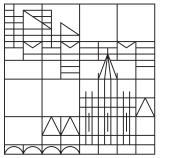
Universität Konstanz



Konzepte der Informatik

Informationscodierung und -speicherung

Barbara Pampel

Universität Konstanz, WiSe 2023/2024

Reelle Zahlen

- Festkommadarstellung
  - feste Aufteilung der k Bits in Vor- und Nachkommabits
  - unflexibel
- Fließ- oder Gleitkommadarstellung
  - flexible Aufteilung in Vorzeichen (s), Mantisse (m) und Exponent (e)
  - $z = s \cdot m \cdot \beta^e$

Standardisierung nach IEEE 754

$z = s \cdot m \cdot \beta^e$

- $\beta = 2$
- Einfache und doppelte Genauigkeit (32 bzw. 64 Bit)
- 1 Bit Vorzeichen: $s = (-1)^V$

Standardisierung nach IEEE 754

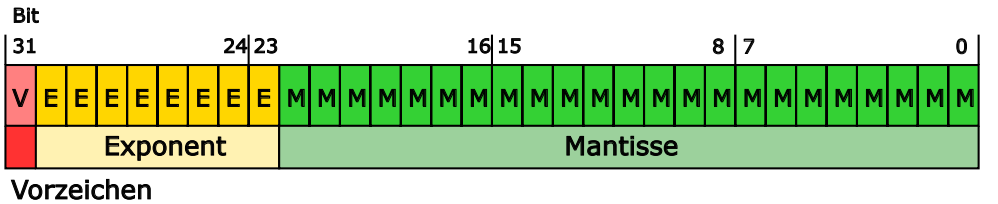
$z = s \cdot m \cdot \beta^e$

- $\beta = 2$
- Einfache und doppelte Genauigkeit (32 bzw. 64 Bit)
- 1 Bit Vorzeichen: $s = (-1)^V$
- 23/52 Bit Mantisse, mit impliziter 1: $m = 1, M$
  - führende 1 wird nicht mitgespeichert
  - Mantisse und Exponent werden entsprechend angepasst (normalisiert)

