

Tutorat 5

Zusatzautomat, Kontrolllogik, RETI übersetzen

Vorbereitung

Vorbereitung

"Xor-Terme"

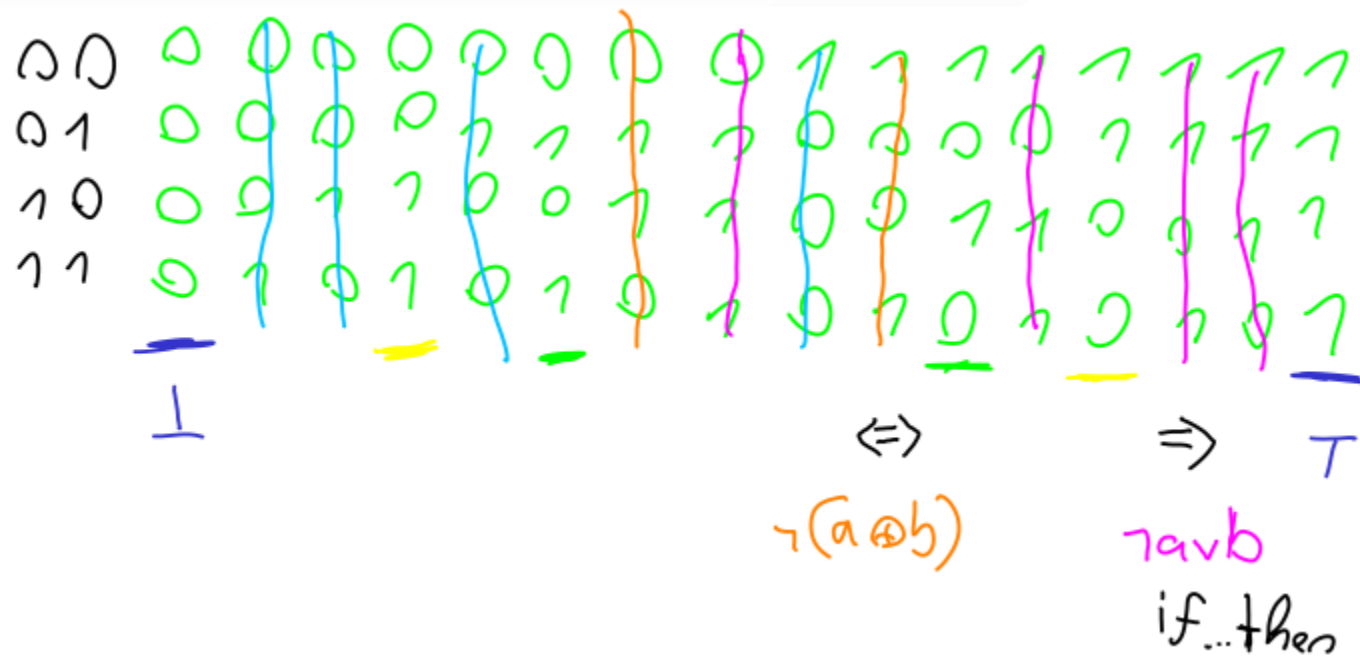
a	b	\bar{a}	\bar{b}	$a \oplus b$	$\bar{a} \oplus b$	$a \oplus \bar{b}$	$\bar{a} \oplus \bar{b}$
0	0	1	1	0	1	1	0
0	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1
1	1	0	0	0	1	1	0

$$\neg(a \oplus b) \equiv \neg(\neg a \oplus \neg b) \equiv \neg a \oplus b \equiv a \oplus \neg b$$

$$\neg(\neg a \oplus b) \equiv \neg(a \oplus \neg b) \equiv a \oplus b \equiv \neg a \oplus \neg b$$

Vorbereitung "Xor-Terme"

- damit sind die meisten **Variationen** aus **2 Aussagenlogischen Variablen** durch **Minterme**, **Maxterme**, **"Xor-Terme"**, **Top** und **Bottom** darstellbar:



