Programmierkurs

Vorlesung 01

M.Sc. Laslo Hunhold

Department Mathematik/Informatik
Abteilung Informatik
Universität zu Köln

9. November 2020

Letzte Vorlesung

- Begriff der Informatik
- Programmierablauf
- Programmübersetzung (Compiler, Interpreter, Quellcode, Maschinencode, Bytecode)
- Java-„Hallo Welt“-Programm und -Programmstruktur

Daten im Computer

- Binär (0 oder 1)
- 1 Ziffer = 1 bit („binary digit“), 8 bit = 8b = 1 Byte = 1B
- 1 Speicherzelle hält 1 bit
- Typen: Arbeitsspeicher (RAM), CPU-Cache, ROM-Chips, ...

```
01000101011000010111001101110100011001010111 0010
```

- Speicherzellen werden gruppiert
- Kleinste Gruppe von $n > 1$ Speicherzellen = „Wort“
- Jedes Wort hat eine Adresse, kleinste adressierbare Einheit
- n = „Wortbreite“, z.B. n = 8

```
01000101 01100001 01110011 01110100 01100101 01110010
0        1        2        3        4        5
```

- Computer liest und schreibt in Adressen
- Beispiel: Schreibe 00000000 00000000 (16 bit = 2 Byte) in Adresse 3

```
01000101 01100001 01110011 00000000 00000000 01110010
0        1        2        3        4        5
```

