

CHIP-8

CHIP8 ist ein Interpreter für die [CHIP-8-Sprache](#) 🎮 [CHIP-8](#). Entwickelt habe ich diesen Interpreter 2013 für das [KC-Club Treffen 2013](#).

Speicherbelegung

| | |
|------------------|-------------|
| Dateiname | CHIP8.COM |
| Laden in | OS |
| Programmstandort | 0300h-10FFh |
| Arbeitsspeicher | 2000h-2EFFh |
| OS-Kommando | CHIP8 |

Anleitung

Nach Start meldet sich das Programm mit der Oberfläche und einer kurzen Hilfe:



Mit STOP (Strg-C) wird der CHIP8-Interpreter beendet.

LIST startet den HEX-Editor.



Mit den Cursortasten bewegt man sich im virtuellen RAM-Bereich des CHIP-8-Systems. Hex-Werte werden einfach überschrieben, mit STOP oder ESC beendet man den HEX-Editor. RUN startet das Programm direkt.

Durch Betätigen der Taste <L> werden CHIP8-Programme vom Datenträger geladen; mit <S> wird das aktuelle Programm gespeichert. Bei der Arbeit mit USB oder Diskette werden zuvor die vorhandenen Programme aufgelistet:



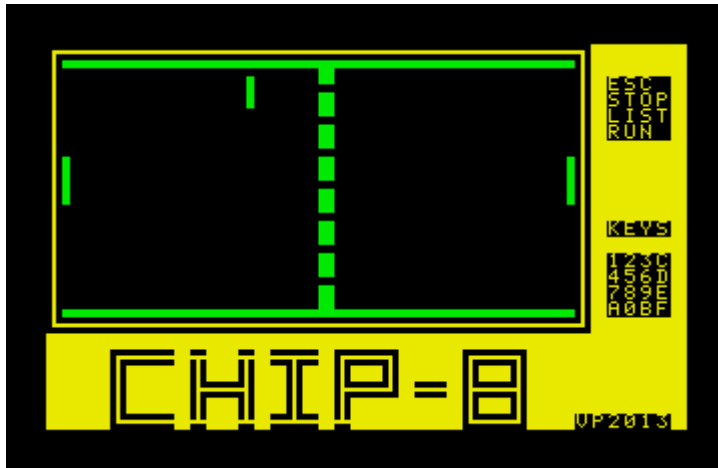
Als Dateierdung wird automatisch CH8 vergeben.

Hinweis: Bei der Arbeit mit USB oder Diskette werden die Programme OHNE den typischen KC-Vorblock gespeichert. Die CH8-Dateien sind hier reine Binärdateien und kompatibel zu den im Internet verfügbaren CHIP-8-Programmen.

RUN startet das aktuelle Programm. Im Fenster laufen die CHIP8-Programme auf einem 64×32 Pixel großen „Bildschirm“.

Mit den Tasten <1>..<<4> werden 4 integrierte Programme geladen und gleich gestartet.

1. IBM-Logo
Anzeige des Logos, Ende mit ESC
2. Breakout
der Schläger wird links-rechts mit <4>-<6> bewegt.
3. Panzer (s. [KC-Club Treffen 2013](#))
Panzer bewegen hoch <2> - rechts <6> - runter <8> - links <4>
4. Pong
linker Spieler hoch <1> runter <4> rechter Spieler hoch <C> runter <D>



Innerhalb des Programmlaufs wird mit den 16 Tasten <0>..<<9>, <A>..<<F> gesteuert. Die konkrete Bedienung ist beim jeweiligen Programm aufgeführt. Mit ESC wird das Programm beendet, man landet wieder im Startbildschirm.

Die 16 Tasten sind beim Original in folgender Matrix angeordnet. Bei einigen Programmen spielt diese Anordnung eine Rolle, z.B. als Steuerkreuz mit den Tasten 2 - 6 - 8 - 4 oder links-rechts mit A und B und 0 als Aktionstaste. Die Tastenanordnung ist auch auf dem Bildschirm dargestellt.

```

+---+---+---+---+
| 1 | 2 | 3 | C |
+---+---+---+---+
| 4 | 5 | 6 | D |
+---+---+---+---+
| 7 | 8 | 9 | E |
+---+---+---+---+
| A | 0 | B | F |
+---+---+---+---+

```

Links

- <http://www.chip8.com> Die größte Sammlung an Programmen und Infos zu CHIP-8, Handbuch des COSMAC VIP
- <http://mattmik.com/documents.html> die komplette Sammlung der VIPER-Magazine für die COSMAC VIP-Computer.