Vorbereitung

"Xor-Ausdrücke" mit mehreren Aussgnl. Variablen

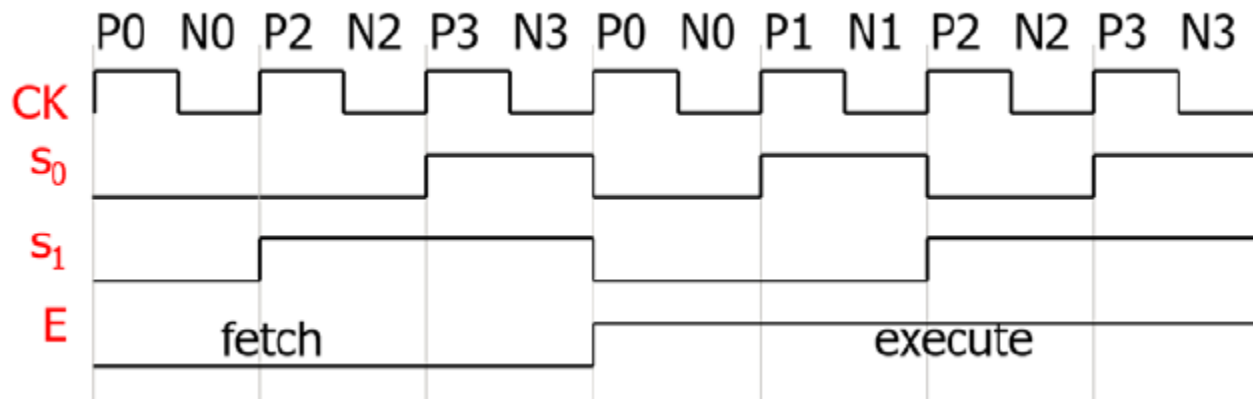
a	b	c	$a \oplus b$	$a \oplus b \oplus c$
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	1
1	1	1	0	0

- ist **true** gdw. $\#1$ en **ungerade** ist:
 - 1 ist **controlling value** und switcht $1 \rightarrow 0$ und $0 \rightarrow 1$
 - 0 ist **non-controlling value** und belässt es so wie es ist
 - **Bsp.** 01011 : $0 \xrightarrow{1} 1 \xrightarrow{0} 0 \xrightarrow{1} 1 \xrightarrow{1} 0$

Übungsblatt

Übungsblatt

Aufgabe 1

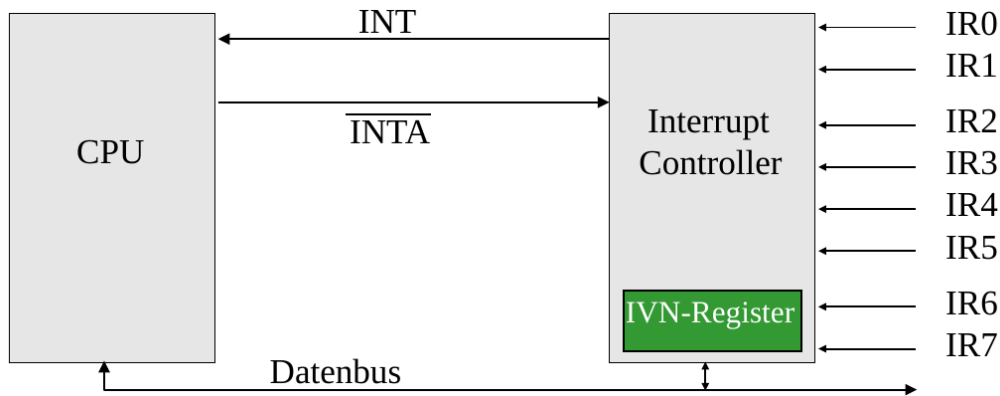


- CNTinc, CNTdec, CNTzero

Typ	J	Befehl	Wirkung	Typ	J	Befehl	Wirkung
1 1	0 1	INT i	Interruptroutine Nr. i wird ausgeführt	1 1	1 0	RTI	Rücksprungadresse vom Stack entfernt, in PC geladen, Wechsel in Usermodus

