

PySNES – EIN SNES Emulator in PYTHON

Homebrew

Inhalt

- Homebrew
- CC65 Intro
- CC65 Code
- CC65 Konfigurationsdatei
- CC65 Header

Homebrew

- Homebrew ist das Schreiben von „selbstgebrauten“ SNES ROMs. Diese werden i.d.R. in einem Emulator oder sogar auf echter Hardware ausgeführt
- Motivation: Testcases für Emulator schreiben

CC65 Intro

- CC65 ist eine Sammlung von Tools um 62X Programm zu erzeugen, also auch SNES ROMs
- Download: <https://sourceforge.net/projects/cc65/>
- CC65 Tools im Ordner bin
 - Assembler mit `ca65 XXX.s`
 - Linker mit `ld65 -C CONF.cfg -o XXX.smc XXX.o`
- Dokumentation im Ordner `html/index.html`
- Beispiel: <https://wiki.superfamicom.org/basic-ca65-usage-for-snes-programming>

CC65 Intro

Linker Konfig Datei:

```
1 # ca65 linker config for 128K SMC
2 # https://wiki.superfamicom.org/basic-ca65-usage-for-snes-programming
3 # Physical areas of memory
4 # Names need not match, but it makes it easier to remember if they do.
5 MEMORY {
6     ZEROPAGE:    start =      0, size = $100;
7     BSS:         start =   $200, size = $1800;
8     ROM:         start =  $8000, size = $8000, fill = yes;
9     BANK1:       start = $18000, size = $8000, fill = yes;
10    BANK2:       start = $28000, size = $8000, fill = yes;
11    BANK3:       start = $38000, size = $8000, fill = yes;
12 }
13
14 # Logical areas code/data can be put into.
15 SEGMENTS {
16     ZEROPAGE:    load = ZEROPAGE,    type = zp;
17     BSS:         load = BSS,         type = bss, align = $100;
18
19     CODE:        load = ROM,         align = $8000;
20     RODATA:      load = ROM;
21     HEADER:      load = ROM,         start = $FFC0;
22     ROMINFO:     load = ROM,         start = $FFD5, optional = yes;
23     VECTORS:     load = ROM,         start = $FFE0;
24
25     # The extra three banks
26     BANK1:       load = BANK1,       align = $8000, optional = yes;
27     BANK2:       load = BANK2,       align = $8000, optional = yes;
28     BANK3:       load = BANK3,       align = $8000, optional = yes;
29 }
```

CC65 Intro

65816 Assembler Datei:

```
1 ; Minimal example of using ca65 to build SNES ROM.
2 ;
3 ; ca65 ca65.s
4 ; ld65 -C lorom128.cfg -o ca65.smc ca65.o
5
6 .p816      ; 65816 processor
7 .il16      ; X/Y are 16 bits
8 .a8        ; A is 8 bits
9
10 .segment "HEADER"      ; +$7FE0 in file
11     .byte "CA65 EXAMPLE" ; ROM name
12
13 .segment "ROMINFO"      ; +$7FD5 in file
14     .byte $30           ; LoROM, fast-capable
15     .byte 0             ; no battery RAM
16     .byte $07          ; 128K ROM
17     .byte 0,0,0,0
18     .word $AAAA,$5555   ; dummy checksum and complement
19
20 .segment "VECTORS"
21     .word 0, 0, 0, 0, 0, 0, 0, 0
22     .word 0, 0, 0, 0, 0, 0, 0, reset, 0
```

```
24 .segment "CODE"
25
26 reset:
27     clc                ; native mode
28     xce
29     rep #$10           ; X/Y 16-bit
30     sep #$20           ; A 8-bit
31
32     ; Clear PPU registers
33     ldx #$33
34 @loop: stz $2100,x
35     stz $4200,x
36     dex
37     bpl @loop
38
39     ; Set background color to $03E0
40     lda #$E0
41     sta $2122
42     lda #$03
43     sta $2122
44
45     ; Maximum screen brightness
46     lda #$0F
47     sta $2100
48
49 forever:
50     jmp forever
```

