD}$	0	



1) 100 MHz Takt = $100 \cdot 10^6$ Hz = 10^8 Hz

$\frac{1}{10^8 \text{Hz}} = 10^{-8}$ s $= 10 \cdot 10^{-9}$ s = 10 ns Taktzyklus

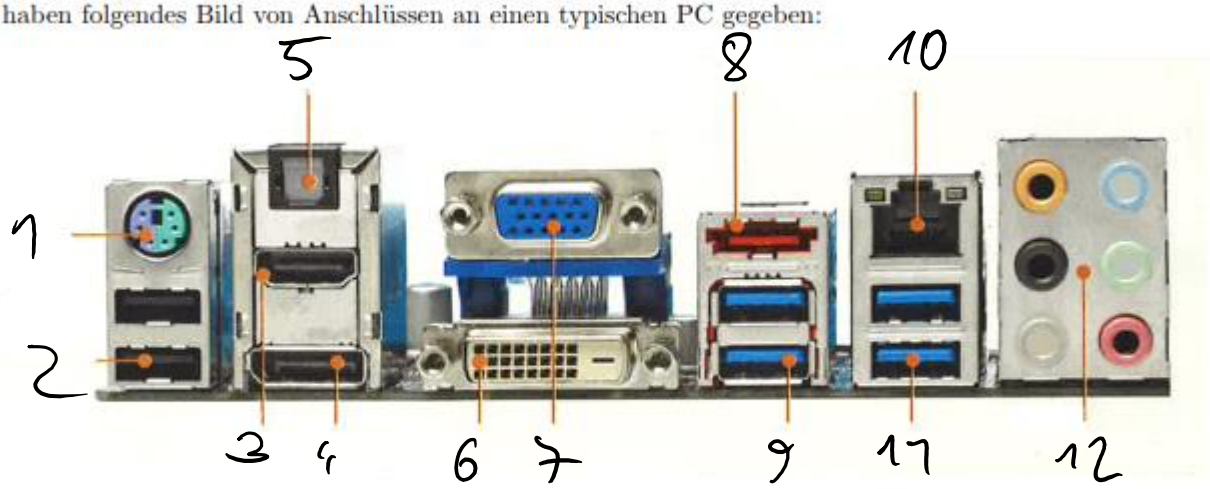
2) 5 ns

3) $20\text{ns} - T_{RL} - T_{DS} = 20\text{ns} - 3\text{ns} - 2\text{ns} = 15\text{ns}$

4) Bei einem Syncronen Takt wird immmer 15ns gewartet, auch wenn der Speicher die Daten bereits in 10ns zur verfügung stellt. Bei einem Asyncronen takt hingegen, kann der Prozessor nach diesen 10ns bereits mit den Daten weiterarbeiten, da nicht auf ablauf des Taktes gewartet werden muss.

Aufgabe 5: Anschlüsse

Sie haben folgendes Bild von Anschlüssen an einen typischen PC gegeben:



Beschriften Sie zumindest 8 verschiedene Anschlüsse und erklären Sie in ein bis zwei Sätzen den Sinn und Zweck dieser Anschlüsse.

1) PS/2 Schnittstelle: Anschluss von Tastatur oder Maus, heute kaum mehr verwendet, hat aufgrund der einfachheit der Übertragung aber den Vorteil, dass es garantiert auch im BIOS funktioniert

2) USB 2.0: Universial Serial Bus, überträgt beliebige daten mit bis zu 480 Mbit/s, nahezu unbegrenzte vielfalt an Geräten.

3) HDMI: High Definition Media Interface: Zum Anschluss von modernen Digitalmonitoren
4) Displayport: Ebenfalls für moderne Digitalmonitore

5) S/PDIF-TOSLINK: Digitaler Optischer Audioanschluss: Audiodaten werden z.b. an eine HiFi Anlage über Lichtwellenleiter übertragen

6) DVI-D (Dual-Link): Digitaler Displayanschluss (eher outdated, nicht so hochauflösend wie HDMI/Displayport)

7) VGA: Analoger Displayanschluss: Schlechte bildqualität, dafür einfacher anzusteuern, hilfreich um z.b. einen PC ohne Grafiktreiber zu verwenden

8) eSATA: external SATA: nach außen geführter SATA-Anschluss für externe Festplatten

