Beispiele zur Codetransformation

Variableninhalte und Zugriffsoperatoren:

Konsequenz:
Typen ordnen Variablen

- Größe im Speicher und
- Interpretation zu

2. Darstellungen:

Zeichen		**ASCII**
Unsigned		**Betragsdarstellung**
Signed	→	**2-er Komplement**
Float		**single precision format**
Double		**double precision format**
		nach IEEE 754

2. Darstellungen

ASCII - American Standard Code for Information Interchange

ASCII-Zeichentabelle, hexadezimale Nummerierung

Code	…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…	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
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

2. Darstellungen

Betragsdarstellung

$$z_m z_{m-1} \cdots z_0$$

Interpretation

$$Z = \sum_{i=-n}^{m} z_i \cdot 2^i$$

2. Darstellungen

2-er Komplement

$$z_m z_{m-1} \ldots z_0$$

Interpretation

$$\text{2-er Komplement } Wert(b) = \begin{cases} \sum_{i=0}^{n-2} b_i * 2^i & , falls \quad b_{n-1} = 0 \\ -(\sum_{i=0}^{n-2} \bar{b}_i * 2^i + 1) & , sonst \end{cases}$$

2. Darstellungen

IEEE 754 Single Precision Format

Single Precision Floating Point (IEEE 754):
Darstellung mit:

- Vorzeichen → 1 Bit
- Exponent mit Bias–Darstellung →8 Bit Exponent, Bias 127
- normierter Mantisse →23 Bit

$$Float(b) = (-1)^{VZ} * 2^{(\sum_{i=0}^{7} e_i * 2^i - 127)} * (1 + \sum_{i=1}^{23} m_i * 2^{-i})$$

b = VZ | $e_7 \ldots e_0$ | $m_1 \ldots m_{23}$

IEEE 754 Double Precision Format

Double Precision Floating Point (IEEE 754):
Darstellung mit:

- Vorzeichen → 1 Bit
- Exponent mit Bias–Darstellung →11 Bit Exponent, Bias 1023
- normierter Mantisse →52 Bit

Konvertiere float in Binärdarstellung (C)

```
main() {
        float f;
        printf („Bitte float-Zahl eingeben: „);
        scanf („%f“, &f);
        fausgeben(f);
        return 0;
}
```

Konvertiere float in Binärdarstellung (C)

```
void fausgeben(float f) {
      int i, w;

      w= * ((int *)(&f));

      for (i=31; i>=0; i--) {
            switch (i) {
            case31:     printf(„Vorzeichen:“);
                        break;
            case30:     printf(„Exp:“);
                        break;
            case22:     printf(„Mantisse:“);
            }
```

Konvertiere float in Dezimaldarstellung (C)

```
                printf(„%c“, ((w&(1<<i)) !=0) + ’0’ );
                switch (i) {
                case31:
                case23:
                case0: printf(„\n“);
                }
        }
}
```

Instruktions Codierung

Bestandteile:

1. Opcode
2. Operandenspezifikation
3. Operanden

Welche Operation ist auszuführen

wo stehen Operanden,

Operanden - Register, Speicher, ….

Instruktions Codierung: Beispiel

```
push    5
```

2. Darstellungen

Instruktions Codierung: Beispiel

push 5

IA-32 Intel® Architecture Software Developer's Manual

Volume 2: Instruction Set Reference

2. Darstellungen

Instruktions Codierung: Beispiel

push 5

IA-32 Intel® Architecture Software Developer's Manual

Volume 2: Instruction Set Reference

PUSH—Push Word or Doubleword Onto the Stack

Opcode	Instruction	Description
FF /6	PUSH *r/m16*	Push *r/m16*
FF /6	PUSH *r/m32*	Push *r/m32*
50+*rw*	PUSH *r16*	Push *r16*
50+*rd*	PUSH *r32*	Push *r32*
6A	PUSH *imm8*	Push *imm8*
68	PUSH *imm16*	Push *imm16*
68	PUSH *imm32*	Push *imm32*
0E	PUSH CS	Push CS
16	PUSH SS	Push SS
1E	PUSH DS	Push DS
06	PUSH ES	Push ES
0F A0	PUSH FS	Push FS
0F A8	PUSH GS	Push GS