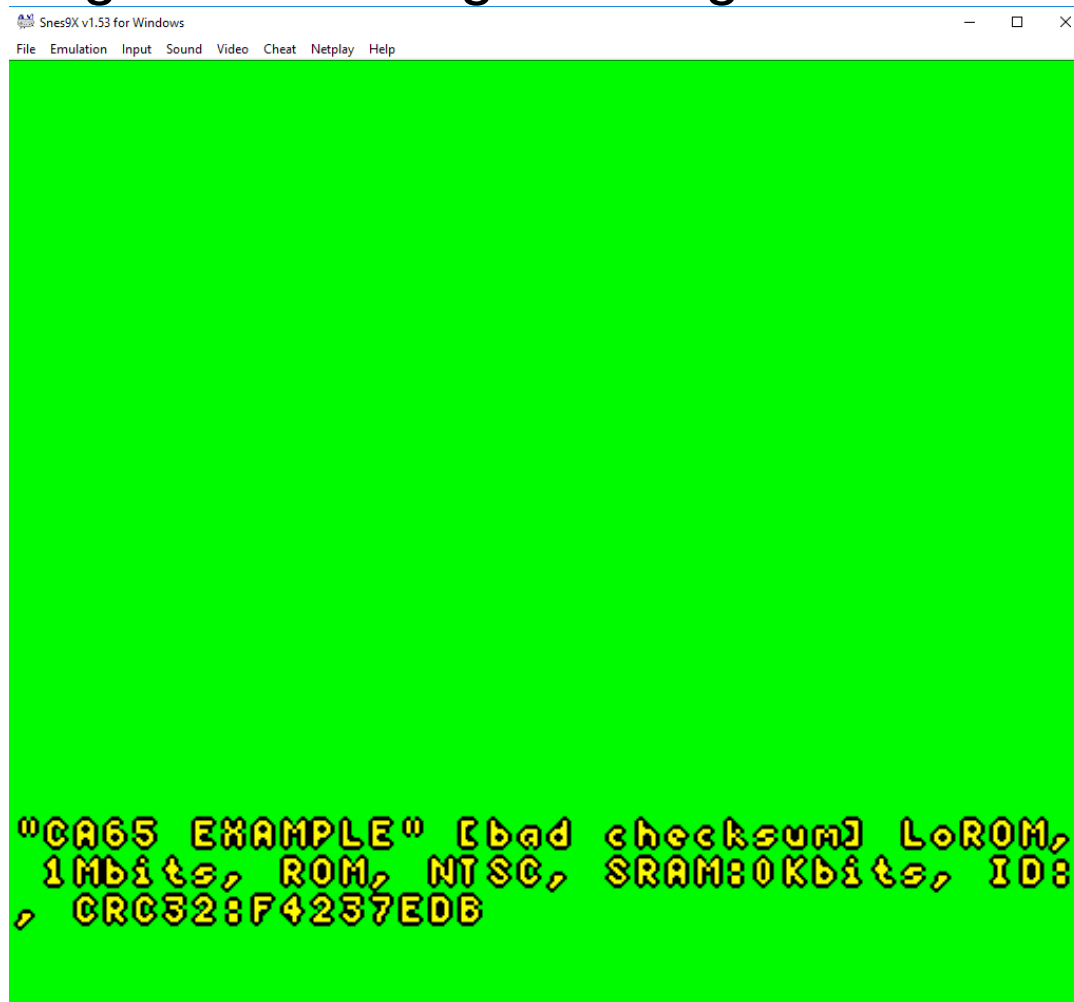
ca65 ca65.s

ld65 -C lorom128.cfg -o ca65.smc ca65.o

CC65 Intro

Erzeugte ca65.smc Datei testen (Bsp. Snes9X)

