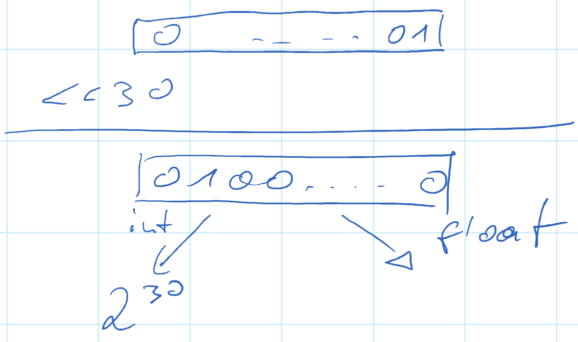
Einleitung und Zielsetzung

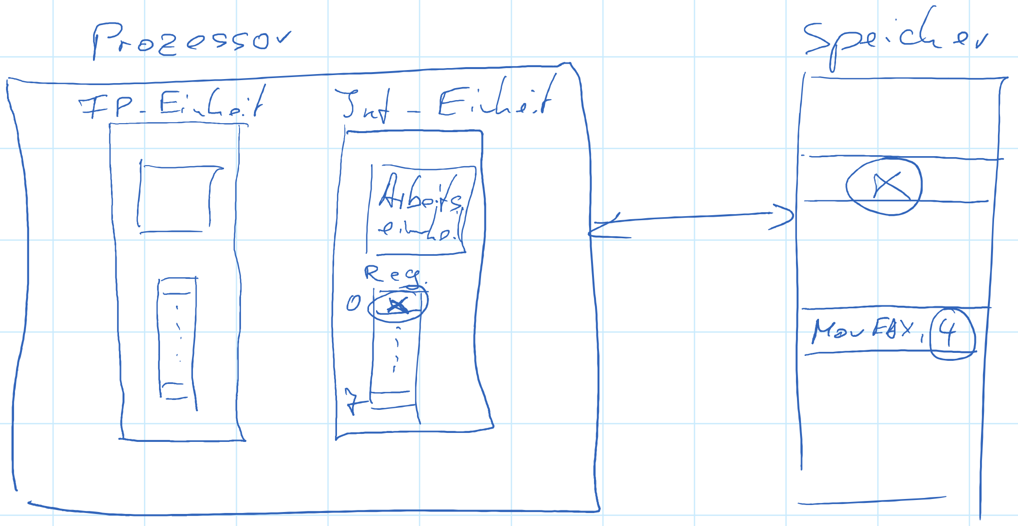
Sonntag, 17. März 2019 20:25

$1 << 30$



$$(-1)^{VZ} \cdot 2^{(B(8Bit)-127)} \cdot (1 + \text{Mant.Bits})$$

$$(-1)^0 \cdot 2^1 \cdot 1 = 2$$



Betragsdarstellung:

n-Bit: $b_{n-1}\ b_{n-2} \cdots b_0 \xrightarrow[\text{Betrag}]{\text{Interp.}} \sum_{i=0}^{n-1} b_i * 2^i$

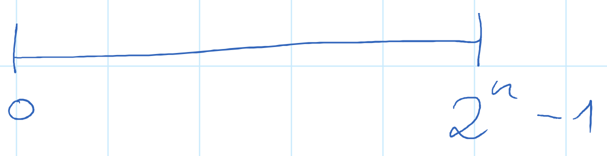
Bsp.:

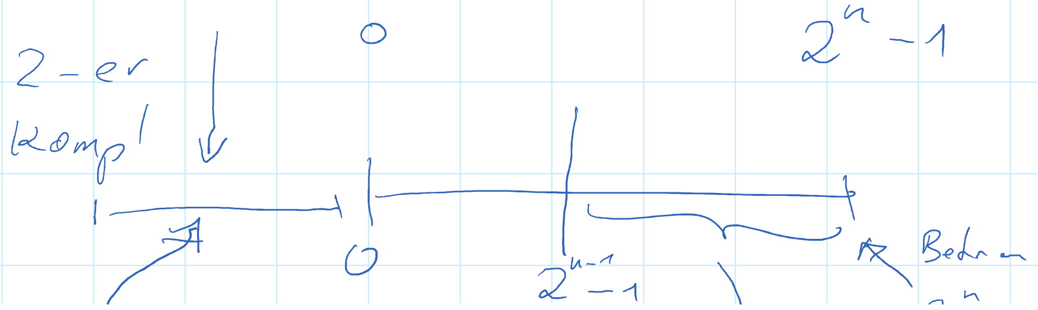
$01010101 \longrightarrow 2^6 + 2^4 + 2^2 + 2^0$