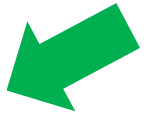
2. Darstellungen

Instruktions Codierung: Beispiel

push 5

IA-32 Intel® Architecture Software Developer's Manual

Volume 2: Instruction Set Reference



PUSH—Push Word or Doubleword Onto the Stack

Opcode	Instruction	Description
FF /6	PUSH *r/m16*	Push *r/m16*
FF /6	PUSH *r/m32*	Push *r/m32*
50+*rw*	PUSH *r16*	Push *r16*
50+*rd*	PUSH *r32*	Push *r32*
6A	PUSH *imm8*	Push *imm8*
68	PUSH *imm16*	Push *imm16*
68	PUSH *imm32*	Push *imm32*
0E	PUSH CS	Push CS
16	PUSH SS	Push SS
1E	PUSH DS	Push DS
06	PUSH ES	Push ES
0F A0	PUSH FS	Push FS
0F A8	PUSH GS	Push GS

2. Darstellungen

Instruktions Codierung: Beispiel

6A 05 ⟺ push 5

PUSH—Push Word or Doubleword Onto the Stack

Opcode	Instruction	Description
FF /6	PUSH *r/m16*	Push *r/m16*
FF /6	PUSH *r/m32*	Push *r/m32*
50+*rw*	PUSH *r16*	Push *r16*
50+*rd*	PUSH *r32*	Push *r32*
6A	PUSH *imm8*	Push *imm8*
68	PUSH *imm16*	Push *imm16*
68	PUSH *imm32*	Push *imm32*
0E	PUSH CS	Push CS
16	PUSH SS	Push SS
1E	PUSH DS	Push DS
06	PUSH ES	Push ES
0F A0	PUSH FS	Push FS
0F A8	PUSH GS	Push GS

IA-32 Intel® Architecture Software Developer's Manual

Volume 2: Instruction Set Reference

2. Darstellungen

Instruktions Codierung: Beispiel

83 C4 04  add esp,4

2. Darstellungen

Instruktions Codierung: Beispiel

83 C4 04 ⟺ add esp,4

ADD—Add

Opcode	Instruction	Description
04 *ib*	ADD AL,*imm8*	Add *imm8* to AL
05 *iw*	ADD AX,*imm16*	Add *imm16* to AX
05 *id*	ADD EAX,*imm32*	Add *imm32* to EAX
80 /0 *ib*	ADD *r/m8,imm8*	Add *imm8* to *r/m8*
81 /0 *iw*	ADD *r/m16,imm16*	Add *imm16* to *r/m16*
81 /0 *id*	ADD *r/m32,imm32*	Add *imm32* to *r/m32*
83 /0 *ib*	ADD *r/m16,imm8*	Add sign-extended *imm8* to *r/m16*
83 /0 *ib*	ADD *r/m32,imm8*	Add sign-extended *imm8* to *r/m32*
00 /r	ADD *r/m8,r8*	Add *r8* to *r/m8*
01 /r	ADD *r/m16,r16*	Add *r16* to *r/m16*
01 /r	ADD *r/m32,r32*	Add r32 to *r/m32*
02 /r	ADD *r8,r/m8*	Add *r/m8* to *r8*
03 /r	ADD *r16,r/m16*	Add *r/m16* to *r16*
03 /r	ADD *r32,r/m32*	Add *r/m32* to *r32*

2. Darstellungen

Instruktions Codierung: Beispiel

83 C4 04 ⟺ add esp,4

ADD—Add

Opcode	Instruction	Description
04 *ib*	ADD AL,*imm8*	Add *imm8* to AL
05 *iw*	ADD AX,*imm16*	Add *imm16* to AX
05 *id*	ADD EAX,*imm32*	Add *imm32* to EAX
80 /0 *ib*	ADD *r/m8,imm8*	Add *imm8* to *r/m8*
81 /0 *iw*	ADD *r/m16,imm16*	Add *imm16* to *r/m16*
81 /0 *id*	ADD *r/m32,imm32*	Add *imm32* to *r/m32*
83 /0 *ib*	ADD *r/m16,imm8*	Add sign-extended *imm8* to *r/m16*
83 /0 *ib*	ADD *r/m32,imm8*	Add sign-extended *imm8* to *r/m32*
00 /r	ADD *r/m8,r8*	Add *r8* to *r/m8*
01 /r	ADD *r/m16,r16*	Add *r16* to *r/m16*
01 /r	ADD *r/m32,r32*	Add r32 to *r/m32*
02 /r	ADD *r8,r/m8*	Add *r/m8* to *r8*
03 /r	ADD *r16,r/m16*	Add *r/m16* to *r16*
03 /r	ADD *r32,r/m32*	Add *r/m32* to *r32*

