Chip-8 techn. Referenz

http://devernay.free.fr/hacks/chip8/C8TECH10.HTM

Cowgod's Chip-8 Technische Referenz v1.0

ins Deutsche übersetzt und leicht gekürzt: V.Pohlers 2021

Verwendung dieses Dokuments

Bei der Erstellung dieses Dokuments habe ich alle Anstrengungen unternommen, um das Lesen zu vereinfachen und das Gesuchte zu finden. In den meisten Fällen, wenn ein Hexadezimalwert angegeben wird, folgt auf den entsprechenden Dezimalwert in Klammern. Zum Beispiel „0x200 (512)“.

In den meisten Fällen bezieht sich ein kursiv geschriebenes Wort oder ein kursiver Buchstabe auf einen variablen Wert. Wenn ich beispielsweise „Vx“ schreibe, verweist das x auf einen 4-Bit-Wert.

Über Chip-8

Immer wenn ich jemandem erwähne, dass ich einen Chip-8-Interpreter schreibe, ist die Antwort immer dieselbe: „Was ist ein Chip-8?“

Chip-8 ist eine einfache, interpretierte Programmiersprache, die erstmals in den späten 1970er und frühen 1980er Jahren auf einigen Do-it-yourself- Computersystemen verwendet wurde. Die Computer COSMAC VIP, DREAM 6800 und ETI 660 sind nur einige Beispiele. Diese Computer wurden normalerweise für die Verwendung eines Fernsehgeräts als Anzeige entwickelt, hatten zwischen 1 und 4 KB RAM und verwendeten eine hexadezimale Tastatur mit 16 Tasten für die Eingabe. Der Interpreter nahm nur 512 Byte Speicherplatz ein, und Programme, die hexadezimal in den Computer eingegeben wurden, waren noch kleiner.

In den frühen neunziger Jahren wurde die Chip-8-Sprache von einem Mann namens Andreas Gustafsson wiederbelebt. Er erstellte einen Chip-8-Interpreter für den HP48-Grafikrechner namens Chip-48. Dem HP48 fehlte zu dieser Zeit eine Möglichkeit, schnell schnelle Spiele zu machen, und Chip-8 war die Antwort. Chip-48 brachte später Super Chip-48 hervor, eine Modifikation von Chip-48, die Grafiken mit höherer Auflösung sowie andere grafische Verbesserungen ermöglichte.

Chip-48 inspirierte eine ganze Reihe neuer Chip-8-Interpreter für verschiedene Plattformen, darunter MS-DOS, Windows 3.1, Amiga, HP48, MSX, Adam und ColecoVision. Ich habe mich mit Chip-8 beschäftigt, nachdem ich auf Paul Robsons Interpreter im World Wide Web gestoßen bin. Kurz danach begann ich meinen eigenen Chip-8-Interpreter zu schreiben.

Dieses Dokument ist eine Zusammenstellung aller verschiedenen Informationsquellen, die ich beim Programmieren meines Interpreters verwendet habe.

Chip-8-Spezifikationen

In diesem Abschnitt werden der Chip-8-Speicher, die Register, die Anzeige, die Tastatur und die Timer beschrieben.

Speicher

Die Chip-8-Sprache kann von Speicherplatz 0x000 (0) bis 0xFFF (4095) auf bis zu 4 KB (4.096 Byte) RAM zugreifen. In den ersten 512 Bytes von 0x000 bis 0x1FF befand sich der ursprüngliche Interpreter und sollte nicht von Programmen verwendet werden. Die meisten Chip-8-Programme beginnen bei 0x200 (512), einige jedoch bei 0x600 (1536). Programme, die bei 0x600 beginnen, sind für den ETI 660-Computer vorgesehen.