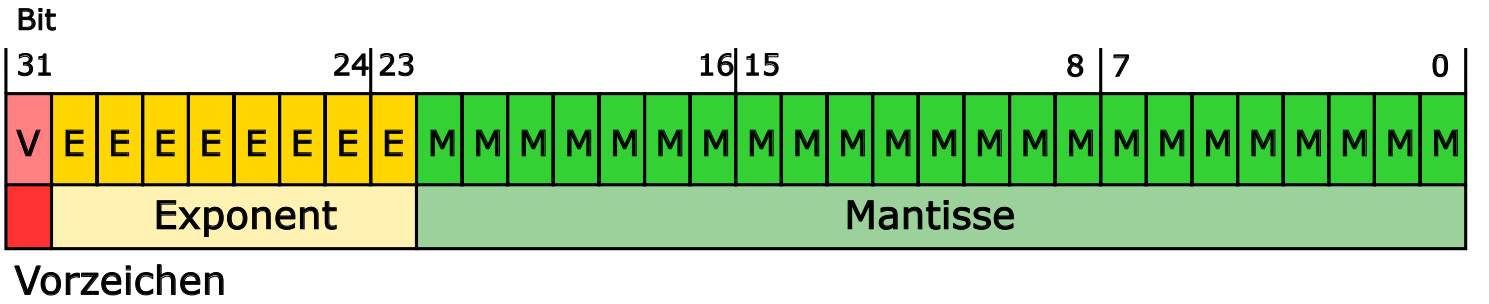
Standardisierung nach IEEE 754

$z = s \cdot m \cdot \beta^e$

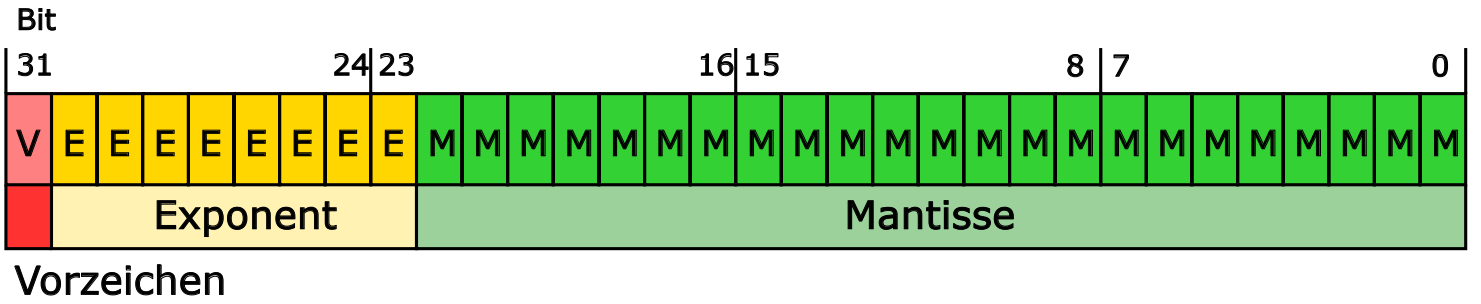
- $\beta = 2$
- Einfache und doppelte Genauigkeit (32 bzw. 64 Bit)
- 1 Bit Vorzeichen: $s = (-1)^V$
- 23/52 Bit Mantisse, mit impliziter 1: $m = 1, M$
  - führende 1 wird nicht mitgespeichert
  - Mantisse und Exponent werden entsprechend angepasst (normalisiert)
- 8/11 Bit Exponent mit Bias: $e = E - B$
  - Bias erlaubt vorzeichenlose Speicherung von negativen Exponenten
  - $B = 127$ bzw. $B = 1023$



IEEE 754


Bit
31
24
23
16
15
8
7
0
V
E E E E E E E E
M M M M M M M M M M M M M M M M M M M M M M M M
Exponent
Mantisse
Vorzeichen

IEEE 754



- Spezielle Darstellungen für
  - 0: Exponent und Mantisse beide 0; es gibt positive und negative 0!
  - ∞: Exponent E mit lauter 1 gefüllt, Mantisse $M = 0$
  - NaN (not a number): Exponent E mit lauter 1 gefüllt, Mantisse $M \neq 0$
- Wertebereiche
  - einfache Genauigkeit $1,4 \cdot 10^{-45} \leq |z| \leq 3,4 \cdot 10^{38}$
  - doppelte Genauigkeit $4,9 \cdot 10^{-324} \leq |z| \leq 1,7 \cdot 10^{308}$

Berechnung der IEEE 754-Darstellung

Beispiel

$18,625_{10}$ in einfacher Genauigkeit

Berechnung der IEEE 754-Darstellung

Beispiel

$18,625_{10}$ in einfacher Genauigkeit

- Bias $B = 127 = 2^7 - 1$
- Umwandlung in eine Binärzahl $18,625_{10} = 10010,101000\ldots_2$
- Normalisieren $10010,101000\ldots \cdot 2^0 = 1,0010101000\ldots \cdot 2^4$
- Exponent $E = 4_{10} + 127_{10} = 131_{10} = 1000\ 0011_2$
- Vorzeichen $V = 0$
- Ergebnis $0|10000011|00101010000000000000000$

Berechnung der Dezimaldarstellung

Beispiel

1|10000001|10100000000000000000000 in einfacher Genauigkeit

Berechnung der Dezimaldarstellung

Beispiel

$1|10000001|10100000000000000000000$ in einfacher Genauigkeit

- Vorzeichen $V = 1 \Rightarrow$ negative Zahl
- Exponent $E = 129 = 127 + 2 \Rightarrow e = 2$
- Mantisse $m = 1 + 0,5 + 0,125 = 1,625$
- Ergebnis $-1,625 \cdot 2^2 = -6,5$

Berechnung der Dezimaldarstellung

Beispiel

1|10000001|10100000000000000000000 in einfacher Genauigkeit

- Vorzeichen $V = 1 \Rightarrow$ negative Zahl
- Exponent $E = 129 = 127 + 2 \Rightarrow e = 2$
- Mantisse $m = 1 + 0,5 + 0,125 = 1,625$
- Ergebnis $-1,625 \cdot 2^2 = -6,5$

Beispiel

0|10000100|11101010000000000000000 in einfacher Genauigkeit

Berechnung der Dezimaldarstellung

Beispiel

$1|10000001|10100000000000000000000$ in einfacher Genauigkeit

- Vorzeichen $V = 1 \Rightarrow$ negative Zahl
- Exponent $E = 129 = 127 + 2 \Rightarrow e = 2$
- Mantisse $m = 1 + 0,5 + 0,125 = 1,625$
- Ergebnis $-1,625 \cdot 2^2 = -6,5$

