

Aufgabe 1

6/6

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:

$$\begin{aligned} \text{a) } SP_{\text{cken}}_{\text{pre}} &= \left([E \cdot s_0 \cdot \bar{s}_1] \cdot \{NB \cdot [i_{24} \cdot i_{23} \cdot i_{22} \cdot (i_{31} \cdot \bar{i}_{30} + \bar{i}_{31} \cdot i_{30} + i_{31} \cdot \bar{i}_{30} \cdot i_{25} \cdot i_{28})] + \bar{h}_2 \cdot h_1 \cdot \bar{h}_0 + h_2 \cdot h_1 \cdot h_0\} \right) \\ \text{b) } IVN_{\text{cken}}_{\text{pre}} &= \left([E \cdot s_0 \cdot \bar{s}_1] \cdot \{NB \cdot i_{31} \cdot i_{30} \cdot \bar{i}_{26} \cdot i_{25} + h_2 \cdot h_1 \cdot h_0\} \right) \end{aligned}$$

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

Aufgabe 2

6/8

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 berechnen

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;

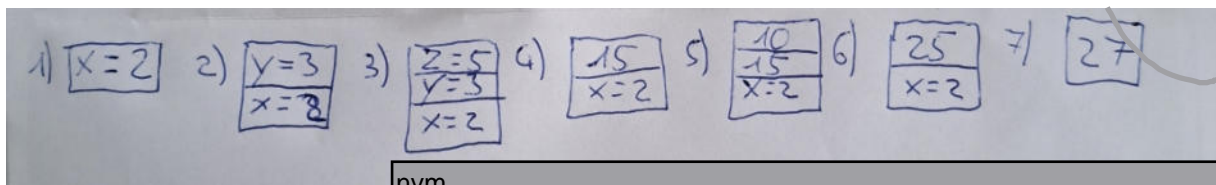
LOADIN SP IN2 1;

ADD ACC IN2;

STOREIN SP ACC 2;

ADDI SP 1; // Endergebnis

Stack:



nvm

-2 keine c)

Aufgabe 3

Vergleichsoperation $x \leq y$
wird zu $(x - y) \geq 0$

also:

$5 \leq 10 \rightarrow 5 - 10 = -5 \geq 0$ wahr also Ergebnis 1

$10 \leq 5 \rightarrow 10 - 5 = 5 \geq 0$ 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

$-10 \leq -5 \rightarrow -5 \geq 0$ wahr also ergebnis 1

$-5 \leq -10 \rightarrow 5 \geq 0$ 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

Ich hab den RETI-Code aller Studenten die Aufgabe 3 bearbeitet haben (Aufgabe 3 war diesmal die beliebteste Aufgabe) mithilfe des im PicoC-Compilers <https://github.com/matthejue/PicoC-Compiler/releases> eingebauten RETI-Interpreters ausgeführt, genauer mittels des Befehls `picoc_compiler -b -p c.reti -S -P 2 -D 15`. Ich habe versucht den Code von euch Studenten lauffähig zu machen, sodass dieser die Aufgabenstellung erfüllt. Die Datei <fruit>.in enthält Eingaben für CALL INPUT REG. Die Datei <fruit>.out enthält die Ausgaben der CALL PRINT REG bei der Ausführung. Die Datei <fruit>.out_expected enthält die erwarteten Ausgaben. Eure Korrektur ist unter https://github.com/matthejue/Abgaben_Blatt_3/tree/main/Blatt5/wassermelone.reti zu finden.

```

// 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

```