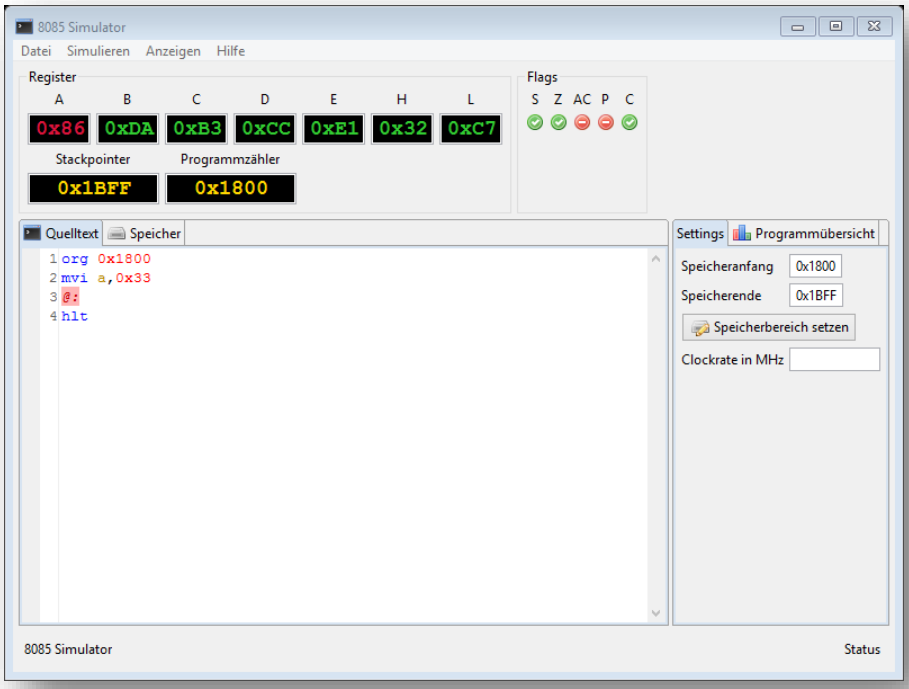
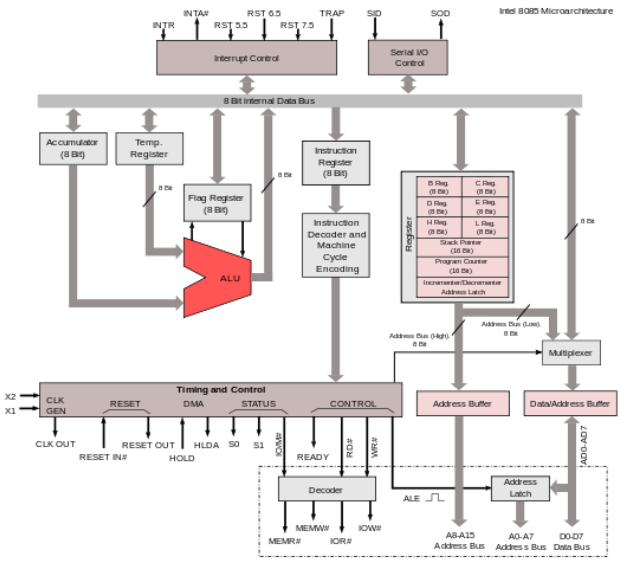
8085 Simulator



Florian Schleich
Hochschule Fulda
22.06.2015
florian.schleich@gmail.com

Was ist der 8085 Simulator?

Die vorliegende Software soll möglichst genau einen Prozessor der Reihe 8085 nachbilden. Der Simulator ist grundlegend dafür gedacht, eigene Programme zu kompilieren und für den weiteren Gebrauch auf dem Prozessor zu testen.

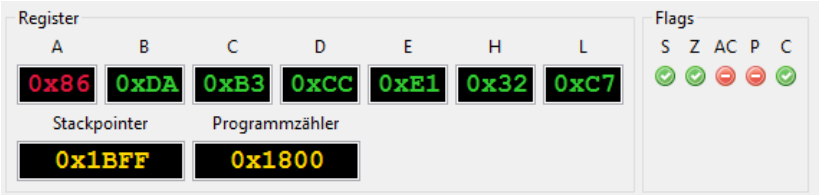


Inhaltsverzeichnis

Teile der Bedienoberfläche

Die Bedienoberfläche ist übersichtlich aufgebaut.

