68000 Assembler

WAS ist ein Assembler ?

- Ein System, das den Programmierer hilft, eine maschinennahe Programmierung zu realisieren.

Ein Programm liegt der CPU in binärer Form vor und wird durch den Assembler in einer primitiven symbolischen Sprache dargestellt. Damit ist er das User-Interface zur ISA (Instruction Set Architecture). Ein Assembler ist so etwas wie ein „Übersetzer“ und Zusammensetzer von Programmcode.

Der Benutzer-Aspekt:

Im Folgenden nehmen wir den Standpunkt ein, dass ein Assembler ein Programm verarbeitet, in dem niedergeschriebene Befehle die untere Struktur der Maschine widerspiegeln. Sie sind damit die symbolische Darstellung der Inhalte von Speicherzellen oder Gruppen von Zellen und erlauben dem Programmierer, sich direkt auf interne Funktionscodes, auf Akkumulatoren und Register der CPU zu beziehen.

Technische Aspekte:

Die einfachen Assembler waren von Typ „Laden und Starte“ (Load and Go) d.h. eine kodierte Befehlszeile wird direkt übersetzt und unmittelbar von der CPU ausgeführt. (keine Unterprogrammtechnik möglich).

Die in der Entwicklung nachfolgenden Assembler konnten Unterprogramme „einbinden“, dafür war eine Assemblierung in mehreren Stufen notwendig.

- In der ersten Phase baut der Assembler eine Symboltabelle auf und alle Definitionen von Symbolen in den Programmen(explizite und implizite) werden gesammelt.

- In der zweiten Phase wird die eigentliche Übersetzung ausgeführt; dabei werden die in der ersten Phase den Symbolen zugeordneten Werte verwendet.

Der Assembler Befehl:

- Die Information, die das Steuerwerk / Control Unit einer CPU benötigt, um Befehle auf Maschinenebene auszuführen, liegen zunächst in geeigneter Form /Codierung als Bitmuster im Speicher als „Befehls / Instruktionswort“ vor.

- Die Länge eines Befehlswortes kann mehrere Speicherworte umfassen.

- Die Menge der zulässigen Befehlsworte bezeichnet man als Befehlssatz (Instructionset)

68000 Assembler

- **Die Befehls-Struktur sieht wie folgt aus:**

  Operationscode ; Adressteil

  Opcode: Hier wird festgelegt WAS zu tun ist und damit auch die Zahl der Operanden.

  Adressteil: Hier wird der Operandenzugriff beschrieben (WOMIT) Aber auch WOHIN verzweigt werden soll im Befehlsablauf.

Logische Aspekte des Assembler

Der Assembler hat zwischen 3 Arten von Informationen zu unterscheiden:

Statische Information:

Elemente wie der Operationscode sowie Konstanten, deren Wert nicht von der Lage des Programm im Speicher abhängt.

Darum ist diese Information unabhängig von der Art und Weise wie das Programm mit anderen Programmen zusammengefügt wird.

Verschiebbare Information:

Ferner gibt es Symbole, wie die Namen von Befehlen oder Namen (Labels) von Speicherplätzen innerhalb des Programms, auf die man in anderen Programmen aber keinen Bezug nimmt.
Obwohl diese Speicheradressen von der Lage des Programms im Speicher abhängen, daher auch in der Art und Weise wie das Programm mit anderen Programmen zusammengefügt wird, liegt jedoch die Position der Speicherstellen relativ zum Anfang des Programms fest !

Externe Information (externe Referenz)

Schließlich gibt es noch Symbole von Querbezügen / Referenzen, die in anderen Programmen definiert werden und auf die dort Bezug genommen wird.
Die ihnen zugeordneten Werte sind solange unbekannt, bis das gesamte Programm zusammengesetzt ist.

68000 Assembler

Für die MOTOROLA Familie gibt es mehrere Assembler, mit wenigen Unterschieden. Allen gleich sind die Schreibweise der Befehle und die Register. Die syntaktische Schreibweise ist zeilenorientiert. Die Befehle, die mehrer Operanden haben, werden durch Kommata getrennt nach dem Befehl angegeben, wobei generelle die Festlegung gilt, dass der linke Operand die Quelle und der rechte die Senke ist.

Ein Beispiel

. Marke: MOV.W Quelle, Ziel ;Kommentar

mit der Wirkung, dass der Zieloperand als Inhalt den Wert des Quelloperanden (ein Wort) bekommt. Danach sollte ein Kommentar folgen zur Dokumentation.

Schauen wir uns in diesem Zusammenhang Assemblerdirektiven an, wie sie ähnlich in den SCHABAAN – Folien auftreten.

Direktiven sind keine Befehle mit Operationscode, sie sind Anweisungen an den Assembler, wie Variablen strukturiert sein sollen, ob als Wort, als Byte oder als Formelwert, oder als Folge von Worten in Form einer Tabelle.

