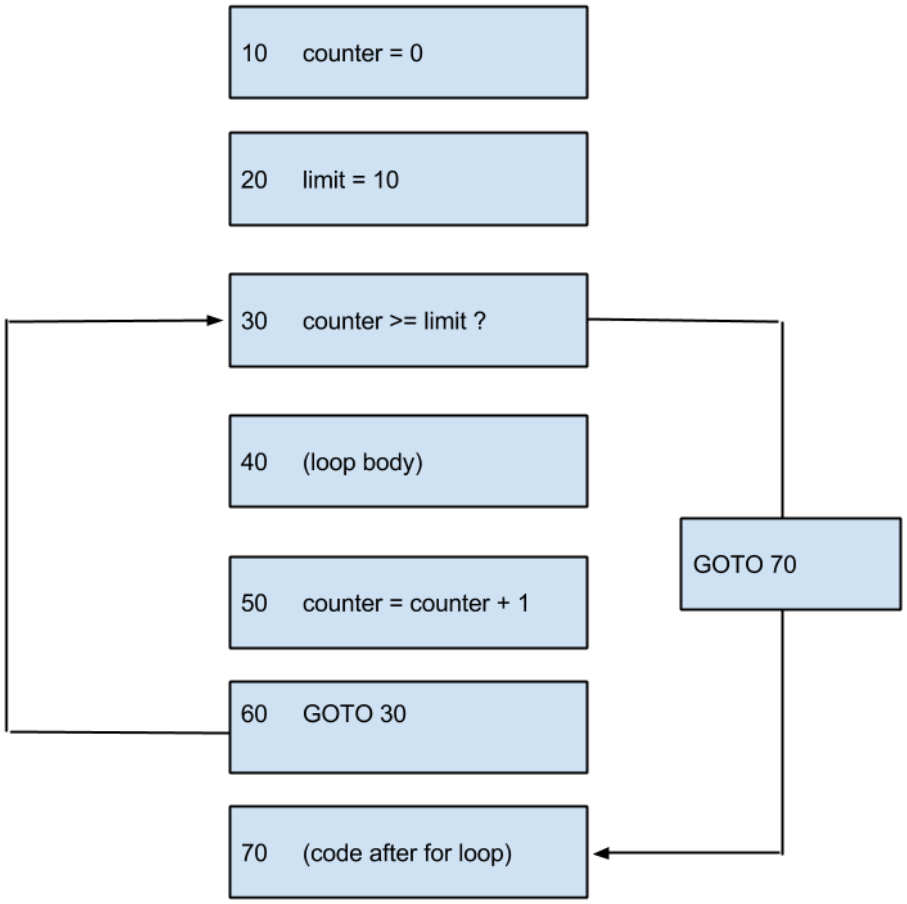
Constantin Lazari, Marco Wettstein 21. Oktober 2013

1. Kontrollstrukturen, wie z. B. Schleifen in höheren Programmiersprachen, werden in Maschinensprache über Sprungbefehle realisiert.

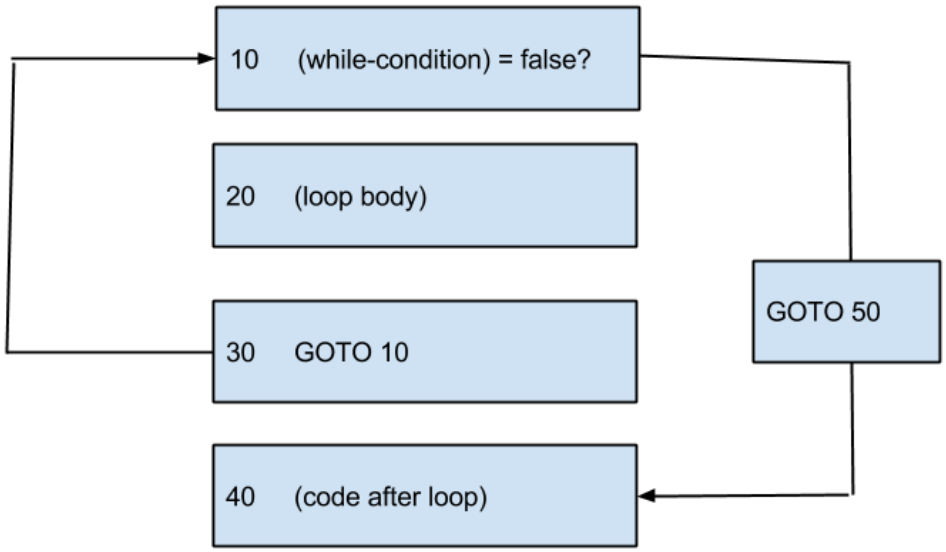
   (a) Skizzieren Sie graphisch (schematisch), wie eine For- und eine While-Schleife, die 10-mal ($n \geq 0$) durchlaufen wird, umgesetzt werden könnte.

