2010

RiSa

Tutorial

http://risa.gym-vilshofen.de
r i s a @ g m x . e u

Letzte Änderung: 27.06.2010

Inhalt

1 RiSa: Tutorial

Ein neues Anwendungsprogramm erlernt man am schnellsten, wenn man es benutzt. Zwei Beispiele sollen dabei helfen.

1.1 Addition zweier Zahlen mit dem Befehlssatz: Standard

Die Aufgabenstellung ist simpel: Es sollen zwei positive, natürliche Zahlen addiert werden. Das Ergebnis soll am Ende in einer Zelle des Speichers stehen.

Zuerst startet man RiSa. Standardmäßig legt RiSa 32 Zellen im Speicher an. Dies kann man einfach durch das Auswahlmenü rechts oben ändern, was nun gleich ausprobiert werden soll, indem man die Anzahl der Speicherstellen auf 8 verändert.

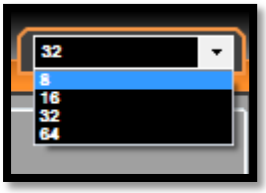

Abbildung 1 - Durch das Auswahlmenü kann die Speichergröße geändert werden

Nun wechselt man in die Editor-Ansicht durch einen Klick auf den Button „Editor".

Abbildung 2 - RiSa in der Editor-Ansicht

Das zu erstellende Programm soll den bezeichnenden Namen „Addition" erhalten. Also wird dies in das Feld hinter „Name" eingetippt.

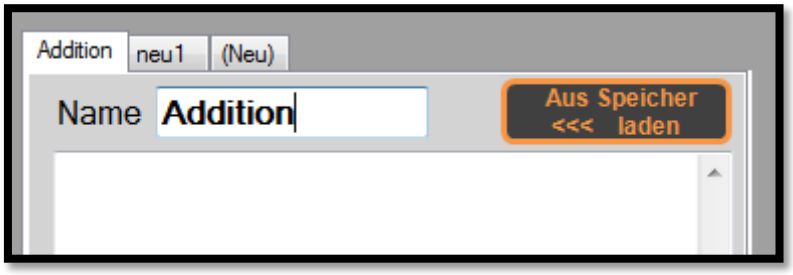

Abbildung 3 - Eingabe des Namens

Man beachte: Der Name des Tabs ändert sich automatisch passend.

Als Nächstes gibt man den Code ein, der zwei Zahlen addiert. Es wird dabei angenommen, dass die beiden zu addierenden Zahlen in den Speicherzellen 5 und 6 stehen. Das Ergebnis soll in Zelle 7 geschrieben werden.

```
LOAD 5
ADD 6
STORE 7
HLT
// Dies ist ein Kommentar
```

Der erste Befehl kopiert den Inhalt von Zelle 5 in den Akkumulator (AKK). Dann wird mit dem ADD-Befehl der Wert der Speicherzelle 6 zum Wert des Akkumulators hinzuaddiert und das Ergebnis im Akkumulator gespeichert. Nun ist es noch nötig, den Inhalt von AKK an die Speicherstelle 7 zu kopieren. Am Ende wird das Programm korrekt mit einem HLT beendet. Kommentare werden durch „//" gekennzeichnet.

Abbildung 4 - Im Editor eingegebener Code

Bevor der Code ausprobiert wird, soll er zur Sicherheit gespeichert werden. Ein Klick auf den „Speichern"-Button bewerkstelligt dies.

RiSa schlägt nun einen Dateinamen vor, der dem Namen des Programms entspricht.

Dieser kann jedoch beliebig geändert werden.

Übrigens:

RiSa speichert seine Dateien mit der Endung „.ris" in reiner Textform. Die Dateien können mit jedem beliebigen Editor bearbeitet, betrachtet oder ausgedruckt werden.

Damit der eingegebene Code nun ausgeführt werden kann, muss er zuerst in den Speicher übertragen werden. Nach einem Klick auf den Button „...in Speicher laden >>>" sollte RiSa nun folgendermaßen aussehen:

Abbildung 5 - RiSa nach dem Laden von Code in den Speicher

Die Speicherzellen, in welchen der Code steht, sollten blau gefärbt sein. Das bedeutet, dass die Eingabe in der Zelle syntaktisch korrekt ist. Ist dies nicht der Fall, wird die Zelle rot gefärbt.

