Verwendung der virtuellen UMach Maschine

20. September 2012

Inhaltsverzeichnis

1 Einführung **3**
1.1 Beispiel 3

2 Assembler **5**

3 Debuggen **6**
3.1 Debug-Befehle 6

1 Einführung

1.1 Beispiel

Gleich am Anfang soll ein Beispiel für die Verwendung der UMach virtuellen Maschine und des Assemblers gegeben werden.

Wir haben ein UMach-Programm in eine normale Textdatei geschrieben. Das Programm kann sich über mehrere Dateien erstrecken, aber hier verwenden wir nur eine Datei. Das Programm sieht wie folgt aus:

```
    set r1 3
loop:
    dec r1
    cmp r1 zero
    be  finish
    jmp loop
finish:
    EOP
```

Dieses Programm wurde in der Datei `text.um` gespeichert (die Endung ist egal). Wir gehen davon aus, dass der Assembler `uasm`, die Programmdatei `test.um` und die virtuelle Maschine `umach` sich in dem selben Verzeichniss befinden. Sonst muss man die Befehle entsprechend anpassen.

Das Programm kann wie folgt assembliert werden:

```
./uasm -o prog.ux test.um
```

Die Option `-o` gibt die Ausgabedatei an. Wird diese Option nicht angegeben, so wird das assemblierte Programm in die Datei `u.out` geschrieben. Ergebnis des Assemblers is also die Datei `prog.ux`. Jetzt kann man diese Datei „ausführen“:

```
./umach prog.ux
```

Das Programm beendet sich ohne Ausgabe. Starten wird also das Programm im Debug-Modus (Option `-d`):

```
./umach -d prog.ux
```

Es wird die erste Anweisung angezeigt, der aktuelle Programm-Counter (Inhalt des Registers PC) und ein Prompt, der auf eine Eingabe von uns wartet. So könntes es weiter gehen:

```
[256]    SET    R1     3
umdb > show reg r1
R1 = 0x00000000 = 0
umdb > s
[260]    DEC    R1
umdb > show reg r1
R1 = 0x00000003 = 3
umdb > s
[264]    CMP    R1     ZERO
umdb > s
[268]    BE     2
umdb > s
[272]    JMP    -3
umdb > s
[260]    DEC    R1
umdb > s
[264]    CMP    R1     ZERO
umdb > show reg r1
R1 = 0x00000001 = 1
umdb > s
[268]    BE     2
umdb > s
[272]    JMP    -3
umdb > s
[260]    DEC    R1
umdb > s
[264]    CMP    R1     ZERO
umdb > s
[268]    BE     2
umdb > s
[276]    EOP
umdb > s
umdb > s
The maschine is not running.
umdb > show reg r1 cmpr
R1 = 0x00000000 = 0
CMPR = 0x00000000 = 0
umdb > q
```

2 Assembler

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 Debuggen

3.1 Debug-Befehle

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.