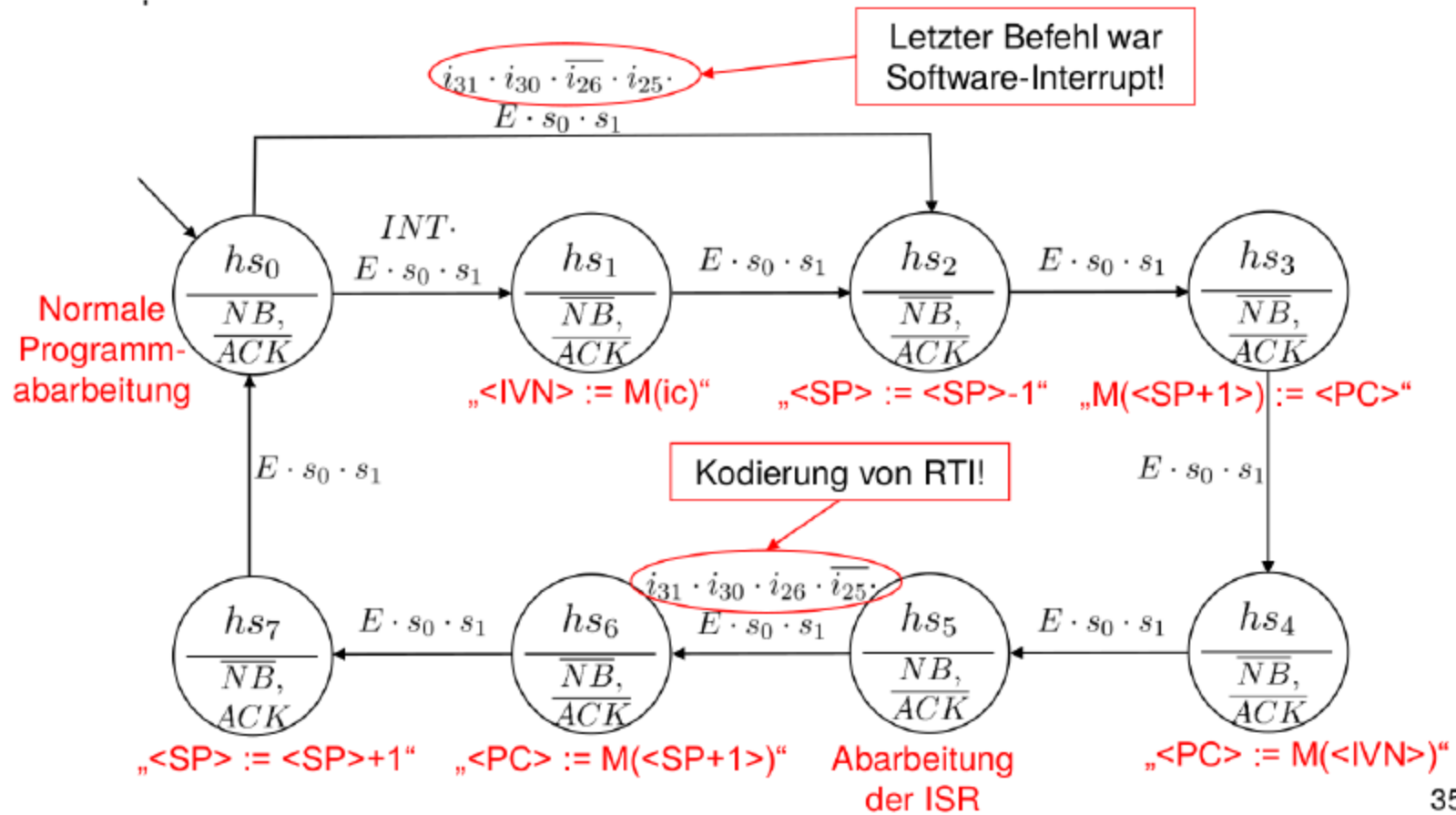
Übungsblatt

Aufgabe 1



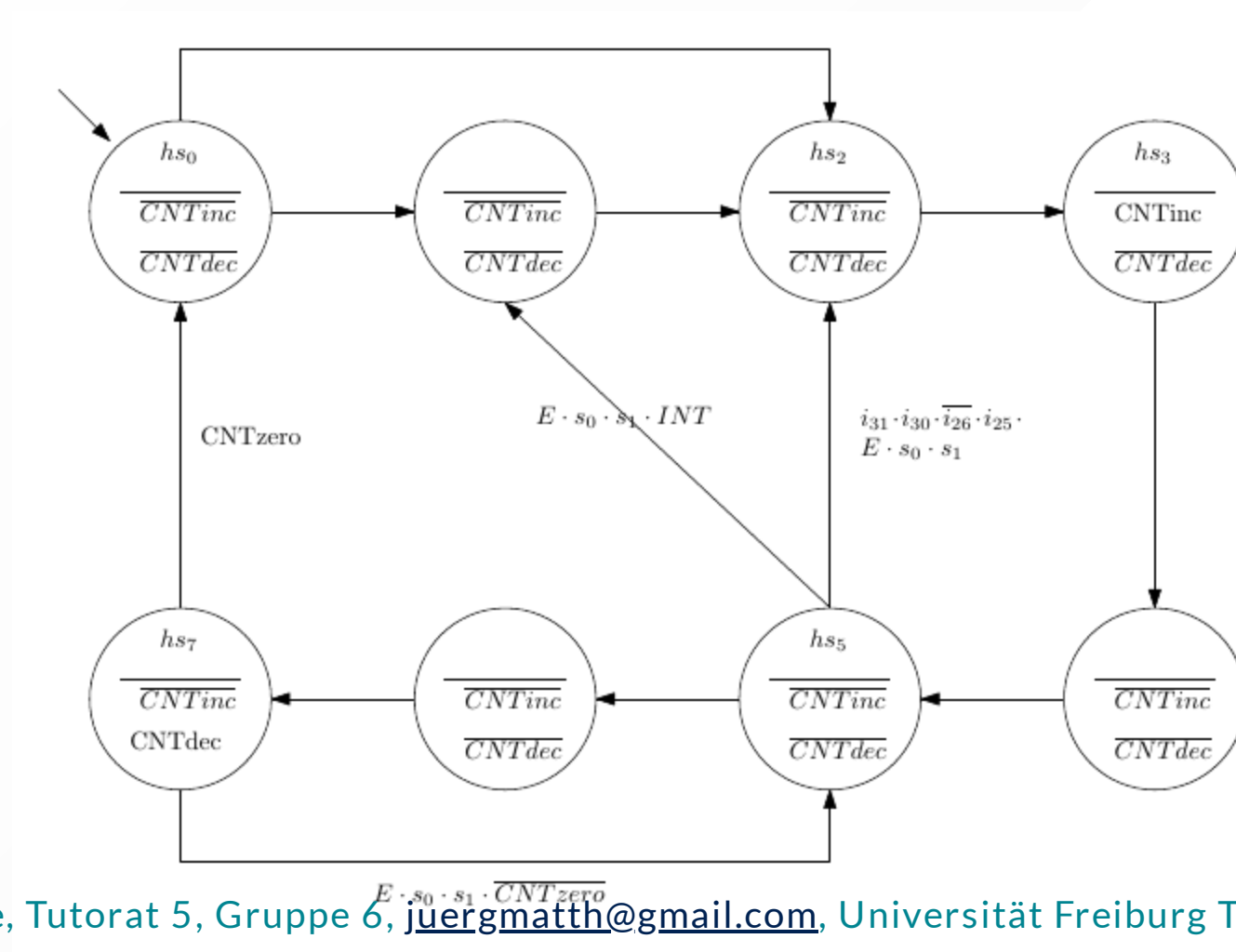
- **Normalbetrieb** sind Befehle `ADDI D i`, `STOREIN D S i`, `INT i` usw., die der User über **Bibliothekfunktionen** indirekt aufruft mit **Fetch** und **Execute Phase** usw. `INT i` ist nur der Befehl der eine **ISR** einleitet und läuft selber im **Normalbetrieb**. Die **ISR** selber ist dann im **Nichtnormalbetrieb** und läuft den Zustandsautomaten durch. Neben **Software-Interrupt** gibt es noch **Hardware Interrupt**, wenn man z.B. eine Taste auf dem Keyboard drückt.