Speicherbelegung:

```
+ --------------- + = 0xFFF (4095) Ende des Chip-8-RAM
|                 |
|                 |
|                 |
|                 |
|                 |
| 0x200 bis 0xFFF |
|     Chip-8      |
| Programm/Daten- |
|     Platz       |
|                 |
|                 |
+ --------------- + = 0x200 (512) Start der meisten Chip-8-Programme
| 0x000 bis 0x1FF |
| Reserviert für  |
| Interpreter     |
+ --------------- + = 0x000 (0) Start des Chip-8-RAM
```

Register

Chip-8 verfügt über 16 Allzweck-8-Bit-Register, die üblicherweise mit Vx bezeichnet werden, wobei x eine hexadezimale Ziffer (0 bis F) ist. Es gibt auch ein 16-Bit-Register namens I. Dieses Register wird im Allgemeinen zum Speichern von Speicheradressen verwendet, es werden normalerweise nur die niederwertigen (rechten) 12 Bits verwendet. Das VF-Register sollte von keinem Programm verwendet werden, da es von einigen Anweisungen als Flag verwendet wird. Weitere Informationen finden Sie in Abschnitt 3.0, Anweisungen.

Chip-8 verfügt außerdem über zwei spezielle 8-Bit-Register für die Verzögerungs- und Sound-Timer. Wenn diese Register nicht Null sind, werden sie automatisch mit einer Rate von 60 Hz dekrementiert. Weitere Informationen hierzu finden Sie in Abschnitt 2.5, Timer und Sound.

Es gibt auch einige „Pseudoregister“, auf die über Chip-8-Programme nicht zugegriffen werden kann.

Der Programmzähler (PC) sollte 16-Bit sein und wird zum Speichern der aktuell ausgeführten Adresse verwendet. Der Stapelzeiger (SP) kann 8-Bit sein und wird verwendet, um auf die oberste Ebene des Stapels zu zeigen.

Der Stapel ist ein Array von 16 16-Bit-Werten, die zum Speichern der Adresse verwendet werden, zu der der Interpreter zurückkehren soll, wenn er mit einem Unterprogramm fertig ist. Chip-8 ermöglicht bis zu 16 Ebenen verschachtelter Unterprogramme.

Tastatur

Die Computer, die ursprünglich die Chip-8-Sprache verwendeten, hatten eine hexadezimale Tastatur mit 16 Tasten und folgendem Layout:

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

Dieses Layout muss in verschiedene andere Konfigurationen abgebildet werden, um auf die Tastaturen heutiger Plattformen zu passen.

Anzeige

Bei der ursprünglichen Implementierung der Chip-8-Sprache wurde eine 64 x 32 Pixel große monochrome Anzeige mit diesem Format verwendet:

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

Einige andere Interpreter, insbesondere der des ETI 660, hatten ebenfalls die Modi 64×48 und 64×64. Meines Wissens unterstützt kein aktueller Interpreter diese Modi. In jüngerer Zeit hat Super Chip-48, ein Interpreter für den HP48- Rechner, einen 128×64-Pixel-Modus hinzugefügt. Dieser Modus wird jetzt von den meisten Interpretern auf anderen Plattformen unterstützt.

Chip-8 zeichnet mithilfe von Sprites Grafiken auf dem Bildschirm. Ein Sprite ist eine Gruppe von Bytes, die eine binäre Darstellung des gewünschten Bildes sind. Chip-8-Sprites können bis zu 15 Byte groß sein, bei einer möglichen Sprite-Größe von 8×15.

Programme können sich auch auf eine Gruppe von Sprites beziehen, die die hexadezimalen Ziffern 0 bis F darstellen. Diese Sprites sind 5 Byte groß (8×5 Pixel). Die Daten sollten im Interpreterbereich

des Chip-8-Speichers (0x000 bis 0x1FF) gespeichert werden.

Timer & Sound

Chip-8 bietet 2 Timer, einen Delay-Timer und einen Sound-Timer.

Der Verzögerungszeitgeber ist immer dann aktiv, wenn das Verzögerungszeitgeberregister (DT) ungleich Null ist. Dieser Timer subtrahiert lediglich 1 mit einer Rate von 60 Hz vom Wert von DT. Wenn DT 0 erreicht, wird es deaktiviert.