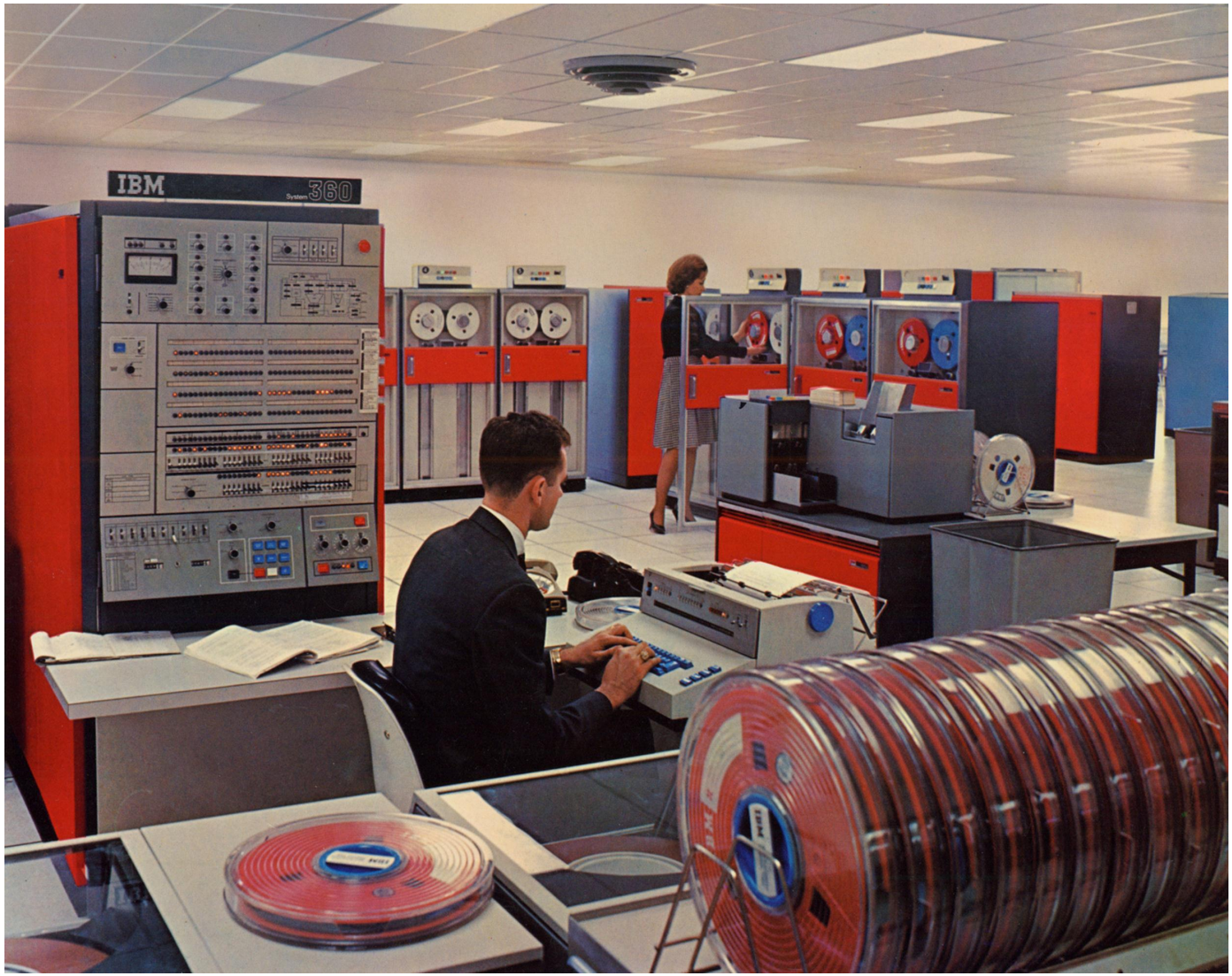

Abbildung: IBM System 360 (1964), Magnetbänder im Vordergrund (Quelle: IBM100 Archive)

Binäre Kodierungen

Bitgruppen Bedeutung geben

Unsigned Integer

- Schulwissen: Basis-2-Darstellung natürlicher Zahlen

$$26 = 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^1 = 11010_2 = 00011010_2$$

- Trage Stellen rechtsbündig in Bitgruppe von n bit ein, z.B. $n = 8$:

  00011010 entspricht der Zahl 26
- Wertebereich: 0 (00...00) bis $2^n - 1$ (11...11)
- Ganzzahlüberlauf: Sei $n = 8$. Der Wertebereich ist 0 bis 255.
  Was ist 11111110 + 00000100 ($254 + 4 = 258$)? 0000010 (2)
  $\rightarrow$ Ergebnis ist Rest der Division durch $2^n = 256$
- Bezeichnung: uintn, z.B. uint8, uint16, ...

Was ist mit negativen Zahlen?

Signed Integer (Zweierkomplement)

- Betrachte wieder Bitgruppe aus n bit
- Idee: Reserviere das höchstwertige bit als „Vorzeichenanzeiger“ (0 entspricht positiv, 1 entspricht negativ)
- Weise Werte wie folgt zu:

100...00	-2^{n-1}
100...01	$-2^{n-1}+1$
⋮	
111...10	$-2^{n-1}+(2^{n-1}-2)=-2$
111...11	$-2^{n-1}+(2^{n-1}-1)=-1$
000...00	0
000...01	1
⋮	
011...11	$2^{n-1}-1$

- Wertebereich: -2^{n-1} bis $2^{n-1}-1$
- Ganzzahlüberlauf: i.A. nicht definiert
- Bezeichnung: intn, z.B. int8, int16, ...

Fließkommazahlen

- Darstellung von Nicht-Ganzzahlen mit Vorzeichen $s \in \{0, 1\}$, Exponent $e \in \mathbb{Z}$ und Mantisse $m \in [1, 2)$ (Fließkommadarstellung):

$$(-1)^s \cdot m \cdot 2^e$$

- Beispiele (mit 10 statt 2, also $m \in [1, 10)$)
  - $-55.4 = (-1)^1 \cdot 5.54 \cdot 10^1$
  - $0.002 \ = (-1)^0 \cdot 2 \cdot 10^{-3}$
- Betrachte Bitgruppe aus n bit der Form `se...em...m` mit

s	Vorzeichenbit (1 bit)
e...e	Exponent (z.B. 8 bit für n=32, 11 bit für n=64)
m...m	Mantisse (z.B. 23 bit für n=32, 52 bit für n=64)

- Details sehr komplex, hier nur eine Skizze
- Genauigkeit: ∼8 Mantissenstellen (n=32), ∼17 Mantissenstellen (n=64)
- Bezeichnung: floatn, z.B. float32, float64, ...

Typen, Variablen und Operatoren

Typen

01000101011000010111001101110100

Ist das eine int32, uint32, float32, etwas anderes?