Im oberen Teil des Programmfensters befinden sich Textfelder und Anzeigen, die über den aktuellen Status des Prozessors informieren.

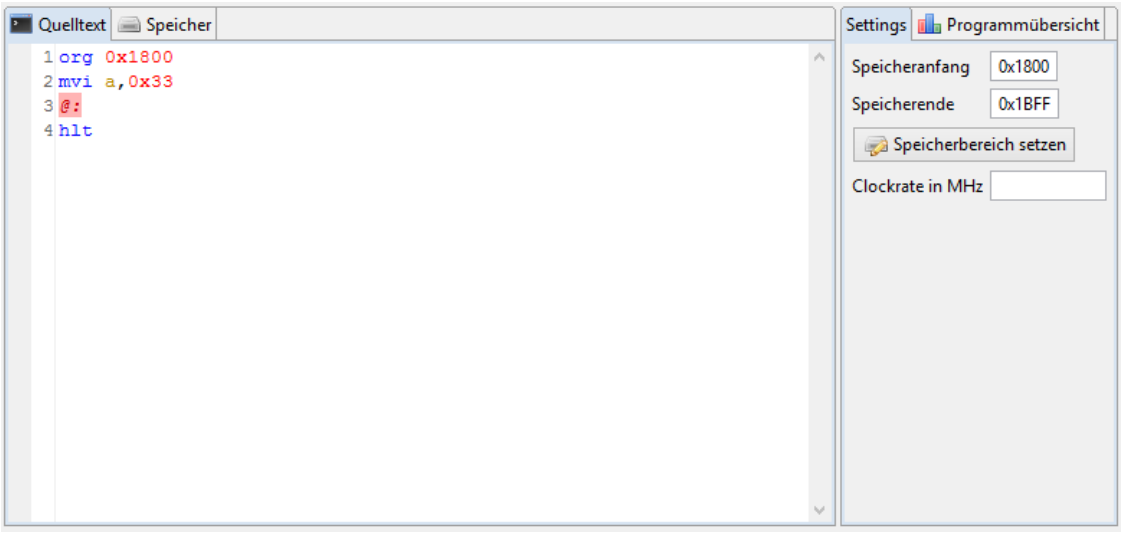


Register

Die Register A-E, H und L werden beim Programmstart automatisch mit Zufallswerten initialisiert. Der Stackpointer und der Programmzähler werden jeweils mit den sinnvollen Werten aus dem Speicherbereich befüllt. Der Stackpointer wird dem entsprechend mit dem Ende des Speicherbereiches initialisiert, der Programmzähler startet am Anfang des vorgegebenen Speichers.

Flags

Die farbigen Anzeigen auf der rechten Seite des oberen Programmfensters informieren über den aktuellen Zustand der Flags. Diese werden während der Ausführung eines Programmes im Simulator entsprechend ihrer Bedeutung gesetzt oder gelöscht. Ein gesetztes Flag erkennen Sie an einem grünen Haken. Ist unter dem Flag ein roter Hinweis, so ist das Flag nicht gesetzt.



Im unteren Teil befinden sich die Eingabemaske für den Quelltext und den Speicherbreich, wobei der Speicherbereich hier lediglich angezeigt werden kann. Eine Manipulation der Speicherzellen erfolgt über die Ausführung der jeweiligen Mnemonics. Der Simulator unterstützt Syntaxhighlighting bei der Eingabe des Quelltextes. Mnemonics werden derzeit nur erkannt, wenn die Eingabe durch Kleinbuchstaben erfolgt. Siehe dazu das Kapitel Syntax.

Im unteren Bereich befinden sich auch einige Einstellmöglichkeiten für den simulierten Prozessor.

Speicherbereich anpassen

Der Speicherbereich kann unter „Settings“ angepasst werden. In den Feldern Speicheranfang und Speicherende kann mittels hexadezimaler Zahlen ein Speicherbereich definiert werden, auf den der Prozessor zugreifen kann. Wenn der Speicherbereich geändert wurde muss diese Änderung mit einem Klick auf „Speicherbreich ändern“ bestätigt werden.

