```
*********************************************************************
***            Z 8 0  -  C r o s s a s s e m b l e r              ***
*********************************************************************
***                                                               ***
***                         Version 4.07                          ***
***                        (c)  Juli  '91                         ***
***                     Tel.: (05 71) 505 220                     ***
***       ZeitControl, Hauptstr. 16, 4952 Porta-Westfalica        ***
***                                                               ***
*********************************************************************
```

an OPENOFFICE.ORG angepasst (Umlaute ersetzt)
Stand 29.11.2010 Dr. Hehl Hans

ASS ist ein Z80 Crossassembler. Er ist im Wesentlichen kompatibel mit den Befehlen des M80-CPM Assemblers. Die Opcode's des HD64180 werden zusätzlich verarbeitet. ASS kann nur ABSOLUTEN Code erzeugen. EXTERNAL deklarierte Variablen sind ebenfalls nicht möglich, daher existiert kein Linker. M80 Befehle wie DSEG und CSEG führen zu Fehlern. Tritt während der Assemblierung ein Fehler auf, so wird ein DOS-Exitcode ungleich 0 geliefert.

Das Programm ist komplett in PASCAL geschrieben. Es nutzt den gesamten noch zur Verfügung stehenden Platz im Hauptspeicher aus. Falls ein Runtime-Fehler 203 oder 202 auftritt, reicht der Speicher nicht aus. Viel Speicher wird durch den ORG-Befehl verbraucht (siehe weiter unten). 5000 Zeilen Text, in denen nur ein ORG vorkommt, können noch mit weniger als 200 K-Byte freiem Speicher assembliert werden.

Weitere Eigenschaften und Unterschiede werden im folgenden beschrieben:

Zur Aufrufsyntax:

Der Aufruf erfolgt mit
ASS <Textfilename> [<Codefilename>] [/Parameter1 /Parameter2...]

Wenn in den Filenamen keine Typen angegeben werden, wird an den Textfilenamen xxxx.MAC und an den Codefilenamen xxxx.Z80 angehängt. Codefilename und Parameter sind optional.

Parameter können an beliebiger Stelle, durch Leerstellen getrennt und mit einem vorangestellten "/" eingegeben werden.

Folgende Parameter sind möglich:

/L	Erzeugt ein Listing. Das gesamte Listfile wird erst in eine Zwischendatei (ASSTEMP.$$$) geschrieben. In einem zweiten Durchgang werden die fehlenden Referenzen aufgelöst. Das Zwischenlisting ist identisch mit der 'Quicklist' Option /Q.
/R	Crossreferenz Liste ausgeben (bisher nur alphabetisch sortiert m”glich)
/D	Codedump ausgeben (Hier wird der Code, ausgegeben, bevor er gespeichert wird. Ein * hinter der Adresse in der ersten Zeile weist auf einen ORG Befehl hin.
/Oxxxx	Die Ausgabe des Assemblers wird in die Datei xxxx.LST ausgegeben. Wird für xxxx nichts angegeben, so wird <Source-name>.LST benutzt
/Cxxxx	Erzeugt bei erfolgreicher Assemblierung die Datei xxxx.CLF (CodeLengthFile), in der die

Startadressen, Längen und Offset's der erzeugten Codeblöcke enthalten sind. Offset beschreibt die Adresse, ab der der Codeblock im Codefile zu finden ist. Wird xxxx nicht angegeben, so wird <Codefilename>.CLF benutzt.

	(Org)	(Länge)	(Offset)
Bsp:	0	128	0
	16384	2048	128

Die Angaben sind hier dezimal.

/Q (Quicklist) Gibt das Zwischenlisting aus. Hier sind keine FORWARD-Referenzen aufgelöst. /Q wird bei Benutzung von /L ignoriert.

/E Ermöglicht Mehrfachdefinitionen mit EQU. Statt einer Fehlermeldung wird eine Warnung ausgegeben (.EQU liefert keine Fehlermeldung oder Warnungen).

/Ixxxx Erzeugt bei erfolgreicher Assemblierung die Datei xxxx.HEX im INTEL-HEX Format. Der Dateiname xxxx muß nicht mit angegeben werden, die Datei wird dann unter dem Namen <Sourcename>.HEX abgespeichert.

Bsp: in TEST.MAC steht der Quelltext
Eingabe: "ASS Test /I"
danach steht in TEST.Z80 der assemblierte OBJECTCODE.
Die Datei TEST.HEX enthält die Codedatei im INTEL-HEX Format.

Falsche Parameter, oder völlig fehlende Parameter führen zur Ausgabe der Aufrufsyntax und der Optionen.

