Technische Informatik: Übungsblatt 01

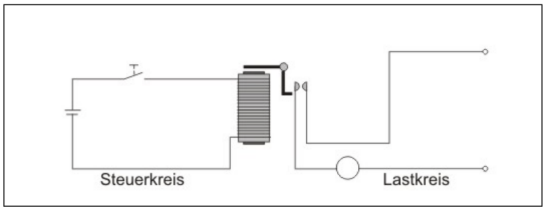
Benzschawel, Tittebrandt, Well 4. Dezember 2024

1. In der Vorlesung wurde die sogenannte **Von-Neumann-Architektur** vorgestellt. Erläutern Sie den so genannten **Von-Neumann-Flaschenhals**.

2. Welche alternative Architektur vermeidet den Von-Neumann-Flaschenhals? Skizzieren Sie diese alternative Architektur.

3. Erläutern Sie die Unterschiede zwischen einer L/S-Architektur, einer R/M-Architektur und einer R+M-Architekur.

4. In der Vorlesung wurde die Funktionsweise eines Relais erläutert. Wird der Schalter im Steuerkreis geschlossen, dann wird der Elektromagnet aktiviert und schaltet mechanisch den Schalter des Lastkreis.



   (a) Modifizieren Sie die Zeichnung bitte so, dass eine logische **UND**-Schaltung realisiert ist. Sie benötigen dazu zwei Schalter im Steuerkreis. Denken Sie z. B. an eine Heckenschere oder eine Stanzmaschine, bei denen der Benutzer zwei Schalter drücken muss um die Maschine zu bedienen.

   (b) Modifizieren Sie die Zeichnung bitte so, dass eine logische **ODER**-Schaltung entsteht. Sie benötigen wieder Schalter im Steuerkreis. Denken Sie an den Notrufknopf von Patienten im Krankenhaus. Die rote Lampe im Schwesternzimmer (Lastkreis) erleuchtet wenn einer der Patienten den Schalter drückt. (Was würde hier eine UND-Schaltung bedeuten?)

   (c) Modifizieren Sie die Zeichnung bitte so, dass eine logische **NOT**-Schaltung entsteht. Sie brauchen nur einen Schalter im Steuerkreis. Solange der Schalter geschaltet ist, bleibt die Maschine im Lastkreis aus. Wird der Schalter im Steuerkreis geöffnet, läuft die Maschine. Denken Sie z. B. an die Luftdruckbremse beim LKW. Solange Druck auf dem Kessel ist, bleibt die Bremsanlage untätig (Schalter im Steuerkreis geschlossen) und der LKW kann bewegt werden. Ohne Druck ist der Schalter offen und die Bremsanlage (Lastkreis) schaltet. Durch Betätigen des Bremspedals oder durch einen Defekt an der Luftdruckanlage würde der Schalter im Steuerkreis geöffnet.

(d) Jetzt bauen Sie noch bitte eine Schaltung mit zwei Schaltern A und B im Steuerkreis, so dass die Maschine im Lastkreis läuft, solange weder A noch B geschlossen sind. Wird einer der beiden, oder werden beide geschlossen, wird der Lastkreis ausgeschaltet. Welche logische Funktion ist entstanden?

5. Erklären Sie mit eigenen Worten die Bedeutung der Komponenten einer CPU.

   (a) Steuerwerk

   (b) ALU

   (c) Adresswerk

6. Wozu werden die folgenden Komponenten einer CPU verwendet?

   (a) Befehlsregister

   (b) Programm-Counter (PC)

   (c) AC-Register

7. Es existieren verschiedene Architekturmodelle mit unterschiedlichen Adressformaten. Schreiben Sie für die nachfolgende Berechnung jeweils ein Programm unter Verwendung von 3-Adress-, 2-Adress- und 1-Adress-Instuktionen:

$$x = (a \cdot b \cdot c) - (d + e)$$

   Hinweis: Die Variablen a, b, c, d, e können Sie als Registerbezeichnung interpretieren, die Sie nutzen können, um das jeweilige Register anzusprechen. Möchten Sie für die Berechnung in dem Programm ein zusätzliches Hilfsregister verwenden, können Sie dies im Programm durch Neueinführung einer Variablenbezeichnung definieren.

zulässige 3-Adress-Befehle:

add a, b, c	Bedeutung:	a	$\leftarrow$	$b + c$
sub a, b, c	Bedeutung:	a	$\leftarrow$	$b - c$
mul a, b, c	Bedeutung:	a	$\leftarrow$	$b \cdot c$

zulässige 2-Adress-Befehle:

move a, b	Bedeutung:	a	$\leftarrow$	b
add a, b	Bedeutung:	a	$\leftarrow$	$a + b$
sub a, b	Bedeutung:	a	$\leftarrow$	$a - b$
mul a, b	Bedeutung:	a	$\leftarrow$	$a \cdot b$

zulässige 1-Adress-Befehle:

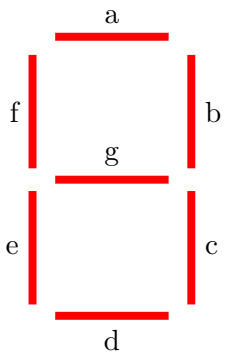
load a	Bedeutung:	ACC	$\leftarrow$	a
store a	Bedeutung:	a	$\leftarrow$	ACC
add a	Bedeutung:	ACC	$\leftarrow$	$ACC + a$
sub a	Bedeutung:	ACC	$\leftarrow$	$ACC - a$
mul a	Bedeutung:	ACC	$\leftarrow$	$ACC \cdot a$

Technische Informatik: Übungsblatt 02

Benzschawel, Tittebrandt, Well 4. Dezember 2024

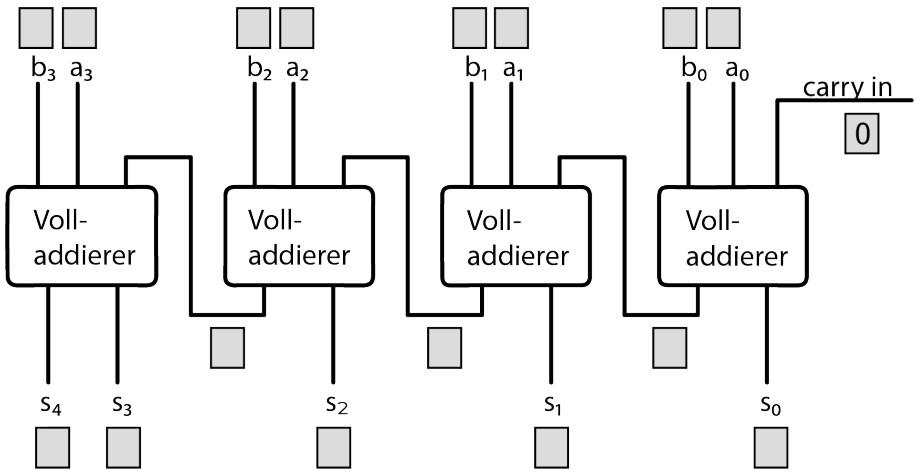
1. In der Wertetabelle fehlen die Werte für das Segment d. Ergänzen Sie diese. Welche Segmente müssen „on “ sein um das Zeichen für 5 darzustellen?

x_3	x_2	x_1	x_0	Zeichen	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1		1	1	0
0	0	0	1	1	0	1	1		0	0	0
0	0	1	0	2	1	1	0		1	0	1
0	0	1	1	3	1	1	1		0	0	1
0	1	0	0	4	0	1	1		0	1	1
0	1	0	1	5							
0	1	1	0	6	1	0	1		1	1	1
0	1	1	1	7	1	1	1		0	0	0
1	0	0	0	8	1	1	1		1	1	1
1	0	0	1	9	1	1	1		0	1	1



2. Informieren Sie sich über die **BCD Codierung**. Schauen Sie in der gegebenen Wertetabelle von Aufgabe 1 nach, bei welchen Eingangskonstellationen für x_3, x_2, x_1, x_0 das Segment e leuchten soll. Stellen Sie mit den digitalen Basis-Gattern eine Schaltung auf, deren Ausgang das Segment e ansteuert. (Ohne vorherige Optimierung, nur Gatter mit 1 oder 2 Eingägen verwenden.)

3. Spielen Sie die Addition der beiden Zahlen A $= 10_{10} = 1010_2$ und B $= 7_{10} = 0111_2$ mit dem 4 Bit Paralleladdierer durch. Schreiben Sie dazu die Bits aus der Rechnung jeweils an die Ein- und Ausgänge des Addierers.



4. (a) Wie viele Steuerleitungen benötigt ein Multiplexer für 14 Eingänge?