9/11) USB 3.?: wie USB 2.0 mit mehr leitungen zur schnelleren Datenübertragung, häufig genutzt für externe Festplatten und USB-Sticks

10) Ethernet: Netzwerkanschluss ermöglicht die Verbindung mit einem Netzwerk / dem Internet

12) Analoges 3,5mm Klinkenstecker Audio: ermöglicht das direkte anschließen von Lautsprechern

Aufgabe 1: Netzwerke - Grundlagen

Zwei Hosts A und B seien über eine Leitung mit R bps verbunden. Gehen Sie davon aus, dass die Hosts m Meter voneinander entfernt sind und dass die Ausbreitungsgeschwindigkeit eines Signals über diese Leitung s Meter/Sekunden beträgt. Host A möchte ein Paket mit der Größe L zu Host B schicken.

a) Geben Sie die Ausbreitungsverzögerung d_{prop} mit Hilfe von m und s an!

b) Bestimmen Sie die Übertragungszeit für ein Paket d_{trans} mit Hilfe von L und R!

c) Geben Sie eine allgemeine Formel für die Ende-zu-Ende-Verzögerung zwischen diesen beiden Hosts an (ohne Verarbeitungsverzögerung und Wartezeit in den Puffern)!

d) Wenn Host A zum Zeitpunkt $t = 0$ mit dem Versenden des Pakets beginnt, wo befindet sich dann das letzte Bit des Pakets zum Zeitpunkt $t = d_{trans}$?

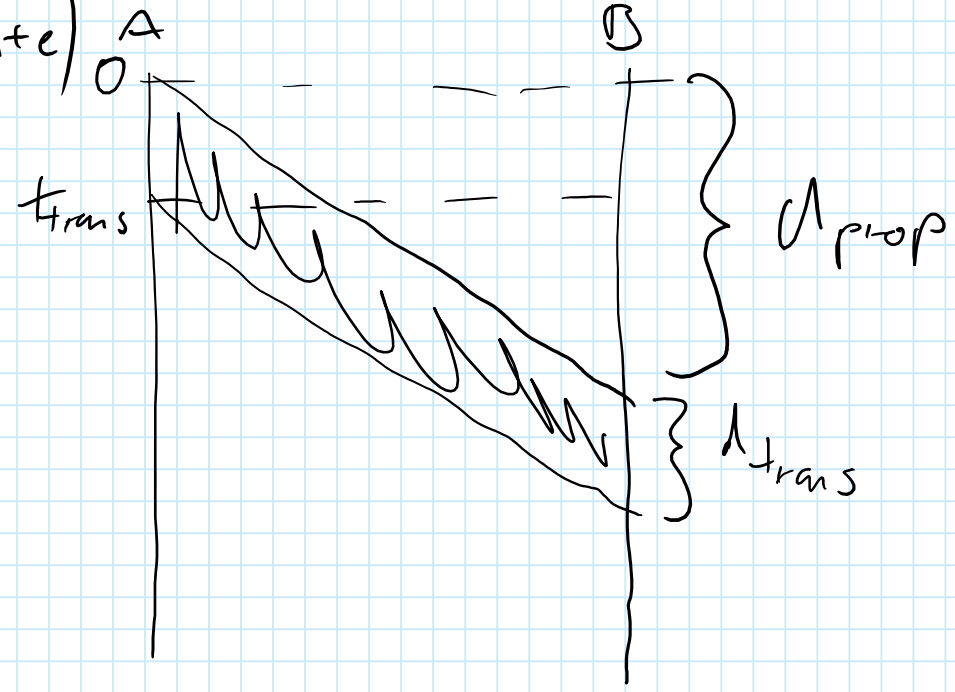
e) Nehmen Sie nun $d_{prop} > d_{trans}$ an. Wo befindet sich das erste Bit des Pakets zum Zeitpunkt $t = d_{trans}$ (Host A beginnt wieder mit dem Senden bei $t = 0$)?

a) $d_{prop} = \frac{m}{s}$

b) $d_{trans} = \frac{L}{R}$

c) $d_{ges} = d_{prop} + d_{trans} = \frac{m}{s} + \frac{L}{R}$

d+e)



d) Zum Zeitpunkt t_trans ist das letzte Bit gerade gesendet worden

e) Zum Zeitpunkt t_trans ist das erste Bit gerade in der Leitung, wenn d_prop>d_trans