Es fehlen noch die zu addierenden Zahlen. Diese werden nun direkt in den Speicher eingetragen. Dazu klickt man auf die Speicherstelle 5 bzw. 6 und gibt die Werte ein:

```
16
26
```

RiSa sieht nun so aus:

Abbildung 6 - RiSa nach Eingabe der zu addierenden Zahlen

Da alle Speicherzellen Korrektheit signalisieren, kann die Simulation durch einen Klick auf den blauen „Play-Button" gestartet werden. Durch erneutes Klicken auf diesen Button wird nun Schritt für Schritt das Programm ausgeführt. Um sich Veränderungen zwischen zwei Schritten zu verdeutlichen, kann man mit dem „Zurück-Button" jederzeit zurückgehen. Ungeduldige Anwender starten einen schnellen Vorlauf oder springen gleich zum Ende der Simulation. Ist das Programm am Ende angelangt, steht das Ergebnis der Addition („42") in Speicherzelle 7 und der Prozessor befindet sich im Zustand „HALT".

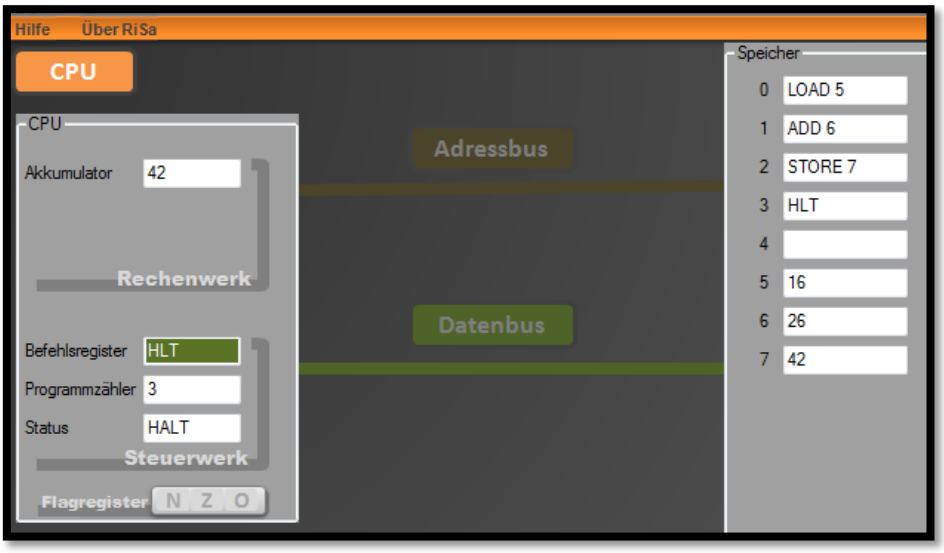

Abbildung 7 - Das Ergebnis des Simulationslaufs

Mittels des Buttons „Simulation beenden" wechselt man nun wieder in den Editier-Modus zurück. Um die zu editierenden Zahlen auch in den Code im Editor aufzunehmen, klickt man nun im Editor auf „Aus Speicher laden". Zum Beachten hierbei ist, dass eventuell bestehende Kommentare im Editor auf diesem Wege überschrieben werden, da in den Speicherstellen selbst keine Kommentare gespeichert werden können.