   (b) Zeichnen Sie das logische Diagramm für einen Multiplexer mit 1 Steuerleitung und 2 Eingängen. Verwenden Sie nur AND, OR und NOT Gatter. AND und OR Gatter dürfen beliebig viele Eingänge haben.
   Hinweis: Dann sollte ein NOT, zwei AND und ein OR ausreichen.

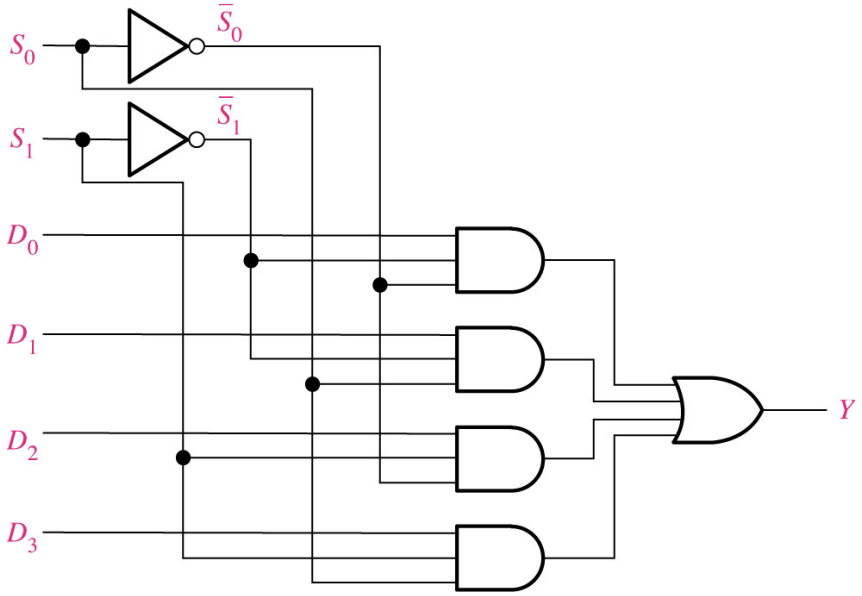
Technische Informatik: Zusatzblatt

Benzschawel, Tittebrandt, Well — 4. Dezember 2024

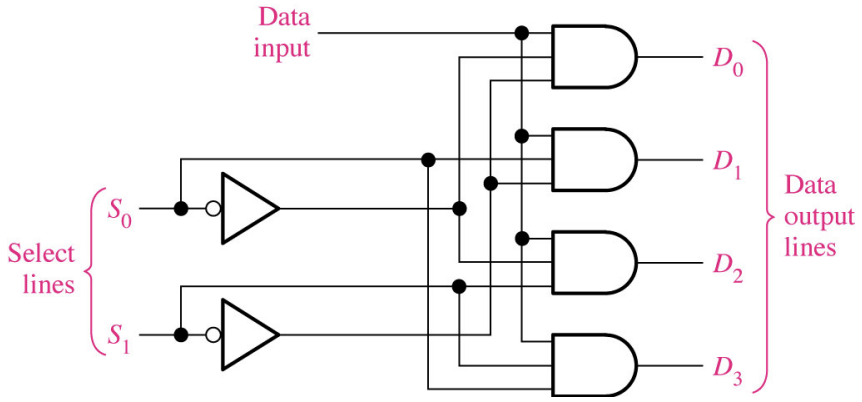
1. Unter `http://www.cburch.com/logisim/de/download.html` finden Sie das Programm Logisim. Machen Sie sich damit etwas vertraut. Bauen Sie mit Standardgattern folgende Bausteine auf:

   (a) Einen Multiplexer mit 2 Eingängen D_0 und D_1 und einer Steuerleitung S. Nutzen Sie dazu ein NOT, zwei AND und ein OR Gatter. Der Ausgang soll Y heißen.

   (b) Den aus aus der Vorlesung bekannten Multiplexer mit 2 Steuerleitungen.



   (c) Den aus der Vorlesung bekannten Demultiplexer mit 2 Steuerleitungen.

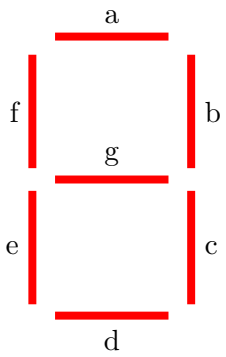


   (d) Erkunden Sie das Verhalten der Bausteine.

2. (a) Nutzen Sie aus den angebotenen Auswahlbausteinen einen fertigen MUX und einen fertigen DEMUX mit jeweils zwei Steuerleitungen. Erkunden Sie das Verhalten. Ach-

tung: Sie benötigen eine Enable-Leitung eine 2-Bit breite Eingabe für die Steuerungsbits.

(b) Unter den Ein-Ausgabe-Bausteinen finden Sie eine 7-Segmentanzeige. Verwenden Sie 7 Eingänge, die Sie von a bis g durchnummerieren und damit die Segmente so benennen wie in der Vorlesung.



(c) Bauen Sie die Schaltung zur Ansteuerung des Segments e nach. Hinweis: Übungsblatt 02, Aufgabe 2.

Technische Informatik: Übungsblatt 03

Benzschawel, Tittebrandt, Well 4. Dezember 2024

1. Welchen dezimalen Wert hat die hexadezimale Zahl $2AB,F$. Sie können die folgende Tabelle als Hilfe nutzen.

Wertigkeit	$16^2 = 256$	$16^1 = 16$	$16^0 = 1$	$16^{-1} = 0,0625$
Ziffer	2	A	B,	F

2. Bestimmen Sie den Wert der Zahlen:

0153	(octal)	=	001 101 011
0x153	(hex)	=	0001 0101 0011
153	(decimal)	=	1001 1001

   (a) Geben Sie die Werte in dezimaler Schreibweise an.

   (b) Welchen dezimalen Wert hat 153 in einem 6-er System?

   (c) Welchen dezimalen Wert hat 153 in einem 4-er System?

3. In C++ deklariert ein Programmierer gerne besonders strukturiert:

```
int y = 120;
int x = 012;
int z = 222;
```

   Sehen Sie ein Problem?

4. Konvertieren Sie die folgenden Dezimalzahlen nach BCD:

   (a) 35

   (b) 98

   (c) 170

   (d) 2469

   Konvertieren Sie folgende BCD Codes zu einer Dezimalzahl:

   (a) 10000110

   (b) 001101010001

   (c) 1001010001110000

5. Wandeln Sie die beiden Dezimalzahlen 65 und 77 in die binäre Darstellung um. Addieren Sie die Binärzahlen von Hand durch untereinander schreiben.

6. Wandeln Sie folgende Dezimalzahlen in Binärzahlen um indem Sie jeweils die Summe der 2-er Potenzen bilden.
   Beispiel:

$$61 = 32 + 16 + 8 + 4 + 1 = 2^5 + 2^4 + 2^3 + 2^2 + 2^0 \qquad = 111101$$
$$0,0981 \approx 0,0 + 0,0 + 0,0 + 0,0625 + 0,03125 + 0,0 + 0,0 + 0,00390625 \quad = 0,0001101$$

   (a) 10

   (b) 17

   (c) 24

   (d) 48

   (e) 93

   (f) 125

   (g) 186

   (h) 0,32

   (i) 0,246

   Üben Sie mit angemessen vielen Beispielen. Bei den beiden Kommazahlen sind 5 Nachkommastellen ausreichend.

7. Wandeln Sie die folgenden Dezimalzahlen in Binärzahlen um durch die Methode der wiederholten Division durch 2.

   (a) 15

   (b) 21

   (c) 28

   (d) 34

   (e) 40

8. Wandeln Sie die Zahl $(59)_{10}$ um in

   (a) eine Binärzahl.

   (b) eine Zahl im 4-er System.

   (c) eine Zahl im 8-er System.

   Nutzen Sie jeweils die Methode der wiederholten Division. Denken Sie an die elegante Umrechnung einer Oktalzahl in eine Hexadezimalzahl und wenden Sie dieses Wissen auf das obige Beispiel an.

9. Wandeln Sie die folgenden Dezimalzahlen in Binärzahlen um durch die Methode der wiederholten Multiplikation mit 2. Sechs binäre Nachkommastellen sind dabei ausreichend.

   (a) 0,98

(b) $0,347$

(c) $0,9028$

Wandeln sie die letzte Dezimalzahl auch in eine Oktalzahl um. Nutzen Sie dazu auch die Methode der wiederholten Multiplikation. Zwei oktale Nachkommastellen sind ausreichend.

