



<http://jump.to/oblivion>

Die Grundlage des Assemblers

Als Grundlage für die Assemblierung verwendet MICAsm eine Datei namens opcodes.ini.

Der Assembler kennt nur Befehle, die dort spezifiziert sind.

Format der Datei: Jede Zeile beschreibt einen Opcode wie folgt...

```
opcode mnemo[ parameter1[ parameter2]]
```

gültige Parameter sind: RA, RB, [RA], [RB], disp[RA], disp[RB], imm, [addr]

```
ad loopnz imm RB
```

```
40 inc RB
```

Assembler Sourcecodes

Assemblerbefehle und ihre Parameter

Ein Assemblerbefehl beginnt mit dem jeweiligen Mnemo.

Es folgen die Parameter; sind es zwei Parameter, so werden sie durch ein Komma getrennt.

Operanden im Sourcecode:

RA, RB	R0-R7
imm, addr, disp	eine Zahl oder die Bezeichnung eines Labels
[...], disp[...]	wie oben - nur mit entsprechenden Klammern; der Assembler unterstützt von sich aus disp[addr]

```
move 3, r5
```

```
ldsp 0x10[label1]
```

Achtung:

MICAsm verwendet die dem AT&T-Syntax ähnliche Parameterreihenfolge src-dst anstelle von dst-src, da auch die TGI-Beschreibungen diese Reihenfolge verwenden.

Zahlen

Zahlen werden standardmäßig als dezimal interpretiert. Für Hex-Zahlen muss das Präfix "0x" verwendet werden

Labels

Um auf die Adressen von Code oder Daten verweisen zu können, werden Labels verwendet.

Eine Zeile, die ein Label enthält, darf zusätzlich weder Code noch Daten enthalten (nur Kommentare). Ein Label verweist stets auf die nächste Code oder Daten enthaltende Zeile.

```
label1:
```

Daten

Neben Code können Daten angegeben werden, in der aktuellen Version nur Words.

Ein Datenteil wird durch "dw" (data word) eingeleitet; danach folgen eine oder mehrere Zahlen (durch Kommata getrennt).

```
dw 0x10
```

```
dw 123,23223,0xfefa
```



<http://jump.to/oblivion>

Kommentare

Kommentare werden durch einen Strichpunkt eingeleitet und können überall im Sourcecode auftauchen.

```
move 0x12a4,r0 ;lade Register 0 mit 0x12a4
```

Assemblierung

MICAsm wird von der Kommandozeile aus aufgerufen

Syntax: **micasm** <dateiname> [optionen]

Optionen:

- os Output auf dem Bildschirm (mit Sourcecode) - default
- ot:file.txt Output in eine Textdatei
- om:file.mpr Output in eine mpr-Datei des Simulators

```
micasm example.asm -om:MI-Befehlssatz.mpr
```

Ein Beispielprogramm

```
; Berechnung der Summe aller Zahlen von 1 bis R1
; und anschließendes Abspeichern des Ergebnisses im Hauptspeicher

        move 150, R1          ; berechne 1...150 als Beispiel
        move 0, R0            ; Die Summe wird in R0 gespeichert, initialisiere
                               ; mit 0
loop1:
        add R1, R0            ; addiere zur Summe
        loop loop1, R1        ; R1 dekrementieren und zurück zu loop1 springen
                               ; wenn 0 noch nicht erreicht ist
        move R0, [memory_destination] ; Speichern des Ergebnisses

        ; Das Programm endet hier...

memory_destination:
dw 0 ; In dieser Speicherzelle wird das Ergebnis gespeichert
```

Die Ausgabe von MICAsm mit -os:

```
MICAsm v0.04 (C) 2002 by Dominik Jain

0000: 3801  move    15,R1
0001: 000F
0002: 3800  move    0,R0
0003: 0000
loop1:
0004: 0C10  add     R1,R0
0005: A501  loop   loop1,R1
0006: 0004
0007: 0500  move   R0,[memory_destination]
0008: 0009
memory_destination:
0009: 0000  dw
```