Das Programm erzeugt einen grünen Bildschirm



CC65 Code

ca65.smc Inhalt:

```
18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80
```

Inhalt mit Hexeditor: (Bsp HxD)

[illegible]

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21
80

Disassembliert:

0x00: CLC

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21

Disassembliert:

0x00: CLC

0x01: XCE

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21
80

Disassembliert:

0x00: CLC

0x01: XCE

0x02: REP

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21
80

Disassembliert:

0x00: CLC

0x01: XCE

0x02: REP 0b00010000

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21
80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21
80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21
80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21
80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL 0xF7

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL 0xF7
0x12: LDA

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL 0xF7
0x12: LDA 0xE0

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL 0xF7
0x12: LDA 0xE0
0x14: STA

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL 0xF7
0x12: LDA 0xE0
0x14: STA 0x2122

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL 0xF7
0x12: LDA 0xE0
0x14: STA 0x2122
0x17: LDA

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL 0xF7
0x12: LDA 0xE0
0x14: STA 0x2122
0x17: LDA 0x03

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL 0xF7
0x12: LDA 0xE0
0x14: STA 0x2122
0x17: LDA 0x03
0x19: STA

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL 0xF7
0x12: LDA 0xE0
0x14: STA 0x2122
0x17: LDA 0x03
0x19: STA 0x2122

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL 0xF7
0x12: LDA 0xE0
0x14: STA 0x2122
0x17: LDA 0x03
0x19: STA 0x2122
0x1B: LDA

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL 0xF7
0x12: LDA 0xE0
0x14: STA 0x2122
0x17: LDA 0x03
0x19: STA 0x2122
0x1C: LDA 0x0F

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL 0xF7
0x12: LDA 0xE0
0x14: STA 0x2122
0x17: LDA 0x03
0x19: STA 0x2122
0x1C: LDA 0x0F
0x1E: STA

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL 0xF7
0x12: LDA 0xE0
0x14: STA 0x2122
0x17: LDA 0x03
0x19: STA 0x2122
0x1C: LDA 0x0F
0x1E: STA 0x2100

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL 0xF7
0x12: LDA 0xE0
0x14: STA 0x2122
0x17: LDA 0x03
0x19: STA 0x2122
0x1C: LDA 0x0F
0x1E: STA 0x2100
0x21: JMP

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

```
0x00: CLC
0x01: XCE
0x02: REP 0b00010000
0x04: SEP 0b00100000
0x06: LDX 0x0033
0x09: STZ 0x2100
0x0C: STZ 0x4200
0x0F: DEX
0x10: BPL 0xF7
0x12: LDA 0xE0
0x14: STA 0x2122
0x17: LDA 0x03
0x19: STA 0x2122
0x1C: LDA 0x0F
0x1E: STA 0x2100
0x21: JMP 0x8021
```

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Disassembliert:

0x00: CLC	; Clear Carry Flag
0x01: XCE	; Switch Carry Flag and E Flag. E=0 means 16 Bit native Mode
0x02: REP 0b00010000	; Reset Processor Status Bits. Sets X Flag=0. X and Y operate in 16 Bit Mode
0x04: SEP 0b00100000	; Set Processor Status Bits. Set M Flag=1. A operate in 8 Bit Mode
0x06: LDX 0x0033	; Store a 0x33 (0b00110011) in the X register
0x09: STZ 0x2100	; Move a Zero to memory address 0x2100+x
0x0C: STZ 0x4200	; Move a Zero to memory address 0x4200+x
0x0F: DEX	; Decrement X. X=X-1
0x10: BPL 0xF7	; Jump to 0x09 if N Flag is 0 (That means X>0) 0xF7 is a -9 0x12+0xF7=0x09
0x12: LDA 0xE0	; Load A. Store a 0xE0 (0b11100000) in the A register
0x14: STA 0x2122	; Move a 0xE0 to the memory adress 0x2122
0x17: LDA 0x03	; Load A. Store a 0x03 (0b00000111) in the A register
0x19: STA 0x2122	; Move a 0x03 to the memory adress 0x2122
0x1C: LDA 0x0F	; Load A. Store a 0x0F (0b00001111) in the A register
0x1E: STA 0x2100	; Move a 0x0F to the memory adress 0x2100
0x21: JMP 0x8021	; Jump to 0x8021 (im LoRom = 0x0021)