Beispiel

$0|10000100|11101010000000000000000$ in einfacher Genauigkeit

- Vorzeichen $V = 0 \Rightarrow$ positive Zahl
- Exponent $E = 132 = 127 + 5 \Rightarrow e = 5$
- Mantisse $m = 1,1110101_2$
- Ergebnis $1,1110101_2 \cdot 2^5 = 11\ 1101,01_2 = 61,25_{10}$

Probleme mit Zahlendarstellungen

- Begrenzter Wertebereich
- Begrenzte Genauigkeit, Rundungsfehler
  - sehr große bzw. sehr kleine Zahlen lassen wenig Platz für Nachkommastellen
  - bestimmte Dezimalbrüche nicht exakt als Binärbruch darstellbar (z.B. $0,1$)
  - Akkumulation von kleinen Rundungsfehlern

```
float f = 2f;
for (int i = 0; i < 100; i++) {
    f += 0.1f;
    System.out.println(f);
}
```

Das Wichtigste in Kürze

- Zahlen
  - beliebige Basis möglich, im Computer Basis 2
  - Umrechnung zwischen Zahlensystemen
  - Negative Zahlen, Reelle Zahlen

Inhalt


PRIOR
RIORITY
РОССИЯ МОСКВА
ÒÏÒÒÉÑ, ÍÏÒÉ×Á, 119415
ÐÒ. ÷ÅÒÎÁÄÓËÉÊ, 37,
Ë. 1817-1,
ÓÌÅÄÎÉ× ÏÊ
ÓÓÅÒÌÁÈ
RUSSIE.
RÉPUBLIQUE FRANÇAISE
PARIS
MERCŒUR
04/07/02
5,34 EUR
602 RC75712
(c) mpak

ASCII-Code

- *American Standard Code for Information Interchange*
- Standard von 1963, heute immer noch in Benutzung

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

ASCII-Code

- *American Standard Code for Information Interchange*
- Standard von 1963, heute immer noch in Benutzung

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

  - zu Beginn 7-Bit Code zur Verwendung in Fernschreibern
    - 128 Zeichen, davon 33 Kontrollzeichen, 95 druckbare Zeichen

Codepages

- Erweiterung des ASCII-Codes auf 8 Bit (1 Byte)
- Codepages oder character sets
  - CP850, CP437, ISO-8859-1 und ISO-8859-15 mit westeuropäischen Sonderzeichen

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	‚	-	·	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	¯
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
9	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
A	NBSP	¡	¢	£	€	¥	Š	§	š	©	ª	«	¬	SHY	®	¯
B	°	±	²	³	Ž	µ	¶	·	ž	¹	º	»	Œ	œ	Ÿ	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Unicode

- Probleme mit ASCII und Codepages
  - nicht alle möglichen Zeichen darstellbar, z.B. Kanji
  - Bedeutung eines Zeichens hängt von der zu Grunde gelegten Codepage ab
- Entwicklung von Unicode
  - gleichzeitige Darstellung aller möglichen Zeichen
  - eine feste Übersetzungstabelle
  - erste Fassung im Oktober 1991 mit 65.536 verschiedenen Zeichen
  - aktuelle Version 14.0 definiert 144.697 der 1.114.112 möglichen Zeichen

Unicode-Bereiche

- Einteilung des gesamten Codebereichs in
  17 Planes à $2^{16} = 65.536$ Zeichen
  - Beschreibung durch *U+* und mind. 4 Hexadezimalzahlen
  - ein Code beschreibt genau ein Zeichen, z.B. U+00DF das β
- Bereich 0, Basic Multilingual Plane (BMP)
  - aktuell verwendete Schriftsysteme, Satzzeichen, Kontrollzeichen, ...
  - stark fragmentiert und größtenteils belegt
- Bereich 1, Supplementary Multilingual Plane (SMP)
  - historische Schriftzeichen
  - Domino- und Mahjonggsteine
- Bereich 3, Supplementary Ideographic Plane (SIP)
  - chinesische, japanische und koreanische Schriftzeichen (CJK)
- Bereich 4, Supplementary Special-purpose Plane (SSP)
  - Kontrollzeichen zur Sprachmarkierung

