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/kiwis zu finden.

12.25 / 14

BETRIEBSSYSTEME

12.25/20

Blatt 03

Students:
Julian Polzer jp390, David Janzen dj57

Tutor:
GRUPPE 7

1 Aufgabe

1. RETI Code für Empfang

CODE	COMMENT
1. LOADI IN1 0	IN1 auf 0 setzen
2. LOADI DS 0	Zugriff auf Daten im EPROM
3. LOAD DS r	Konstante r in DS laden
4. LOAD ACC 2	Satusregister R2 in ACC
5. ANDI ACC 2	AND mit ..010 to check for b1 = 1
6. SUBI ACC 2	-2; weiter mit check for b1 = 1
7. JUMP>= +2	b1 is 1 -> go on
8. JUMP< -6	b1 is not 1 -> go back
(–. LOADI DS 0)	Zugriff auf Daten im Eprom
(–. LOAD DS r)	Konstante r in DS laden
9. ADD IN1 1	R1 in IN1 speichern
(–. LOADI DS 0)	Zugriff auf Daten im EPROM
(–. LOAD DS r)	Konstante r in DS laden
10. LOAD ACC 2	R2 in ACC laden
11. SUBI 2	sub 2 to set b1 to 0

2. CODE für Bitshift

CODE	COMMENT
1. LOADI IN2 4	IN2 = 4 (Schleifencountdown)
2. Pulling-Loop	CODE aus 1. liefert 4 bits
3. MULI IN1 256	Bitshift um 8 bit mit 2 hoch 8
4. SUBI IN2 1	IN2 um 1 reduzieren
5. JUMP= +2	falls IN2 = 0 sind 32 bin in IN1
6. JUMP -4	zurück. Neue 8 Bits
7. END	32 Bit Befehl is in IN1 gespeichert

3. CODE für Speichern

CODE	COMMENT
1. Aufgabenteil b	Befehl in IN1 speichern
2. LOADI DS 0	Daten im EPROM
3. LOAD DS s	Konstante s für Hauptspeicher
4. STORE IN1 a	erste Speicheradresse mit Befehl laden
5. a+1	wir erhöhen a um 1 für nächste Speicheradresse?
6. LOADI DS 0	Zugriff EPROM
7. LOAD ACC t	Zugriff auf Programmende
8. SUB ACC IN1	Vergleich, ob es Ende ist
9. JUMP= +2	Ende wenn ja
10. JUMP -9	loop wenn nicht
11. END	

2 Aufgabe