Direktive	**Effekt**	**Bezeichner**	**Direktive**	**Operanden**
EQU	Setzt einen Bezeichner Gleich dem Wert der absoluten Formel	N M	EQU EQU	4 4+(2*N)
DC.B	Definiert Bytedaten-Bereich mit Initialisierung	Zeichen String Bytevar	DC.B DC.B DC.B	„a“ „Januar“ 2+(4*N)
DC.W	def. Wortdatenbereich Mit Initialisierung	var Reihung	DC.W DC.W	2+(4*N) 2,5,6,7
DC.L	def. Doppelwortbereich	DVAR	DC.L	$0A123456
DS.B	Definiert einen Datenbereich ohne Initialisierung mit den im Operanden (absolute Formel) angegebenen Anzahl von Einheiten:			
	.B = Byte .W = Wort .L = Doppelwort	 Bytevar DVAR	DS.B DS.W DS.L	1 10 1
ORG	setzt den Adresszähler auf den im Operanden angegebenen Wert (absolute Formel)			

68000 Assembler

Ausrichtung:
Wie oben, mit Blick auf die Verschiebbarkeit von Information im Speicher ausgeführt, betrachten wir nun die Datendefinitionen DC.W, DC.L sowie DS.L und DS.W

Bei diesen Definitionen wird der Datenbereich automatisch ausgerichtet angelegt werden.
Das bedeutet: Auf Wort bzw. Doppelwortgrenze! beginnt der Bereich. Gegebenenfalls werden dazu leere Bytes eingefügt. Der Grund dafür: Der Motorola Prozessor verlangt diese Ausrichtung und verbietet den Zugriff auf nichtausgerichtete Speichereinheiten (sonst erfolgt ein Adressierungsfehler)

Adressoperator:
Wie bei Assemblern üblich, gibt es einen Adressoperator, der angewandt auf einen Datenbenzeichner (zur Assemblierzeit) die (verschiebbare) Adresse liefert.
Das Symbol für den Adressoperator ist „#“.

Symbol für den *unmittelbaren* Operator:
Bei 68000 wird zur Kennzeichnung eines unmittelbaren Operators ein *Symbol für unmittelbarer Operand* vorgestellt: „#“

Hier scheinen Verwechslungen möglich. Es wir aber durch die Art der Verwendung klar, was gemeint ist, wie das folgende Programmteil verdeutlicht:

```
Val     EQU     4       constante mit n = 4;
Var     DC.W    10      definiere Constante 10 in Wortbreite

        MOV.W   #Val, D0
        MOV.L   #Var, A0
        LEA.L   Var, A0
```

Beim ersten MOV Befehl wird der Wert Val als unmittelbarer Operand verwendet. Hier kennzeichnet das # den unmittelbaren Operanden. Beim zweiten MOV Befehl ist Var der Bezeichner eines Datenbereiches (ein Wort initialisiert mit 10). Das Symbol # angewendet auf den Bezeichner liefert - jetzt als Adressoperator – die Adresse von Var.
Diese Adresse wird nun als unmittelbarer Operand in das Befehlsformat des MOV-Befehls eingetragen und das zum Zeitpunkt der Assemblierung.
Bei der Ausführung des Befehls, zur Laufzeit, wird dann die Adresse in das Register A0 geladen.
Der letzte Befehl LEA hat die gleiche Wirkung, allerdings wird hier erst zur Laufzeit die Adresse beschafft (nicht zur Assemblierzeit wie zuvor beschrieben).

68000 Assembler

Konstante Ausdrücke
Absoluter Ausdruck
Wie bei allen Assemblern üblich, existiert hier auch die Möglichkeit, auf der Position von Operanden *Konstanten* oder Ausdrücke mit Konstanten, sogenannte *Konstanten Ausdrücke* oder *Absolute Ausdrücke* anzugeben.

Konstante	**Beispiele**	**Bemerkungen**
Binär	%1010101010	% steht für binär
Hexadezimal	$0AFF	$ steht für hexadezimal
Dezimal	19	dezimal
Zeichen	„a“	ASCII Code
Text	„Text“	

Solche Konstanten können mittels Operatoren zur Bildung von *numerischen Ausdrücken* verwendet werden.
Der Wert eines solchen Ausdrucks muss zur Assemblierzeit bestimmbar sein – denn er wird zur Assemblierzeit und nicht zur Ausführungszeit bestimmt.
Dabei können, wie bei arithmetischen Ausdrücken, runde Klammern angegeben werden, um eine andere Auswertungsfolge ablaufen zu lassen.
Die Definition der Auswertungsreigenfolge ist gleich der bei höheren Programmiersprachen.

Operator	**Bemerkung**
-; +; NOT	nur unäre Operatoren, NOT führt bitweise Negation durch
***; /; MOD**	/ ist die ganzzahlige Division MOD ist die Modulo Funktion (Rest bei einer ganzzahligen Division)
+; -	Diese Operationen arbeiten analog wie bei den Maschinenbefehlen (werden nicht übersetzt) und arbeiten bitweise.