Speicherung von Unicode

- 3 Bytes für alle möglichen Zeichen nötig
  - ungeschickt zu verarbeiten, da kein Maschinenwort
  - Platzverschwendung im europäischen Raum
- Definition von *Unicode Transformation Format*s (UTF)
- UTF-16
  - 2 Byte pro Zeichen, deckt die BMP ab
  - andere Bereiche werden durch Kombination von zwei UTF-16-Zeichen abgedeckt
  - immer noch Speicherverschwendung
- UTF-8
  - 1 Byte pro Zeichen, deckt die wichtigsten westlichen Zeichen ab
  - weitere Zeichen werden durch Kombination von bis zu drei UTF-8-Zeichen abgedeckt
- Daneben noch UTF-7 und UTF-32

Das Wichtigste in Kürze

- Zahlen
  - beliebige Basis möglich, im Computer Basis 2
  - Umrechnung zwischen Zahlensystemen
  - Negative Zahlen, Reelle Zahlen
- Zeichen
  - ASCII, Codepages, Unicode

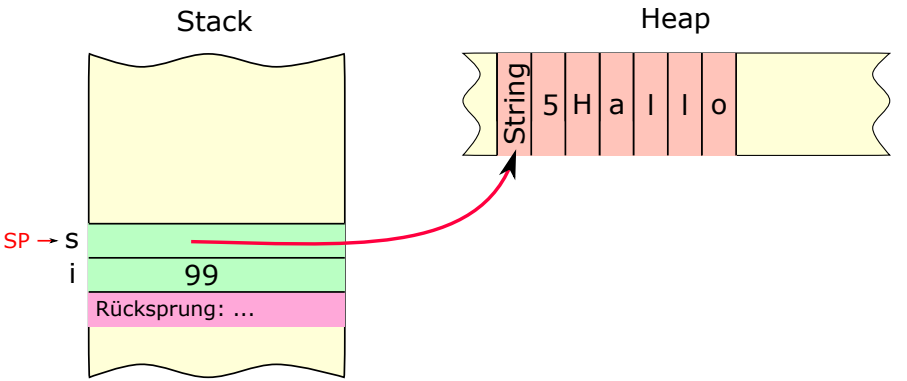
Inhalt

Speicherbereiche

- Programmbereich (Code Segment, CS)
  - enthält das von der Festplatte geladene Programm
  - ist i.d.R. nicht beschreibbar
- Stapelbereich (Stack Segment, SS)
  - enhält lokale Variablen, Parameter, Rückgabewerte, Rücksprunginformationen
  - ist i.d.R. nicht ausführbar (NX – no execute)
  - in der Größe beschränkt, z.B. 8MB unter Linux, 1MB unter Windows und Java
- Daten-/Haldenbereich (Data Segment, DS; Heap)
  - enthält globale Daten und in Java sämtliche Objekte
  - ist i.d.R. nicht ausführbar (NX – no execute)
  - Größe nur durch Hauptspeicher begrenzt bzw. in Java durch Angabe bei Programmstart
    - `java -Xmx2048m` reserviert 2GB für Heap

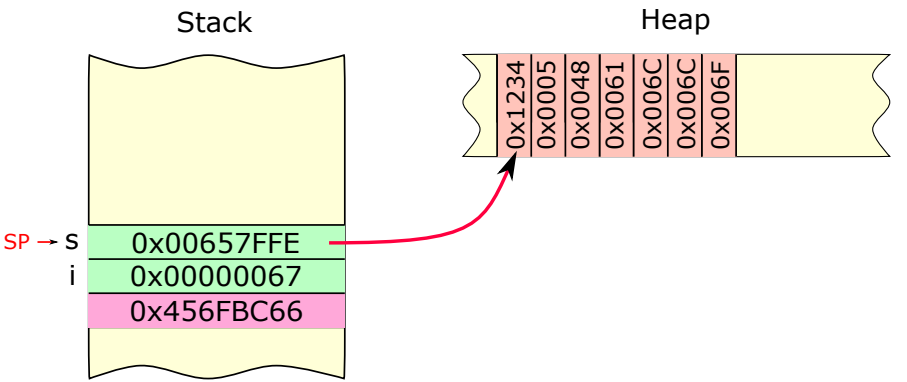
Heap und Stack I

```
public void foo(int i) {
  String s = new String("Hallo");
}
```



