NAME

pot − Format der Quelldateien für den **pot**(1) Assembler.

BESCHREIBUNG

Dieses Dokument erläutert das Format der Quelldateien, die der **pot**(1) Assembler verarbeiten kann.

ZEILENFORMAT

Eine **pot**(1) Quelldatei besteht aus Zeilen, von denen jede *Symbole*, *Befehle* (und *Pseudobefehle*), *Oktalzahlen* und *Kommentare* enthalten kann. Wenn eine Zeile zumindest einen *Befehl* oder eine *Oktalzahl* enthält, wird ein Speicherwert generiert und der Speicherzeiger erhöht.

Der Speicherzeiger verweist dabei auf die Speicherzelle, in die der nächste Wert abgelegt werden soll. Zu Beginn verweist der Speicherzeiger auf die Speicherzelle mit der Adresse null.

Der Speicherzeiger kann über zwei verschiedene Pseudobefehle manipuliert werden:

***** <*nnnn*>
: Setzt den Speicherzeiger auf den oktalen Wert <*nnnn*>.

PAGE
: Setzt den Speicherzeiger auf den Anfang der nächsten Speicherseite. Eine Speicherseite besteht aus 128 Worten.

SYMBOLE

Symbole stehen am Anfang einer Zeile. Damit ein Symbol korrekt erkannt wird, dürfen vor ihm höchstens Leer- und Tabulatorzeichen stehen. *Adressymbole* enden mit einem **:** (Doppelpunkt). Ihnen wird der aktuelle Wert des Speicherzeigers zugewiesen. *Wertsymbole* weden mit **=** abgeschlossen. Ihr Wert wird bestimmt, indem der Rest der Zeile ausgewertet und das Ergebnis dem Symbol zugewiesen wird. Der Speicherzeiger wird dabei nicht erhöht.

Wenn ein Symbol nicht durch **:** oder **=** terminiert wird, so wird angenommen das es referenziert wird und wird duch seinen Wert ersetzt.

ANWEISUNGEN

Eine Anweisung besteht aus einem oder mehr *Befehlen*, *Symbolen* und/oder *Oktalzahlen*. Zusätzlich können Symbole und Oktalzahlen mit arithmetischen Operationen (+, -) zu Ausdrücken zusammengefasst werden. Der Ausdruck muß dazu in runde Klammern **( )** gesetzt werden. Ausdrücke werden berechnet und durch ihren Wert ersetzt. Der aktuelle Wert des Speicherzeigers kann über das Symbol **.** (Punkt) in die Berechnung einbezogen werden (z.B. **JMP I (.+1)**).

Die Befehle (Siehe **pep8**(7)) und Symbole werden durch ihre Werte ersetzt. Die sich ergebende Menge von Oktalzahlen wird oder-verküpft um ein einzelnes Anweisungswort zu erhalten.

Die Adressierung der Zero−Page wird automatisch erkannt und berücksichtigt. Sie bedarf keines speziellen Symbols. Indirekte Adressierung wird durch das spezielle Symbol **I** angezeigt.

KOMMENTARE

Kommentare werden durch **;** eingeleitet und erstrecken sich bis zum Ende der Eingabezeile.

PSEUDOBEFEHLE

Zusätzlich zur Manipulation des Speicherzeigers mit ***** und **PAGE** kennt **pot**(1) weitere Pseudobefehle zur Erzeugung spezieller Speicherwerte, zum Einbinden von Quelldateien und für die Definition von Makros.

TEXT

Mit dem Pseudobefehl **TEXT** kann Text in Form von gepackten **sixbit**(7) Zeichen im Speicher abgelegt werden. Das erste Zeichen nach dem **TEXT** Befehl, das kein Leerschritt oder Tabulator ist, wird als Trennzeichen aufgefasst und darf im Text nicht vorkommen. Dieses Zeichen beendet den abzulegenden Text. Zunächst wird ein Wort mit der Anzahl der Zeichen im Text abgelegt, dann die Zeichen, je zwei pro Wort. Das erste Zeichen wird in den Bit 0-5 der ersten Wortes abgelegt, das zweite in Bit 6-11 der ersten Wortes. Dann wird das zweite Wort gefüllt usw.

Beispiel:

```
TEXT    /HELLO WORLD!/    ; Begruessung
```

Beinhaltet zwölf Zeichen und erzeugt damit sieben Worte im Speicher.

FILE

Der Pseudobefehl **FILE** erlaubt das Einbinden von Quelltext aus weiteren Dateien. Hinter dem Befehl wird der Name der einzubindenden Datei angegeben der um **.pps** erweitert wird. Die Datei wird im gleichen Verzeichnis gesucht in dem sich die Datei die den -B FILE Befehl enthält befindet. Die Verarbeitung geschieht als ob der Inhalt der benannten Datei an Stelle des **FILE** Befehls eingefügt orden wäre. Mit **FILE** eingebundene Dateien können ihrerseits weitere Dateien mit **FILE** einbeinden. Die maximale Verschachtelungstiefe ist zehn.

