# V E R G L E I C H von S P R A C H E und S Y S T E M - S O F T W A R E der R E C H N E R 301/306 mit 330 aus der S I C H T des A N W E N D E R S

E. Bachner Kernforschungsanlage Jülich / ZAT

Vorbemerkungen:

Mit meinen Ausführungen will ich keinen objektiven Vergleich von Sprache und Systemsoftware der beiden Prozeßrechnerfamilien unternehmen, sondern ausgehend von den Erfahrungen mit den alten Rechnern die Vor- und Nachteile aus der Sicht eines Prozeßrechner-Anwenders schildern. Für andere Anwendungsfälle, z. B. für die Fachhochschulen kann sich durchaus ein anderes Bild ergeben.

## Assembler-Sprache

Das auffälligste Merkmal der neuen Sprache ist ihre Abkehr von der mnemotechnischen Notierung der Befehle. Die Befehle werden als Operatoren angegeben, ähnlich wie in FORTRAN oder ALGOL. Das ist in meinen Augen ein so gewaltiger Fortschritt für eine Assemblersprache, daß der Mann, der sich das ausgedacht hat, einen Orden verdient hat.

Durch die Matrixbefehlsliste ergeben sich statt der 210 Befehle für den PR 330 nur 26 Operatoren, von denen einige nur erweiterte Funktionen eines anderen Operators darstellen:

| | |
|---|---|
| Addieren | + |
| Addieren mit Übertrag | +. |

Einige Bespiele für Operatoren: Für die früheren Befehle TEP, TAS, TEX, TAX, LAP gibt es nur einen Operator

: =

wobei links vom Operator das Ziel, rechts die Quelle steht. Eine weitere Angabe muß der Assembler noch haben, um den richtigen Befehl absetzen zu können: die Art des Zahlentyps, mit dem wir arbeiten wollen. Diese Angabe steht am Anfang der Assemblerzeile:

```
'G'       GØ : = (GK)
```

In diesem Fall wird eine Gleitpunktzahl (G) aus den Zellen GK und GK + 1 in die Register GØ und G 1 geladen. Das Speichern sieht dann so aus:

```
'G'       (GK) : = GØ
```

Die arithmetischen Operatoren werden wie im FORTRAN geschrieben, allerdings ist keine Klammerung möglich.

Für die Verschiebe-Operationen gibt es auch nur einen Operator

```
'F'       .V        VAL  VAR
'A'       .V        VLL  VLR
'D'       .V        VDR  VDL
```

Ein erfreuliches Kapitel sind die Bitbefehle:
Es gibt nicht nur den VSE in ähnlicher Form wieder, sondern man kann ganz gezielt bits abfragen, auf Ø oder 1 setzen. Sie sind nicht nur interessant für gezielte Anzeigen-Auswertung von Aufrufen, sondern auch, um z. B. Markierungen für irgendwelche Zustände zu setzen oder zu löschen:

```
(MARK (10))     : T Ø
(MARK (10))     = 1  : SP FORT
```

Durch die teilweise sehr elegante Methode der Programmierung wird man leicht dazu verführt, sie für eine höhere Programmiersprache zu halten, und auf sie zu schimpfen, wenn es nicht immer so elegant und einfach geht, wie im überwiegenden Teil der Fälle.

Trotzdem ist diese Assemblersprache eine der erfreulichsten Eigenschaften der neuen Rechnergeneration.

ORG 330 PP

Das Kapitel ORG 330 hat erfreuliche Aspekte, aber leider auch einige weniger schöne Seiten. Das erste, was einem auffällt, wenn man sich die ORG-Beschreibung durchliest, ist, daß die Einfachheit und Übersichtlichkeit des alten ORG I oder selbst des ORG E 301 nicht mehr vorhanden ist. Betrachtet man die historische Entwicklung, wie sie sich einem Außenstehenden darstellt, dann kommt man zu folgendem Schluß: die ORG-Entwickler haben ein ORG-Konzept entwickelt, das alles nur Mögliche und Wünschenswerte enthalten sollte. Als es an die Realisierung ging, merkten sie, daß das ORG wohl doch ein bißchen sehr lang werden würde. Aber anstatt nur einige Funktionen zu vereinfachen, wurden einige Funktionen ganz einfach gestrichen. Ich kann mir nicht vorstellen, daß für einen Prozeßrechner ein Benutzerkennzeichen für eine Datei oder einen Plattenstapel interessant ist, kann mir aber sehr wohl vorstellen, daß der leider gestrichene dynamische Prioritätswechsel für viele wichtig wäre.