10. Wandeln Sie die folgenden Hexadezimalzahlen in Binärzahlen um.

    (a) 38_{Hex}

    (b) 59_{Hex}

    (c) $A14_{\text{Hex}}$

    (d) $5C8_{\text{Hex}}$

    (e) 4100_{Hex}

    (f) $FB17_{\text{Hex}}$

    (g) $8A9D_{\text{Hex}}$

    Üben Sie mit angemessen vielen Beispielen.

11. Wandeln Sie die folgenden Hexadezimalzahlen in Dezimalzahlen um.

    (a) 23_{Hex}

    (b) 92_{Hex}

    (c) $1A_{\text{Hex}}$

    (d) $8D_{\text{Hex}}$

    (e) $F3_{\text{Hex}}$

    (f) EB_{Hex}

    (g) $5C2_{\text{Hex}}$

    (h) 700_{Hex}

12. Stellen Sie die Zahlen 10, 13, 18, 21, 25, 36, 44, 125, 156

    (a) in der BCD Form und daneben

    (b) als Binärzahl dar.

    (c) Für die beiden ersten und die beiden letzten Zahlen: Wie viele Bits benötigt die Darstellung als Binärzahl im Vergleich zur BCD Form?

13. (a) Jeder 7 Bit Block stellt ein ASCII Zeichen dar. Was bedeutet

    1010100 1110010 1101001 1100101 1010010?

    Die ASCII-Tabelle finden Sie im Anhang 1.

    (b) Nehmen Sie die ASCII Tabelle zur Hand und codieren Sie die 6 Zeichen „Hello.“ mit 7 Bit Blöcken.

(c) Zur Fehlererkennung bei der Übermittlung von „Hello.“ ergänzen Sie den ASCII Code der Zeichen am Ende um ein 8. Bit als **even parity bit**.

(d) Ergänzen Sie ebenso ein **even parity bit** bei Aufgabe (a).

Technische Informatik: Übungsblatt 04

Benzschawel, Tittebrandt, Well 4. Dezember 2024

1. (a) Auf welche beiden Arten kann die Zahl 0 im 1-er Komplement dargestellt werden?
   (b) Wie wird die 0 im 2-er Komplement dargestellt?

2. Bestimmen Sie das 1-er Komplement der nachfolgenden Binärzahlen:
   (a) 101
   (b) 110
   (c) 1010
   (d) 11010111
   (e) 1110101
   (f) 00001

3. Bestimmen Sie das 2-er Komplement der folgenden Binärzahlen:
   (a) 111
   (b) 1101
   (c) 10011
   (d) 00111101

4. Wandeln Sie jedes Paar von Dezimalzahlen in ein Paar aus Binärzahlen um und addieren beide Summanden mit Hilfe des 2-er Komplements:
   (a) 33 und 15
   (b) 56 und -27
   (c) -46 und 25
   (d) -110 und -84

5. Wandeln Sie die dezimale Zahl $3,248 \cdot 10^4$ in eine **single-precision binäre Fließkommazahl** um. Erläutern Sie die einzelnen Schritte.

6. Welchen dezimalen Wert hat die IEEE-754 Fließkommazahl?

0	01111110	10000000000000000000000

7. Welchen dezimalen Wert hat die IEEE-754 Fließkommazahl?

0	10000001	11000000000000000000000

8. Stellen Sie den dezimalen Wert als IEEE-754 Fließkommazahl dar.

   (a) $-0,75$

   (b) $0,375$

   (c) $98,7$

9. Die größte (normalisierte) IEEE-754 Zahl bei 32-Bit Darstellung ist

$$\left(1 - 2^{-24}\right) \cdot 2^{128}.$$

   Erläutern Sie bitte, wie Sie auf diesen Wert kommen. Achten Sie auf den Spezialfall mit dem Exponentenwert 255, also binär 11111111.

Technische Informatik: Übungsblatt 05

Benzschawel, Tittebrandt, Well 4. Dezember 2024

1. (a) Welchen dezimalen Wert hat die angegebene IEEE-754 Fließkommazahl?

1	10000010	10000000000000000000000

   (b) Der Wert dieser Zahl wird mit $(-16)_{10}$ multipliziert. Wie sieht die entstandene Zahl in IEEE-754 (32-Bit) Notation aus.

2. Die größte positive darstellbare Fließkommazahl in IEEE-754 (32-Bit) Notation ist:

0	11111110	11111111111111111111111

   (a) Der Exponent hat dabei den Wert 2^n. Denken Sie an den Bias-Wert. Wie groß ist n?

   (b) Bestimmen Sie den **dezimalen** Wert dieser Fließkommazahl. Als Ergebnis sind $(1 - 2^h) \cdot 2^k$ oder $(2^m - 2^n) \cdot 2^y$ erlaubt, wobei Sie h, k oder m, n, y durch die richtigen Werte ersetzen. Das Ergebnis muss nicht dezimal notiert werden.

3. Die kleinste darstellbare positive denormalisierte Fließkommazahl ($\neq 0$) in IEEE-754 (32-Bit) Notation ist:

0	00000000	00000000000000000000001

   (a) Wie groß ist der Wert des Exponenten als 2-er Potenz?

   (b) Wie lautet der (dezimale) Wert der Mantisse?

   (c) Weisen Sie nach, dass diese Fließkommazahl den Wert 2^{-149} hat.

4. Die kleinste positive normalisierte Fließkommazahl in IEEE-754 (32-Bit) Notation ist:

0	00000001	00000000000000000000000

   (a) Wie groß ist der Wert des Exponenten als 2-er Potenz?

   (b) Wie lautet der (dezimale) Wert der Mantisse?

   (c) Welchen Wert hat diese Fließkommazahl? Sie dürfen den Wert als 2-er Potenz notieren.

5. Die größte positive denormalisierte Fließkommazahl in IEEE-754 (32-Bit) Notation ist:

0	00000000	11111111111111111111111

   (a) Wie groß ist der Wert des Exponenten als 2-er Potenz?

(b) Die Mantisse hat dabei den dezimalen Wert $\left(2^{-1} + 2^{-2} + \ldots + 2^{-23}\right)$. Notieren Sie den Wert der Mantisse in der Form $(1 - 2^k)$, wobei Sie k angeben.

(c) Bestimmen Sie nun den dezimalen Wert der Fließkommazahl. Das Ergebnis sollte $(2^a - 2^b)$ lauten, wobei Sie die richtigen Werte für a und b einsetzen.

6. Konvertieren Sie die Dezimalzahl $1,2077 \cdot 10^4$ in eine binäre, single precision Fließkommazahl. Als Hilfestellung ist die Darstellung in einfacher Binärform bereits angegeben.

$$(12077)_{10} = (0010111100101101)_2$$

7. Die angegebene IEEE-754 Darstellung repräsentiert den dezimalen Wert $(20,00)_{10}$.

0	10000011	01000000000000000000000

Modifizieren Sie diese bitte so, dass der dezimale Wert durch $(-8)_{10}$ dividiert wird. Das geänderte Bitmuster soll also den neuen dezimalen Wert von $(-2,5)_{10}$ darstellen.

8. Wie groß ist die Differenz (als 2-er Potenz) zwischen der größten positiven **normalisierten** Fließkommazahl in IEEE-754 (32-Bit) Notation und der zweitgrößten Fließkommazahl?

9. Wie groß ist die Differenz (als 2-er Potenz) zwischen der zweitkleinsten positiven **normalisierten** Fließkommazahl in IEEE-754 (32-Bit) Notation und der kleinsten positiven **normalisierten**?

10. Wie groß ist die Differenz (als 2-er Potenz) zwischen der größten positiven **denormalisierten** Fließkommazahl in IEEE-754 (32-Bit) Notation und der zweitgrößten **denormalisierten** Fließkommazahl?