Lösung:

1. For-Schleife

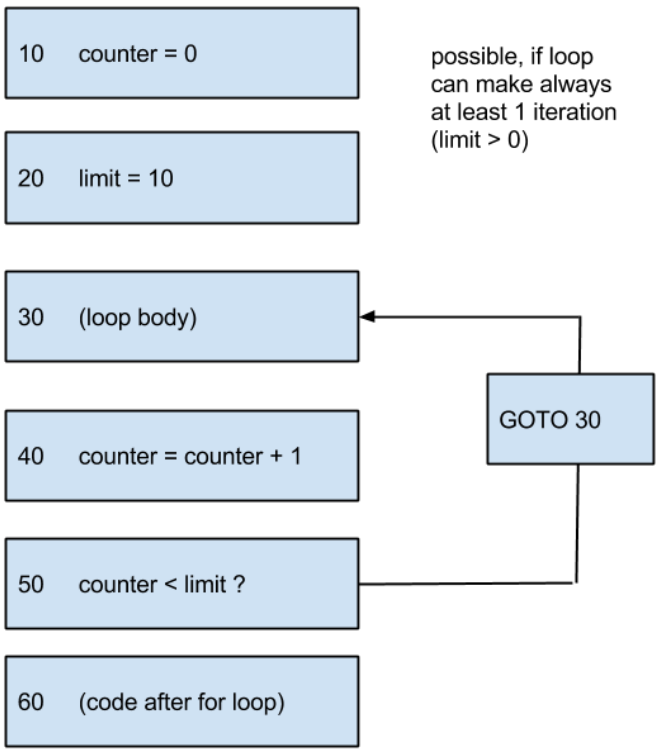


2. While-Schleife



(b) Mit welcher Einschränkung ist es möglich, die FOR-Schleife mit nur einem Sprungbefehl zu realisieren? Skizzieren Sie auch diesen Fall.

Lösung:



2. Das Steuerwerk eines Rechners dekodiert die Befehle aus dem OP-Code bzw. dem Maschinencode; für Benutzer sind Befehle mit mnemonische Symbolen leichter zu lesen (und schreiben). Gehen Sie vom Befehlssatz für den „Mini-Power-PC“ aus.

   (a) Geben Sie für die folgenden Befehle mit mnemonische Symbolen den Maschinencode an:

   - LWDD 1, #240
   - ADDD #62
   - Not
   - BCD #15

   Lösung:

   - LWDD 1, #240 $\mapsto$ 01000101 10100100
   - ADDD #62 $\mapsto$ 10000000 00111110
   - Not $\mapsto$ 00000000 10000000
   - BCD #15 $\mapsto$ 00111000 00001111

   (b) „Dekodieren“ Sie die folgenden Befehle in Maschinencode so weit es möglich ist (mnemonische Symbol und Beschreibung):

   - 00011111 11101111
   - 01011010 00000000
   - 00001011 00011010
   - 01100001 11110110

   Lösung:

   - 00011111 11101111 $\mapsto$ BC R3; (Falls Carry-Flag gesetzt, verzweige zur Adresse von Register 3)
   - 01011010 00000000 $\mapsto$ LWDD R3, #512; (Lade den Wert an Adresse 512 in das Register 3)
   - 00001011 00011010 $\mapsto$ OR R2; (Führe eine OR-Verknüpfung zwischen dem Akku und Register 2 aus)
   - 01100001 11110110 $\mapsto$ SWDD R0, #502; (Speichere den Wert aus Register 0 (=Akku) an die Speicheradresse 502)

3. Der Befehlssatz für den „Mini-Power-PC“ ist sehr klein / eingeschränkt. Sie haben gelernt, dass Zugriffe auf den Arbeitsspeicher sehr langsam sind (zum Teil deutlich mehr als 100 Zyklen).

