Diesmal sieht die Korrektur etwas anders aus als sonst. Ich hab den RETI-Code aller Studenten 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. Alle Korrekturanmerkungen sind in der `c.reti`-Datei als Kommentare zu finden. Die Dateien `c.uart_r` und `c.uart_s` sind zur Simualation einer UART da und stehen für das Empfangs- und Statusregister und die darin enthalten Zahlen werden sobald auf die entsprechendedn Register zugegriffen wird gepopt. Eure Korrektur ist unter https://github.com/matthejue/Abgaben_Blatt_3/tree/main/Blatt3/aprikosen zu finden.

11.11.2022

13/14

19/20

Aufgabe 1:

a)

```
LOAD ACC 2     // Statusregister R2 in Akkumulator laden.                  <--
SUBI ACC 2     // Subtrahiere ...0010 vom Akkumulator                         |
JUMP/= -2      // Wenn Akkumulator ungleich Null Schleife von vorne.       ---

ADD IN1 1      // Daten aus Empfangsregister R1 in Indexregister 1 laden.
LOADI ACC 0    // 0 in den Akkumulator laden.
STORE ACC 2    // Akkumulator abspeichern in Statusregister R2.
```

b)

```
LOADI IN2 4     // Benutze IN2 als Schleifenzaehler.
SUBI IN2 1      // Ziehe 1 von IN2 ab pro Schleifendurchlauf.               <--
MULI IN1 <100000000>    // Linksshift von IN1 um 8.                          |
POLLING-LOOP    // Aufgabenteil a)                                           |
MOVE IN1 ACC    // Schleifenzähler in Akkumulator laden,                     |
JUMP/= -4       // falls Zähler /= 0 Schleife von vorne durchlaufen.       ---
```

c)

```
LOADI IN1 0     // IN1 auf 0 setzen.
LOADI BAF 0     // Zähler für Speicher Offset in Register BAF auf 0 setzen.
LOADI DS 0      // Zugriff auf Daten im EPROM                               <--
LOAD DS r       // Konstante 010...0 in DS laden --> Zugriff auf UART          |
                                                                               |
READING-LOOP    // Aufgabenteil b)                                           |
                                                                               |
LOADI DS 0      // Zugriff auf Daten im EPROM                                |
LOAD IN2 t      // "LOADI PC 0" Kodierung aus EPROM in Indexregister 2 laden. |
OPLUS IN2 IN1   // XOR von Indexregister 1 & 2 in 2.                         |
LOAD DS s       // Konstante 100...0 in DS laden --> Zugriff auf SRAM        |
STOREIN BAF IN1 a   // Indexregister 1 an Stelle a+BAF im SRAM abspeichern.  |
ADDI BAF 1      // Register BAF inkrementieren.                              |
MOVE IN2 ACC    // Aktuelles Wort XOR Endwort in Akkumulator laden.          |
JUMP/= -8       // Wenn aktuelles Wort XOR Endwort /= 0: Schleife von vorne.  ---
LOADI DS 0      // Zugriff auf Daten im EPROM.
LOAD ACC s      // Konstante s von EPROM in Akkumulator laden.
ADDI ACC a      // Konstante a zu s im Akkumulator hinzuaddieren.
MOVE ACC PC     // Programmzähler auf a+s setzen, um das Programm auszuführen.
```

Aufgabe 2:

6/6

Das Datensegmentregister kann auf EPROM eingestellt bleiben, wenn statt LOAD D i der Befehl LOAD ACC s(z.B.) und dann LOADIN ACC D i verwendet wird, da dann im Akkumulator der Präfix für den Speicherbereich steht und für den LOADIN Befehl genutzt wird, Gleiches bei STORE und STOREIN.