# Rechnerstrukturen Hausaufgaben zum 26. Oktober 2016

Ali Ebrahimi Pourasad, Moritz Lahann, Matz Radloff

26. Oktober 2016

## Inhaltsverzeichnis

**1** **1**
1.1 . . . 1
1.1.1 (a) Interpreter . . . 1
1.1.2 (b) Compiler . . . 1
1.1.3 (c) Virtuelle Maschine . . . 1
1.2 . . . 1
1.3 . . . 2
1.4 . . . 2
1.4.1 (a) . . . 2
1.4.2 (b) . . . 3
1.5 . . . 3
1.5.1 (a) . . . 3
1.5.2 (b) . . . 4
1.5.3 (c) . . . 4
1.5.4 (d) . . . 5

# 1

## 1.1

### 1.1.1 (a) Interpreter

Ein Interpreter übersetzt einzelne Befehle einer höheren Programmiersprache in eine Befehlsfolge, die direkt auf einer niedrigeren Spracheebene ausgeführt wird. Z.B. könnte ein Interpreter Anweisungung einer Sprache L1 sequentiell nach L0 übersetzen, sodass diese direkt ausgeführt werden können.

### 1.1.2 (b) Compiler

Ein Compiler übersetzt auch Befehle einer höheren in Befehlsfolgen einer niedrigeren Sprache. Allerdings wird dabei ein neues Programm erzeugt, das erst nach dem Generieren ausgeführt werden kann.

### 1.1.3 (c) Virtuelle Maschine

Auf der untersten Sprachebene L0 können Anweisungen direkt durch elektronische Schaltungen ausgeführt werden. Um einem Interpreter zu simulieren, dass auch höhersprachige Befehle ausgeführt werden können (um eine direkte Übersetzung von z.B. L5 zu L0 zu vermeiden) kann eine virtuelle Maschine eingesetzt werden, da diese die Rechenarchitektur simuliert. Dadurch können neue, komplexere und besser verständliche Sprachen eingesetzt werden, die jeweils eine Abstraktionseben darstellen. So ist es möglich, auch ohne Verständnis der untersten Schichten, in höheren Sprachen zu programmieren. Der größte Nachteil ist dabei, dass eine virtuelle Maschine meistens langsamer ist, als es eine echte. Auch die unterste Maschine M0 kann eine virtuelle Maschine sein (z.B. Betriebssystem-Virtualisierung). Soft- und Hardware sind in diesem Fall äquivalent anzusehen.

## 1.2

$$k_0 = k \tag{1}$$
$$k_1 = nk \tag{2}$$
$$k_2 = n^2 k \tag{3}$$
$$k_3 = n^3 k \tag{4}$$

Für den allgemeinen Fall erhält man:

$$k_i = n^i k \tag{5}$$

Ali Ebrahimi Pourasad(6948107)
Moritz Lahann(6948050)
Matz Radloff(6946325)

## 1.3

Die Vorteile dieses Konzepts liegen in der Dynamik und Flexibilität des Computers. Programme müssen diesem nicht getrennt von anderen Daten zugeführt werden. Durch die rein logische Trennung vereinfacht sich der Aufbau stark. Da sich Programme dadurch auch selbst verändern können, ist es mögliche diese sehr schnell und wenig speicherbedürftig zu entwickeln.

Nachteile liegen vor allem im Bereich der Sicherheit, die bei dem Aufkommen des von-Neumann Rechners noch nicht ersichtlich waren. So könnte eine Schadsoftware sich selbst oder andere Programme umschreiben und diese so unbrauchbar machen bzw. deren Funktionalität verändern. Um zu gewährleisten, dass bösartige Programme Daten nicht manipulieren, auslesen oder anderweitig Schaden anrichten, muss eine Kontrollebene die Ausführung sowie Zugriffe auf die korrekten Speicherbereiche der Programme kontrollieren (Betriebssystem). Trotz dieser Gegenmaßnahmen schafft es Software immer wieder Sicherheitslücken zu finden und aktiv auszunutzen.

## 1.4

### 1.4.1 (a)

$f(x) = (ax^5 + bx^4 + cx^3 + dx^2 + ex + f)$
$\rightarrow (6 + 5 + 4 + 3 + 2) \cdot 5\,\text{ns} + (5 \cdot 1\,\text{ns}) = 105\,\text{ns}$

Ohne Horner-Schema würde bei naiver Herangehensweise die Berechnung des Polynoms 105ns dauern. Bei der Berechnung der Potenzen $x^k, k \geq 2$, kann man zuerst die niedrigen und dann die höheren Potenzen berechnen. Damit macht man sich jeweils zunutze, dass $x^{k-1}$ schon berechnet ist, wenn $x^k$ gebraucht wird. Für $x^k$ braucht man daher nur eine weitere Multiplikation und nicht deren $(k-1)$. Laut wikipedia sind es also nur eine Multiplikation pro Potenz, da die vorherigen aus dem Speicher ausgelesen werden können[1]. Nach der Methode wären es $(9 \cdot 5\,\text{ns} + 5\,\text{ns}) = 50\,\text{ns}$ für die Multiplikationen.