2. Darstellungen

Instruktions Codierung: Beispiel

83 C4 04 ⟺ add esp,4

ADD—Add

Opcode	Instruction	Description
04 *ib*	ADD AL,*imm8*	Add *imm8* to AL
05 *iw*	ADD AX,*imm16*	Add *imm16* to AX
05 *id*	ADD EAX,*imm32*	Add *imm32* to EAX
80 /0 *ib*	ADD *r/m8,imm8*	Add *imm8* to *r/m8*
81 /0 *iw*	ADD *r/m16,imm16*	Add *imm16* to *r/m16*
81 /0 *id*	ADD *r/m32,imm32*	Add *imm32* to *r/m32*
83 /0 *ib*	ADD *r/m16,imm8*	Add sign-extended *imm8* to *r/m16*
83 /0 *ib*	ADD *r/m32,imm8*	Add sign-extended *imm8* to *r/m32*
00 /r	ADD *r/m8,r8*	Add *r8* to *r/m8*
01 /r	ADD *r/m16,r16*	Add *r16* to *r/m16*
01 /r	ADD *r/m32,r32*	Add r32 to *r/m32*
02 /r	ADD *r8,r/m8*	Add *r/m8* to *r8*
03 /r	ADD *r16,r/m16*	Add *r/m16* to *r16*
03 /r	ADD *r32,r/m32*	Add *r/m32* to *r32*

2. Darstellungen

Instruktions Codierung: Beispiel

83 C4 04 ⟺ add esp,4

ADD—Add

Opcode	Instruction	Description
04 *ib*	ADD AL,*imm8*	Add *imm8* to AL
05 *iw*	ADD AX,*imm16*	Add *imm16* to AX
05 *id*	ADD EAX,*imm32*	Add *imm32* to EAX
80 /0 *ib*	ADD *r/m8,imm8*	Add *imm8* to *r/m8*
81 /0 *iw*	ADD *r/m16,imm16*	Add *imm16* to *r/m16*
81 /0 *id*	ADD *r/m32,imm32*	Add *imm32* to *r/m32*
83 /0 *ib*	ADD *r/m16,imm8*	Add sign-extended *imm8* to *r/m16*
83 /0 *ib*	ADD *r/m32,imm8*	Add sign-extended *imm8* to *r/m32*
00 /r	ADD *r/m8,r8*	Add *r8* to *r/m8*
01 /r	ADD *r/m16,r16*	Add *r16* to *r/m16*
01 /r	ADD *r/m32,r32*	Add r32 to *r/m32*
02 /r	ADD *r8,r/m8*	Add *r/m8* to *r8*
03 /r	ADD *r16,r/m16*	Add *r/m16* to *r16*
03 /r	ADD *r32,r/m32*	Add *r/m32* to *r32*

2. Darstellungen

Instruktions Codierung: Beispiel

83 C4 04 ⟷ add esp,4

Table 2-2. 32-Bit Addressing Forms with the ModR/M Byte

r8(/r)			AL	CL	DL	BL	AH	CH	DH	BH
r16(/r)			AX	CX	DX	BX	SP	BP	SI	DI
r32(/r)			EAX	ECX	EDX	EBX	ESP	EBP	ESI	EDI
mm(/r)			MM0	MM1	MM2	MM3	MM4	MM5	MM6	MM7
xmm(/r)			XMM0	XMM1	XMM2	XMM3	XMM4	XMM5	XMM6	XMM7
/digit (Opcode)			0	1	2	3	4	5	6	7
REG =			000	001	010	011	100	101	110	111
Effective Address	**Mod**	**R/M**	**Value of ModR/M Byte (in Hexadecimal)**							
.....										
EAX/AX/AL/MM0/XMM0	11	000	C0	C8	D0	D8	E0	E8	F0	F8
ECX/CX/CL/MM/XMM1		001	C1	C9	D1	D9	E1	E9	F1	F9
EDX/DX/DL/MM2/XMM2		010	C2	CA	D2	DA	E2	EA	F2	FA
EBX/BX/BL/MM3/XMM3		011	C3	CB	D3	DB	E3	EB	F3	FB
ESP/SP/AH/MM4/XMM4		100	C4	CC	D4	DC	E4	EC	F4	FC
EBP/BP/CH/MM5/XMM5		101	C5	CD	D5	DD	E5	ED	F5	FD
ESI/SI/DH/MM6/XMM6		110	C6	CE	D6	DE	E6	EE	F6	FE
.....										