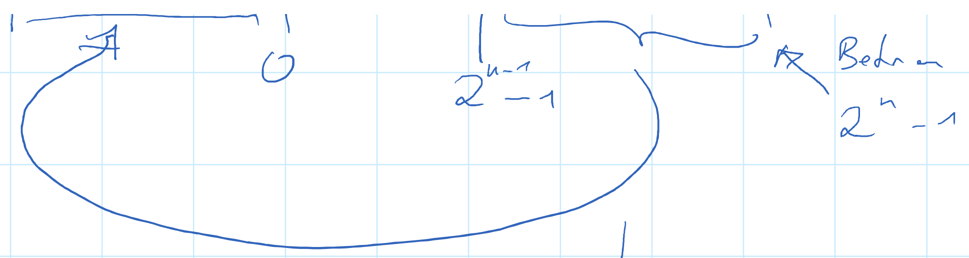
$= 64 + 16 + 4 + 1$

$= 85$

$87 \longrightarrow 2^6 + 2^4 + 2^2 + 2^1 + 2^0$







2-er Komplement $b_{n-1} | b_{n-2} \cdots b_0$

$$\text{Wert}(b) = \begin{cases} \sum_{i=0}^{n-2} b_i \cdot 2^i & \text{, falls } b_{n-1} = 0 \\ -\left(\sum_{i=0}^{n-2} \overline{b_i} \cdot 2^i + 1\right) & \text{, falls } b_{n-1} = 1 \end{cases}$$

$-1 \;\hat{=}\; 1111\ 1111$

$-17 \;\hat{=}\; -(\underset{\underset{2^4}{\|}}{16} + 1)$

$1\ 1\ 1\ 0\ \ 1\ 1\ 1\ 1$

$17 \;\hat{=}\; 0001\ 0001$

Neg. $1110\ 1110$

$+ \qquad\qquad 1$

$\overline{1110\ 1111}$

Operanden:

Konstanten – imm, unmittelbar in Instruktion

Register – r, 3 Bit

Speicher – m, für memory

- Konstante Adresse (globale Variablen)
- Variable Adresse (z. B. Pointer) in Register
- Kombination:

- Konstante Anfangsadr. + Variable * Skalierungsfaktor (Arrayzugriff, 1-dim.)
- Konstante Anf.adr. + Basis + Index * Scale

Implizite
Explizite } Operanden

Bsp.:

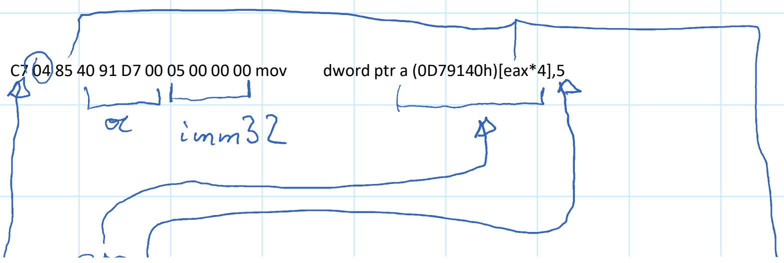
NOP — 0 explizite Operanden
Implizit: EIP

Push 5 — Explizit: 5
Implizit: ESP
[ESP]
EIP

mov EAX, 5 — Explizit: EAX
5
Implizit: EIP

iMUL EAX, EBX, 5
Explizit: EAX
EBX
5
Implizit: EIP

0 bis 3 Adressoperationen

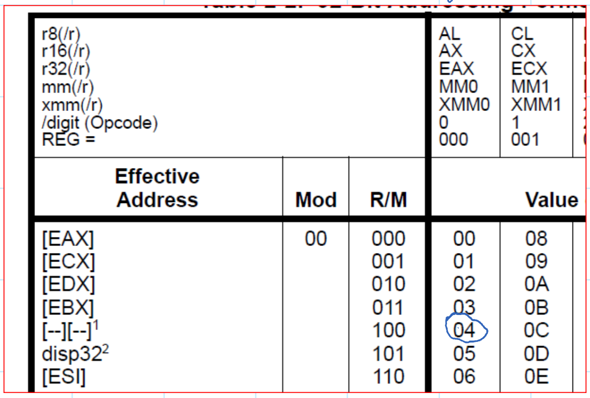


```
C7 04 85 40 91 D7 00 05 00 00 00 mov      dword ptr a (0D79140h)[eax*4],5
```

Table 2-3. 32-Bit Addressing Forms with the SIB Byte

r32 Base = Base =			EAX 0 000	ECX 1 001	EDX 2 010	EBX 3 011	ESP 4 100	[*] 5 101	ESI 6 110	EDI 7 111
Scaled Index	**SS**	**Index**	**Value of SIB Byte (in Hexadecimal)**							

C7 /0 MOV r/m32,imm32 Move imm32 to r/m32



r8(/r)
r16(/r)
r32(/r)
mm(/r)
xmm(/r)
/digit (Opcode)
REG =
			AL
			AX
			EAX
			MM0
			XMM0
			0
			000
				CL
				CX
				ECX
				MM1
				XMM1
				1
				001