Aufgabe 2: Netzwerke - Grundlagen

Gehen Sie davon aus, dass zwei Rechner A und B 100 Kilometer voneinander entfernt und über eine einzige Leitung mit R=4 Mbps verbunden sind. Die Ausbreitungsgeschwindigkeit für das Signal beträgt $2.0 * 10^8$ Meter/Sekunde.

a) Berechnen Sie das Produkt $R \times d_{prop}$ (Verzögerung-Bandbreite-Produkt).

b) Sie wollen 8000 Bits als eine einzige große Nachricht über diese Verbindung schicken. Was ist die maximale Anzahl an Bits, die sich zu einem Zeitpunkt auf der Leitung befinden kann?

c) Nehmen Sie nun an, dass die Nachricht in 10 Pakete (jeweils 800 Bit) aufgeteilt wird. Wird ein Paket geschickt, dann wartet der Sender bis er vom Empfänger eine Bestätigung erhält und schickt erst dann ein weiteres Paket. Wie lange dauert es, bis man die gesamte Nachricht übertragen hat (in ms)? Sie können für diese Aufgabe die Größe der Bestätigungen und eventuelle Header bei der Berechnung vernachlässigen!

a) $R \times d_{prop} = R \cdot \frac{d}{s} = 4\,\text{Mb/s} \cdot \frac{100\,\text{km}}{2 \cdot 10^8\,\text{m/s}}$

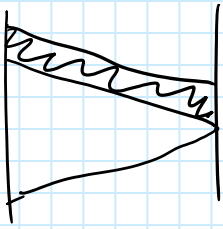
$= 4\,\text{Mb} \cdot \frac{100 \cdot 10^3}{2 \cdot 10^8}$

$= \frac{4}{2} \cdot \frac{10^5}{10^8} \cdot 10^6$ bit

$= 2 \cdot 10^3$ bit

$= 2000$ bit

b) $\underline{2000 \text{ bit}}$



c) $10 \cdot (2 d_{prop} + d_{trans})$

$= 10 \cdot \left(2 \cdot \frac{100\,\text{km}}{2 \cdot 10^8\,\text{m/s}} + \frac{800\,\text{bit}}{4\,\text{Mb/s}}\right)$

$= 10 \cdot \left(2 \cdot \frac{10^5}{2 \cdot 10^8}\,\text{s} + \frac{8 \cdot 10^2}{4 \cdot 10^6}\,\text{s}\right)$

$= 10 \cdot \left(10^{-3} + 2 \cdot 10^{-4}\right)\text{s}$

$= (10+2) \cdot 10^{-3}\,\text{s} = \underline{\underline{12\,\text{ms}}}$

Aufgabe 3: Netzwerke - Header

Auf der Anwendungsschicht soll eine Nachricht mit 7140 Byte Nutzdaten zuverlässig übertragen werden. Es gelten die bekannten Header-Größen für TCP und IP. Zusätzlich wird angenommen, dass der Header der Sicherungsschicht eine Größe von 20 Bytes hat.
Für diese Aufgabe gehen wir davon aus, dass die Transportschicht die Größe eines übertragenen Segments (TCP-Header + TCP-Nutzdaten) auf maximal 1400 Byte beschränkt. Alle darunterliegenden Schichten unterliegen keiner weiteren Beschränkung.

a) In wie viele Einzelpakete muss die Transportschicht die ursprüngliche Nachricht aufteilen?

b) Wie viel Prozent Overhead (= Σ Header / Bytegröße der Gesamtnachricht) fällt bei der Übertragung insgesamt an?

c) Nehmen Sie an, Sie können die Pakete im Netzwerk mit 20 Mbit/s übertragen. Wie lange dauert die Übertragung?