Mit Horner-Schema, bei dem das Ausklammern einzelner $x$ ausgenutzt wurd, erhält man insgesamt:
$y = f + x(e + x(d + x(c + x(b + a \cdot x)))) \rightarrow 30\,\text{ns}$

### 1.4.2 (b)

**Gegeben:**

$$y = (x^2 + 2 \cdot x + 1)^{11} \tag{6}$$

**Rechnung:** Vereinfachung mit Ausnutzung der 1. binomischen Formel:

$$a = (x^2 + 2x + 1) = (x + 1)^2 \tag{7}$$
$$b = (x + 1) \tag{8}$$

→ Eine Addition

$$c = (b)^2 = (x + 1) \cdot (x + 1) \tag{9}$$

→ Eine Multiplikation und nur eine Addition, da b nur einmal ausgerechnet werden muss

$$y = d = (c)^{11} \tag{10}$$
$$t = 12 \cdot 5ns + 1 \cdot 1ns = 61\,\text{ns} \tag{11}$$

**Antwort:** Eine Addition und 12 Multiplikationen benötigen eine Rechenzeit von 61 ns.

## 1.5

Gegeben ist eine mittlere Datenrate von $5\,\text{MB/s}$.

### 1.5.1 (a)

**Rechnung:**

$$5\,\text{MB/s} \cdot 60 = 300\,\text{MB/min} \tag{12}$$
$$30\,\text{MB/min} \cdot 60 = 18.000\,\text{MB/h} \tag{13}$$
$$18.000\,\text{MB/h} \cdot 24 = 432.000\,\text{MB/d} \tag{14}$$
$$432.000\,\text{MB/d} \cdot 365 = 157.680.000\,\text{MB/a} \tag{15}$$
$$157.680.000\,\text{MB/a} \cdot 80 = 12.614.400.000\,\text{MB/Leben(80Jahre)} \tag{16}$$

**Antwort:**

- Pro Tag werden 432.000 MB (430 GB) abgespeichert.
- Pro Jahr werden 157.680.000 MB (157,68 TB) abgespeichert.
- Pro Leben werden 12.614.400.000 MB (12614,4 TB) abgespeichert.

### 1.5.2 (b)

**Gegeben:**

- Kapazität Anfang 2016 = 4 TiB
- Kapazitätzuwachs von 38% pro Jahr

**Gesucht:**

- Kapazität im Jahr $t = 12.614.400.000\,\text{MB} \approx 11472,73\,\text{TiB}$

**Rechnung:** $t$: Zeit in Jahren, $f(t)$: Kapazität im Jahr $t$

$$f(t) = 4\,\text{TiB} \cdot 1,38^t \tag{17}$$
$$4\,\text{TiB} \cdot 1,38^t = 11472,73\,\text{TiB} \tag{18}$$
$$t = \log_{1,38}(\frac{11472,73\,\text{TiB}}{4\,\text{TiB}}) \tag{19}$$
$$= \log_{1,38}(2868,1825) \tag{20}$$
$$= 24,72 \tag{21}$$

**Antwort:** Nach ca. $24\frac{3}{4}$ Jahren, also in 2040 (bzw. Anfang des Jahres 2041) ist die Kapazität einer typischen Festplatte, groß genug um ein ganzes Leben (80 Jahre) aufzuzeichnen.

### 1.5.3 (c)

**Gegeben:**

- Kapazität Anfang 2016 = 128 GiB
- Kapazitätzuwachs = 50% pro Jahr

**Gesucht:** Kapazität im Jahr $t = 12.614.400.000\,\text{MB} \approx 11748075,5\,\text{GiB}$

**Rechnung:** $t$: Zeit in Jahren, $f(t)$: Kapazität im Jahr $t$

$$f(t) = 128\,\text{GiB} \cdot 1,5^t \tag{22}$$
$$128\,\text{GiB} \cdot 1,5^t = 11748075,5\,\text{GiB} \tag{23}$$
$$t = log1,5(\frac{11748075,5\,\text{GiB}}{128\,\text{GiB}}) \tag{24}$$
$$= log1,5(91781,83984) \tag{25}$$
$$= 28,18 \tag{26}$$
$$\tag{27}$$

**Antwort:** Nach ca. 28 Jahren, also 2044 (bzw. Anfang 2045) ist die Kapazität einer typischen SD-Card, groß genug um ein ganzes Leben (80 Jahre) aufzuzeichnen.

### 1.5.4 (d)

**Gegeben:** Magnetband Kapazität: 140 MB

**Gesucht:** Anzahl der Magnetbänder, um 12.614.400.000 MB zu speichern.

**Rechnung:**

$$140\,\text{MB} = 12.614.400.000\,\text{MB} \tag{28}$$

$$x = \frac{12.614.400.000\,\text{MB}}{140\,\text{MB}} \tag{29}$$

$$x = 90.102.857,14 \tag{30}$$

**Antwort:** Man bräuchte 90.102.858 Magnetbänder, um eine Datenmenge von 12.614.400.000 MB zu speichern.

## Literatur

[1] Wikipedia, “Horner-schema — wikipedia, die freie enzyklopädie,” 2016. [Online; Stand 25. Oktober 2016].