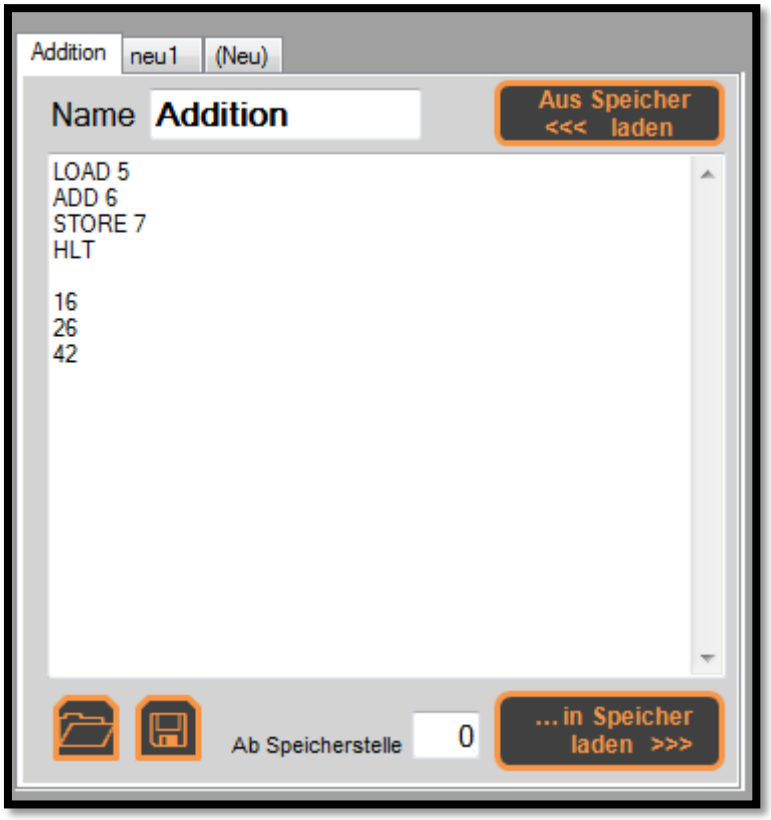

Abbildung 8 - Editor mit aus dem Speicher geladenen Werten

Das geänderte Programm kann erneut gespeichert werden.

1.2 Addition zweier Zahlen mit dem Befehlssatz: Minimal

Um die Auswirkungen der verschiedenen Befehlssätze auszuprobieren, wechselt man diesen nun auf „Minimal".

Abbildung 9 - Auswählen des Befehlssatzes "Minimal"

Da eine Änderung des Befehlssatzes ein Löschen des Inhalts der Speicherzellen bewirkt, müssen diese neu mit den Werten aus dem Editor belegt werden. Einen Klick im Editor (auf „...in Speicher laden >>>") später, steht man vor einem Problem (oder besser gesagt mehreren roten Feldern). Die Befehle „ADD", „LOAD" und „STORE" werden nicht mehr unterstützt, was die Speicherzellen 0-2 unmissverständlich klar machen.

Abbildung 10 – Mehrere Befehle werden vom Befehlssatz "Minimal" nicht unterstützt

Also muss das Programm entsprechend geändert werden. Die folgende Lösung[1] benötigt aber mehr Platz (sprich Speicherzellen) als die bisherige. 16 Speicherzellen sollten aber genügen, daher wird diese Anzahl ausgewählt.

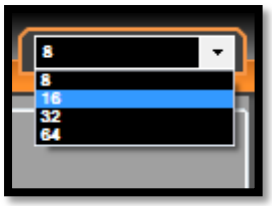

Abbildung 11 - Ändern der Anzahl der Speicherzellen

[1] Ohne Berücksichtigung, dass mindestens einer der Summanden kleiner gleich 0 ist.

Bevor es nun ans Umgestalten des Programms geht, sei noch eine wichtige Änderung, die mit dem Befehlssatz „Minimal" (und auch „Maximal") einhergeht, erwähnt: Die CPU hat zwei Register mehr bekommen. Ein Klick auf „CPU" und man sieht es deutlich.

Abbildung 12 - Die CPU hat nun auch die Register EBX und ECX

Neben diesen neuen Registern verändern sich die Befehle auch insofern, da sie nun stets zwei Operanden benötigen. Das Ziel, z.B. AKK (für den Akkumulator), EBX oder ECX oder eine Speicherstelle. Und die Quelle, welche ebenfalls je nach Befehl ein Register oder eine Speicherstelle sein kann.

Genug der Verwirrung, denn jetzt soll es ans Ausprobieren gehen. Daher ändert man im Editor das Programm wie folgt ab:

```
MOV AKK 7
MOV ECX 8
DEC ECX
INC AKK
JNZ 2
MOV 9 AKK
HLT
```

Zur Erklärung: Der Befehl MOV kopiert in der ersten Zeile den Wert, welcher in Speicherstelle 7 steht, in den Akkumulator. Der Befehl ADD wird durch die Zeilen 2 bis 5 ersetzt. Zuerst wird der Inhalt der Zelle 8 in das Register ECX kopiert, dann dieser um 1 vermindert. Danach wird mit INC AKK der Wert des Akkumulators um 1 erhöht. Nun überprüft JNZ 2, ob ECX null ist. Ist dies nicht der Fall, wird an Speicherstelle 2 gesprungen. Sprich, es wurde hier eine Schleife programmiert, die so lange den Wert von AKK um 1 erhöht und den von ECX um 1 reduziert, bis ECX gleich null ist. Erst wenn dies der Fall ist, wird die Schleife beendet.