Lösung: Wir weisen Bitgruppen „Typen“ zu.

10000101 als uint8 repräsentiert 133
10000101 als int8 repräsentiert -123

Datentypen in Java

Elementare Datentypen (8 Stück)

byte	int8
short	int16
int	int32
long	int64
float	float32
double	float64
boolean	„uint1“, meist uint8; `true` oder `false`
char	uint16; für „Zeichen“

Komplexe Datentypen (z.B. Klassen)

- z.B. String
- Variable Länge

Variablen

- Bitgruppen mit Typ und Bezeichnung
- Deklaration: „Reservierung“, Wert wird nicht festgelegt
  `int j;`
- Zuweisung: Befüllung mit einem Wert (= entspricht $\leftarrow$)
  `j = 10;`
  Erste Zuweisung einer Variable heißt Initialisierung.
- Kombination möglich:
  `int i, j = 10;`
  Deklaration von i und j und Initialisierung von j mit 10.

Bezeichnungsregeln

- Bezeichnungen müssen im Sichtbereich („scope“) eindeutig sein
  `int var; String var;` geht nicht
- Erlaubte Zeichen a-z, A-Z, $, _, 0-9 (bis auf das erste Zeichen)
- Unterscheidung von Groß- und Kleinschreibung („case-sensitive“)
- Keine Schlüsselwörter (`public`, `class`, `static`, `void`, `int`, ...)
- Konvention: Klein geschrieben, Deskriptiv, Hilfsvariablen `i,j,k,...`

Zuweisung

```
int i;
i = 1.5;
```

Fehler: „`Type mismatch: cannot convert from double to int`“

- Implizite Typumwandlung bei Zuweisung (z.B. hier `double` zu `int`); „Typverletzung“, da nicht „verlustfrei“ möglich
- Explizite Typumwandlung

```
int i;
i = (int)1.5;
```

Funktioniert, aber mit „Verlust“. `i` hat den Wert `1`

```
double d;
d = 10;
```

Implizite Typumwandlung funktioniert. `d` hat den Wert `10.0`

Operation

- Betrachte „`z = 0;`“. Zuweisung „=“ ist ein Operator und `z` und `0` sind Operanden
- Operator und Operanden, korrekt kombiniert (vgl. Syntax), bilden Ausdrücke

```
int x = 10;
int y = 20;
int z = x + y;
```

- Präzedenz/Auswertungsordnung: Betrachte
  `int i = 10 * (20 + 4) + 30 + 40;`
  - Auswertung nach Assoziativität des Operators (später)
  - Präzedenzliste von Operatoren (später)

Verschiedenes zu Java

Kontrollstrukturen (1/2)

Anweisungen werden sequentiell abgearbeitet, bis

- das Programm endet
- eine Kontrollstruktur erreicht wird

Eine Kontrollstruktur kann Teile des Codes

- überspringen (Bedingung) (`if...else`)
- wiederholen (Schleifen) (`while`, `for`, `do...while`)

```
1  int i = 1, j = 2, z;
2
3  if (i < j) {
4          z = 2;
5  } else {
6          z = 4;
7  }
8  System.out.println("Wert von z ist " + z);
```

Ausdruck in der `if`-Klammer wird implizit in `boolean` umgewandelt

Kontrollstrukturen (2/2)

```
1  int n = 10;
2
3  while (n > 0) {
4          System.out.println(n);
5          n--;
6  }
```

```
1  int n;
2
3  for (n = 10; n > 0; n--) {
4          System.out.println(n);
5  }
```

Beide Listings sind funktionell identisch

Kommentare

- Code-Lesbarkeit sehr wichtig
- Kommentare helfen bei komplexen Code-Stellen und der Strukturierung

„Inline-Kommentar“ (nur aktuelle Zeile auskommentieren)

```
// Kommentar
```

C-Stil (alles zwischen den Tags auskommentieren)

```
/* Kommentar */
```

„Blockkommentar“ (Konvention)

```
/*
 * Lorem ipsum dolor sit amet, consectetur
 * adipiscing elit, sed do eiusmod tempor
 * incididunt ut labore et dolore magna aliqua.
 */
```