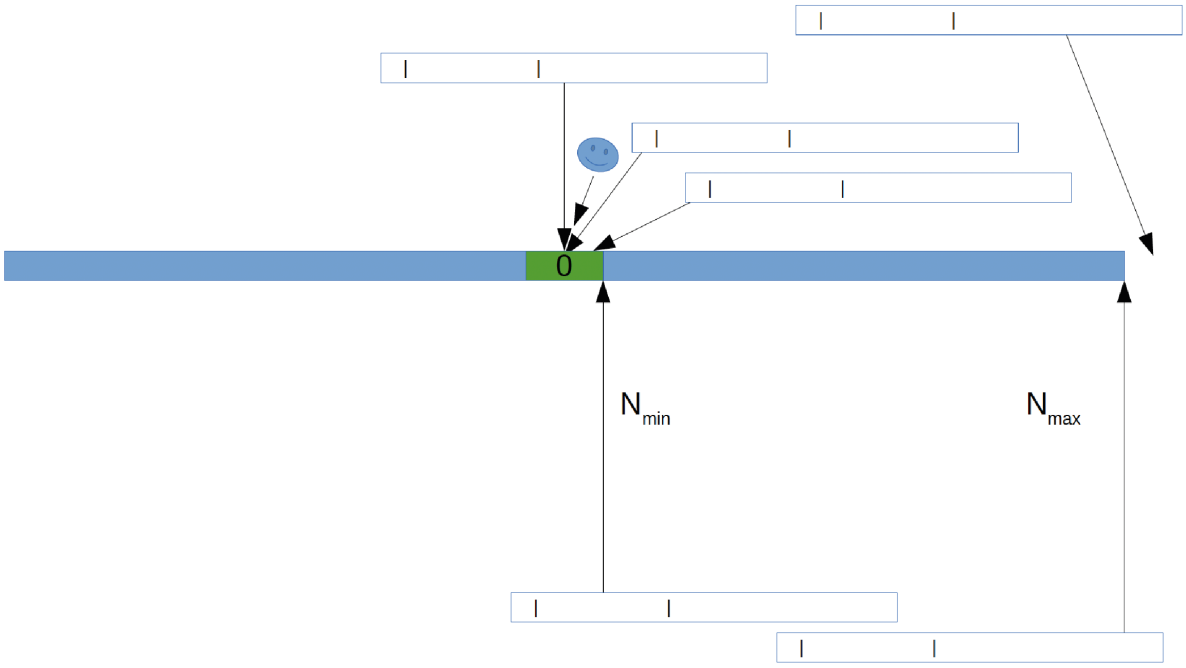
11. Wie groß ist die Differenz (als 2-er Potenz) zwischen der zweitkleinsten positiven **denormalisierten** Fließkommazahl in IEEE-754 (32-Bit) Notation und der kleinsten positiven **denormalisierten** Fließkommazahl ($\neq 0$)?

12. Wie groß ist die Differenz (als 2-er Potenz) zwischen der kleinsten positiven **normalisierten** Fließkommazahl in IEEE-754 (32-Bit) Notation und der größten positiven **denormalisierten** Fließkommazahl?

Technische Informatik: Übungsblatt 06

Benzschawel, Tittebrandt, Well 4. Dezember 2024

1. Bitte notieren Sie in IEEE-754 (32-Bit) Notation im richtigen Kästchen:
   (a) Positiv unendlich
   (b) Eine positive Null
   (c) Kleinste darstellbare denormalisierte positive Fließkommazahl
   (d) Größte darstellbare denormalisierte positive Fließkommazahl
   (e) Kleinste darstellbare normalisierte positive Fließkommazahl
   (f) Größte darstellbare normalisierte positive Fließkommazahl
   (g) Wofür steht der Smiley?



2. (a) Welchen dezimalen Wert hat die angegebene IEEE-754 Fließkommazahl?

1	01111110	10000000000000000000000

   (b) Der Wert dieser Zahl wird durch $(-4)_{10}$ dividiert. Wie sieht die Zahl in IEEE-754 Fließkommazahldarstellung aus?

3. Bestimmen Sie die Werte von A, B, C und D so, dass

(a) $A + \overline{B} + C + \overline{D}$ gleich 0 wird.

(b) $A \cdot \overline{B} \cdot C \cdot \overline{D}$ gleich 1 wird.

(c) $\overline{A} \cdot \overline{B}$ gleich 1 wird.

4. Beweisen Sie mit Hilfe des Distributivgesetzes $A \cdot (B + C) = A \cdot B + A \cdot C$ die Gültigkeit der Gleichung $A + A \cdot B = A$. Verifizieren Sie Ihr Ergebnis zusätzlich mit einer Wertetabelle.

5. Leiten Sie mit Hilfe der bekannten algebraischen Regeln die Gültigkeit der gegebenen Gleichung her. Verifizieren Sie Ihr Ergebnis zusätzlich mit einer Wertetabelle.

$$A + \overline{A} \cdot B = A + B$$

6. In der Booleschen Algebra wird oft ein zweites Distributivgesetz angegeben:

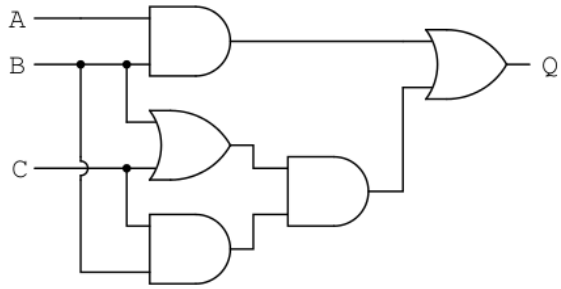
$$(A + B) \cdot (C + D) = A \cdot C + A \cdot D + B \cdot C + B \cdot D$$

Leiten Sie dieses aus dem Ihnen bekannten Distributivgesetz her.

7. Leiten Sie mit Hilfe der bekannten algebraischen Regeln die Gültigkeit der gegebenen Gleichung her.

$$(A + B) \cdot (A + C) = A + B \cdot C$$

8. Stellen Sie anhand der gegebenen Schaltung die logische Gleichung für Q auf. Minimieren Sie danach die erhaltene Gleichung mit den Regeln der Booleschen Algebra.



9. (a) Verwenden Sie die Regeln der Booleschen Algebra zur Vereinfachung des Booleschen Ausdrucks:

$$A \cdot B + A \cdot (B + C) + B \cdot (B + C)$$

(b) Zeichnen Sie die logischen Schaltkreise für den ursprünglichen Ausdruck und den optimierten Ausdruck.

(c) Alternativ verwenden Sie Logisim dazu. Probieren Sie doch einmal unterm Reiter „Fenster“ die „Kombinatorik“ aus. Versuchen Sie es auch einmal mit der „Minimierung“von Logisim.

10. Verwenden Sie die Regeln der Booleschen Algebra zur Vereinfachung der Booleschen Ausdrücke.

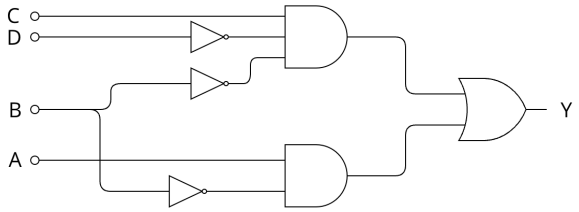
(a) $[A \cdot \overline{B} \cdot (C + B \cdot D) + \overline{A} \cdot \overline{B}] \cdot C$

(b) $A \cdot (A + B)$

(c) $A \cdot (\overline{A} + A \cdot B)$

(d) $B \cdot C + \overline{B} \cdot C$

11. Bestimmen Sie den logischen Ausdruck für:



12. Wandeln Sie die Terme mit DeMorgans Theoremen um, so dass jede Variable nur noch an maximal einer Negation beteiligt ist; also jedes Vorkommen jeder Variablen unter maximal einem Strich steht.

(a) $\overline{(\overline{X \cdot \overline{Y}}) \cdot (\overline{Y} + Z)}$

(b) $\overline{(\overline{X} + Z) \cdot (\overline{X \cdot Y})}$

(c) $\overline{\overline{A + B \cdot \overline{C}} + D \cdot (\overline{E + \overline{F}})}$

13. Wenden Sie DeMorgans Theoreme auf die folgenden Ausdrücke an:

(a) $\overline{A + \overline{B}}$

(b) $\overline{\overline{A} \cdot B}$

(c) $\overline{A + B + C}$

(d) $\overline{A \cdot B \cdot C}$

14. Erstellen Sie eine Wertetabelle für den *Sum-of-Products* Term.

$$\overline{A} \cdot \overline{B} \cdot C + A\overline{B} \cdot \overline{C} + A \cdot B \cdot C$$

15. Erstellen Sie eine Wertetabelle für den *Product-of-Sums* Ausdruck.

$$(A + B + C) \cdot (A + \overline{B} + C) \cdot (A + \overline{B} + \overline{C}) \cdot (\overline{A} + B + \overline{C}) \cdot (\overline{A} + \overline{B} + C)$$

16. Ermitteln Sie aus der gegebenen Wertetabelle

(a) einen *SOP* Ausdruck.

(b) einen *POS* Ausdruck.

Dabei sind A, B und C die Inputs und X der Output.