Der Sound-Timer ist immer dann aktiv, wenn das Sound-Timer-Register (ST) ungleich Null ist. Dieser Timer verringert sich ebenfalls mit einer Rate von 60 Hz. Solange jedoch der ST-Wert größer als Null ist, ertönt der Chip-8-Summer. Wenn ST Null erreicht, wird der Sound-Timer deaktiviert.

Der vom Chip-8-Interpreter erzeugte Ton hat nur einen Ton. Die Frequenz dieses Tons wird vom Autor des Interpreters festgelegt.

Chip-8-Anweisungen

Die ursprüngliche Implementierung der Chip-8-Sprache enthält 36 verschiedene Anweisungen, einschließlich Mathematik-, Grafik- und Flusssteuerungsfunktionen.

Super Chip-48 fügte weitere 10 Befehle hinzu, insgesamt 46.

Alle Befehle sind 2 Byte lang und werden zuerst mit dem höchstwertigen Byte gespeichert. Im Speicher sollte sich das erste Byte jedes Befehls an einer geraden Adresse befinden. Wenn ein Programm Sprite-Daten enthält, sollten diese aufgefüllt werden, damit alle darauf folgenden Anweisungen ordnungsgemäß im RAM gespeichert werden.

Dieses Dokument enthält keine Beschreibungen der Super Chip-48-Anweisungen.

In diesen Auflistungen werden die folgenden Variablen verwendet:

nnn oder addr - Ein 12-Bit-Wert, die niedrigsten 12 Bits des Befehls
n oder Nibble - Ein 4-Bit-Wert, die niedrigsten 4 Bits des Befehls
x - Ein 4-Bit-Wert, die unteren 4 Bits des High-Bytes des Befehl
y - Ein 4-Bit-Wert, die oberen 4 Bits des unteren Bytes des Befehls
kk oder Byte - Ein 8-Bit-Wert, die niedrigsten 8 Bits des Befehls

Standard-Chip-8-Befehle

0nnn - SYS addr

Springt zu einer Maschinencode-Routine bei nnn.

Diese Anweisung wird nur auf den alten Computern verwendet, auf denen Chip-8 ursprünglich implementiert war. Es wird von modernen Interpretern ignoriert.

00E0 - CLS

Löscht die Anzeige.

00EE - RET

Rückkehr von einem Unterprogramm.

Der Interpreter setzt den Programmzähler auf die oberstes Adresse im Stapel und subtrahiert dann 1 vom Stapelzeiger.

1nnn - JP addr

Wechsel zu Adresse nnn.

Der Interpreter setzt den Programmzähler auf nnn.

2nnn - CALL addr

Aufruf Unterprogramm bei nnn.

Der Interpreter erhöht den Stapelzeiger und legt dann den aktuellen PC oben auf den Stapel. Der PC wird dann auf nnn gesetzt.

3xkk - SE Vx, byte

Nächste Anweisung überspringen, wenn Vx = kk.

Der Interpreter vergleicht das Register Vx mit kk und erhöht, wenn sie gleich sind, den Programmzähler um 2.

4xkk - SNE Vx, byte

Nächste Anweisung überspringen, wenn Vx != kk.

Der Interpreter vergleicht das Register Vx mit kk, und wenn sie nicht gleich sind, erhöht er den Programmzähler um 2.

5xy0 - SE Vx, Vy

Nächste Anweisung überspringen, wenn Vx=Vy.

Der Interpreter vergleicht das Register Vx mit dem Register Vy und erhöht, wenn sie gleich sind, den

Programmzähler um 2.

6xkk - LD Vx, Byte

Setzt Vx = kk.

Der Interpreter legt den Wert kk in das Register Vx.

7xkk - ADD Vx,Byte

Setzt Vx = Vx + kk.

Addiert den Wert kk zum Wert des Registers Vx und speichert das Ergebnis in Vx.

8xy0 - LD Vx, Vy

Setzt Vx = Vy.

