UMach Spezifikation

8. April 2012

Inhaltsverzeichnis

1 Einführung

UMach ist eine einfache virtuelle Maschine (VM), die einen definierten Instruktionssatz und eine definierte Architektur hat. UMach orientiert sich dabei an Prinzipien von RISC Architekturen: feste Instruktionslänge, kleine Anzahl von einfachen Befehlen, Speicherzugriff durch Load- und Store-Befehlen, u.s.w.

Für den Anwender der virtuellen Maschine wird zuerst eine Assembler-Sprache zur Verfügung gestellt. In dieser Sprache werden Programme geschrieben die anschließend kompiliert werden. Die kompilierte Dateien (Maschinen-Code) wird von der virtuellen Maschine ausgeführt.

1.1 Anwendungsbeispiel

```
LOAD R1 90
LOAD R2 09
REV  R3 R1
```

2 Struktur der UMach VM

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig ob ich schreibe: »Dies ist ein Blindtext« oder »Huardest gefburn«?. Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muß keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie »Lorem ipsum« dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

2.1 Betriebsmodi

Ein Betriebmodus bezieht sich auf die Art, wie die UMach VM läuft. Die UMach VM kann in einem der folgenden Betriebsmodi laufen:

1. Normalmodus
2. Einzelschrittmodus

Normalmodus Die virtuelle Maschine führt ohne Unterbrechung ein Programm aus. Nach der Ausführung befindet sich die Maschine in einem Wartezustand, falls sie nicht ausdrücklich ausgeschaltet wird.

Einzelschrittmodus Die virtuelle Maschine führt immer eine einzige Instruktion aus und nach der Ausführung wartet sie auf einen externen Signal um mit der nächsten Instruktion fortzufahren. Dieser Modus soll dem Entwickler erlauben, ein Programm schrittweise zu debuggen.

2.2 Aufbau

Dieser Abschnitt beschreibt den Aufbau der UMach virtuellen Maschine. Die virtuelle Maschine besteht aus internen und aus externen Komponenten. Dabei sind die externen Komponenten nicht wesentlich für die Funktionsfähigkeit der gesamten Maschine, d.h. die Maschine kann im Prinzip auch ohne die externen Komponenten funktionieren – in diesem Fall fehlt ihr eine Menge von Funktionen.

Interne Komponenten sind diejenigen Komponenten, die für die Funktionsfähigkeit der UMach Maschine wesentlich sind:

1. Recheneinheit
2. Logische Einheit
3. Register

Externe Komponenten

1. Anbindung an einem I/O-Port

2.2.1 Register

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig ob ich schreibe: »Dies ist ein Blindtext« oder »Huardest gefburn«?. Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muß keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie »Lorem ipsum« dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

2.2.2 Rechen- und logische Einheit

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so?

Ist es gleichgültig ob ich schreibe: »Dies ist ein Blindtext« oder »Huardest gefburn«?. Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muß keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie »Lorem ipsum« dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

2.2.3 I/O-Einheit

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig ob ich schreibe: »Dies ist ein Blindtext« oder »Huardest gefburn«?. Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muß keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie »Lorem ipsum« dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

2.3 Speichermodell

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig ob ich schreibe: »Dies ist ein Blindtext« oder »Huardest gefburn«?. Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muß keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie »Lorem ipsum« dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

2.4 Datentypen

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig ob ich schreibe: »Dies ist ein Blindtext« oder »Huardest gefburn«?.

Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muß keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie »Lorem ipsum« dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

3 Instruktionssatz

In diesem Abschnitt werden alle Instruktionen (Befehle) der UMach VM vorgestellt.

3.1 Instruktionsformat

Befehlsbreite Jeder UMach-Befehl hat eine feste Bitlänge von 32 Bit (4 mal 8 Bit). Das gilt unabhängig davon, was der Befehl tut. Befehle, die für ihren Informationsgehalt weniger als 32 Bit brauchen, wie z.B. `NOP`, werden mit Nullbits gefüllt. Alle Daten und Informationen, die mit einem Befehl übergeben werden, müssen in diesen 32 Bit untergebracht werden.

Byte Order Die Byte Order (Endianness) der gelesenen Bytes ist big-endian, die Byte-Reihenfolge, die für den Mensch selbstverständlich wäre (von links nach rechts lesen): die zuerst gelesenen 8 Bits sind die 8 höchstwertigen (Wertigkeiten 2^{31} bis 2^{24}) und die zuletzt gelesenen Bits sind die niedrigstwertigen (Wertigkeiten 2^7 bis 2^0). Bits werden in Stücken von n Bits gelesen, wobei $n = k \cdot 8$ und $k \in \mathbb{N} \setminus \{0\}$ (Byte für Byte).

Allgemeines Format Jeder Befehl (Instruktion) besteht aus zwei Teilen: der erste Teil ist 8 Bit lang und beinhaltet den tatsächlichen Befehl, bzw. die Operation, die von der UMach virtuellen Maschine ausgeführt werden soll. Diese 8 Bits belegen also die 8 höchstwertigen Bits in einem Befehl (Instruktionswort). Es gibt 256 Befehle, gemäß $2^8 = 256$. Die übrigen 24 Bit werden für Operanden oder Daten benutzt.

Binär	`0000`	`0001`	`0010`	`0011`
Hexa	`00`	`01`	`02`	`03`
Byte Order	erstes Byte	zweites Byte	drittes Byte	viertes Byte
Interpretation	Befehl	Operanden, Daten oder Füllbits		

Die Instruktionsformaten unterscheiden sich lediglich darin, dass sie die 24 Bits nach dem 8-Bitigen Befehl untershiedlich verwenden.

3.2 Instruktionen

Zur besseren Übersicht der verschiedenen UMach-Befehlen, unterteilen wir den Instruktionssatz der UMach virtuellen Maschine in den folgenden Kategorien:

1. Kontrollbefehle: das sind Befehle, die die Maschine selbst kontrollieren, wie z.B. den Betriebsmodus umschalten oder Ausschalten.

2. Arithmetische Befehle: Addieren, Subtrahieren etc. Alle arithmetische Befehle operieren auf Register Inhalte.

3. Logische Befehle, die eine logische Verknüpfung zwischen den Inhalten von Registern veranlassen.

4. Vergleichsbefehle: Vergleichen von Registerinhalten.

5. Sprünge.

6. Load- und Store-Befehle: die einzigen Befehle, die einen Zugriff auf den Speicher ausführen.

7. Input-Output-Befehle: operieren auf die I/O-Einheit der UMach VM.

Glossar

Befehl Die ersten 8 Bits in einer Instruktion. Operation code.

Byte eine Reihe oder Gruppe von 8 Bit.

Instruktion Eine Anweisung an die UMach VM etwas zu tun. Besteht aus einem Befehl (Operation Code) und eventuellen Argumenten.

Instruktionssatz Die Menge aller Instruktionen.

Index