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

17. Vereinfachen Sie die folgenden Ausdrücke mit Hilfe der Umformungen der Booleschen Algebra.

    (a) $A \cdot B + \overline{A \cdot B} \cdot C + A$

    (b) $(A + \overline{A}) \cdot (A \cdot B + A \cdot B \cdot \overline{C})$

    (c) $A \cdot B + (\overline{A} + \overline{B}) \cdot C + A \cdot B$

18. Minimieren Sie folgenden *SOP* Ausdruck mit einem Karnaugh Diagramm:

$$A \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot C + \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C}$$

    Sie finden eine Vorlage für ein Karnaugh Diagramm im Anhang 2.

19. Minimieren Sie folgenden *SOP* Ausdruck mit einem Karnaugh Diagramm:

$$\overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} + A \cdot B \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C \cdot D + A \cdot \overline{B} \cdot C \cdot D + \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D}$$
$$+ \overline{A} \cdot B \cdot C \cdot \overline{D} + A \cdot B \cdot C \cdot \overline{D} + A \cdot \overline{B} \cdot C \cdot \overline{D}$$

    Sie finden eine Vorlage für ein Karnaugh Diagramm im Anhang 2.

Technische Informatik: Übungsblatt 07

Benzschawel, Tittebrandt, Well 4. Dezember 2024

1. Erstellen Sie mit Logisim

   (a) einen Multiplexer mit 2 Eingängen und 1 Steuerleitung. Nutzen Sie dazu ein NOT, zwei AND und ein OR Gatter.

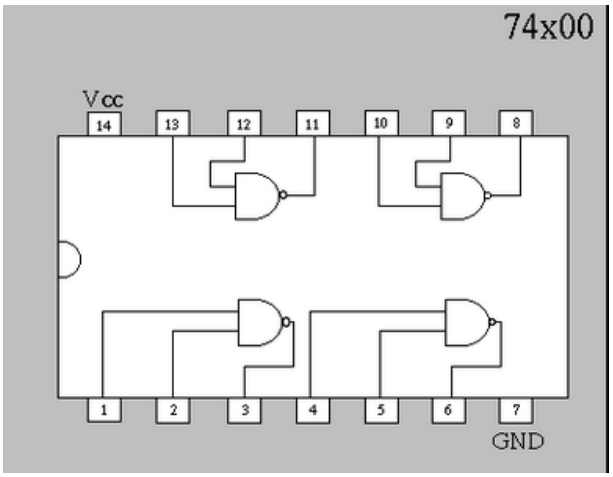
   (b) einen S-R Latch.

   (c) einen gated S-R Latch.

   (d) einen (gated) D Latch.

   Spielen Sie für das D Latch die Eingaben aus der Vorlesung nach.

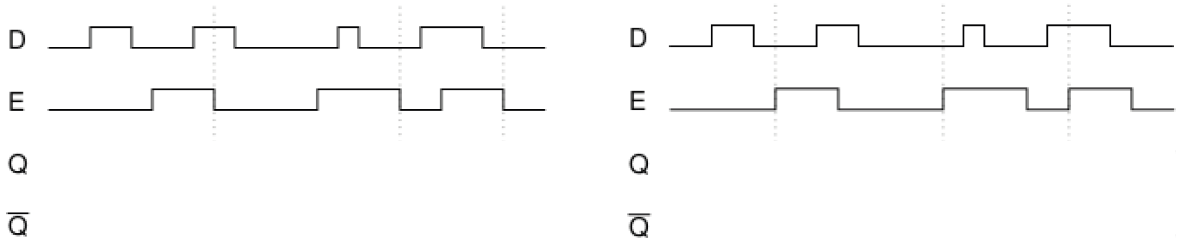
2. Sie haben folgenden Baustein zur Verfügung. Realisieren Sie eine „OR“ Logik: $Y = A + B$.



3. (a) Erläutern Sie den Unterschied zwischen einem D Latch und einem S-R Latch.

   (b) Welchen Vorteil hat ein gated S-R Latch gegenüber einem S-R Latch?

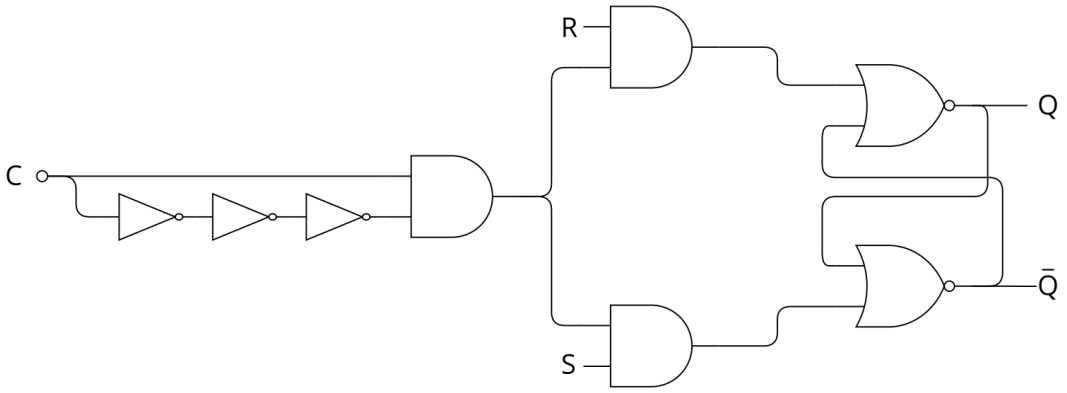
4. Unten ist das Verhalten von zwei unterschiedlichen D-Latches dargestellt. Bitte vervollständigen Sie die Output-Linie jeweils für Q und $\overline{Q}$.



5. (a) Zeichnen Sie ein S-R Flip-flop mit 2 NOR und 2 AND Gattern. Die Flankenerzeugung können Sie als Blackbox einzeichnen oder mit 3 Negierern und einem AND eine aufsteigende Taktflanke anlegen.
   Danach bauen Sie das S-R Flip-flop um zu einem J-K Flip-flop.

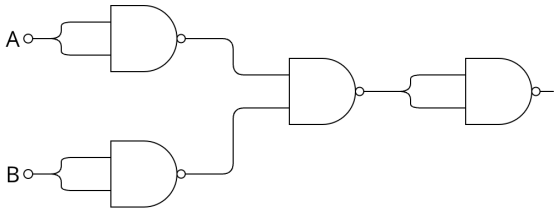
   (b) Welchen Vorteil hat das J-K gegenüber einem S-R Flip-flop?

6. Ändern Sie die Schaltung so ab, dass Zustand Q sich mit abfallender Flanke von $C(1 \rightarrow 0)$ ändern kann.

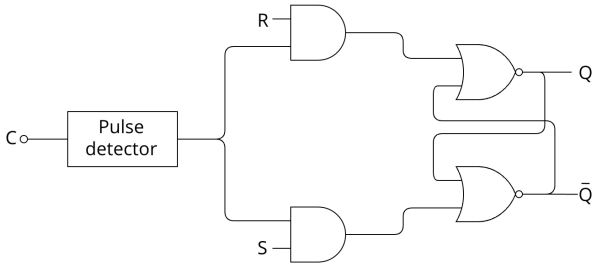


7. (a) Welche Gatter werden auch als „Universal Gates“ bezeichnet?

   (b) Die Logik welches Basis-Gatters ist hier realisiert?



   (c) Unten ist ein S-R Flip-Flop mit NOR- und AND-Gattern gezeigt. Erweitern Sie dieses S-R Flip-Flop zu einem J-K Flip-Flop. (Hilfestellung: R wird zu K).



8. In Stud.ip finden Sie einen Logisim-Registersatz. Es sind 4 Register, die mit 00, 01, 10, 11 adressierbar sind. Die Wortbreite jedes Registers ist 3 Bit, realisiert mit je 3 D Flip-flops. Laden Sie bitte eine binäre 3 in das Register 2 und eine 5 in das Register 3. Lesen Sie dann den Inhalt von Register 2 aus.

Technische Informatik: Übungsblatt 08

Benzschawel, Tittebrandt, Well 4. Dezember 2024

1. Ergänzen Sie den Programmausschnitt zur Berechnung von

$$a = (b + c) - (d - e).$$

Kommentieren Sie Ihre Schritte.

```
# a, b, c, d, e in $s0, $s1, $s2, $s3, $s4

add $t0,
```

2. Erläutern Sie die Abarbeitung der ersten beiden unten genannten Assembler-Instruktionen anhand des Architekturbildes (Anhang 4). Beschränken Sie sich auf den Registersatz und den Datenspeicher.

```
lw $t0, 8($s0)
lw $t1, 12($s0)
add $t2, $t0, $t1
sw $t2, 4($s0)
```

   (a) Erläutern Sie zuerst die erste Instruktion und heben Sie die benötigten Datenleitungen hervor.