Beispiele:

ASS test	Assembliert TEST.MAC und speichert den erzeugten Code in TEST.Z80. Alle sonstigen Meldungen werden auf dem Bildschirm ausgegeben.
ASS P /l /op.txt	Assembliert P.MAC nach P.Z80 und schreibt Listing und Fehlermeldungen in P.TXT.
ASS M D.COD /o	Assembliert M.MAC nach D.COD; Fehlermeldungen in M.LST.
ASS T.ASS	Assembliert T.ASS nach T.Z80; Fehlermeldungen erscheinen am Bildschirm.
ASS D /L /R /O /D	Assembliert D.MAC nach D.Z80. Ein vollständiges Listing mit Referenzliste und Codedump wird in die Datei D.LST ausgegeben.
ASS D /LROD	ist nicht erlaubt. Hier wird nur die Option /L erkannt.

Zur Syntax des Sourcetextes:

Es ist nicht vorgeschrieben,in welcher Spalte ein Label oder Befehl beginnen muss. (Vielleicht kann jemand die Programme dadurch strukturieren). Das Label kann mit oder ohne ":" eingegeben werden; im MASM haben Namen von Macro's und EQU's keinen ":", alle anderen haben ihn. Eine Zeile, in der nur ein Label steht, ist auch erlaubt.

Folgende Pseudo Operatoren sind möglich:
- EQU, DB, DEFB, DW, DEFW, DC, DEFC, DZ, DEFZ, DS, ORG, (ASEG),

MACRO, ENDM, REPT, END, IF, ELSEIF, ENDIF, INCLUDE,
(innerhalb eines MACRO's sind keine lokalen Labeldefinitionen möglich!)
.LIST, .XLIST, .CREF, .XCREF
ACHTUNG: Bei den Operatoren DS, ORG, REPT, IF dürfen die Ausdrücke keine Forward-Referenzen enthalten.

- EQU weist einem <label> einen <ausdruck> zu.
  Falls die Option /E gesetzt, sind Mehrfachdefinitionen
  erlaubt; sie führen dann nur noch zu Warnungen.
  Anwendung: <label> EQU <ausdruck>

- .EQU analog zu EQU, jedoch keine Fehler oder Warnungen bei
  Mehrfachdefinitionen. Entspricht dem M80 Befehl SET im 8080
  Modus.

- DB schreibt nur das niederwertige Byte jedes Ausdrucks in den
  Programmcode. Ist das höherwertige Byte nicht 0 oder 0FFh
  (bei negativen Werten), so kommt eine Warnung.

  Anwendung: DB <ausdruck1>,<ausdruck2>,<ausdruck3>,.....
             DB "Test"
             DB -5, -0Fh
             DEFB "Dies ist ein TEST"

- DW schreibt erst das niederwertige, dann das höherwertige Byte
  von jedem Ausdruck in den Programmcode.
  Anwendung: DW <ausdruck1>,<ausdruck2>,.....
             DEFW <ausdruck1>,...

- DC schreibt nur das niederwertige Byte jedes Ausdrucks in den
  Programmcode. Ist das höherwertige Byte nicht 0 oder 0FFh
  (bei negativen Werten), so kommt eine Warnung.
  Desweiteren wird das höherwertigste BIT (BIT 7) des letzten Bytes
  gesetzt (z.B. als Endekennung im String)
  Anwendung: DC "Test"

- DZ oder auch DEFZ, Anwendung und Funktion wie DB (DEFB), nur wird
  hinter das letzte Byte eine Null gesetzt (z.B. als Endekennung)
  Anwendung: DZ "Test"
             belegt im Speicher 5 Bytes, wie DB "Test",0

- DS setzt den Programmzähler um die angegebenen Bytes hoch
  oder herunter. (Alternativ kann man $ .EQU $+<ausdruck> verwenden.
  Anwendung: DS <ausdruck>

- ORG kann ebenfalls mehrmals angewandt werden. Der nächste Code-
  block wird auf der Platte direkt an den vorhergehenden angehängt.
  Ein Codeblock darf maximal 32 Kilobytes lang sein. Jeder ORG Be-
  fehl führt intern zu einem Speicherverbrauch von 32 KByte. Dadurch
  kann schnell die Speicherkapazität des Rechners erschöpft werden.
  Als Alternative kann der DS Befehl verwendet werden:
  Statt "ORG 400h" kann z.B. "DS 400h-$" benutzt werden.
  Die Länge eines zusammenhängendes Codestücks darf 32 K-Byte nicht
  überschreiten. Hier muss ein zweiter ORG verwendet werden.
  Anwendung: ORG <ausdruck>

