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/klementinen zu finden.

10/14

10/20

Betriebssysteme

Übungsblatt 3

Anne Ross Diana Hörth

November 10, 2022

Aufgabe 1

a)

PC	Befehl	Kommentare
0	LOADI IN1 0	IN1 auf 0 setzen (hier kann spaeter Inhalt aus R1 addiert werden).
1	LOADI DS 0	Zugriff auf Daten im EPROM
3	LOAD DS r	Konstante 010...0 in DS laden –> Zugriff auf UART
4	LOAD ACC 2	Statusregister R2 in Akkumulator laden.
5	SUBI ACC $010^{27}010$	Wenn B1 = 1 dann kommt 0 raus wenn B1 = 0 dann eine negtive Zahl.
6	$\text{JUMP}_{<}$-2	Wenn B1 = 0, dann weiter abfragen
7	ADD IN1 1	Daten aus R1 in IN1 laden
8	SUBI 2 10	Damit B1 = 1 zu B1 = 0 im R2 wird.
9	JUMP -3	Zurück zum abfragen

b)

PC	Befehl	Kommentare
0	LOADI IN2 4	Benutze IN2 als Schleifenzaehler
1	POLLING-LOOP	Code aus Teil a)
3	MULTI IN1 100000000	Eine Zahl mit 100 Mio. multipliziert simuliert Linksshift.
4	SUBI IN2 1	Zähler um eins verringern
5	LOAD ACC IN2	
6	$\text{JUMP}_{=}$ -4	Wenn es nicht der 4te Durchlauf war, den Code aus a) durchlaufen.

c)

PC	Befehl	Kommentare
0	LOADI IN2 4	Benutze IN2 als Schleifenzaehler
1	LOADI IN1 0	IN1 auf 0 setzen (hier kann spaeter Inhalt aus R1 addiert werden).
2	LOADI DS 0	Zugriff auf Daten im EPROM
3	LOAD DS r	Konstante 010...0 in DS laden –> Zugriff auf UART
4	LOAD ACC 2	Statusregister R2 in Akkumulator laden.
5	SUBI ACC $010^{27}010$	Wenn B1 = 1 dann kommt 0 raus wenn B1 = 0 dann eine negtive Zahl.
6	$JUMP_{<}$ -2	Wenn B1 = 0, dann weiter abfragen
7	ADD IN1 1	Daten aus R1 in IN1 laden
8	SUBI 2 10	Damit B1 = 1 zu B1 = 0 im R2 wird.
9	JUMP -3	Zurück zum abfragen
10	MULTI IN1 100000000	Eine Zahl mit 100 Mio. multipliziert simuliert Linksshift.
11	SUBI IN2 1	Zähler um eins verringern
12	LOAD ACC IN2	
13	$JUMP_{=}$ -4	Wenn es nicht der 4te Durchlauf war, den Code aus a) durchlaufen.
14	LOADI c a	a in Speicherzelle c laden damit man da 1 drauf addieren kann für die nächste Z
15	ADD IN1 s	10^{32} aus s auf IN1 addieren damit der Prafix 01 zu 11 wird für DS.
16	STOREIN c IN1 0	Speichern bei Adresse c.
17	ADDI c 1	Damit wir das nächste aus IN1 im darauf folgenden c speichern können.
18	LOAD ACC 2	Statusabchecken.
19	SUBI ACC $110^{27}001$	Abchecken von B0. Wenn Null raus kommt ist B0 = 1 und bei B0 = 0 ist es neg
20	$JUMP_{<} - 18$	Bei B0 = 0 springe zurück auf PC = 2
21	LOAD ACC t	Ende

Aufgabe 2

Man kann MOVE benutzen, da wir später nicht nochmal auf die Informationen zugreifen, sondern sie nur hin und her schieben müssen.

leider nicht >_< 0/6