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Pseudo Code:

```
0x00: CLC                ; Modus: Native 16 Bit,
0x01: XCE                ; Modus: X/Y Register 16 Bit, A Register 8 Bit
0x02: REP 0b00010000    ;
0x04: SEP 0b00100000    ; do {
0x06: LDX 0x0033        ;     X = 0x33
0x09: STZ 0x2100        ;     RAM[2100+x] = 0
0x0C: STZ 0x4200        ;     RAM[4200+x] = 0
0x0F: DEX                ;     X=X-1
0x10: BPL 0xF7          ; while(X>0)
0x12: LDA 0xE0          ; RAM[2122] = 0x30E0
0x14: STA 0x2122        ;
0x17: LDA 0x03          ;
0x19: STA 0x2122        ;
0x1C: LDA 0x0F          ; RAM[2100] = 0x0F
0x1E: STA 0x2100        ;
0x21: JMP 0x8021        ; while(true) {}
```

CC65 Code

ca65.smc Inhalt:

18 FB C2 10 E2 20 A2 33 00 9E 00 21 9E 00 42 CA 10 F7 A9 E0 8D 22 21 A9 03 8D 22 21 A9 0F 8D 00 21 4C 21 80

Pseudo Code:

```
0x00: CLC                ; Modus: Native 16 Bit,
0x01: XCE                ; Modus: X/Y Register 16 Bit, A Register 8 Bit
0x02: REP 0b00010000    ;
0x04: SEP 0b00100000    ; do {
0x06: LDX 0x0033        ;     X = 0x33
0x09: STZ 0x2100        ;     RAM[2100+x] = 0
0x0C: STZ 0x4200        ;     RAM[4200+x] = 0
0x0F: DEX                ;     X=X-1
0x10: BPL 0xF7          ; while(X>0)
0x12: LDA 0xE0          ; RAM[2122] = 0x30E0
0x14: STA 0x2122        ;
0x17: LDA 0x03          ;
0x19: STA 0x2122        ;
0x1C: LDA 0x0F          ; RAM[2100] = 0x0F
0x1E: STA 0x2100        ;
0x21: JMP 0x8021        ; while(true) {}
```

```
24  .segment "CODE"
25
26  reset:
27      clc                ; native mode
28      xce
29      rep #$10          ; X/Y 16-bit
30      sep #$20          ; A 8-bit
31
32      ; Clear PPU registers
33      ldx #$33
34 @loop: stz $2100,x
35      stz $4200,x
36      dex
37      bpl @loop
38
39      ; Set background color to $03E0
40      lda #$E0
41      sta $2122
42      lda #$03
43      sta $2122
44
45      ; Maximum screen brightness
46      lda #$0F
47      sta $2100
48
49  forever:
50      jmp forever
```


CC65 Konfigurationsdatei

lorom128.cfg:

