Rechnerarchitektur - Übungsblatt 9
Übungsgruppe 8 - Di, 15-17 Uhr
Steffen Heil

Jan Hendrik Dithmar (Matrikelnummer 2031259)
Carsten Saar (Matrikelnummer 2030107)
Thomas Ospelt (Matrikelnummer 2029959)
Timo Adam (Matrikelnummer 2031946)

Aufgabe 1

Die Aufgabe war nicht ganz schlüssig gestellt. Es fehlte die Angabe, ob Worte, Halbworte oder Bytes bearbeitet bzw. untersucht werden sollen. Aufgrund der Tatsache, dass die letzte Adresse 32767 weder Wort- noch Halbwort-aligned ist (und somit weder ein Wort- noch Halbwort-Zugriff auf diese Adresse möglich ist), habe ich vorausgesetzt, dass Bytes betrachtet werden sollen. Für das Resultat des Programms aus Aufgabenteil (b) habe ich einen Wort-Zugriff angenommen, da dann das Resultat direkt vor dem untersuchten Speicherbereich steht.

(a) In Speicherzellen 1024 bis 32767 (Byte) jeweils die Adresse modulo 2^8 hineinschreiben.

```
R1 = R0 + 1024                  // lade Register R1 mit Konstante 1024
                                // Laufvariable für die zu bearbeitende Adresse
R2 = (R1 > 32767 ? 1 : 0)       // WHILE: Schleife abbrechen, wenn
                                // Adresse hinter zu bearbeitendem Bereich liegt
PC = PC + (R2 ≠ 0 ? 16 : 4)     // wenn ja, springe aus Schleife nach DONE
M1(R1 + 0) = R1                 // untere 8 Bit der Adresse (Adresse modulo 2^8) schreiben
                                // obere 24 Bit werde einfach abgeschnitten
R1 = R1 + 1                     // Laufvariable mit Adresse um 1 erhöhen
PC = PC - 16                    // unbedingter Sprung nach WHILE
                                // DONE: Programm beendet
```

(b) Prüfen, ob in den Speicherzellen 1024 bis 32767 (Byte) jeweils die Adresse modulo 2^8 steht. Wenn ja, in Speicherzelle 1020 (Wort) den Wert 1 ablegen, ansonsten den Wert 0.

```
M4(R0 + 1020) = R0              // initialisiere Ergebnis mit 0
                                // Annahme „gescheitert“, bis alle Speicherzellen geprüft
R1 = R0 + 1024                  // lade Register R1 mit Konstante 1024
                                // Laufvariable für die zu prüfende Adresse
R2 = M1(R1 + 0)                 // LOOP: Speicherzelle an Adresse R1 nach R2 laden
                                // Achtung: R2 enthält den Wert mit sign extension
R2 = R2 ∧ 255                   // obere 24 Bits ausmaskieren (sign extension aufheben)
R3 = R1 ∧ 255                   // Adresse modulo 2^8 in R3, Soll-Wert
R2 = (R2 = R3 ? 1 : 0)          // vergleiche Ist- mit Soll-Wert
PC = PC + (R2 = 0 ? 24 : 4)     // wenn ungleich, Schleife abbrechen (springe zu DONE)
                                // in diesem Fall bleibt Ergebnis 0 (siehe oben) unverändert
R1 = R1 + 1                     // Laufvariable mit Adresse um 1 erhöhen
R2 = (R1 > 32767 ? 1 : 0)       // liegt nächste Adresse hinter zu prüfendem Bereich?
PC = PC + (R2 = 0 ? -28 : 4)    // wenn nein, zu Beginn der Schleife (LOOP) springen
R1 = R0 + 1                     // Wert 1 in R1 laden
M4(R0 + 1020) = R1              // Ergebnis 1 in die geforderte Speicherzelle schreiben
                                // DONE: Programm beendet,
                                // Ergebnis in Speicherzelle 1020 (Wort)
```

Aufgabe 2

(a) siehe Anhang

(b) Sei $z = \langle z_1\ z_0 \rangle$ der alte Zustand, $z_0, z_1, \text{up} \in \{0, 1\}$. Der neue Zustand z' ergibt sich wie folgt

$$z' = \langle z'_1\ z'_0 \rangle \quad \text{mit} \quad z'_1 = z_0 \oplus z_1 \oplus (\sim \text{up}) \quad \text{und} \quad z'_0 = \sim z_0$$

(c) Sei $z = \langle z_1\ z_0 \rangle$ der aktuelle Zustand des Automaten, $z_0, z_1, \text{output} \in \{0, 1\}$. Das Ausgangssignal $\langle x \rangle$ ergibt sich wie folgt

$$\langle x \rangle = \langle (z_1 \wedge \text{output}), (z_0 \wedge \text{output}) \rangle$$