- Wird bei (IX+d) bzw. (IY+d) ein negativer Operand verwendet, muss
  trotzdem die Z80 Syntax beibehalten werden!
  Anwendung: LD A,(IX+-4)
             LD B,(IY+222)
             LD B,(IX+-20) nicht möglich ist LD B,(IX-20), da dann
                            die INTEL Syntax nicht eingehalten wird

- ASEG wird ignoriert

- MACRO
  - Parameter müssen in Definition und Aufruf die gleiche
    Anzahl haben.
  - Es sind noch keine lokalen Labels in einem Makro möglich.
    Ersatzweise muss der Programmzähler benutzt werden.
    (z.B. JP $+10)

```
    - Jedes Macro muss vor seinem ersten Aufruf definiert werden.
    - Jede Macrodefinition muss mit ENDM abgeschlossen werden.
    - Anwendung:
      - Definition: M1    MACRO   <parameter1>, <parameter2>,....
                          <auszuführende Befehle>
                          ENDM
      - Aufruf:           M1      <ausdruck1>,  <ausdruck2> ,....

    - Macro-Aufrufe können beliebig tief geschachtelt werden;
      Macro-Definitionen und REPT-Anweisungen jedoch überhaupt nicht.

  - REPT

    - Wiederholt den zwischen REPT und ENDM stehenden Text
      <ausdruck> mal
    - Wie bei Macro sind auch hier noch keine lokalen Labels möglich.
    - Anwendung           REPT    <ausdruck>
                          <zu vervielfältigender Programmtext>
                          ENDM
    - Das folgende Beispiel :   hilf  EQU   "0"
                                      REPT  10
                                      DB    hilf
                                hilf .EQU   hilf + 1
                                      ENDM
                erzeugt den Code "0123456789"

  - IF
    - Textblöcke können bei der Assemblierung ausgelassen werden.
    - Die Bedingung muss direkt auswertbar sein
    - Bedingungen können beliebig tief verschachtelt werden
    - Achtung: Es wird nur durch das niederwertigste Bit entschieden,
      ob die Bedingung erfüllt ist ( Bit0=1 => Bedingung erfüllt).
      Im Gegensatz dazu dreht NOT alle Bit's um.

    - Anwendung           IF      <Bedingung>
                          <Anweisungsteil, falls wahr>
                          ENDIF
    - oder                IF      <Bedingung>
                          <Anweisungsteil, falls wahr>
                          ELSEIF
                          <Anweisungsteil, falls falsch>
                          ENDIF

  - INCLUDE
    - An der Stelle dieses Befehls wird eine weitere Datei eingefügt.
    - Der Dateiname muss auch die Extension enthalten.
    - Includefiles können beliebig tief geschachtelt werden.
    - Anwendung:          INCLUDE  "<Dateiname>"

  - KEYB
    - Werte können während der Assemblierung über die Tastatur einge-
      geben werden.
    - Bei Eingabe von Return entspricht KEYB dem EQU Befehl,
      andernfalls ersetzt der eingegebene Wert den Operanden.
    - Anwendung   <label> KEYB   <default>

  - .LIST / .XLIST entspricht dem Setzen / Löschen der /L Option.

  - .CREF /.XCREF entspricht dem Setzen  / Löschen der /R Option.
    Das letzte mal, das es im Programm auftaucht legt fest, ob eine
    Referenztabelle ausgegeben wird oder nicht.

  - END muss am Ende des zu assemblierenden Textes stehen.
    Zeilen die danach kommen werden nicht mehr gelesen.

---------------------------------------------------------------------

Zur Syntax der Ausdrücke:

  In den Ausdrücken werden folgende Operationssymbole ausgewertet:
```

- 1 stellige Operationssymbole (Syntax: <Op1> <ausdruck>):

LOW		Liefert das niederwertige Byte des Ausdrucks
HIGH		Liefert das höherwertige Byte
NOT		Invertiert alle Bits des Ausdrucks

!!!!

- 2 stellige Operatoren: (Syntax: <ausdruck1> <Op2> <ausdruck2>):
  In den Ausdrücken sind folgende 2-stelligen Operationssymbole möglich:

+		Addition
-		Subtraktion
*		Multiplikation
/	DIV	Division (ganzahlig)
MOD		Rest der ganzzahligen Division
SHL		links ÄÄÂÄÄÄÂÄÄ Verschieben von <ausdruck1>
SHR		rechts ÄÄÙ ÀÄÄ um <ausdruck2> Bits
AND	&	logische UND Verknüpfung \
OR		logische ODER Verknüpfung Alle Bits
XOR		logische Exklusiv ODER Verknüfung /

  Folgende Operatoren sind hauptsächlich für die bedingte Assemblierung gedacht. Sie können jedoch auch in Ausdrücken benutzt werden. Wenn die Berechnung zutrifft, ist das Ergebnis 1, sonst 0.