Zweifellos hat man mit dem neuen ORG sehr viel mehr Möglichkeiten, um trickreiche und raffinierte Systeme zu erstellen. Ich befürchte nur, daß bis auf wenige Ausnahmen, weder die zeitlichen noch die personellen Voraussetzungen gegeben sind, um Programmsysteme zu erstellen, die so raffiniert sind, daß selbst der Entwickler hinterher nicht mehr weiß, wie er am besten eine kleinere Änderung eines Kunden anbauen kann. Werden Programme von verschiedenen Leuten erstellt, kann das System nicht laufen, oder die Abstimmung über die Parameter nimmt einen sehr großen Teil der Entwicklungszeit in Anspruch (Beispiel: Änderung von Parameter in YXANWP mit DIPOS).

Doch nun zu einigen Eigenschaften des ORG 330 PP, die von der ORG-Konzeption der 24-bit- Reihe abweichen:

1. Das ORG 330 ist generierbar. Der Kunde erhält den gesamten Masterstapel der ORG-Bausteine und kann sich jederzeit nach seinen Erfordernissen ein ORG generieren. Wenn die ORG-Bausteine auf der Platte stehen, dauert eine Generierung ohne Protokoll etwa 2 min.

2. Eine sehr schöne Funktion ist der Koordinierungszähler. Mit ihm kann man einfache Koordinierungsaufgaben sehr elegant lösen, die früher z. B. durch doppeltes Belegen mit mehr Aufwand verbunden waren. Darüber hinaus kann man auch aufwendigere Koordinierungsprobleme lösen, z. B. Synchronisierung von Programmen mit Datenübergabe von einem zum anderen Programm.

3. Sehr viel flexibler ist die Bedienfunktion geworden (BEWA). Der Aufruf BEDIEN meldet dem ORG nur die Bereitschaft, Bedienungsanweisungen anzunehmen. Das Warten ist nicht implizit enthalten. Trifft eine Bedienungsanweisung für dieses Programm ein, wird sie in den programmeigenen Bedienungspuffer geschrieben. Das Programm braucht nicht auf den Aufruf zu warten, sondern kann irgend etwas anderes machen. Ein Beispiel zur Erläuterung: Der Übersetzer protokolliert das übersetzte Programm.

Bei dem 24-bit-Rechner mußte er eine zweite Programmnummer haben, wenn er das Protokoll abbrechen wollte. Das ist beim ORG 330 nicht mehr nötig. Der Übersetzer gibt eine Anweisung BEDIEN ab ohne Warten und schaut während des Protokollierens nach, ob eine Bedienungsanweisung eingetroffen ist. Das kann im Extremfall soweit gehen, daß jedes Anwenderprogramm gleichzeitig Bedienungsprogramm sein kann.

4. Sehr viel Aufwand und Komfort wurde investiert, um bei Programmierfehlern bei ORG-Aufrufen trotzdem ein lauffähiges Programm zu erhalten. Es wird z. B. in der Beschreibung darauf hinge wiesen, daß Geräte und Dateien grundsätzlich belegt werden sollen. Wird es vergessen, holt das ORG trotzdem das Belegen nach. Es wird dann allerdings erst bei Programmende freigegeben. Will man in eine Datei schreiben, braucht im Extremfall nur der Transferaufruf gegeben zu werden. Das Einrichten, Belegen, Eröffnen, Schließen und Freigeben wird vom ORG automatisch durchgeführt. Das ist zum Teil möglich geworden durch eine andere Aufruf-Struktur, die in den letzten Jahren schon Eingang in die 24 - bit Modelle gefunden hat: die S-Aufrufe.