   - Wozu dient das Register $\$s0$?
   - Wie und wo wird im Bild $\$s0$ adressiert?
   - Welche Aufgabe hat die 8?
   - Wozu dient das Register $\$t0$?
   - Welche Daten sollen von wo nach wo geladen (load) werden?

   (b) Erläutern Sie nun den Unterschied zur zweiten Instruktion.

   - Was muss an der Adressierung des Registersatzes geändert werden?
   - Warum 12 anstatt 8?
   - Was würde es bedeuten, wenn die zweite Instruktion wie folgt geändert wäre:

```
lw $t0, 12($s0)
```

   - Was würde es bedeuten, wenn die zweite Instruktion wie folgt geändert wäre:

```
lw $t1, 8($s0)
```

3. Erläutern Sie die Abarbeitung der letzten beiden unten genannten Assembler-Instruktionen anhand des Architekturbildes (Anhang 4). Beschränken Sie sich auf den Registersatz und den Datenspeicher.

```
lw $t0, 8($s0)
lw $t1, 12($s0)
add $t2, $t0, $t1
sw $t2, 4($s0)
```

   (a) Die add Instruktion spricht drei Register an.

   - In welches soll das Ergebnis geschrieben werden?
   - Woher weiß die ALU welche Operation sie rechnen soll?
   - Welche Bit-Breite haben die beiden Leitungen bei Read-data1 und Read-data2 im Registerblock?
   - Wann liegt das Ergebnis der Berechnung am Ausgang der ALU bereit?
   - Heben Sie den Datenpfad hervor, den das Ergebnis nun nehmen soll.
   - Annahme: Die Instruktion lautet etwas anders:

     ```
     add $t0, $t0, $t1
     ```

     Wäre das noch korrekt?
   - Was würde die folgende Änderung bedeuten?

     ```
     add $t2, $t2, $t2
     ```

   (b) Erläutern Sie abschließend die letzte Instruktion und heben Sie die benötigten Datenleitungen hervor.

   - Wozu dient das Register $$s0$?
   - Wie und Wo wird im Bild $$s0$ adressiert?
   - Welche Aufgabe hat die 4?
   - Wozu dient das Register $$t2$?
   - Welche Daten sollen von wo nach wo gespeichert (store) werden?

4. Betrachten Sie das Architekturbild und denken Sie an **add $t2, $t0, $t1**. Die Hardware versteht diese Instruktion noch nicht. Sie benötigt 0-en und 1-en. Im Registersatz haben Sie 32 Register verfügbar. Die Tabelle (Anhang 3) ordnet jedem nummerierten Register $0 bis $31 einen schöneren Namen zu.
   Schreiben Sie die Instruktion mit den nummerierten Registern ($0 bis $31) auf.
   Mit dem Operator besteht das gleiche Problem. Die Hardware will 0-en und 1-en. Jeder Operator bekommt ein bestimmtes Bitmuster zugewiesen. Das Bitmuster für add, sowie ein paar nicht benötigte Bits sind unten bereits in einer 32-Bit breiten Instruktion eingetragen.

   000000 _____ _____ _____ 00000 100000

Es fehlen nur noch die beteiligten Register. Diese werden jeweils mit 5 Bit angegeben. Eine kleine Änderung muss jedoch bedacht werden: Zuerst werden die beiden Operanden-Register benannt, dann das Register für das Ergebnis. Tragen Sie die fehlenden 0-en und 1-en ein.

Technische Informatik: Übungsblatt 09

Benzschawel, Tittebrandt, Well 4. Dezember 2024

1. Zeichnen Sie die unten genannten Komponenten mit dem vereinfachten Datenpfad der MIPS-Architektur.

   - PC
   - Programmspeicher und Befehlsregister
   - 32-er Registersatz
   - ALU und Datenspeicher

2. (a) Welche Register werden in dem genannten Maschinenbefehl verwendet? Benutzen Sie zur Ermittlung die Tabelle im Anhang 3.

Opcode	RS	RT	RD	SHAMT	FUNCT
000000	01000	01001	10001	00000	100000

   (b) Ergänzen Sie die Assembleranweisung. Achten Sie auf die Reihenfolge.

   add _____ _____ _____

3. Gegeben sind die unten stehenden Maschineninstruktionen. Um welche Instruktionsformate handelt es sich? Zerlegen Sie den Maschinencode und identifizieren Sie die relevanten Register/Felder. Geben Sie die resultierenden MIPS-Assemblerbefehle vollständig an.

   (a) 00000001000010011000000000100010

   (b) 10101101101001010000000000101000

4. (a) Eine C/C++ Zuweisung $x = y + z$; wird z. B. übersetzt in den MIPS-Assemblerbefehl **add $s0, $t0, $t1** unter Nutzung der genannten Register. Der Assembler erzeugt daraus wiederum den angegebenen Maschinenbefehl.

Opcode	RS	RT	RD	SHAMT	FUNCT
000000	01000	01001	10000	00000	100000

Prüfen Sie anhand des Core-Instruction-Set (Anhang 5), ob das stimmt. Tragen Sie bitte jedes Bit dieses Maschinenbefehls in die MIPS-Architektur (Anhang 4) ein. Färben Sie bitte die verwendeten Datenpfade ein.

(b) Wiederholen Sie diese Übung auf der Software-Hardware-Schnittstelle mit dem weiter gegebenen Maschinenbefehl. Färben Sie wieder die Datenpfade ein.

Opcode	RS	RT	Constant/Address/Offset
100011	01001	01000	0000000000010000

Technische Informatik: Übungsblatt 10

Benzschawel, Tittebrandt, Well 4. Dezember 2024

1. Gegeben ist der folgender Assembler-Code:

```
0x80000:    Loop:   sll $t1, $s3, 2
0x80004:            add $t1, $t1, $s6
0x80008:            lw $t0, 0($t1)
0x8000C:            bne $t0, $s5, Exit
0x80010:            addi $s3, $s3, 1
0x80014:            j Loop
0x80018:    Exit:
```

   (a) Geben Sie für jeden Befehl an um welches Instruktionsformat es sich handelt.

   (b) Zerlegen Sie die Assemblerbefehle aus den Codezeilen **0x80004 - 0x80014** in die einzelnen Felder der entsprechenden Maschinenbefehle. Die Werte der einzelnen Felder innerhalb der Maschineninstruktion können Sie als Dezimalzahl angeben.

   (c) Welches Statement einer höheren Programmiersprache liegt diesem Assembler-Code zugrunde: *while*, *if*, *repeat*, *for*, *do-while*?

2. Laden Sie zwei Ganzzahlen in die Register $\$t0$ und $\$t1$ der MIPS. Addieren Sie beide und legen Sie das Ergebnis in ein Register. Achten Sie dabei darauf, dass er Wertebereich nicht überschritten wird. Speichern Sie das Ergebnis an der Speicherstelle **0x10000100**.

3. Laden Sie die binäre Zahl 0000 0000 0011 1101 0000 1001 0000 0100 in das Register $\$s0$.

   (a) Mit dezimaler Wertangabe
   lui \$s0, _____
   ori \$s0, \$s0, _____

lui	ori
0000 0000 0011 1101	0000 1001 0000 0100

   (b) Mit hexadezimaler Wertangabe
   lui \$s0, _____
   ori \$s0, \$s0, _____

4. Es soll ein 32-Bit Wort aus dem Register $\$s0$ an die Speicheradresse **0x00040014** geschrieben werden. Nutzen Sie das Register $\$t1$ als Hilfsregister zur Adressierung. Pseudoanweisungen sind nicht erlaubt. Ergänzen Sie bitte die 3 MIPS-Assembleranweisungen.

lui
ori
sw

Lösen Sie die gleiche Aufgabe mit nur 2 MIPS-Assembleranweisungen. Pseudoanweisungen sind auch hier nicht erlaubt.

5. Gegeben sind die folgenden Registerbelegungen und die unten gezeigte Speicherbelegung.

   Register $\$t2$ enthält **0xA9040FFF**
   Register $\$t3$ enthält **0x00040000**

   Schreiben Sie mit der lw Instruktion den Wert **0x00000002** ins Register $\$t2$. Hinweis: Speicheradressen und Werte sind Hexadezimal angegeben.