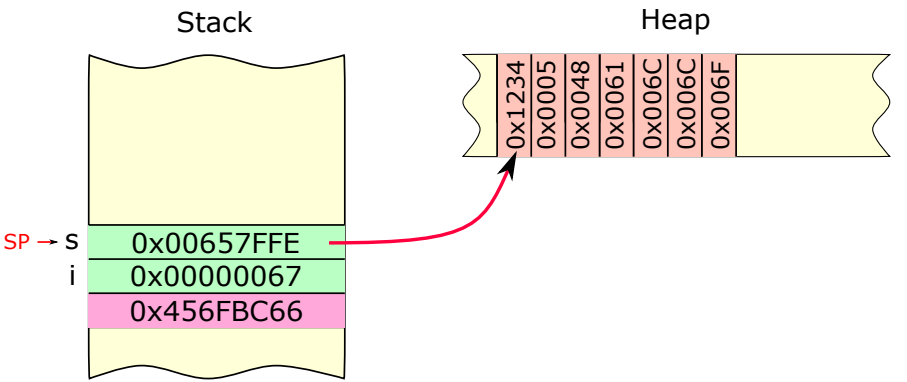
Heap und Stack II

- Im Speicher stehen nur Bitmuster
- Interpretation erfolgt durch das Programm



Heap und Stack II

- Im Speicher stehen nur Bitmuster
- Interpretation erfolgt durch das Programm



- Im Heap können Lücken entstehen ⇒ Speicherverwaltung notwendig

Das Wichtigste in Kürze

- Zahlen
  - beliebige Basis möglich, im Computer Basis 2
  - Umrechnung zwischen Zahlensystemen
  - Negative Zahlen, Reelle Zahlen
- Zeichen
  - ASCII, Codepages, Unicode
- Speicherbereiche
  - Stack u.a. für lokale Variablen und Parameter
  - Heap für sämtliche Objekte

Inhalt

Primitive Datentypen

zum Beispiel in Java

Typ	Bits	Kodierung	Minimalwert	Maximalwert
`boolean`	1/8	Wahrheitwert	`false`	`true`
`byte`	8	2er-Komplement	-128	127
`short`	16	2er-Komplement	-32 768	32 767
`int`	32	2er-Komplement	-2 147 483 648	2 147 483 647
`long`	64	2er-Komplement	-9 223 372 036 854 775 808	9 223 372 036 854 775 807
`char`	16	Unicode	0 (\u0000)	65 536 (\uffff)
`float`	32	IEEE 754	$\pm 1,4 \cdot 10^{-45}$	$\pm 3,4 \cdot 10^{38}$
`double`	64	IEEE 754	$\pm 4,9 \cdot 10^{-324}$	$\pm 1,7 \cdot 10^{308}$

Überläufe

- Wertebereiche sind begrenzt
- Beim Verlassen des Wertebereiches tritt ein *Überlauf* auf

Beispiel

für `byte`

$$\begin{aligned} 90_{10} &= 0101\ 1010_2 \\ +40_{10} &= 0010\ 1000_2 \\ 130_{10} &= 1000\ 0010_2 \end{aligned}$$

Aber: $1000\ 0010_2 = -126_{10}$ im Zweierkomplement!

- Überläufe bei Ganzzahlen werden nicht erkannt!
- Überläufe bei Fließkommazahlen ergeben ∞

Das Wichtigste in Kürze

- Zahlen
  - beliebige Basis möglich, im Computer Basis 2
  - Umrechnung zwischen Zahlensystemen
  - Negative Zahlen, Reelle Zahlen
- Zeichen
  - ASCII, Codepages, Unicode
- Speicherbereiche
  - Stack u.a. für lokale Variablen und Parameter
  - Heap für sämtliche Objekte
- Datentypen
  - Elementare Datentypen (in Java)

Literatur

- H. P. Gumm und M. Sommer.
  Einführung in die Informatik.
  Oldenburg Verlag, 7. Ausgabe, 2006, ISBN 978-3-486-58115-7.
- H. Herold, B. Lurz, und J. Wohlrab.
  Grundlagen der Informatik.
  Pearson Studium, 2007, ISBN 978-3-8273-7305-2.

Bildquellen

Foto Russischer Großbrief:
`https://unicodebook.readthedocs.io/_images/Letter_to_Russia_with_krakozyabry.jpg`
zuletzt geöffnet am 10. November 2020