VERKÜRZUNG DER PROGRAMMLAUFZEIT VON FORTRAN-UNTERPROGRAMMEN DURCH AUSNUTZUNG DER R3OE-MASCHINENREGISTER AUF ASSEMBLEREBENE

Dietrich Schlichthärle, T&N GmbH, Frankfurt/Main

1. EINLEITUNG

Der folgende Beitrag wendet sich vor allem an jene Benutzer, die vor der Aufgabe stehen, sich mit einem Prozessrechner aus der R-Serie vertraut zu machen. Die Erfahrungen mit dem Rechner, die hier dargestellt werden sollen, wurden in einem Zeitraum von einem guten halben Jahr gesammelt. In dieser aus terminlichen Gründen recht knapp bemessenen Zeit mußten das Betriebssystem und die Assemblersprache erlernt sowie Erfahrungen mit dem Fortrancompiler gesammelt werden, um dann einige fortranaufbare Assemblerunterprogramme zu erstellen, die zum ORG-gerechten Anschluß einer speziellen Anwenderhardware an eine Schnittstelle vom Typ WAPO notwendig war. Mit diesen Unterprogrammen sollten folgende ORG-Funktionen aus dem Fortran heraus aktiviert werden:

1) POA-Bearbeitung
2) SEAP-Betrieb
3) I/O mit zentraler Initiative
4) Koordinationsfunktionen (Warten, Platzwechselsperre)
5) Dateiverkehr

Für die Handhabung des Rechners erwies sich eine sinnvolle Anwendung der Kurzzeichenbedienung besonders in Hinsicht auf einen Mehrbenutzerbetrieb gerade für den Anfänger als äußerst hilfreich. Beim Erlernen des Assemblers leistete die Übersetzungsmöglichkeit von Fortran in Assembler wertvolle Dienste.

2. KURZZEICHENBEDIENUNG

Einem Anfänger kann das Erlernen der doch recht komplizierten Bediensyntax erhebliche Schwierigkeiten bereiten. Bei Mehrbenutzerbetrieb kommt noch die Schwierigkeit hinzu, daß häufig die Masken für Compiler und Binder neu ausgefüllt werden müssen. Durch sinnvolle Anwendung der Kurzbe-

dienzeichen ist es jedoch möglich, auch auf dem Siemensrechner einen Bedienungskomfort zu erreichen, wie man ihn z.B. von amerikanischen Rechnern wie DEC oder HP gewohnt ist.

Für die Entwicklung von Fortranprogrammen müssen dem Anwender folgende Funktionen zur Verfügung stehen: Editieren, Compilieren, Binden, Laden und Auslisten von Text- und Quellsprachedateien

Während der Editor lediglich durch das betätigen einer einzigen Taste aufrufbar ist (einfacher geht es nicht), wurden für das Aktivieren der übrigen Funktionen Kurzbedienzeichen eingeführt.

Jedem Benutzer wurde zunächst eine Text-, eine Quellsprache- und eine Grundsprachebibliothek eingerichtet. Die dreibuchstabigen Bibliotheksnamen ergaben sich dabei folgendermaßen: Die ersten beiden Buchstaben (xy) charakterisieren den Benutzer, der dritte den Typ der Bibliothek.

xyT , xyQ , xyG

Der Fortrancompiler wird durch das Kurzzeichen

FORTxy,Dateiname;

aufgerufen. Durch das Kurzbedienzeichen wird der Monitor dazu veranlaßt, die Compilermaske auf den jeweiligen Benutzer umzustellen und den Compilerlauf zu starten.

Mit dem Aufruf

BINDxy;

wird entsprechend die Maske des Binders auf den jeweiligen Benutzer umgestellt und ein Bindelauf gestartet. Die notwendigen Informationen stehen in einem Bindefile mit fest vereinbartem Namen.

Für das Laden und Protokollieren wurden folgende Kurzbedienzeichen eingeführt:

LADExy,Grundsprachedatei,LBn,PP;

PRxyT,Textdatei;

PRxyQ,Quellsprachedatei;

Es zeigte sich, daß durch diese Handvoll einprägsamer Bedienkurzzeichen auch dem Anfänger ein rascher Einstieg in den Umgang mit dem Rechner ermöglicht wird.

3. ÜBERSETZUNG VON FORTRAN IN ASSEMBLER

Der Anfänger auf der Assemblerebene, der nach Studium des Manuals nun seine ersten kleinen Assemblerprogramme schreiben und ausprobieren möchte, trifft auf die Schwierigkeit, daß es keine einfache Ein- und Ausgabemöglichkeit gibt, die ihm eine Kontrolle seiner Programme ermöglicht. Zur Realisierung von I/Os wäre es notwendig, auch noch die Makrosprache zu erlernen, was bei einem Anfänger nur noch zu zusätzlicher Verwirrung führen würde. Da das Fernziel darin bestand, fortranaufrufbare Unterprogramme zu erstellen, wurde das Problem folgendermaßen gelöst: In Fortran wurde ein Hauptprogramm erstellt, das alle für die Kontrolle des Assemblerprogramms notwendigen Ein- und Ausgaben enthält sowie ein Unterprogramm, das im wesentlichen nur die Parameterübergabe realisiert. Das Hauptprogramm wurde mit dem Fortrancompiler FC30E in Grundsprache, das Unterprogramm in Assembler übersetzt. Dieser Assemblerunterprogrammrumpf ist ein sehr geeignetes Übungsobjekt, um erste eigene Assemblerbefehlsfolgen auszuprobieren. Sie werden einfach mit dem Unterprogrammrumpf in Grundsprache übersetzt und mit dem Hauptprogramm zusammengebunden. Nebenbei erlernt man auf diese Art recht schnell, wie fortrangerechte Assemblerunterprogramme zu schreiben sind.