5. Die Anzahl der Parameter ist bei der 16-bit-Reihe stark ausgeweitet worden. In dem GEDA-Block für Dateien kann man nicht weniger als 45 Parameter angeben. Damit komme ich zu den weniger erfreulichen Seiten des ORG. Von diesen 45 Parametern des GEDA-Blocks sind 34 echte ORG-Parameter, die das ORG auch auswerten muß (ORG-Zeit). Außerdem muß der Programmierer über alle Bescheid wissen, und alle GEDA-Blöcke für diese Datei in verschiedenen Programmen müssen übereinstimmen, sonst meckert das ORG.

   Gegenüber der alten Rechnerfamilie hat sich die Programmier-Sprache vereinfacht, Aufrufe an das ORG sind aber wesentlich komplizierter geworden.

6. Schwer durchschaubar ist, zumindestens für mich, der Belegmechanismus. Es gibt drei verschiedene Belegmodi. Außerdem kann man noch nach Priorität und nach zeitlicher Reihenfolge belegen. Bis jetzt konnte mir noch keiner den Belegmechanismus verständlich erklären.

7. Eine andere zweifelhafte Angelegenheit wird der Platzwechselmechanismus von plattenspeicherresidenten Programmen (PRP) in Laufbereichen sein. Beim ORG 306 und ORGE 301 werden PRP an jeder Unterbrechbarkeitsstelle streng nach Priorität gewechselt. Beim ORG 330 PP II soll es anders werden.

   Die Prioritäten werden in 2 Klassen geteilt (evtl. in 3, wie ich hörte). Innerhalb einer Prioritätsklasse wird ein Programm nicht mehr durch ein anderes, selbst höherpriores unterbrochen. Damit hat man pro Laufbereich praktisch nur noch zwei echte Prioritäten. Das scheint mir ganz entschieden zu wenig zu sein. Wir haben die verschiedensten Experimente und Geräte am Rechner, die alle ihre eigenen Programme haben, außerdem sollen auch noch Background-Arbeiten gemacht werden, um den Rechner halbwegs auszulasten. Ich glaube, an anderen Anlagen sieht es oft nicht anders aus. Deshalb scheint mir die Idee der Prioritätsklassen, wenigstens in dieser Form, für einen Prozeßrechner nicht geeignet zu sein.

8. Probleme gibt es im Augenblick noch mit der ORG-Beschreibung. Obwohl sie gegenüber früheren Beschreibungen stark an Umfang zugenommen hat (über 500 Seiten), gibt es immer noch Dinge, die zu kurz kommen und deshalb unverständlich sind (Belegen von Dateien, Belegen) bzw. die Fehler enthalten (EINREROEF). Erst nach zahllosen Telefonaten mit Karlsruhe konnten wir unser erstes Programm zum Laufen bringen.

9. Zum Schluß noch eine Bemerkung zu einer vertanen Chance. Die 330 bietet hardwaremäßig die Möglichkeit des Speicherschutzes, dessen Grenze im Gegensatz zur 304 - 306 durch die Software eingestellt werden kann.

Leider wird diese Möglichkeit nur zum Schutz des ORG benutzt und nicht für Anwendungsprogramme, obwohl der Aufwand nach meiner Meinung nicht allzu groß sein kann. Dieser dynamische Speicherschutz hätte es ermöglicht, neu-entwickelte Programme on-line zu testen.

Bevor ich nach dem ORG auf einige Systemprogramme eingehe, muß ich auf eine weitere Eigenschaft des Software-Systems eingehen:

die Bibliotheksstruktur. ORG, AS 30 und SM 30 arbeiten nur noch mit Bibliotheken zusammen,d. h. das ORG kann Programme von Externspeichern nur aus einer Grundsprache - Bibliothek bereitstellen, nicht mehr aus einer Datei.

Der Makro - Übersetzer holt das zu übersetzende Programm aus der Quellsprache - Bibliothek, schreibt den Output wieder in dieselbe Bibliothek, in der auch die Makros stehen müssen. Das Protokoll wird auf Wunsch in der Textbibliothek abgelegt. Bei dem AS 30 ist es ähnlich. Die Bibliothekselemente, z. B. Programme sind zwar auch Dateien, aber nicht unter ihrem Programmnamen, sondern mit speziellen Bibliotheksnamen. Der Zugriff zu diesen Dateien geht über die Bibliotheksbuchführung. Dieses System hat einige Vorteile, setzt aber ein lauffähiges Programm DIPOS voraus.

