Technische Universität München

Fakultät für Informatik

Rechnerarchitektur - Praktikum (IN0005)

Gruppe 12

Projekt 2 - Spezifikation

Verfasser

Vasil Sarafov (vasil.sarafov@tum.de)
Franz Fuchs (franz.fuchs@tum.de)
Yoav Schneider (yoav.schneider@tum.de)

Tutor

Artur Faltenberg (artur.faltenberg@tum.de)

5. Juni, 2016

Sommersemester 16

Inhalt

1 Einleitung

Im Rahmen des ERA-Praktikums an der TU München sollen zwei Projekte in Dreier-Teams ausgearbeitet werden. Unsere Gruppe muss die folgenden Aufgaben bearbeiten:

- Assembler Realisierung der Funktion $y = arcsin(x)$
- *strcpy* (Kopieren von Zeichenketten) als Mikro- und Assemblerprogramm umsetzen

Im folgenden soll die Aufgabe zur Mikroprogrammierung in Ausführung einer Spezifikation beschrieben werden. Dabei werden zwei Lösungsansätze für das Thema vorgestellt, auf ihre Tauglichkeit überprüft und verglichen. Anschließend wird dann ein Ansatz ausgewählt, der für die Implementierung verwendet wird.

2 Aufgabenkurzbeschreibung

In diesem Projekt soll die Funktion *strcpy* verwirklicht werden, bei der es um das Kopieren von Zeichenketten geht. Dabei soll der Befehl strcpy einmal direkt als Mikroprogramm und einmal als Maschinenprogramm implementiert werden. Beide Implementierungen sollen anhand ihrer Effizienz verglichen werden.

3 Analyse der Funktion strcpy

Die Laufzeit des Algorithmus kann nicht in einer geringeren Laufzeit als $\Theta(n)$ sein, wobei n gleich die Länge des zu kopierenden Textes ist, denn jedes Symbol aus der Zeichenkette muss mindestens einmal ausgelesen werden.

Außerdem kennt man die Länge des Strings nicht, weil er als ein $C - String$ gespeichert ist, wobei das Ende mit einem Nullsymbol angegeben wird. Dabei ist zu beachten, dass die zu kopierenden Zeichen als 16bit Unicode Symbole gespeichert sind. Das bedeutet, dass sie genau in einem Takt ausgelesen werden können, da die MI-Maschine über 16 Bit große Register, Speicherzellen und Busse verfügt.

Die *strcpy* Funktion wird als Maschinenprogramm und als Mikroprogramm realisiert.

Für die Realisierung als Maschinenprogramm, kann man z.B davon ausgehen, dass die Anfangs- und Zieladressen in zwei spezifizierten Registern bereits angelegt worden sind.

Für die Realisierung als Mikroprogramm, muss man zunächst das Format definieren. Eine Möglichkeit wäre, die Anfangs- und die Zieladresse in verschiedenen Registern anzulegen und dann die Nummer der Register im Befehl zu übergeben.

$$strcpy \quad RA \quad RB \tag{1}$$

Hierbei ist RA die Anfangsadresse und RB die Zieladresse.

4 Lösungsvorschläge

Bei dieser Aufgabenstellung springt einem eine iterative Lösung förmlich ins Auge, wenn man sich den String Zeichen für Zeichen anschaut. Lediglich das mit einem Nullzeichen terminierte Ende kennt man nicht. Deswegen haben wir zwei Lösungsvorschläge ausgearbeitet, die jeweils die Aufgabenstellung erfüllen, aber die unterschiedliche Herangehensweisen bezüglich des Endes aufweisen.

4.1 Direktes Kopieren

Dieser Lösungsansatz basiert auf die Idee, dass man jedes Symbol direkt nach seinem Auslesen kopieren kann. Deshalb werden Anfangs- und Zieladresse in zwei Zählern gespeichert und schrittweise inkrementiert bis das letzte Symbol gefunden ist. Der Pseudocode unseres Algorithmus zu diesem Vorschlag sieht folgendermaßen aus:

Algorithm 1 Unicode Zeichenkette Kopieren als MI-Maschinenenprogramm

```
1:  function STRCPY(ra, rb)
2:      ri ← ra                          ▷ Anfangsadresse
3:      rz ← rb                          ▷ Zieladresse
4:      rc ← load(ra)                    ▷ Zu kopierendes Symbol
5:      while rc ≠ 0 do
6:          store(rz, rc)
7:          ri ← ri + 1
8:          rz ← rz + 1
9:          rc ← load(ri)
10:     end while
11:     store(rz, rc)                    ▷ Füge die Null hinzu
12: end function
```

4.2 Invertiertes Kopieren

Dieser Lösungsansatz findet zuerst das Ende der Zeichenkette und kopiert dann in der invertierten Reihenfolge. Anschließend geht man wieder den String Zeichen für Zeichen durch und kopiert die einzelnen Zeichen. Dieser Algorithmus ist also ziemlich ähnlich dem ersten. Der Pseudocode des von uns vorgeschlagenen Algorithmus zur Lösung der Aufgabe sieht folgendermaßen aus:

Algorithm 2 Unicode Zeichenkette Kopieren als MI-Maschinenenprogramm

```
1:  function STRCPY(ra, rb)
2:      ri ← ra                                  ▷ Anfangsadresse
3:      rc ← load(ri)
4:      while rc ≠ 0 do                          ▷ Finde das Ende
5:          ri ← ri + 1
6:          rc ← load(ri)
7:      end while
8:      rl ← ri − ra                             ▷ Länge der Zeichenkette
9:      rz ← rb
10:     rz ← rz + rl                             ▷ Ende der Zielzeichenkette
11:     while ri > ra do
12:         store(rz, rc)
13:         ri ← ri − 1
14:         rz ← rz − 1
15:         rc ← load(ri)
16:     end while
17:     store(rz, rc)                            ▷ Kopiert das erste Symbol
18: end function
```

5 Bewertung der Ansätze und Auswahl der Implementierung

	DirektesKopieren	*InvertiertesKopieren*
Schwierigkeitsgrad	niedrig	hoch
Effizienz	hoch	niedrig

Da das direkte Kopieren lediglich einen Schleifendurchlauf benötigt, ist es offensichtlich besser geeignet für diese Funktion. Man könnte sich allerdings vorstellen, dass das indirekte Kopieren gewisse Vorteile mit sich bringt, wenn fehlerhafte Eingaben zu erwarten wären. Da die Effizienz wieder im Vordergrund stand, haben wir uns für das direkte Kopieren entschieden, sowohl für die Maschinenprogramm als auch für die Implementierung des Mikroprogramm-Befehls.

6 Verfasser

Die Spezifikation der Mikroprogrammieraufgabe zum *strcpy* wurde von Franz Fuchs, Yoav Schneider und Vasil Sarafov verfasst.