EQ	=	
NE	#	<>
LT	<	
LE	<=	
GT	>	
GE	>=	

  Die Symbole in der 2. Spalte können ersatzweise benutzt werden; sie sind im M80-Assembler nicht vorhanden.
  Prioritäten: NOT LOW HIGH vor * / DIV MOD vor allen anderen.

Merkmale des erzeugten Codes:

- ASS speichert das erste Codeerzeugende Byte auch als erstes Byte im Code ab. Benutzt man M80 und L80 und hat am Anfang den Befehl ORG 1000h, so erzeugt L80 vor dem eigentlichen Code 0F00h Nullen. Lädt er das Programm dann ab Adresse 100h, so paßt der Befehl ORG 1000h. Die Länge des Code, die im DOS-Inhaltsverzeichnis angegeben wird, ist exakt die Anzahl der erzeugten Codebytes; ohne irgendwelche Füll- oder Daten-Bytes wie beim M80, der noch die Linker Informationen speichert.

==========

Änderungen, Korrekturen:

Version 1.1 ..... 17.10.88

Version 1.2 ..... 8.11.88
- Fehler bei der Übergabe von Strings mit Länge > 1 an Macro's behoben.

Version 1.3 ..... 14.12.88
- RST Befehl auch mit FORWARD Referenzen
- Sprungbedingungen auch als Labels möglich
- Ausgabe von Aufrufsyntax / Optionen bei falschem Aufruf
- Das Programm ASSLISTER macht ein Listing mit aufgelösten Referenzen.

Version 2.0 ..... 2. 1.89

- Fehler -keine Warnung bei mehrfacher EQU Anweisung- behoben
- Mnemonics der HD64180 CPU werden verarbeitet:
  SLP; MLT; IN0 g,(m); OUT0; OTIM; OTIMR; OTDM; OTDMR;
  TSTIO; TST g; TST m; TST (HL)
- Bedingte Assemblierung (auch verschachtelt)
- Eingabe von Werten während der Assemblierung (KEYB)

Version 2.01 ..... 9. 1.89
- Fehler bei der Auswertung von komplizierten EQU's behoben
- Bei allen 8 Bit Werten Bereichsüberprüfungen eingeführt

Version 2.1 ..... 5. 2.89
- INCLUDE Befehl eingeführt
- EQU erzeugt Fehler bei Mehrfachdefinition durch EQU
- .EQU erlaubt Mehrfachdefinitionen ohne Warnung
- Die Option /E erlaubt wie früher Mehrfachdefinitionen
  mit EQU, jedoch mit Warnung.

Version 2.2 ..... 16. 3.89
- IN Befehl mit Vorwärtsbezug erlaubt
- Fehler beim Auflösen von Forwardreferenzen bei mehrfachen
  ORG's.
- Alle Codeblöcke werden im Codefile aneinandergehängt, statt
  in getrennte Files geschrieben
- Verbesserung der Ausgabe

Version 3.01 ..... 27. 4.89
- INTEL-HEX Format wird erzeugt, bei Angabe des Parameters /I

Version 4.01 ..... 30. 5.89
- Listing wird jetzt durch den Assembler erzeugt.
- Diverse Bug's mit Runtime-Fehlern behoben.

Version 4.02 ..... 5. 7.89
- Ausgabe der bei EQU berechneten Werte

Version 4.03 ..... 16. 4.90
- Fehler bei Bereichsüberprüfung von JR behoben;
  Bisher -256..+255 ohne Fehlermeldung möglich,
  aber nur -128..+127 real möglich.
- Bei Fehlern in Forwardreferenzen werden die Zeilen-
  nummern ausgegeben.

Version 4.04 24.06.90
- Fehler beim Befehl TST m behoben, es wurde falscher
  OP-Code erzeugt

Version 4.05 28.06.90
- Fehler beim Erzeugen eines Listfiles oberhalb der
  Adresse 8000h behoben. Listing ist jetzt möglich

Version 4.06 07.09.90
- Fehler beim Abspeichern eines Listfiles
  CR wurde erst nach dem LF ausgegeben

Version 4.07 16.07.91
- Bei den Befehlen DB, DW ist jetzt auch die Syntax
  DEFB und DEFW erlaubt.
  Neu eingeführt wurden die Befehle DC und DZ.
  Die Programmcodelänge von ASS wurde verkleinert.
  Die Geschwindigkeit wurde deutlich erhöht.
  Bei den Indexregistern IX+ und IY+ ist auch ein
  negativer Operand erlaubt: IX+-Operand