Zusammenfassung

- Daten im Computer
- Binäre Kodierungen
- Typen, Variablen und Operatoren
- Kontrollstrukturen

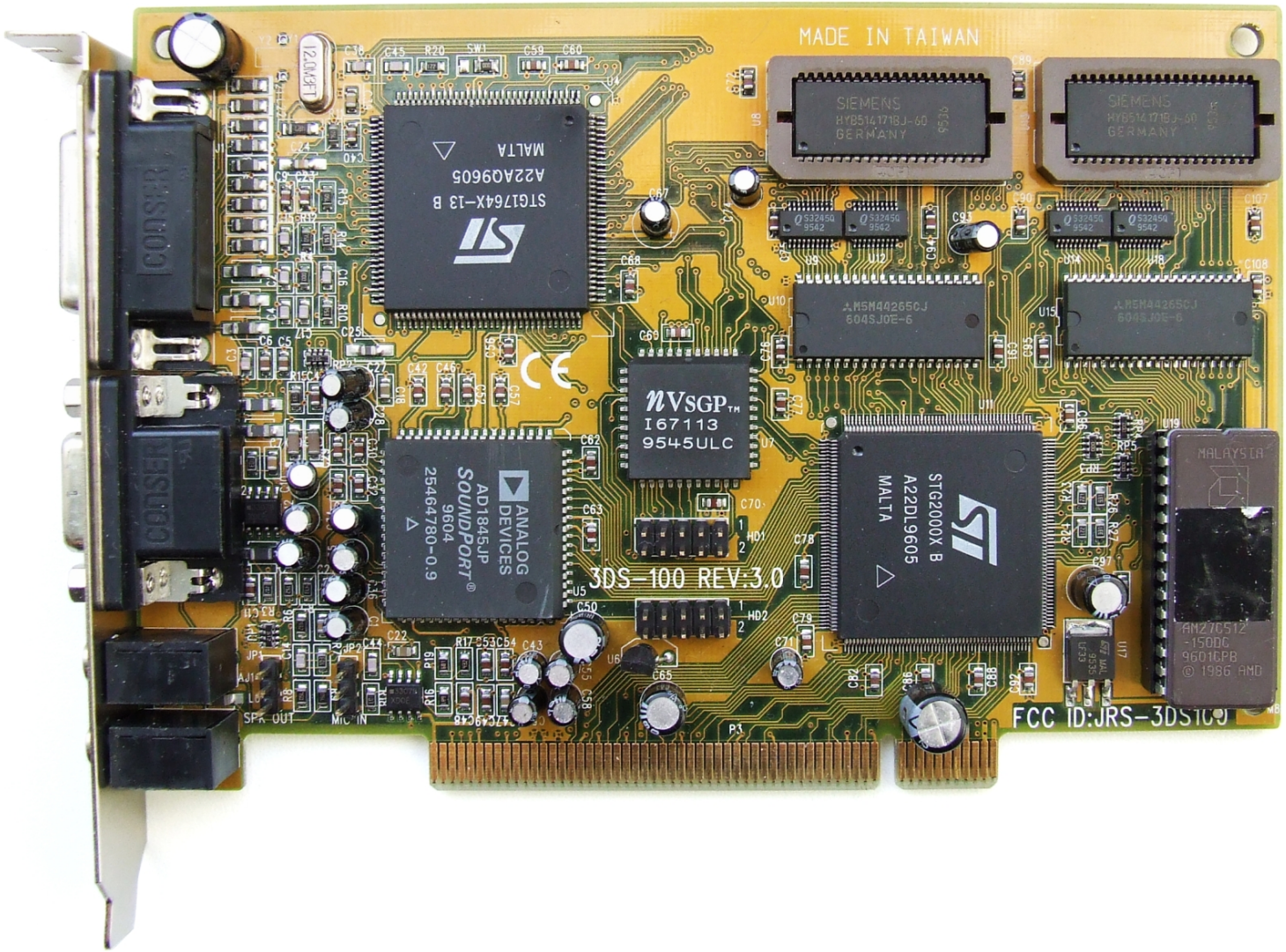

Abbildung: Nvidia NV1 (1995), eine der ersten Beschleunigerkarten (Quelle: Trio3D, Wikimedia Commons)