Nach dem Laden des Programms in den Speicher und Eintragen der zu addierenden Werte in die Speicherzellen an Adresse 7 und 8 sieht RiSa so aus:

Abbildung 13 - Addition mit Befehlssatz "Minimal"

Nach dem Starten der Simulation und einem signifikant längeren Programmablauf steht das Ergebnis der Addition in Zelle 9.

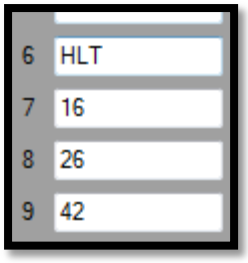

Abbildung 14 - Auch mit dem Befehlssatz "Minimal" lässt sich korrekt addieren

Mittels der Auswahl von verschiedenen Befehlssätzen lässt sich also gut aufzeigen, dass so simpel anmutende Befehle wie das Addieren intern in einer Zentraleinheit durchaus aufwendig realisiert sein können. Der dabei erhöhte Speicherstellenbedarf zeigt sich klar dadurch, dass der Benutzer gezwungen ist, die Größe des Speichers zu ändern.

1.3 Multiplikation zweier Zahlen

Diese Aufgabe erscheint ebenfalls simpel: Es sollen zwei positive, natürliche Zahlen multipliziert werden und das Ergebnis am Ende in einer Speicherzelle stehen.

Mit Hilfe des Befehlssatzes „Standard" ist das Programm dazu schnell erstellt:

```
LOAD 5
MUL 6
STORE 7
HLT
```

Mit den in die Speicherzellen 5 und 6 eingetragenen Werten „7" und „6" liefert das Programm auch das bekannte Ergebnis „42".

Abbildung 15 - Multiplikation mit Befehlssatz "Standard"

Schwieriger wird es allerdings, wenn man das gleiche Programm mit dem Befehlssatz „Minimal" realisieren will. Die Idee der Lösung ist, den ersten zu multiplizierenden Wert so oft zum Akkumulator (der initial **0** ist) zu addieren, wie der Wert der zweiten zu multiplizierenden Zahl ist. Um 7 * 6 zu lösen, addiert man also 6-mal die Zahl 7 zu AKK:

```
0 + 7 + 7 + 7 + 7 + 7 + 7 = 42
```

Da das Addieren zweier Zahlen bereits in der vorhergehenden Aufgabe gelöst worden ist, muss nun nur noch eine Schleife „außen herum" implementiert werden, die die entsprechende Anzahl an Additionen regelt.

```
((((((0 + 7) + 7) + 7) + 7) + 7) + 7) = 42
```

Diese Überlegungen führen zu folgendem Programm[2]:

```
MOV EBX 17
MOV 18 AKK
MOV ECX EBX
DEC EBX
JNZ 7
HLT

MOV AKK 18
MOV ECX 16
DEC ECX
INC AKK
JNZ 9
JMP 1
```

Die zu multiplizierenden Werte sind in die Speicherzellen 16 und 17 einzutragen. Das Ergebnis der Berechnung wird in Zelle 18 kopiert.

Die Größe des Speichers ist entsprechend auf 32 Stellen zu erweitern. Die Beispielwerte „6" und „7" illustrieren, dass zur Berechnung des Ergebnisses („42") nun eine große Anzahl an Einzelschritten nötig ist.

Abbildung 16 - Multiplikation mit Befehlssatz "Minimal"

[2] Ohne Berücksichtigung, dass mindestens einer der beiden zu multiplizierenden Werte kleiner gleich 0 ist. Des Weiteren ist diese Lösung nicht die effizienteste, erfüllt aber gut den Aspekt der Veranschaulichung.

2 Links

RiSa Homepage

http://risa.gym-vilshofen.de