Achtung!
Dabei wir ein bereits kompiliertes Programm aus dem Speicher gelöscht und durch Zufallswerte neu initialisiert.

Taktfrequenz einstellen

Die Anpassung der Taktfrequenz erfolgt ohne Bestätigung. Geben Sie lediglich die Taktfrequenz in MHz in das dafür vorgesehene Textfeld ein. Eine Taktfrequenz wird lediglich bei der Simulation bis zu dem nächsten Breakpoint benötigt.

Programmübersicht

Im Reiter „Programmübersicht“ erfahren Sie nach der Übersetzung des Programmes wie viele Anweisungen im Quelltext zu übersetzen waren und wie viele Byte das Programm benötigt. In Anhängigkeit des verwendeten Speicherbereichs wird auch die Prozentuale Speicherauslastung des Programmes gemessen.

Ein Programm simulieren

In diesem Kapitel erfahren Sie wie Sie ein Programm eingeben, danach übersetzen und schließlich simulieren lassen können.

Den Quelltext eingeben

Im Reiter „Quelltext“ können Sie den Programmablauf in Assembler eingeben. Das Syntaxhighlighting unterstützt Sie dabei Mnemonics und Zahlenwerte voneinander zu unterschieden.

Wenn Mnemonics in Kleinbuchstaben eingegeben werden und der Simulator dieses Mnemonic unterstützt wird es als blaues Wort dargestellt. Sollte das eigegebene Kommando nicht blau dargestellt werden, so liegt wahrscheinlich ein Schreibfehler vor. So können im Vorfeld schon einige Übersetzungsfehler vermieden werden.

Für jedes Kommando muss eine eigene Zeile verwendet werden. Nur so lässt sich ein Programm fehlerfrei übersetzen.

Damit der Simulator weiß, welche die Basisadresse des Programmes ist, ist es nötig die Pseudoanweisung „org“ zu verwenden. Diese muss immer als erste Anweisung im Quelltext stehen. Anhand dieser Basisadresse werden nachher alle Unteradressen des Programmes berechnet.

Den Quelltext kompilieren

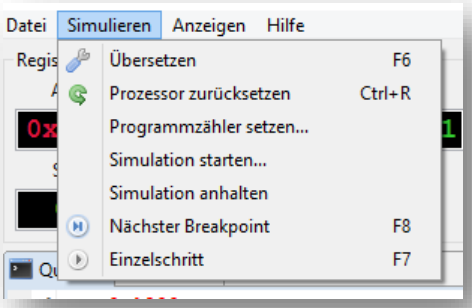
Ist das Programm vollständig eingegeben, kann es in die Maschinensprache übersetzt werden. Öffnen Sie dazu das Menü „Simulieren“ und wählen danach den Menüpunkt „Übersetzen“. Das Programm wird nun auf Syntaxfehler untersucht. Sollten keine Syntaxfehler gefunden werden, wird das Programm automatisch übersetzt und in den Speicher eingetragen.

Achtung!
Eine fehlerhafte Syntax führt derzeit noch zu einem kompletten Programmabsturz. Daher sollte ein eingegebenes Programm immer vorher in eine Datei gespeichert werden.

Sie können das Programm dann im Reiter „Speicher“ kontrollieren.

Programmzähler einstellen

Für eine sinnvolle Simulation sollte zunächst der Programmzähler auf die Startadresse des Programmes eingestellt werden. Dies ist standardmäßig die Adresse 0x1800. Sollte die Startadresse davon abweichen kann über den Menüpunkt „Simulation > Programmzähler setzen…“ eine andere Startadresse gewählt werden.



Das Programm simulieren

Der Simulator unterstützt einige Simulationsmodi. Alle Simulationsmodi haben aber gemeinsam, dass sie nur funktionieren, wenn der Quelltext vorher übersetzt wurde.

Automatische Simulation

Eine vollständig automatisch ablaufende Simulation starten Sie über das Menü „Simulation“ und den Menüpunkt „Simulation starten…“. Danach wird das Programm an der Stelle auf die der Programmzähler zeigt gestartet. Sollte in dem Programm eine Endlosschleife existieren oder der Programmauflauf verläuft nicht wie gedacht, kann die Simulation mit dem Kommando „Simulation > Simulation anhalten“ abgebrochen werden. Der Prozessor behält dabei aber seine aktuelle Konfiguration.