Speichert den Wert des Registers Vy im Register Vx.

8xy1 - OR Vx, Vy

Setzt Vx = Vx ODER Vy.

Führt ein bitweises ODER für die Werte von Vx und Vy durch und speichert dann das Ergebnis in Vx. Ein bitweises ODER vergleicht die entsprechenden Bits von zwei Werten, und wenn eines der Bits 1 ist, ist das gleiche Bit im Ergebnis auch 1. Andernfalls ist es 0.

8xy2 - AND Vx, Vy

Setze Vx = Vx UND Vy.

Führt ein bitweises UND für die Werte von Vx und Vy durch und speichert dann das Ergebnis in Vx. Ein bitweises UND vergleicht die entsprechenden Bits von zwei Werten, und wenn beide Bits 1 sind, ist das gleiche Bit im Ergebnis auch 1. Andernfalls ist es 0.

8xy3 - XOR Vx, Vy

Setze Vx = Vx XOR Vy.

Führt ein bitweises exklusives ODER für die Werte von Vx und Vy durch und speichert das Ergebnis in Vx. Ein exklusives ODER vergleicht die entsprechenden Bits von zwei Werten, und wenn die Bits nicht beide gleich sind, wird das entsprechende Bit im Ergebnis auf 1 gesetzt. Andernfalls ist es 0.

8xy4 - ADD Vx, Vy Set Vx = Vx + Vy, setze VF = Carry/Überlauf.

Die Werte von Vx und Vy werden addiert. Wenn das Ergebnis größer als 8 Bits ist (d.h. > 255), wird VF auf 1 gesetzt, andernfalls auf 0. Nur die niedrigsten 8 Bits des Ergebnisses werden beibehalten und in Vx gespeichert.

8xy5 - SUB Vx, Vy

Setze Vx = Vx - Vy, setze VF = KEIN Borrow/Unterlauf.

Wenn Vx > Vy ist, wird VF auf 1 gesetzt, andernfalls auf 0. Dann wird Vy von Vx subtrahiert und die Ergebnisse in Vx gespeichert.

8xy6 - SHR Vx {, Vy}

Setze Vx = Vx SHR 1.

Wenn das niedrigstwertige Bit von Vx 1 ist, wird VF auf 1 gesetzt, andernfalls 0. Dann wird Vx durch 2 geteilt.

8xy7 - SUBN Vx, Vy

Setze Vx = Vy - Vx, setze VF = KEIN Borrow/Unterlauf.

Wenn Vy > Vx ist, wird VF auf 1 gesetzt, andernfalls auf 0. Dann wird Vx von Vy subtrahiert und die Ergebnisse in Vx gespeichert.

8xyE - SHL Vx {, Vy}

Setze Vx = Vx SHL 1.

Wenn das höchstwertige Bit von Vx 1 ist, wird VF auf 1 gesetzt, andernfalls auf 0. Dann wird Vx mit 2 multipliziert.

9xy0 - SNE Vx, Vy

Nächste Anweisung überspringen, wenn Vx != Vy.

Die Werte von Vx und Vy werden verglichen, und wenn sie nicht gleich sind, wird der Programmzähler um 2 erhöht.

Annn - LD I, addr

Setze I = nnn.

Der Wert von Register I wird auf nnn gesetzt.

Bnnn - JP V0, addr

Zur Position nnn + V0 springen.

Der Programmzähler wird auf nnn plus den Wert von V0 gesetzt.

Cxkk - RND Vx, Byte

Setze Vx = zufälliges Byte UND kk.

Der Interpreter erzeugt eine Zufallszahl von 0 bis 255, die dann mit dem Wert kk UND-verknüpft wird. Die Ergebnisse werden in Vx gespeichert. Weitere Informationen zu UND finden Sie in Anweisung 8xy2.

Dxyn - DRW Vx, Vy, nibble

Anzeige eines n-Byte-Sprites beginnend an Speicherstelle I bei (Vx, Vy), setze VF = Kollision.

