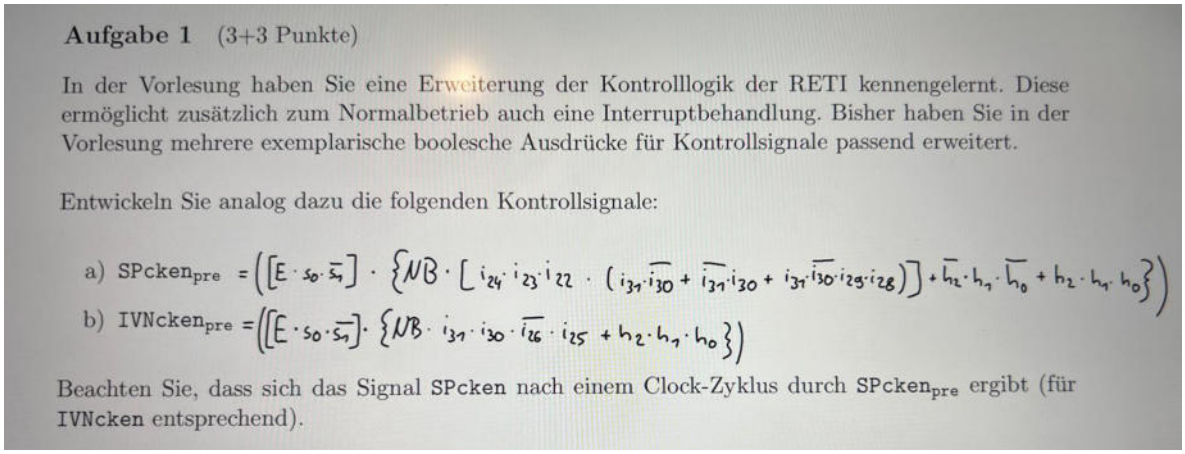
Betriebssysteme Blatt05 **Pascal Stefanelli & Patrick Pankan**

Aufgabe 1



Aufgabe 1 (3+3 Punkte)

In der Vorlesung haben Sie eine Erweiterung der Kontrolllogik der RETI kennengelernt. Diese ermöglicht zusätzlich zum Normalbetrieb auch eine Interruptbehandlung. Bisher haben Sie in der Vorlesung mehrere exemplarische boolesche Ausdrücke für Kontrollsignale passend erweitert.

Entwickeln Sie analog dazu die folgenden Kontrollsignale:

a) $\texttt{SPcken}_{\texttt{pre}} = \left([E \cdot s_0 \cdot \overline{s_1}] \cdot \{ NB \cdot [i_{24} \cdot i_{23} \cdot i_{22} \cdot (i_{31} \cdot \overline{i_{30}} + \overline{i_{31}} \cdot i_{30} + i_{31} \cdot \overline{i_{30}} \cdot i_{29} \cdot i_{28})] + \overline{h_2} \cdot h_1 \cdot \overline{h_0} + h_2 \cdot h_1 \cdot h_0 \} \right)$

b) $\texttt{IVNcken}_{\texttt{pre}} = \left([E \cdot s_0 \cdot \overline{s_1}] \cdot \{ NB \cdot i_{31} \cdot i_{30} \cdot \overline{i_{26}} \cdot i_{25} + h_2 \cdot h_1 \cdot h_0 \} \right)$

Beachten Sie, dass sich das Signal `SPcken` nach einem Clock-Zyklus durch $\texttt{SPcken}_{\texttt{pre}}$ ergibt (für `IVNcken` entsprechend).

Aufgabe 2

a)

st(x) = (var, int, 128)

st(y) = (var, int, 129)

st(z) = (const, int, 5)

b)

```
SUBI SP 1;
LOAD ACC 128;
STOREIN SP ACC 1;       // int x

SUBI SP 1;
LOAD ACC 129;
STOREIN SP ACC 1;       // int y

SUBI SP 1;
LOADI ACC 5;
STOREIN SP ACC 1;       // const int z
```

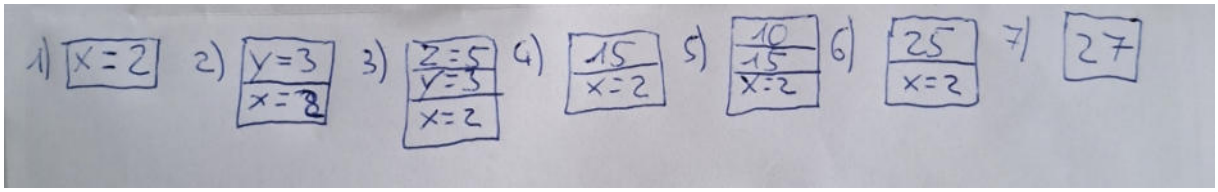
```
LOADIN SP ACC 2;
LOADIN SP IN2 1;
MUL ACC IN2;
STOREIN SP ACC 2;
ADDI SP 1;              // y * z berrechnen

SUBI SP 1;
LOADI ACC 10;
STOREIN SP ACC 1;       // 10 auf stack

LOADIN SP ACC 2;
LOADIN SP IN2 1;
ADD ACC IN2;
STOREIN SP ACC 2;
ADDI SP 1;              // ergebnis von y*z -> + 10 rechnen

LOADIN SP ACC 2;
LOADIN SP IN2 1;
ADD ACC IN2;
STOREIN SP ACC 2;
ADDI SP 1;              // Endergebnis
```

Stack:



Aufgabe 3

Vergleichsoperation x <= y

wird zu (x - y) OP 0

also:

5 <= 10 -> 5 -10 = -5 <= 0 wahr also Ergebnis 1

10 <= 5 -> 10 - 5 = 5 <= 0 Falsch also Ergebnis 0

Wenn x positiv und y negativ dann ist es Ergebnis 0 weil y kleiner ist

Wenn x negativ und y positiv dann ist es Ergebnis 1, weil y größer ist

Beide negativ

-10 <= -5 -> -5 <= 0 wahr also ergebnis 1

-5 <= -10 -> 5 <= 0 falsch also Ergebnis **0**

Code:

```
SUBI SP 1;
LOAD ACC 10;
STOREIN SP ACC 1;       // x auf stack
SUBI SP 1;
LOAD ACC 10;
STOREIN SP ACC 1;       // y auf Stack

LOADIN SP ACC 2;        // x in ACC
AND ACC 1000....0;      // vorderstes Bit prüfen (1 negativ und 0 positiv)
JUMP= 8                 // x positiv
LOADIN SP ACC 1;        // x negativ und y in ACC
AND ACC 1000...0;
JUMP> 12                // wenn y vorderstes bit = 1 also negativ normale Berechnung

LOADI ACC 1;
STOREIN SP ACC 2        // Erg 1
ADDI SP 1;
JUMP 0;                 // Programmende
```

```
                        // x positiv
LOADIN SP ACC 1;        // y in ACC
AND ACC 1000...0;       // vorderes Bit prüfen
JUMP= 5;                // = 0 also y positiv -> x und y positiv

LOADI ACC 0;            //erg 0
STOREIN SP ACC 2;
ADDI SP 1;
JUMP 0;                 // Programmende
                                        // NORMAL x und y positiv oder beide negativ
LOADIN SP ACC 2;        // x in acc
LOADIN SP IN2 1;        // y in IN1
SUB ACC IN2;
JUMP<= 2;
JUMP -8;                // erg 0 weil x - y > 0
JUMP -16;               // erg 1 weil x –y <= 0
```