- Die Datei enthält Konstanten, welche Man später im Code (insb. Header nutzen kann)
- Zwei Abschnitte: MEMORY und SEGMENTS
- Physikalische ROM Adresse (MEMORY): Dort packt der Assembler den Code in die smc Datei
- Logische Adresse (SEGMENTS): Referenziert die Konstanten aus Memory um andere Konstanten Zu definieren. Diese nutzt der Assembler um Adressen umzurechnen.
- Beispiel: Der Code beginnt bei 0x0000 in der smc Datei. Sprünge finden aber zu 0x8000 statt, da 0x8000 (im LoROM) die logische Adresse von 0x0000 ist. (z.B. JMP 0x8021 springt zu 0x0021)
- Die ROM hat eine Größe von 128 KByte. Die letzte Adresse ist 0x1FFFF. Eine Bank ist 32KByte groß. D.h. 4 Bänke (Bank 0 – Bank 3)
- In LoROM beginnt der Header bei 0xFFC0 (siehe MemMap Slides)

```
1 # ca65 linker config for 128K SMC
2 # https://wiki.superfamicom.org/basic-ca65-usage-for-snes-programming
3 # Physical areas of memory
4 # Names need not match, but it makes it easier to remember if they do.
5 MEMORY {
6     ZEROPAGE: start = 0, size = $100;
7     BSS:      start = $200, size = $1800;
8     ROM:      start = $8000, size = $8000, fill = yes;
9     BANK1:    start = $18000, size = $8000, fill = yes;
10    BANK2:    start = $28000, size = $8000, fill = yes;
11    BANK3:    start = $38000, size = $8000, fill = yes;
12 }
13
14 # Logical areas code/data can be put into.
15 SEGMENTS {
16     ZEROPAGE: load = ZEROPAGE, type = zp;
17     BSS:      load = BSS,      type = bss, align = $100;
18
19     CODE:     load = ROM,      align = $8000;
20     RODATA:   load = ROM;
21     HEADER:   load = ROM,      start = $FFC0;
22     ROMINFO:  load = ROM,      start = $FFD5, optional = yes;
23     VECTORS:  load = ROM,      start = $FFE0;
24
25     # The extra three banks
26     BANK1:    load = BANK1,    align = $8000, optional = yes;
27     BANK2:    load = BANK2,    align = $8000, optional = yes;
28     BANK3:    load = BANK3,    align = $8000, optional = yes;
29 }
```

CC65 Header

ca65.s:

- .segment referenziert Konfig Datei
Beispiel: ROMINFO ist 0xFFC0
und CODE ist 0x0000
- Genaue Beschreibung des Headers auf
den Memory Map Folien.
- VECTORS sind 16 Bit Zeiger auf
(Interrupt-)Programmblöcke. Reset
Ist der Entry-Point

```
HxD - [C:\Users\User1\Desktop\Emu\Tools\cc65-snapshot-win32\bin\ca65.smc]
File Edit Search View Analysis Extras Window ?
16 ANSI hex
ca65.smc
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00007FC0 43 41 36 35 20 45 58 41 4D 50 4C 45 00 00 00 00 CA65 EXAMPLE....
00007FD0 00 00 00 00 00 30 00 07 00 00 00 00 AA AA 55 55 .....0.....**UU
00007FE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007FF0 00 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 .....€..
```

```
HxD - [C:\Users\User1\Desktop\Emu\Tools\cc65-snapshot-win32\bin\ca65.smc]
File Edit Search View Analysis Extras Window ?
16 ANSI hex
ca65.smc
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0001FFF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

```
1 ; Minimal example of using ca65 to build SNES ROM.
2 ;
3 ; ca65 ca65.s
4 ; ld65 -C lorom128.cfg -o ca65.smc ca65.o
5
6 .p816 ; 65816 processor
7 .i16 ; X/Y are 16 bits
8 .a8 ; A is 8 bits
9
10 .segment "HEADER" ; +$7FE0 in file
11 .byte "CA65 EXAMPLE" ; ROM name
12
13 .segment "ROMINFO" ; +$7FD5 in file
14 .byte $30 ; LoROM, fast-capable
15 .byte 0 ; no battery RAM
16 .byte $07 ; 128K ROM
17 .byte 0,0,0,0
18 .word $AAAA,$5555 ; dummy checksum and complement
19
20 .segment "VECTORS"
21 .word 0, 0, 0, 0, 0, 0, 0, 0, 0
22 .word 0, 0, 0, 0, 0, 0, 0, reset, 0
```