Memory	**Address**
00000005	00040014
00000004	00040010
00000003	0004000C
00000002	00040008
00000001	00040004
00000000	00040000

   lw _____, _____(_____)

6. Gegeben sind die folgenden Registerbelegungen und die unten gezeigte Speicherbelegung.

   Register $\$s2$ enthält **0xFFFFFFFF**
   Register $\$s3$ enthält **0x00040000**

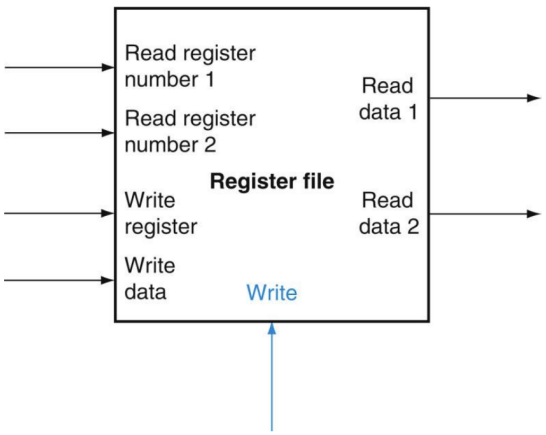
   Schreiben Sie mit der lw Instruktion den Wert **0x00000004** ins Register $\$s2$. Hinweis: Speicheradressen und Werte sind Hexadezimal angegeben.

Memory	**Address**
00000005	00040014
00000004	00040010
00000003	0004000C
00000002	00040008
00000001	00040004
00000000	00040000

   lw _____, _____(_____)

7. (a) Nennen Sie die in MIPS bekannten Befehlsformate (ohne Pseudebefehle).
   (b) Nennen Sie pro Befehlsformat jeweils einen MIPS-Assemblerbefehl.

8. Die Abbildung zeigt den Registersatz einer 64-Bit Architektur mit einem 32×64-Bit General Purpose Registersatz. Schreiben Sie bitte an jede Daten- und/oder Steuerleitung welche Bitbreite diese hat.



9. Betrachten Sie den MIPS-Assemblercode:

```
lw $v0, 0($a0)
lw $v1, 4($a0)
add $t0, $v0, $v1
sw $v0, 4($a0)
sw $t0, 0($a0)
```

   (a) Wie oft wird auf den Programmspeicher zugegriffen?
   (b) Wie oft wird auf den Datenspeicher zugegriffen?

   Zählen Sie jeweils nur die Speicherzugriffe, keine Registerzugriffe.

10. Im Architekturbild (Anhang 6 sind gleich mehrere Leitungen als defekt markiert mit je einem Blitz bei A, B, C, D, E. Welche der defekten Leitungen werden für den genannten MIPS-Befehl benötigt und müssen somit repariert werden?

```
sub $s1, $t0, $t1
```

11. Im Architekturbild (Anhang 6 sind gleich mehrere Leitungen als defekt markiert mit je einem Blitz bei A, B, C, D, E. Welche der defekten Leitungen werden für den genannten MIPS-Befehl benötigt und müssen somit repariert werden?

```
lw $s1, 8($t0)
```

12. Laden Sie in ein beliebiges Register (z. B. $\$t0$) den dezimalen Wert 5. Addieren Sie die dezimale Ganzzahl 65540 (entspricht **0x00010004**) zu dieser hinzu und schreiben Sie das Ergebnis in ein weiteres Register $\$s0$.

13. Überführen Sie das folgende Programm in ein MIPS-Assemblerprogramm.

```
int x = 16;
int y = 131;
int z = x * y;
```

Benutzen Sie zur Implementierung in MIPS nicht die Multiplikationsoperationen.

Technische Informatik: Übungsblatt 11

Benzschawel, Tittebrandt, Well 4. Dezember 2024

1. Die MIPS Pseudoinstruktion *ble* $\$rs, \$rt, Label$

```
# branch less or equal
# if(R[rs] <= R[rt])
#     PC=Label
```

wird vom Assembler in zwei Instruktionen umgesetzt. Bitte kreuzen Sie die richtigen zwei an.

	slt $\$at, \$rt, \$rs$
	slt $\$at, \$rs, \$rt$
	bne $\$at, \$zero, Label$
	beq $\$at, \$zero, Label$

2. Geben Sie die Konvention zum Sichern von Registern bei Prozeduraufrufen an. Kreuzen Sie bitte an, wer welche Register sichern und wiederherstellen sollte. Caller = Aufrufer, Callee = Aufgerufener, nicht-retten = keiner

Name	**nicht retten**	**callee saved**	**caller saved**
$\$zero$			
$\$v0$			
$\$s5$			
$\$t8$			

3. Sehen Sie sich die MIPS-Architektur (Anhang 4) an. Ergänzen Sie die fehlenden Einträge in der Tabelle.

Instr.	Reg-Dst	ALU-Src	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALU Op1	ALU Op2
R-Format		0	0		0	0		1	0
lw		1	1	1		0		0	0
sw	x	1	x	0		1		0	0
beq	x	0	x		0	0		0	1

4. Angenommen $\$t0$ hält den Hexadezimalwert **0x00101000**. Was ist der dezimale Wert von $\$t2$ nach Ausführung des folgenden Codes?

```
        slt   $t2, $zero, $t0
        bne   $t2, $zero, ELSE
        j     DONE
ELSE:   addi  $t2, $t2, 0x4
DONE:
```

5. Im Architekturbild (Anhang 7) sind gleich mehrere Leitungen als defekt markiert mit je einem Blitz bei A, B, C, D, E, F, G, H.

   (a) Welche Breite haben die defekten Leitungen?

   (b) Erklären Sie zu jeder defekt markierten Leitungen wozu die jeweilige Leitung dient und/oder welche Daten drauf fließen.

      i. sub

      ii. addi

      iii. lw

      iv. sw

Technische Informatik: Übungsblatt 12

Benzschawel, Tittebrandt, Well 4. Dezember 2024

1. Implementieren Sie einen Zähler der von 0 bis 100 zählt.

2. Überführen Sie das folgende Programmstück in MIPS-Assembler:

```
int y = 1;
int x = 5;

while (y < 10){
    y *= 2;
    x += y;

    if (x < 10){
        x++;
    }
}
```

   Benutzen Sie für die Multiplikation keine „Mult“-Operationen.

3. Implementieren Sie die folgende Methode in Assembler und rufen Sie diese in Ihrem Programm auf. Geben Sie das Ergebnis auf der Konsole aus. Nutzen Sie dazu den entsprechenden Syscall.

```
int calc (int x, int y, int z)#
{
    return x + 2 * y + 4 * z;
}
```

Technische Informatik: Anhang

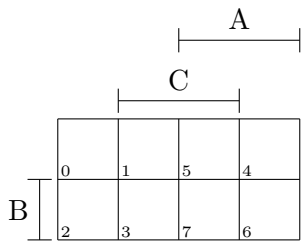
Benzschawel, Tittebrandt, Well 4. Dezember 2024

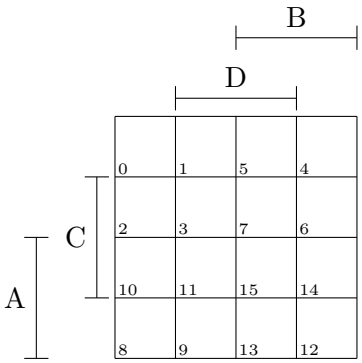
1 ASCII - American Standard Code for Information Interchange

Hex	Name	Meaning	Hex	Name	Meaning
0	NUL	Null	10	DLE	Data Link Escape
1	SOH	Start Of Heading	11	DC1	Device Control 1
2	STX	Start Of Text	12	DC2	Device Control 2
3	ETX	End Of Text	13	DC3	Device Control 3
4	EOT	End Of Transmission	14	DC4	Device Control 4
5	ENQ	Enquiry	15	NAK	Negative AcKnowledgement
6	ACK	ACKnowledgement	16	SYN	SYNchronous idle
7	BEL	BELl	17	ETB	End of Transmission Block
8	BS	BackSpace	18	CAN	CANcel
9	HT	Horizontal Tab	19	EM	End of Medium
A	LF	Line Feed	1A	SUB	SUBstitute
B	VT	Vertical Tab	1B	ESC	ESCape
C	FF	Form Feed	1C	FS	File Separator
D	CR	Carriage Return	1D	GS	Group Separator
E	SO	Shift Out	1E	RS	Record Separator
F	SI	Shift In	1F	US	Unit Separator

Hex	Char	Hex	Char	Hex	Char	Hex	Char	Hex	Char	Hex	Char
20	(Space)	30	0	40	@	50	P	60	‘	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	"	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	’	37	7	47	G	57	W	67	g	77	w
28	(	38	8	48	H	58	X	68	h	78	x
29	)	39	9	49	I	59	Y	69	i	79	y
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