Das Übersetzen aufwendigerer Fortranprogramme in Assembler erlaubt einen guten Einblick in die Wirkungsweise des Fortrancompilers. Das Optimieren derartiger aus Fortran entstandener Programme ist eine weitere gute Übung zum Erlernen der Assemblersprache. Der Benutzer wird dadurch schnell erkennen, daß Assemblerprogramme wesentlich effektiver bezüglich Speicherplatzbedarf und Rechenzeit gestaltet werden können, als dies z.B. in Fortran möglich ist.

Es zeigte sich, daß ein Anfänger mit dieser Methode etwa nach 4 Monaten in der Lage ist, unter Anwendung von Makrobefehlen eine anwenderspezifische Hardware ORG-gerecht an den Rechner anzuschließen.

4. OPTIMALES AUSNUTZEN DER R3O-MASCHINENREGISTER

Eine Analyse der Ausnutzung der R3O-Register durch den Fortrancompiler zeigt, daß diese nicht optimal genutzt werden. Für die Mathematik und die Indexberechnungen werden im wesentlichen die Register GO und G1 genutzt, während die R-Register nur für organisatorische Aufgaben wie Parameterübergabe, Rücksprungadressenkellerung und Unterprogrammsprünge eingesetzt werden. Dem Benutzer stehen jedoch alle acht G-Register uneingeschränkt zur Verfügung. Aus dem Bereich der R-Register kann der Benutzer über die Register RO, R3, R4, R5 und R7 frei verfügen. In einem einwandfrei laufenden, ausgetesteten Programm kann auch noch auf das Register R2 zurückgegriffen werden, wenn bis zum Rücksprung aus dem Unterprogramm die Kelleradresse in einer Hauptspeicherzelle gesichert wird.

Mit diesen zusätzlich zur Verfügung stehenden Registern können Programme in zweierlei Hinsicht optimiert werden:

1) Der Compiler spart lediglich dann einen Lade- oder Speichervorgang ein, wenn die Variable, auf die aufgrund des Übersetzungsvorganges einer Zeile als erstes zugegriffen werden muß, noch von der letzten Zeile her in einem Register steht. Um Hauptspeicherzugriffe zu sparen, sollten daher öfter benötigte Zwischenergebnisse in einem Register abgelegt werden.

2) Für jeden Feldelementzugriff muß die Basisadresse des feldes geladen sowie der Index aufaddiert werden. Bei der Abarbeitung eines Feldes in einer Schleife, was häufig vorkommt, genügt es jedoch in der Regel, die Adresse des vorher benutzten Feldelementes um eins zu inkrementieren oder dekrementieren. Dies kann in einem der noch freien Register geschehen.

Auch der Schleifenaufbau selbst kann in vielen Fällen drastisch vereinfacht werden. In der aus Fortran übersetzten Version sind pro Schleifenumlauf für die Schleifenorganisation 7 Maschinenbefehle nötig:

1x LAF/RAX	1,4 µs	
1x ADF/RC	0,8 µs	
2x SPR/CAX	2x0,8 µs	(Sprung nicht ausgeführt)
1x SPR/CAX	1,6 µs	(Sprung ausgeführt)
1x SPF/RAX	2,1 µs	
1x VGF/RAX	1,4 µs	
	8,9 µs	

Diese Zeit kommt schon in die Größenordnung einer Gleitpunktmultiplikation mit 9,8 µs. Opfert man ein Register für die Laufvariable der Schleife, kann sich der Aufwand bei günstigen Umständen auf einen einzigen Maschinenbefehl (DSP/RRX mit 1,6 µs) reduzieren.

Durch schrittweises Optimieren nach den obengenannten Gesichtspunkten konnte so die Laufzeit eines stark schleifenorientierten mathematischen Algorithmusses wie der Fast-Fourier-Transformation auf 1/3 reduziert werden.

1024-Punkte-FFT komplex	FORTRAN	1,75 s
	optimiert	0,55 s

5. ZUSAMMENFASSUNG

Die recht komplizierte Bediensyntax der R30 kann vor allem Anfängern erhebliche Schwierigkeiten bereiten. Auch potentielle Käufer einer Rechenanlage könnten abgeschreckt

werden, vor allem dann, wenn sie komfortablere Betriebssysteme gewohnt sind. Um diese Schwierigkeiten zu vermeiden, wäre es zu empfehlen, wenn SIEMENS

a) deutlicher auf die weitreichenden Möglichkeiten der Kurzbediensyntax aufmerksam machen würde und
b) dem Anwender eine detaillierte Anleitung für die Erstellung von Kurzbedienzeichensystemen an die Hand geben würde, wie sie etwa in diesem Beitrag vorgeschlagen wurden.

Die Übersetzungsmöglichkeit von Fortran in Assembler stellt eine gute Lernhilfe für die Assemblersprache dar. Vor allem das Optimieren von Fortranprogrammen ist eine interessante Übungsaufgabe. Neben Routine in der Programmierung lernt der Anwender, seine CPU optimal auszunutzen. Weiterhin erhält man einen guten Einblick in die Funktionsweise des Fortrancompilers und lernt recht schnell, fortrangerechte Assemblerunterprogramme zu erstellen.

(Dieser Beitrag beruht auf Erfahrungen, die an der R30E des Fachgebiets Meßtechnik an der Universität Dortmund gesammelt wurden)