Der Interpreter liest n Bytes aus dem Speicher, beginnend mit der in I gespeicherten Adresse. Diese Bytes werden dann als Sprites auf dem Bildschirm bei den Koordinaten (Vx, Vy)angezeigt. Sprites werden auf dem vorhandenen Bildschirm XOR-verknüpft. Wenn dadurch Pixel gelöscht werden, wird VF auf 1 und andernfalls auf 0 gesetzt. Wenn das Sprite so positioniert ist, dass ein Teil davon außerhalb der Koordinaten der Anzeige liegt, wird es auf die gegenüberliegende Seite des Bildschirms gezeichnet. Siehe Anweisung 8xy3 für weitere Informationen zu XOR und Abschnitt 2.4 Anzeige für weitere Informationen zum Chip-8-Bildschirm und zu den Sprites.

Ex9E - SKP Vx

Nächste Anweisung überspringen, wenn die Taste mit dem Wert Vx gedrückt wird.

Überprüft die Tastatur, und wenn sich die dem Wert von Vx entsprechende Taste derzeit in der unteren Position befindet, wird der PC um 2 erhöht.

ExA1 - SKNP Vx

Nächste Anweisung überspringen, wenn die Taste mit dem Wert von Vx nicht gedrückt wird.

Überprüft die Tastatur, und wenn sich die dem Wert von Vx entsprechende Taste derzeit in der oberen Position befindet, wird der PC um 2 erhöht.

Fx07 - LD Vx, DT

Setze Vx = Verzögerungszeitgeberwert.

Der Wert von DT wird in Vx gesetzt.

Fx0A - LD Vx, K

Warten auf einen Tastendruck und Speichern des Werts der Taste in Vx.

Die gesamte Ausführung stoppt, bis eine Taste gedrückt wird, und der Wert dieser Taste wird in Vx gespeichert.

Fx15 - LD DT, Vx

Verzögerungszeitgeber = Vx einstellen.

DT wird gleich dem Wert von Vx gesetzt.

Fx18 - LD ST, Vx

Sound Timer = Vx einstellen.

ST wird gleich dem Wert von Vx gesetzt.

Fx1E - ADD I, Vx

Setze I = I + Vx.

Die Werte von I und Vx werden addiert und die Ergebnisse in I gespeichert.

Fx29 - LD F, Vx

Setze I = Position des Sprites für die Ziffer Vx.

Der Wert von I wird auf die Position für das hexadezimale Sprite gesetzt, die dem Wert von Vx entspricht. Weitere Informationen zur hexadezimalen Chip-8- Schriftart finden Sie in Abschnitt 2.4, Anzeige.

Fx33 - LD B, Vx

Speichern der BCD-Darstellung von Vx an den Speicherplätzen I, I + 1 und I + 2.

Der Interpreter nimmt den Dezimalwert von Vx und platziert die Hunderterstelle im Speicher an der Stelle in I, die Zehnerstelle an der Stelle I + 1 und die Einerstelle an der Stelle I + 2.

Fx55 - LD [I], Vx

Speichern der Register V0 bis Vx im Speicher ab Position I.

Der Interpreter kopiert die Werte der Register V0 bis Vx ab der Adresse in I.

Fx65 - LD in den Speicher Vx, [I]

Lese-Register V0 bis Vx aus dem Speicher ab Position I.

Der Interpreter liest Werte aus dem Speicher ab Position I in die Register V0 bis Vx.

Credits

Dieses Dokument wurde von Thomas P. Greene zusammengestellt.

Quellen sind:

1. Mein eigenes Hacking.
2. E-Mail zwischen David Winter und mir.
3. David Winters Chip-8-Emulator-Dokumentation.
4. Christian Egebergs Chipper-Dokumentation.
5. Marcel de Kogels Vision-8-Quellcode.
6. Paul Hayters DREAM MON-Dokumentation.
7. Paul Robsons Webseite.
8. Andreas Gustafssons Chip-48-Dokumentation.