2 Karnaugh Diagramme

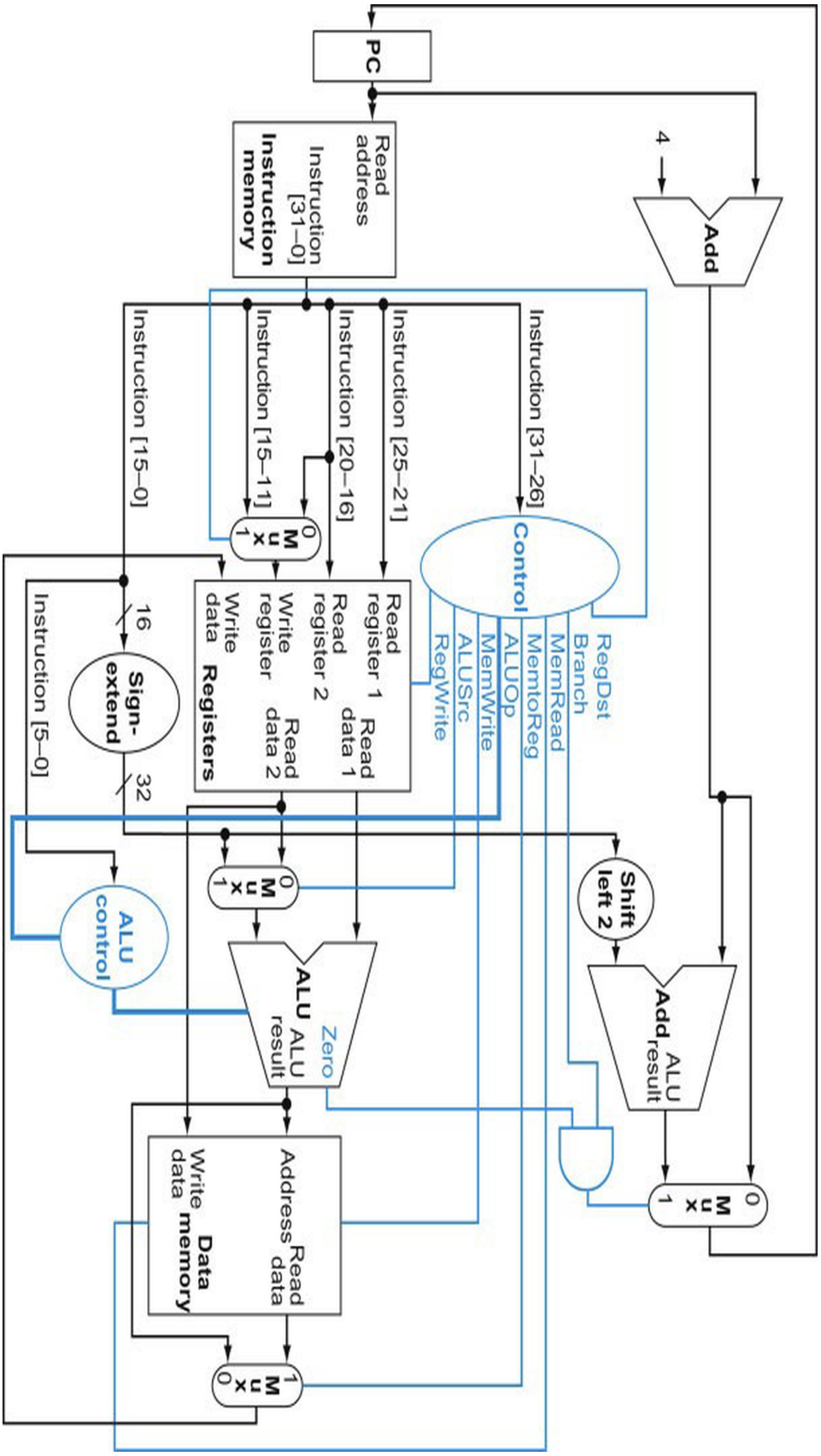




3 Register

Name	Nummer
$zero	0
$at	1
$v0-$v1	2-3
$a0-$a3	4-7
$t0-$t7	8-15
$s0-$s7	16-23
$t8-$t9	24-25
$k0-$k1	26-27
$gp	28
$sp	29
$fp	30
$ra	31

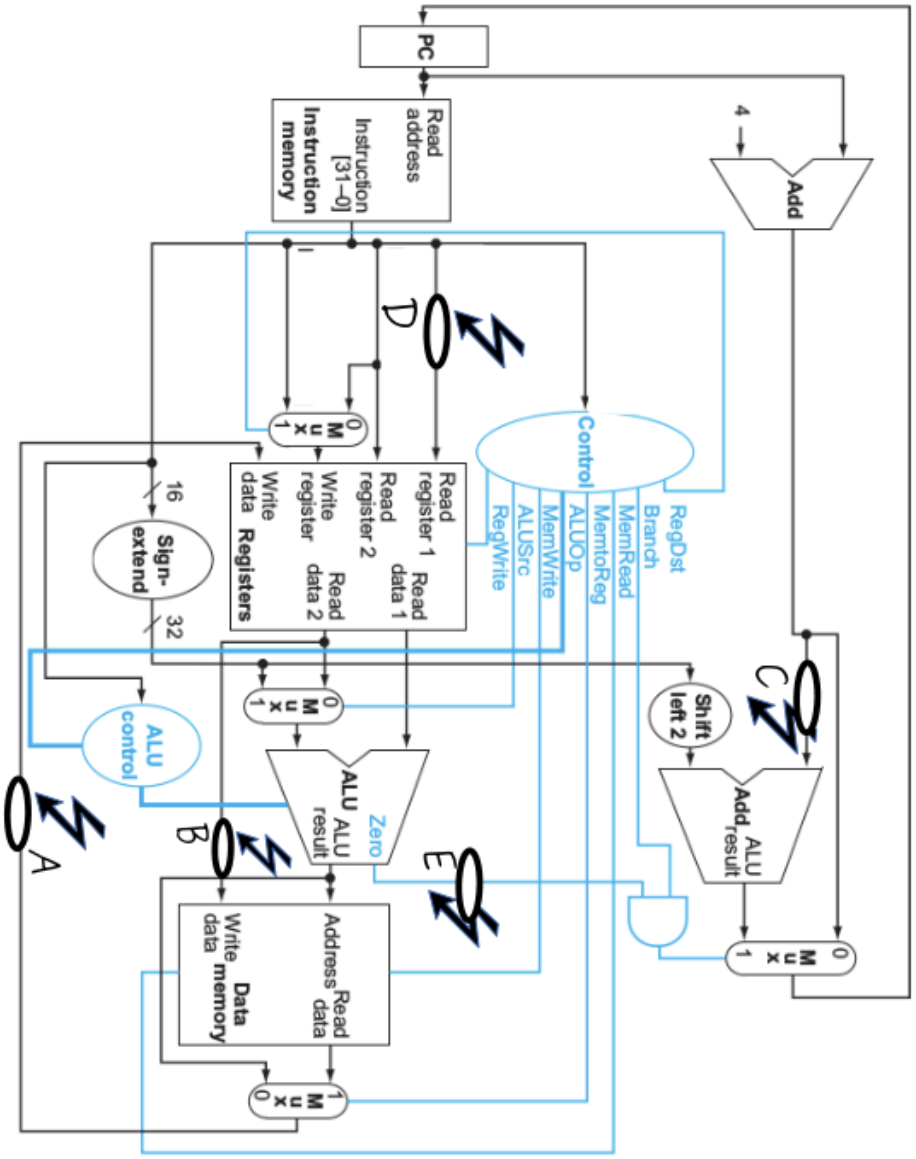
4 MIPS Architektur



5 Core Instruction Set

NAME	MNEMONIC	OPCODE/FUNCT
Add	add	0/100000
Add Immediate	addi	001000
Add Immediate Unsigned	addiu	001001
Add Unsigned	addu	0/100001
Subtract	sub	0/100010
Subtract Unsigned	subu	0/100011
And	and	0/100100
And Immediate	andi	001100
Nor	nor	0/100111
Or	or	0/100101
Or Immediate	ori	001101
Shift Left Logical	sll	0/0
Shift Right Logical	srl	0/000010
Set Less Than	slt	0/101010
Set Less Than Immediate	slti	001010
Set Less Than Immediate Unsigned	sltiu	001011
Set Less Than Unsigned	sltu	0/101011
Branch On Equal	beq	000100
Branch On Not Equal	bne	000101
Jump	j	000010
Jump And Link	jal	000011
Jump Register	jr	0/001000
Load Byte Unsigned	lbu	100100
Load Halfword Unsigned	lhu	100101
Load Upper Immediate	lui	001111
Load Word	lw	100011
Store Byte	sb	101000
Store Halfword	sh	101001
Store Word	sw	101011

6 Defekte MIPS Architektur 1



7 Defekte MIPS Architektur 2