Aufgabe 1



Übungsblatt

Aufgabe 1



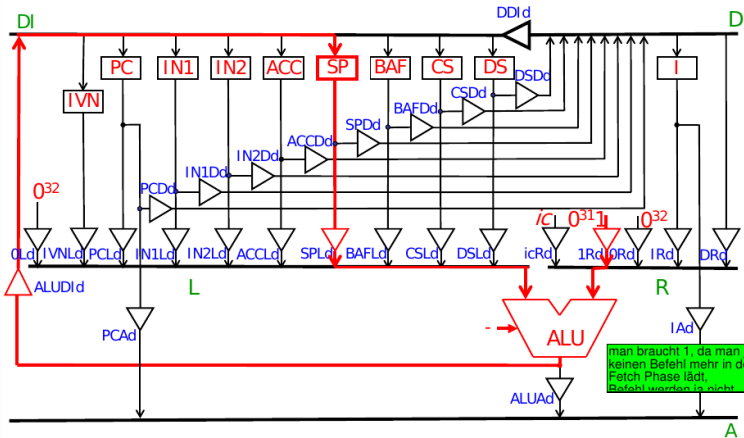
Übungsblatt

Aufgabe 2

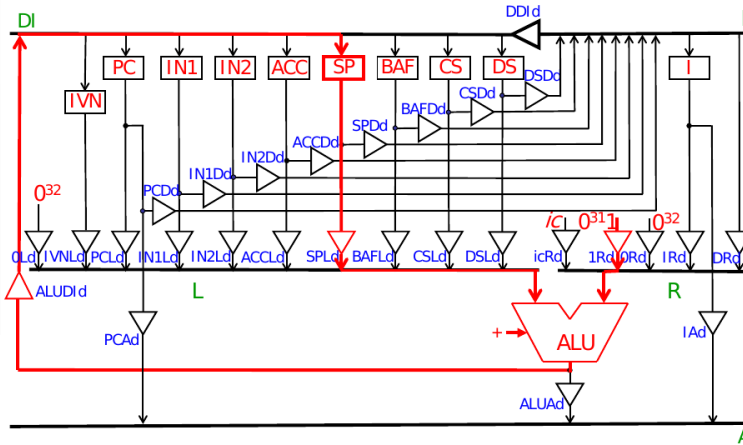
P0 N0 P2 N2 P3 N3 P0 N0 P1 N1 P2 N2 P3 N3

rcken*

Aktive Datenpfade in Zustand *hs₂*



Aktive Datenpfade in Zustand *hs₇*



S, D	Register
0 0 0	PC
0 0 1	IN1
0 1 0	IN2
0 1 1	ACC
1 0 0	SP
1 0 1	BAF
1 1 0	CS
1 1 1	DS

Übungsblatt

Aufgabe 2

Typ	Modus	Befehl	Wirkung		Typ	Modus	Befehl	Wirkung	
0 1	0 0	LOAD D i	D := M(<i>)	<PC> := <PC> + 1	1 0	0 0	STORE S i	M(<i>) := S	<PC> := <PC> + 1
0 1	0 1	LOADIN S D i	D := M(<S> + [i])	<PC> := <PC> + 1	1 0	0 1	STOREIN D S i	M(<D> + [i]) := S	<PC> := <PC> + 1
0 1	1 1	LOADI D i	D := 0 ¹⁰ i	<PC> := <PC> + 1	1 0	1 1	MOVE S D	D := S	<PC> := <PC> + 1