Effective Address	**Mod**	**R/M**	**Value**	
[EAX]	00	000	00	08
[ECX]		001	01	09
[EDX]		010	02	0A
[EBX]		011	03	0B
[--][--][1]		100	04	0C
disp32[2]		101	05	0D
[ESI]		110	06	0E

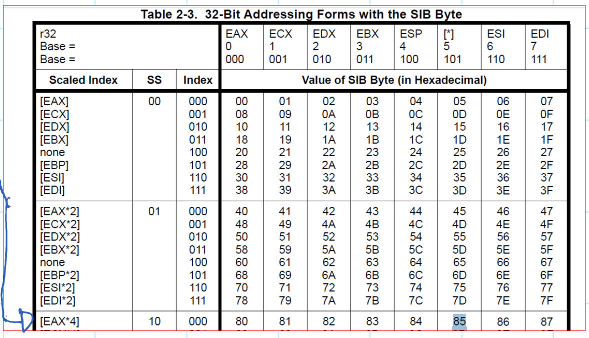
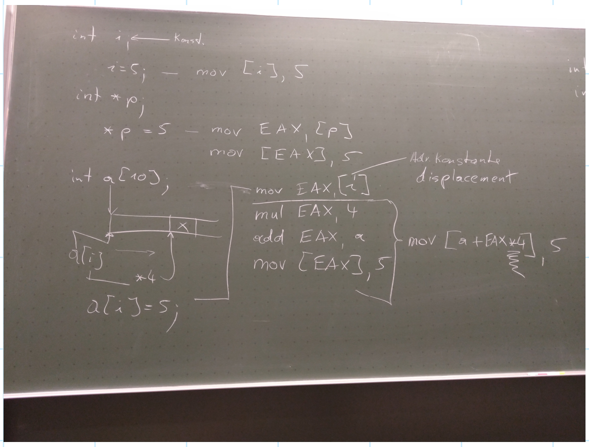
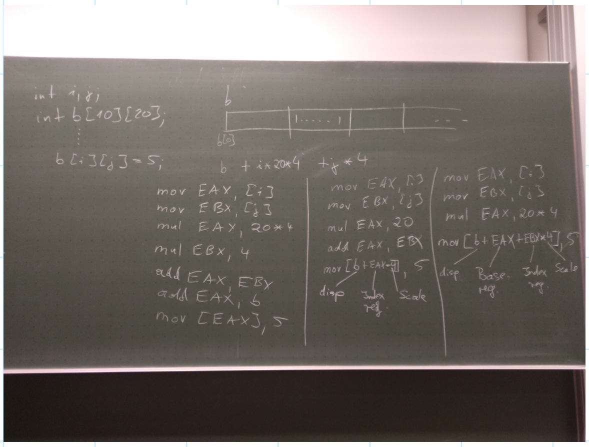


Table 2-3. 32-Bit Addressing Forms with the SIB Byte

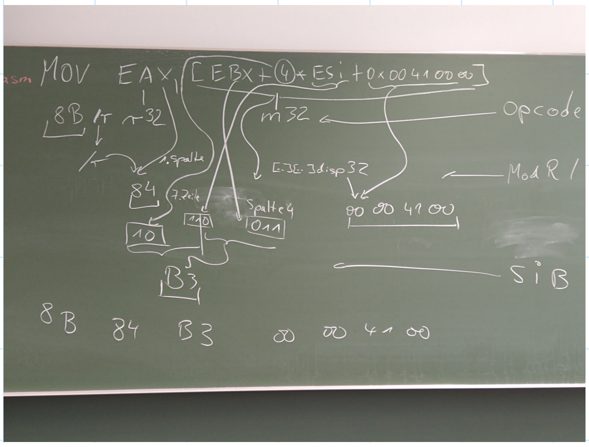
r32 Base = Base =			EAX 0 000	ECX 1 001	EDX 2 010	EBX 3 011	ESP 4 100	[*] 5 101	ESI 6 110	EDI 7 111
Scaled Index	**SS**	**Index**	**Value of SIB Byte (in Hexadecimal)**							
[EAX]	00	000	00	01	02	03	04	05	06	07
[ECX]		001	08	09	0A	0B	0C	0D	0E	0F
[EDX]		010	10	11	12	13	14	15	16	17
[EBX]		011	18	19	1A	1B	1C	1D	1E	1F
none		100	20	21	22	23	24	25	26	27
[EBP]		101	28	29	2A	2B	2C	2D	2E	2F
[ESI]		110	30	31	32	33	34	35	36	37
[EDI]		111	38	39	3A	3B	3C	3D	3E	3F
[EAX*2]	01	000	40	41	42	43	44	45	46	47
[ECX*2]		001	48	49	4A	4B	4C	4D	4E	4F
[EDX*2]		010	50	51	52	53	54	55	56	57
[EBX*2]		011	58	59	5A	5B	5C	5D	5E	5F
none		100	60	61	62	63	64	65	66	67
[EBP*2]		101	68	69	6A	6B	6C	6D	6E	6F
[ESI*2]		110	70	71	72	73	74	75	76	77
[EDI*2]		111	78	79	7A	7B	7C	7D	7E	7F
[EAX*4]	10	000	80	81	82	83	84	85	86	87



```
int i; ← Konst.
  i=5; — mov [i], 5
int *p;
  *p = 5 — mov EAX, [p]
           mov [EAX], 5
int a[10];
           mov EAX, [i]      Adr. konstante
                             displacement
           mul EAX, 4
           add EAX, a        mov [a+EAX*4], 5
           mov [EAX], 5
a[i]=5;
```