Einzelschrittsimulation

Die Einzelschrittsimulation startet ebenso wie die vollautomatische Simulation an der Adresse die der aktuelle Programmzähler vorgibt. Die Einzelschrittsimulation kann entweder mit der Taste „F7“ oder über das Menü „Simulation“ gestartet werden. Bei der Einzelschrittsimulation wird lediglich das Kommando ausgeführt auf das der Programmzähler im Moment zeigt. Dadurch ist ein effizientes Debugging möglich.

Simulation bis zum nächsten Breakpoint

Für diese Simulationsart muss mindestens ein Breakpoint im Quelltext existieren. Die Simulation startet wie gewohnt bei der aktuellen Adresse und wird dann solange ausgeführt, bis der Simulator auf eine Unterbrechung (Breakpoint) stößt. Danach werden die vergangene Simulationszeit und die Anzahl der simulierten Anweisungen angezeigt.

Syntax

Der Simulator benutzt eine eigene Syntax, dazu in diesem Kapitel mehr.

Mnemonics

Mnemonics sollten nur mit Kleinbuchstaben eingegeben werden, da sonst das Syntaxhighlighting nicht funktioniert. Der eingebaute Kompiler macht allerdings keinen Unterschied zwischen Groß- und Kleinbuchstaben.

ORG-Operator

Der ORG-Operator gibt die Basisadresse des Programmes an. org sollte die erste Anweisung des Programmes sein, weil danach die Adressen der Labels und Unterprogrammaufrufe errechnet werden. Der ORG-Operator kann beliebig oft im Programm verwendet werden. Die Programmteile nach dieser Anweisung werden dann in den entsprechenden Speicherbereich geschrieben.

Zahlenwerte

Zahlenwerte werden stets in hexadezimaler Schreibweise in den Quellcode eingefügt. Dabei wird folgende Schreibweise verwendet:

```
0x1E = 30 (dezimal)
```

Labels

Labels kennzeichnen Unterprogramme mittels einem Namen. Ein Label wird mit einem Doppelpunkt am Zeilenende gekennzeichnet. Nach der Übersetzung wird in dieser Zeile auch ein Label-Symbol im Quelltexteditor angezeigt.

Bei Unterprogrammaufrufen (JMP oder CALL) kann dann auf dieses Label Bezug genommen werden. Dabei wird der Name ohne Doppelpunkt verwendet.

```
Label:
mvi a,0x30
…
call Label
```

Breakpoints

Breakpoints dienen dazu, die automatische Simulation an bestimmten Stellen zu unterbrechen und den aktuellen Zustand des Prozessors oder des Speichers zu begutachten.

Ein Breakpoint wird wie folgt in den Quelltext eingelassen.

```
adi b,0x11
@:
mvi b,0x44
```

Die Simulation würde dann vor der Ausführung von mvi b,0x44 stoppen. Mit dem erneuten Start einer Simulation oder der EInzelschrittsimulation würde dann dieser Breakpoint übersprungen

werden. Ideal ist die Verwendung eines Breakpoints während oder nach einer Schleife, um das Ergebnis gleich überprüfen zu können.

In der Zeile die als Breakpoint definiert wurde, dürfen keine weiteren Anweisungen folgen. Erst die nächste Zeile kann wieder normal genutzt werden.

Kommentare

Kommentare werden mit einem Semikolon eingeleitet und im Quelltext in einer grünen Farbe dargestellt. Kommentare werden vom Übersetzer sowie vom Simulator ignoriert und dienen lediglich zur Dokumentation des Quelltextes.

Download des Quelltextes

Der hier beschriebene Simulator ist OpenSource und kann als Git-Repository heruntergeladen werden. Wer den Quelltext nicht selbst übersetzen möchte oder kann, kann sich auch für einige Versionsstände die vorkompilierten Programmpakete herunter laden. Den gesamten Quelltext sowie die Programmpakete finden Sie auf: https://github.com/thetodd/8085-Simulator

Dort ist auch eine Möglichkeit gegeben, Fehler oder Verbesserungsvorschläge einzureichen. Den Bereich finden Sie rechts im Menü unter „Issues“ oder direkt mit diesem Link: https://github.com/thetodd/8085-Simulator/issues