- ComputelmmEDIATE: 0000, Compute Memory: 0010, Compute Register: 00*1

Übungsblatt

Aufgabe 2

$$\begin{aligned} \text{SPcken}_{\text{pre}} = [E \cdot \overline{s_1} \cdot s_0] \{ & [\overline{I_{31}} \cdot I_{30} \cdot I_{24} \cdot \overline{I_{23}} \cdot \overline{I_{22}} + \\ & \overline{I_{31}} \cdot \overline{I_{30}} \cdot I_{24} \cdot \overline{I_{23}} \cdot \overline{I_{22}} + \\ & I_{31} \cdot \overline{I_{30}} \cdot I_{29} \cdot I_{28} \cdot I_{24} \cdot \overline{I_{23}} \cdot \overline{I_{22}}] \cdot NB + \\ & [\overline{h_2} \cdot h_1 \cdot \overline{h_0} + h_2 \cdot h_1 \cdot h_0] \} \end{aligned}$$

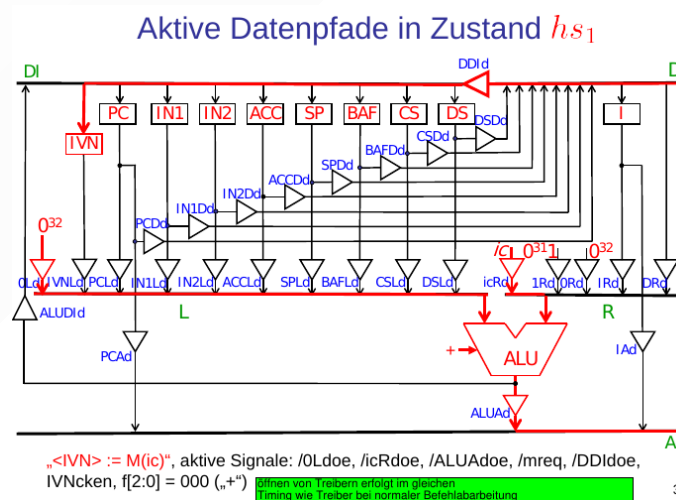
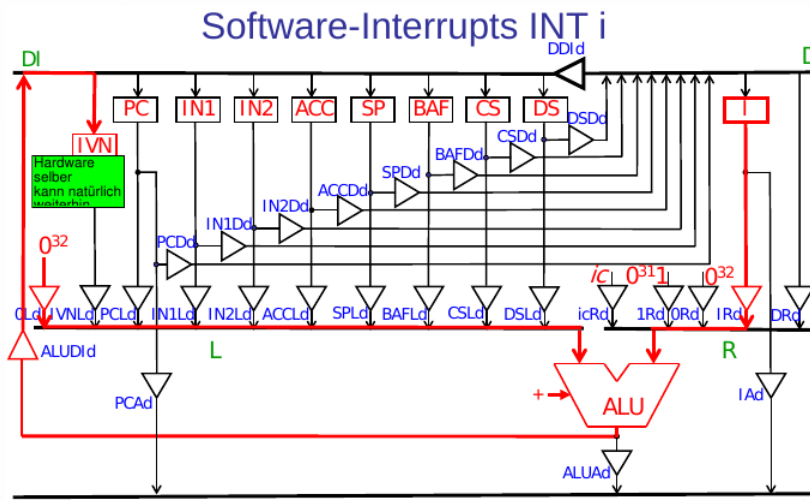
- **true gdw.** bestimmte Phase von *Execute* ($E \sim s_0 s_1$), im *Normalbetrieb* (NB) **und** wenn die *Befehle* [...] kodiert an die Kontrollogik vom Instruktionsregister weitergegeben werden **oder** im Zusatzautomat in den *Zuständen* [...]

Übungsblatt

Aufgabe 2

P0 N0 P2 N2 P3 N3 P0 N0 P1 N1 P2 N2 P3 N3

rcken*



Übungsblatt

Aufgabe 2

Typ	J	Befehl	Wirkung
1 1	0 1	INT i	Interruptroutine Nr. i wird ausgeführt

$$IVNcken_{pre} = [E \cdot \overline{s_1} \cdot s_0] \left\{ [\overline{h_2} \cdot \overline{h_1} \cdot h_0] + NB \cdot I_{31} \cdot I_{30} \cdot \overline{I_{26}} \cdot I_{25} \right\}$$