CHIP-8

Details zur Programmiersprache s. [CHIP-8](#).

| | |
|----------------------|---|
| virtueller Prozessor | 16 Register V0..VF Index-/Adressregister I (Stackregister SP) (Program Counter PC) Delay Timer DT Sound Timer ST |
|----------------------|---|

| | |
|----------|--------------------|
| Grafik | 64×32 Pixel, Torus |
| Tastatur | 4×4-Tastenfeld |
| Sound | Beepton |

Die Virtuelle Maschine

Speicher

Die CHIP-8 Speicher-Adressen liegen im Bereich von 200h bis FFFh, das reicht für 3.584 Bytes. Der Grund für den Speicher ab 200h ist, dass im VIP Cosmac und Telmac 1800 die ersten 512 Byte für den CHIP8-Interpreter reserviert sind. Auf diesen Maschinen wurden die obersten 256 Bytes (F00h-FFFh auf einer 4K-Maschine) für die Anzeige aktualisieren vorbehalten, und die 96 Byte unterhalb (EA0H-EFFh) wurden für den Call-Stack, den internen Gebrauch, und die Variablen vorbehalten.

Register

CHIP-8 verfügt über 16 8-Bit-Register V0..VF. Das VF-Register dient auch als Carry-Flag.

- 16 x 8-Bit-Register V0..VF
- 16-Bit-Index-/Adressregister I
- 8-Bit-Register Delay Timer DT
- 8-Bit-Register Sound Timer ST
- (16-Bit Stackregister SP)
- (16-Bit Program Counter PC)

Stack

Der Stack wird nur verwendet, um die Rückkehr-Adressen zu speichern, wenn Unterprogramme aufgerufen werden. Original ist Speicher für bis zu 12 Verschachtelungsebenen vorhanden.

Timer

CHIP-8 verfügt über zwei Timer. Beide werden automatisch mit 60 Hz dekrementiert, bis sie 0 erreichen. Delay Timer DT: Dieser Timer soll für das Timing der Ereignisse von Spielen verwendet werden. Sein Wert kann eingestellt und gelesen werden. Sound- Timer ST: Dieser Timer ist für Sound-Effekte gedacht. Solange der Wert ungleich Null ist, wird ein Piepton erzeugt.

Tastatur

Die Eingabe erfolgt mit einer Hex-Tastatur mit 16 Tasten von 0 bis F. '8', '4', '6' und '2' dienen in der Regel als Cursortasten. Es gibt drei Opcodes zur Tastaturabfrage. Eine überspringt eine Anweisung, wenn eine bestimmte Taste gedrückt wird, eine weitere überspringt eine Anweisung, wenn eine bestimmte Taste nicht gedrückt wird. Die dritte wartet auf einen Tastendruck, und speichert die Taste

dann in einem der Datenregister.

Grafik und Sound

Die Display-Auflösung beträgt 64×32 Pixel, und die Farbe ist einfarbig.

```
+-----+
| (0,0)          (63,0) | (x,y)
|                  |
|                  |
| (0,31)         (63,31) |
+-----+
```

Grafiken werden auf dem Bildschirm allein durch Sprites gezeichnet, die 8 Pixel breit sind und von 1 bis 15 Pixel hoch sein können. Sprite-Pixel, die gesetzt sind, invertieren die Farbe der entsprechenden Bildschirm-Pixel, während nicht gesetzten Sprite-Pixel nichts verändern.

Beim Zeichnen der Sprites werden 8 Pixel ab Position (x,y) gezeichnet, dann 8 Pixel ab Position (x,y+1) usw.

Wenn beim Zeichnen des Sprites alle Bildschirm-Pixel invertiert wurden, wird das Carry-Flag (VF) auf 1 gesetzt, sonst ist es 0.

Wie zuvor beschrieben, wird ein Signalton abgespielt, wenn der Wert der Sound Timer ungleich Null ist.

Opcode Tabelle

CHIP-8 verfügt über 35 Opcodes, die alle zwei Byte lang sind. Das höchstwertige Byte wird zuerst gespeichert. Die Opcodes sind unten in hexadezimal und mit den folgenden Symbolen aufgelistet:

- mmm: Adresse 200..EFF
- kk: 8-Bit-Konstante 00..FF
- n: 4-Bit-Konstante 0..F
- x und y: 4-Bit Register-Nr 0..F

| Hex | Symbolisch | Assembler | Beschreibung |
|------|----------------|--------------|--|
| 1mmm | GO mmm | JP addr | Go to 0MMM |
| 8mmm | GO mmm+V0 | JP V0, addr | Go to 0MMM + V0 |
| 2mmm | DO mmm | CALL addr | Do subroutine at 0MMM (must end with 00EE) |
| 00EE | RET | RET | Return from subroutine |
| 3xkk | SKIP;Vx EQ kk | SE Vx, byte | Skip next instruction if VX = KK |
| 4xkk | SKIP;Vx NE kk | SNE Vx, byte | Skip next instruction if VX <> KK |
| 5xy0 | SKIP;Vx EQ Vy | SE Vx, Vy | Skip next instruction if VX = VY |
| 9xy0 | SKIP;Vx NE Vy | SNE Vx, Vy | Skip next instruction if VX <> VY |
| Ex9E | SKIP;Vx EQ KEY | SKP Vx | Skip next instruction if VX = Hex key (LSD) |
| ExA1 | SKIP;Vx NE KEY | SKNP Vx | Skip next instruction if VX <> Hex key (LSD) |

| Hex | Symbolisch | Assembler | Beschreibung |
|------|---------------|--------------------|---|
| 6xkk | Vx=kk | LD Vx, byte | Let VX = KK |
| Cxkk | Vx=RND | RND Vx, byte | Let VX = Random Byte (KK = Mask) |
| 7xkk | Vx=Vx+kk | ADD Vx, byte | Let VX = VX + KK |
| 8xy0 | Vx=Vy | LD Vx, Vy | Let VX = VY |
| 8xy1 | Vx=Vx/Vy | OR Vx, Vy | Let VX = VX / VY (VF changed) |
| 8xy2 | Vx=Vx&Vy | AND Vx, Vy | Let VX = VX & VY (VF changed) |
| 8xy4 | Vx=Vx+Vy | ADD Vx, Vy | Let VX = VX + VY (VF = 00 if VX + VY <= FF, VF = 01 if VX + VY > FF) |
| 8xy5 | Vx=Vx-Vy | SUB Vx, Vy | Let VX = VX - VY (VF = 00 if VX < VY, VF = 01 if VX >= VY) |
| Fx07 | Vx=TIME | LD Vx, DT | Let VX = current timer value |
| Fx0A | Vx=KEY | LD Vx, K | Let VX = hex key digit (waits for any key pressed) |
| Fx15 | TIME=Vx | LD DT, Vx | Set timer = VX (01 = 1/60 second) |
| Fx18 | SND=Vx | LD ST, Vx | Set tone duration = VX (01 = 1/60 second) |
| Ammm | I=mmm | LD I, addr | Let I = 0MMM |
| Fx1E | I=I+Vx | ADD I, Vx | Let I = I + VX |
| Fx29 | I=Vx(LSDP) | LD F, Vx | Let I = 5-byte display pattern for LSD of VX |
| Fx33 | MI=Vx(3DD) | LD B, Vx | Let MI = 3-decimal digit equivalent of VX (I unchanged) |
| Fx55 | MI=V0:Vx | LD [I], Vx | Let MI = V0 : VX (I = I + X + 1) |
| Fx65 | V0:Vx=MI | LD Vx, [I] | Let V0 : VX = MI (I = I + X + 1) |
| 00E0 | ERASE | CLS | Erase display (all 0's) |
| DxyN | SHOW nMI@VxVy | DRW Vx, Vy, nibble | Show n-byte MI pattern at VX-VY coordinates. I unchanged. MI pattern is combined with existing display via EXCLUSIVE-OR function. VF = 01 if a 1 in MI pattern matches 1 in existing display. |
| 0mmm | MLS@mmm | SYS addr | Do 1802 machine language subroutine at 0MMM (subroutine must end with D4 byte) |

Die Assemblerbezeichnungen entsprechen <http://devernay.free.fr/hacks/chip8/C8TECH10.HTM>

Beispiel: wandernde Acht

Programm

```
200: A2 10 61 00 62 00 D1 25
208: D1 25 71 01 72 01 12 06
210: F0 90 F0 90 F0 00 00 00
```

Assembler-Code (erstellt mit c8dasm)

```
L200: LD    I,  #210          ; A210    Index auf L210
      LD    V1, #00          ; 6100    Startposition (0,0)
      LD    V2, #00          ; 6200
L206: DRW   V1, V2, #5        ; D125    "8" zeichnen (5 Bytes ab I)
      DRW   V1, V2, #5        ; D125    "8" löschen
      ADD   V1, #01          ; 7101    Position nach rechts und unten
```

```
    ADD  V2, #01          ; 7201
    JP   L206             ; 1206   und neu zeichnen
;
L210: db #F0, #90, #F0, #90, #F0 ;   Sprite "8"
```

From:

<https://hc-ddr.hucki.net/wiki/> - Homecomputer DDR

Permanent link:

https://hc-ddr.hucki.net/wiki/doku.php/z9001/software/chip_8?rev=1613722485

Last update: **2021/02/19 08:14**