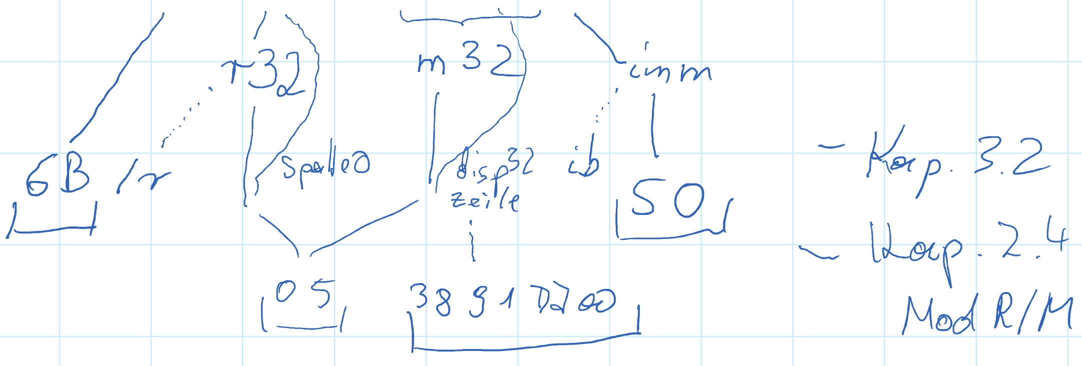


```
int i,j;
int b[10][20];
  b[i][j]=5;          b + i*20*4 +j*4

mov EAX, [i]      | mov EAX, [i]          | mov EAX, [i]
mov EBX, [j]      | mov EBX, [j]          | mov EBX, [j]
mul EAX, 20*4     | mul EAX, 20           | mul EAX, 20*4
mul EBX, 4        | add EAX, EBX          | mov [b+EAX+EBX*4], 5
add EAX, EBX      | mov [b+EAX*4], 5      |  disp Base- Index Scale
add EAX, b        |  disp Index Scale     |       reg.  reg.
mov [EAX], 5      |       reg.            |
```

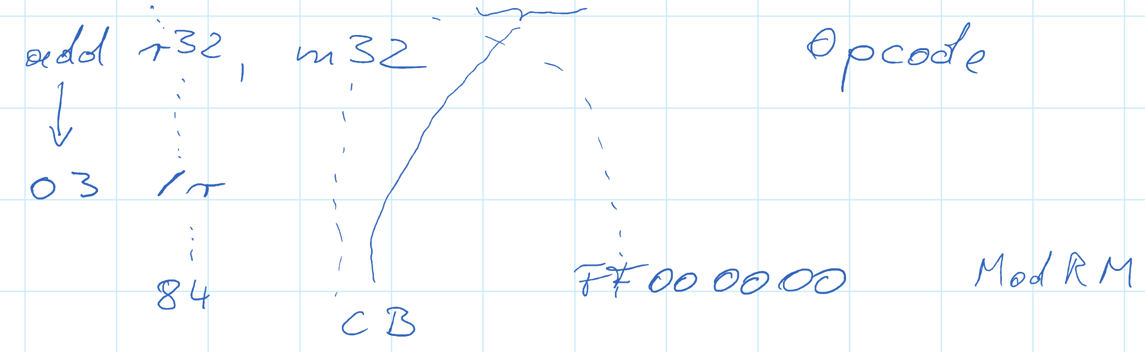


```
asm MOV EAX, [EBX+4*ESi+0x00410000]
    8B /r r32          m32                  ← Opcode
    /r  1.Spalte
    84  7.Zeile  [..][..]disp32             ← ModR/
    10  110  Spalte4 011    00 00 41 00
      B3                                    ← SiB

8B 84 B3 00 00 41 00
```

imul eax,dword ptr [i (0D79138h)],50h



__asm add EAX, DWORD PTR [0x000000FF + EBX + ECX*8]


add r32, m32
03 /r
84
CB
FF000000
Opcode
ModRM