Übungsblatt

Aufgabe 3

```
void main()
{
    //Deklarationsteil
    int x;
    int y;
    const int z = 5;

    //Anweisungsteil
    y = 3;
    x = 2;

    x = (x + ((y * z) + 10)); // vollstaendig geklammerter Ausdruck
}
```

Übungsblatt

Aufgabe 3 a)

Symboltabelle

- $st(x) = (var, int, 128)$
 $st(y) = (var, int, 129)$
 $st(z) = (const, int, '5')$

Übungsblatt

Aufgabe 3 b)

```
SUBI SP 1; // Variablenbezeichner y, Adresse 129
LOADI ACC 3;
STOREIN SP ACC 1;
LOADIN SP ACC 1; // Zuweisung
ADDI SP 1;
STORE ACC 129;
```

```
SUBI SP 1; // Variablenbezeichner x, Adresse 128
LOADI ACC 2;
STOREIN SP ACC 1;
LOADIN SP ACC 1; // Zuweisung
ADDI SP 1;
STORE ACC 128;
```

Übungsblatt

Aufgabe 3 b)

```
SUBI SP 1 // Variablenbezeichner x, Adresse 128  
LOAD ACC 128  
STOREIN SP ACC 1 // x=2 auf Stack
```

```
SUBI SP 1 // Variablenbezeichner y, Adresse 129  
LOAD ACC 129  
STOREIN SP ACC 1 // y=3 auf Stack
```

```
SUBI SP 1  
LOADI ACC 5 // z ist Konstante, also direkt 5 nutzen.  
STOREIN SP ACC 1 // z=5 auf Stack
```

Übungsblatt

Aufgabe 3 b)

```
LOADIN SP ACC 2 // ACC:= y=3
LOADIN SP IN2 1 // IN2:= z=5
MUL ACC IN2 // ACC:= 3*5=15
STOREIN SP ACC 2 // 15 auf den Stack (zweitoberste Stack-Zelle)
ADDI SP 1 // Stack um eins verkuerzen

SUBI SP 1
LOADI ACC 10 // ACC:= 10
STOREIN SP ACC 1 // 10 auf Stack
```


Übungsblatt

Aufgabe 3 b)

```
LOADIN SP ACC 2 // ACC:= 15
LOADIN SP IN2 1 // IN2:= 10
ADD ACC IN2 // ACC:= 15+10=25
STOREIN SP ACC 2 // 25 auf den Stack (zweitoberste Stack-Zelle)
ADDI SP 1 // Stack um eins verkuerzen
```

```
LOADIN SP ACC 2 // ACC:= 2
LOADIN SP IN2 1 // IN2:= 25
ADD ACC IN2 // ACC:= 2+25=27
STOREIN SP ACC 2 // 27 auf den Stack (zweitoberste Stack-Zelle)
ADDI SP 1
```

Übungsblatt

Aufgabe 3 b)

```
LOADIN SP ACC 1;  // Variable x zuweisen  
ADDI SP 1;  
STORE ACC 128;
```

Übungsblatt

Aufgabe 3 c)

- $(x_1 \circ (x_2 \circ (x_3 \circ \dots (x_{n-1} \circ x_n)) \dots)) \rightarrow \text{max. } n \text{ Teilergebnisse}$
- $((\dots (x_1 \circ x_2) \circ x_3) \circ \dots x_{n-1}) \circ x_n \rightarrow \text{max. } 2 \text{ Teilergebnisse}$

Anmerkung

- Compilerpattern sind nicht effizient \rightarrow alle Variablen auf dem Stack gespeichert
- Echte Compiler versuchen allerdings mittels **Graph Coloring** möglichst viele Variablen **Registern** zuzuweisen, was nicht immer so einfach ist, weil man erst die **liveness** (**live gdw.** wenn der momentante Wert der Variablen zu einem späteren Zeitpunkt im Programm genutzt wird) jeder einzelnen Variable herausfinden muss und schauen muss, wo sich diese Zeiträume überschneiden, also nicht dem **gleichen** Register zugewiesen werden können usw.

Quellen

Quellen

Wissenquellen

- 

Quellen

Bildquellen

- 

**Vielen Dank für
eure
Aufmerksamkeit!**