a) $\text{TCP-Header} + \frac{\text{Data}}{n} \leq 1400 \text{ Byte}$

$\frac{\text{Data}}{n} \leq 1400\,\text{B} - \text{TCPH}$

$\frac{\text{Data}}{1400\,\text{B} - \text{TCPH}} \leq n$

$\frac{7140}{1400-20} = n \Rightarrow \frac{7140}{1380} \approx 5{,}1 \leq \underline{\underline{6}}$ min. 6 Pakete

b) $\frac{(3 \cdot 20\,\text{B}) \cdot 6}{7140\,\text{B}} = \frac{360}{7140} = 5\%$

c) $\frac{7140\,\text{B} + 360\,\text{B}}{20\,\text{Mbit/s}} = \frac{7500 \cdot 8\,\text{bit}}{20 \cdot 10^6\,\text{bit/s}} = \frac{60000}{20} \cdot 10^{-6}\,\text{s}$

$= 3 \cdot 10^3 \cdot 10^{-6}\,\text{s} = 3 \cdot 10^{-3} = \underline{3\,\text{ms}}$

Aufgabe 4: Netzwerke - DNS

Für diese Aufgabe lesen Sie zunächst den Wikipedia-Beitrag zu DNS `https://de.wikipedia.org/wiki/Domain_Name_System` und beantworten dann folgende Fragen:

- Welche Aufgabe erfüllt DNS?
- Welche Vorteile bietet die Verwendung von DNS?
- Auf welcher Protokollschicht ist DNS angesiedelt?
- Wie ist der Domain-Namensraum strukturiert?
- Welche zwei Arten von Nameservern gibt es?
- Was versteht man unter einer rekursiven bzw. iterativen Namensauflösung?

1) Namensauflösung von Domains zu IP Adressen
2) Domainnamen sind einfacher zu merken als IPs; wenn sich die IP eines Service ändert, kann man einfach die Domain auf die neue IP verweisen lassen.
3) DNS ist auf der Anwendungsschicht und benutzt meist UDP als Transportschicht
4) Der Domainnamensraum ist Baumförmig Strukturiert, wobei die Domainnamen von links nach rechts gelesen werden.
   d.h. auf der Obersten Ebene exisitert nur ein Baumknoten. Darunter befinden sich alle First-Level-Domains wie z.b. com, org oder at.
   Darunter wiederum befinden sich die second level domains, z.b. google oder wikipedia.
   Darunter kann es noch weitere Ebenen geben, welche allerdings optional sind.
5) Autorative Nameserver sind für eine Zone verantwortlich und sind im direkten Besitz von gesicherter Information
   Nicht-autorative Nameserver beziehen ihre Informationen von anderen Nameservern und speichern diese zwischen.
6) Möchte man eine Domain mit mehreren Ebenen auflösen, gibt es zwei Möglichkeiten: Bei der rekursiven Auflösung stellt man einen Request an den Root Server, dieser fragt bei dem zuständigen Server der nächste ebene an und so weiter. Die aufgelöste Domain wird dann wieder durch alle Ebenen zurückgereicht.
   Bei der iterativen Auflösung antwortet der Root Server mit einem Verweis auf den zuständigen Server der nächsten Ebene. Nun muss der Client den Request nochmal an diesen Server senden (der ihn evtl wieder weiterverweisen könnte)

Aufgabe 5: Netzwerke - Subnetze

Ein ISP besitzt den Adressblock 180.120.70.0/23.

a) Wie viele IP Adressen stehen in diesem Netzwerk zur Verfügung?

b) Wie lautet die Netzwerk- bzw. die Broadcastadresse des Netzwerks?

c) Wie lautet die Netzwerkmaske für dieses Netzwerk?

Dieser Provider will 16 Subnetze einrichten, mit jeweils gleich vielen IP Adressen.

d) Wie lautet der Prefix des ersten Subnetzes (im Format: a.b.c.d/x)?

e) Wie lautet die Netzwerk- bzw. die Broadcastadresse des zweiten Subnetzes?

f) In welchem der 16 Subnetze befindet sich die IP 180.120.71.135?

5 ✓

a) netid: 23 bit
hostid: 9 bit $\Rightarrow 2^9 = \cancel{512}$ Adressen

b) 180.120.70.0 ← netzwerkadresse ✓
180.120.71.255 ← Broadcastadresse

c) 255.255.~~127~~.0
254

d) 180.120.70.0/27 ✓

e) 180.120.70.32 ← netzwerkadresse ✓
180.120.70.63 ← Broadcastadresse

f) $(1;135)_{256} = (\underbrace{10000}_{8}0.0111)_2$ (Nullindizient)

Subnetz 9