DIPOS

Dieses Programm DIPOS enthält eine ganze Reihe von Funktionen, die bei der 24-bit-Familie in getrennten Programmen vorlagen. Es kann u. a.:

- Bibliotheken einrichten, löschen
- Bibliothekselemente löschen, protokollieren
- Datei-Buchführung protokollieren (EXLI)
- Dateien manipulieren
- Grundsprache und Quellsprache zwischen verschiedenen Datenträgern und Bibliotheken umsetzen (MCUM, ANUM)
- Quellsprache editieren (EDITOR)

Auf diesen EDITOR möchte ich kurz eingehen, weil er mir am Herzen liegt. Für denjenigen, der mit dem EDITOR vom Hahn-Meitner-Institut für die 24-bit-Familie gearbeitet hat, ist dieser neue EDITOR eine große Enttäuschung. Es gibt praktisch nur eine EDITOR-Funktion: das Einfügen oder Überschreiben einer Quellsprache- oder Textzeile, und selbst diese Funktion hat noch einen Haken. Man muß hinter den eingegebenen Zeichen mindestens noch soviel Blanks schreiben, daß das abschließende ETX nach dem 74. Zeichen steht, sonst kommt der Makro-Übersetzer nicht damit zurecht.

SM 30

Wie praxisfern die Programmentwickler bei Siemens manchmal sind, kann man am SM 30 bewundern. Sie können sich offenbar überhaupt nicht vorstellen, daß jemand einen Siemens-Rechner ohne Lochkarteneingabe kauft. Der SM 30 verlangt in Spalte 73 der Lochkarte, also auch auf Lochstreifen, bestimmte Zeichen zur Steuerung der Übersetzerfunktionen Können Sie sich vorstellen, wie man auf einem Lochstreifen in Spalte 73 ein Steuerzeichen setzen kann, bzw. in Spalte 80 ein ETX, wenn man zum Schreiben des Programms einen Blattschreiber benutzt? Außerdem darf kein Wagenrücklauf, Zeilenvorschub benutzt werden. An unserer 301 funktioniert es nach diesem Prinzip ausgezeichnet. Mit TABT und dem erwähnten HMI-EDITOR können wir praktisch genauso schnell, vielleicht sogar schneller, Programme erstellen, als mit Lochkarten.

Ein anderes Ärgernis hängt mit der Bibliotheksstruktur zusammen. Wie ich schon erwähnte, bearbeitet der SM 30 ein Element einer Bibliothek und erzeugt ein neues. Dazu belegt er diese Dateien. Daß aber zusätzlich noch die ganze Zeit die Bibliotheksbuchführung belegt wird, ist nicht einzusehen und außerdem ärgerlich. Denn eine Übersetzung dauert ziemlich lange ( bei einem kleinen Programm mit vorübersetzten Makros 15 min ). Während dieser Zeit kann weder DIPOS noch der AS 30 mit dieser Bibliothek arbeiten.

## TEPOS

TEPOS ist zum Abschluß wieder ein erfreuliches Kapitel. Es ist ein Testprogramm wie bei der 24-bit-Familie EXBE, aber mit einigen Zusatzfunktionen (z. B. TRACE-MODE) und wesentlich komfortabler.

Eine sehr wichtige Verbesserung besteht in der Angabe von Adressen. Bei EXBE mußten alle Adressen absolut angegeben werden. Jetzt kann man die Adressen relativ angeben, so wie sie aus dem Übersetzerprotokoll entnommen werden. Wenn man die Kernspeicherverwaltung des ORG benutzt, interessiert es auch gar nicht, wo das Programm im Kernspeicher steht.

## Schluß

Zusammenfassend kann ich von meinem Standpunkt aus sagen:
Die Assemblersprache ist ein großer Wurf. Die Systemsoftware hat gute Anlagen mit einigen Mängeln, die nach meiner Meinung durch Praxisferne entstanden sind und die sobald als möglich beseitigt werden sollten.