Beispiel:

```
FILE    maclib    ; Makro Bibliothek einfügen
```

Liest den Quelltext aus der Datei "maclib.pps".

MACRO / MEND

Der Pseudobefehl **MACRO** definiert ein (neues) Makros. Makros sind Anweisungsfolgen die unter einem Namen, der nach dem **MACRO** Befehl angegegben wird, gespeichert werden und anschließend (mehrfach) unter diesem Namen abgerufen werden können. Makros akzeptieren bis zu neun Parameter die im Textkörper des Makros referenziert werden können. Dabei steht \1 für den ersten, entsprechend \9 für den neunten Parameter.

Um Sprünge innerhalb eines Makros zu erlauben, können lokale Symbole verwendet werden. Lokale Symbole enden mit einem oder mehr Prozentzeichen (%) die beim Aufruf des Makros durch die aktuelle Aufrufnummer des Makros ersetz werden. Somit sind lokale Symbole bei jedem Makroaufruf eindeutig.

Die Definition eines Makros wird mit dem Pseudobefehl **MEND** abgeschlossen.

Beispiel:

```
MACRO   LJP     ; Longjump
JMP I   (.+1)   ; indirect via next word
\1              ; target is here
ENDM
```

Anschließend können mit dem Makro Sprünge zu Adressen auf anderen Seiten eingefügt werden:

```
LJP     SUBPRG    ; Longjump to SUBPRG, takes two words
```

Der Aufruf eines Makros unterscheidet sich nicht vom Aufruf eines Befehls. Währen Befehle immer nur ein Speicherwort belegen kann eine Makroaufruf jedoch prinipell beleibig viele Speicherworte füllen (im Beispiel zwei).

ZMEM

Der Pseudobefehl **ZMEM** wird verwendet um - insbesondere in Makros - dynamisch Konstanten in der Zero-Page abzulegen. Der Konstante wird zusätzlich ein Symbol zugewiesen. Hinter dem befehl steht zunächst das zuzuweisende Symbol und dann der Wert. Es wird zunächst untersucht ob eine Konstante mit dem gewünschten Wert bereits angelegt wurde, dann wird nur dem Symbol die Adresse dieser Konstanten zugewiesen. Existiert die Konstante noch nicht wird die Konstante in die höchste, nicht von Konstanten belegte Adresse in der Zero-Page gespeichert. Damit liegt die erste Konstante im Speicherwort 0177, die zweite in 0176 usw.. Es wird nicht überwacht ob die Speicherzelle bereits von manuell definierten Konstanten oder Befehlen belegt sind!

Beispiel:

```
MACRO   MOVI        ; Move immediate value to address
ZMEM    TMI%% \1
RCL     TMI%%
STO     \2
ENDM
```

In dieser Makrodefinition wird (zur Assemlierungszeit) eine Konstante entsprechend dem ersten Paramter in der Zero-Page abgelegt. Ihr wird ein lokales Symbol zugewiesen. Zur Laufzeit wird die Konstante in den Akkumulator geladen und im Speicherwort das über den zweiten Paramter des Makros fstgelegt ist gespeichert. Das Makro belegt ausser der Speicherzelle in der Zero-Page noch zwei Speicherzellen am Aufrufort.

PROGRAMMBEISPIEL

```
;
; Aufaddieren der Werte einer Tabelle
;

CLLA=   CLA CLL                 ; Wertsymbol zuweisen

        JMP I       VEC         ; Erzeugt indirekten Sprung
VEC:    START                   ; zum Programmstart

RESULT: 0                       ; Platz für das Ergebnis

TAB:    1                       ; Die
        3                       ; ersten
        5                       ; sechs
        7                       ; Primzahlen.
        13                      ; Natürlich
ENDT:   15                      ; oktal.

TLEN:   (ENDT-TAB+1)            ; Länge der Tabelle

TPTR:   TAB                     ; Zeiger zum Anfang
TEND:   ENDT                    ; und Ende der Tabelle

        PAGE                    ; Der Code beginnt auf
                                ; einer neuen Speicherseite

START:  RCL         RESULT      ; Zwischenergebiss holen
        CLL                     ; Link löschen
        TAD I       TPTR        ; Nächsten Wert addieren
        STO         RESULT      ; Neues Zwischenergebniss
        RCL         TPTR        ; Zeiger holen
        CLL CMA IAC             ; Zum Vergleichen subtrahieren
        TAD         TEND        ; Zeiger mit Endezeiger vergleichen
        SNA                     ; Nicht gleich? Weitermachen
        JMP         FINISH      ; Sonst fertig
        RCL         TPTR        ; Zeiger holen
        IAC                     ; erhöhen
        STO         TPTR        ; und zurückspeichern
        JMP         START       ; Nächster Wert
FINISH: HLT
```

Nach dem Ausführen diese Programms steht in der Speicherzelle *0002* (Result) der Wert *0050*.

SIEHE AUCH

pot(1), **pepsi**(1), **pep8**(7)