   (a) Welchen Befehlstyp würden Sie auf jeden Fall ergänzen, um die Code-Ausführung erheblich zu beschleunigen?

   Lösung:
   Befehle, um häufig benötigte Werte temporär zwischenzulagern, also z. B. in ein unbenutztes Register zu verschieben (Register $a \rightarrow$ Register b).

   (b) Kann mit dem Befehlssatz ein Arbeitsspeicher von 16 KiB genutzt werden? Antwort bitte begründen.

   Lösung:
   Nein, den Befehlen `LWDD` und `SWDD` stehen nur 10 Bit zur Adressierung des Speichers zur Verfügung. 10 Bit erlauben als nur 1 024 Speicheradressen.
   Mittels indirekter Adressierung (z.B. via eines 16-Bit Registers), wäre es möglich. Der Befehle müsste dann allerdings von der Form LWDD Rin, Rout sein. Gleiches gilt für SWDD.

4. Gegeben sei der Befehlssatz für den „Mini-Power-PC“. Die Aufgabe Summe $= a + 4 * b + 8 * c$ soll über ein Programm für den „Mini-Power-PC“ berechnet werden.

   (a) Schreiben Sie den Programm-Code mit mnemonische Symbolen (in Assembler)

   Lösung:
   100 LWDD R0 #202; Load b into Akku
   101 SLA; Multiply by 2
   102 BCD 119; Jump to end, on overflow
   103 SLA; Multiply by 2 – So multiplied by 4
   104 BCD 119; Jump to end on overflow
   105 LWDD R1 #200; Load a into Akku
   106 ADD R1; Add a to 4 * b
   107 SWDD R0 #206; Store result as d
   108 LWDD R0 #202; Load c
   109 SLA; Multiply by 2
   110 BCD 119; Jump to end on overflow
   111 SLA; Multiply by 2 – So multiplied by 4
   112 BCD 119; Jump to end on overflow
   113 SLA; Multiply by 2 – So multiplied by 8
   114 BCD 119; Jump to end on overflow
   115 LWDD R1 #206; Load d in Register 1
   116 ADD R1; Add d to 8 * c
   117 BCD 118; Jump to end on overflow
   118 SWDD R0 #206; Write result to 206
   119 END; Program finishes

(b) Übersetzen Sie den Code in Maschinencode

Lösung:
Das würde der „Mini-Power-PC“ machen, er ist aber noch nicht ganz fertig.

(c) Berechnen Sie mit Hilfe Ihres Programms:

- Summe für $a = 14, b = 7$ und $c = 66$
- Summe für $a = 25, b = -14$ und $c = -123$
- Summe für $a = -125, b = 10\,000$ und $c = 16$
- Summe für $a = 1\,000, b = 10\,000$ und $c = -2\,000$

Anmerkungen:

- Das Programm beginnt in der Speicherzelle 100, die Variablen a, b und c liegen in den Speicherzellen 200 / 201 (für a), 202 / 203 (für b) und 204 / 205 (für c).
- Am Ende des Programm soll die Summe in der Speicherzelle 206 / 207 stehen. Bei einem Überlauf soll das Programm abgebrochen werden.
- Sie können selbstverständlich den Emulator der Aufgabenserie 3b für die Verifizierung und Berechnung nutzen - er könnte auch als „Debugger“ sehr hilfreich sein ...

Lösung:
Summe würde der „Mini-Power-PC“ berechnen, er ist aber noch nicht ganz fertig.

- Summe für $a = 14, b = 7$ und $c = 66$
- Summe für $a = 25, b = -14$ und $c = -123$
- Summe für $a = -125, b = 10\,000$ und $c = 16$
- Summe für $a = 1\,000, b = 10\,000$ und $c = -2\,000$