AchtBit - Instructions

Storage	t	t'
r,r	0	0
r,n[n]	1	1
r,(nn)	2	2
r,(r)	3	3
(nn),r	4	-
(r),r	5	-
(r),n	6	-

RRRR=dest. register
/ rrrr=source register

regF	0 / 00
regA	1 / 10
regC	2 / 20
regB	3 / 30
regE	4 / 40
regD	5 / 50
regL	6 / 60
regH	7 / 70
regSpL	8 / 80
regSpH	9 / 90
regAF	A / A0
regBC	B / B0
regDE	C / C0
regHL	D / D0
regSP	E / E0
regIX	F / F0 (with offset o) (f.e. LD (IX-8),DE)

Flags

true	0
Z	1
C	2
P	3
(false	4 unused)
NZ	5
NC	6
NP	7

Memory Map:

```
0000 Program start
...
D200 Stack (unterhalb von...)

Text Input Puffer (1k)
D200 max. Size
D202 Data

D600 Charset(256*10 Byte)

Textmode:
  E000-E320 Char (40x20)
  E400-E720 BG/FG (40x20)

Gfxmode:
  E000-FF3F Grafik
  1: 320x200 s/w, 40x200 Byte
  2: 160x200 4c,  40x200 Byte
  3: 160x100 16c, 80x100 Byte

FF40 Palette (16x3 Byte rgb)

FF70 Mode (0=text, 1..3=gfx)
FF71 BG/FG (in gfxMode 1+2)
FF72 Screen refresh
FF73 Screen Interrupt
FF76 cursor x,y
FF78 BG/FG (TextMode)
FF79 text Cursor (1=enable, 0=disable)
FF7A text Input Puffer Pointer
FF7C cursor speed (25 = 1 sek.;
                   1 = 0.04 sek.)
```

Com	Code		Flags
nop	00		
hlt	01		
clc	02		> NC
reti	03		
setc	04		> C
ld	08+t	Rr[o] | r[o] n | r[o] nn	
add	10+t'	Rr[o] | r[o] n | r[o] nn	> CZP
adc	14+t'	Rr[o] | r[o] n | r[o] nn	C > CZP
sub	18+t'	Rr[o] | r[o] n | r[o] nn	> CZP
sbc	1C+t'	Rr[o] | r[o] n | r[o] nn	C > CZP
mul	20+t'	Rr[o] | r[o] n | r[o] nn	> CZP
div	24+t'	Rr[o] | r[o] n | r[o] nn	> CZP
and	28+t'	Rr[o] | r[o] n | r[o] nn	> ZP
or	2C+t'	Rr[o] | r[o] n | r[o] nn	> ZP
xor	30+t'	Rr[o] | r[o] n | r[o] nn	> ZP
cmp	34+t'	Rr[o] | r[o] n | r[o] nn	> CZP
mod	38+t'	Rr[o] | r[o] n | r[o] nn	> CZP
jp	40+f	nn	[C|Z|P]
call	48+f	nn	[C|Z|P]
jr	50+f	n	[C|Z|P]
ret	58+f		[C|Z|P]
push	60+r		
pop	70+r		>[CZP]
shl	80+r		> CZP
shr	90+r		> CZP
rhl	A0+r		C > CZP
rhr	B0+r		C > CZP
inc	C0+r		> CZP
dec	D0+r		> CZP
in	E0+r	n	
out	F0+r	n	

```
System-Clock:
FF7D sec, min, hour
FF80 day, month, year
FF84 FG2, FG3 (int gfxMode 2)
FF85 ...
..FFFF frei
```

Ports:

```
Keyboard: IN:  (0)            → get next char
          OUT: (0),0xff       → clear buffer

Random:   IN(1)

Sound:    OUT: (2),loudness   → set
               (3),length     → set
               (4),length-unit→ set
               (5),tone       → append tone to
                                queue for playback
               (5),0          → clear queue
               (6),bpm        → set

          IN:  (2),loudness   → get
               (3),length     → get
               (4),length-unit→ get
               (5)!=0?        →finished playback?
               (6),bpm        → get
```