G. Weck

Operationaler Vergleich der Rechner TR 440, Siemens 7.760, VAX11/780 und Modcomp 7870

Zweite, überarbeitete und erweiterte Auflage

Fachbereich
Angewandte Mathematik
und Informatik
der Universität des Saarlandes

D-6600 Saarbrücken

Juni 1980

Bericht Nr. A80-04

GELEITWORT

Den Lesern dieses Berichts seien folgende Hinweise mitgegeben:

- Ein Universalrechner, die Siemensanlage 7.760/7.770, wird verglichen mit dem System TR440 einerseits und Prozeßrechnern (in erster Linie der VAX11) andererseits. Am "unteren" Ende dieser Meßskala steht mit dem TR440 ein System, das in vieljähriger Entwicklung unter nahezu ausschließlicher Orientierung am Hochschulbereich einen sehr hohen Standard erreicht hat.

  Das "obere" Ende dieser Skala bilden Prozeßrechner, die auf Grund ihrer (von vornherein akzeptierten, weil aus ihrer Aufgabenstellung resultierenden) Beschränkung der "Universalität" dafür eine erhebliche Steigerung der Leistung und Flexibilität im überdeckten Bereich erzielen.

  Wir sollten uns der Relativität dieses Maßstabes bewußt sein - eines Maßstabes, bei dem auch andere, daran gemessene Universalrechner ihre Schwierigkeiten hätten.

- Dieser Bericht zielt im wesentlichen auf die Qualität der Rechner als Automaten zur "Programmentwicklung" ab. Für ein Universitätsrechenzentrum, das sämtliche Fachbereiche und Einrichtungen einer Universität bedient, ist natürlich die Rechenleistung im anderen Sinne, d. h. die Eigenschaft, rechen - oder datenintensive Routine - und Produktionsläufe abzuwickeln von gleichrangiger Wichtigkeit.

Der Zweck dieses Berichts besteht nicht darin, spezielle Stärken oder Schwächen spezieller Rechner herauszugreifen. Das Ziel dieses Berichts besteht vielmehr darin, an einem Beispiel Aspekte und Trends von prinzipieller Bedeutung aufzuzeigen. Dies ist eindrucksvoll gelungen:
Herr Dr. Weck zeigt auf, daß infolge des Technologiefortschritts, des Preisverfalls und der raschen Entwicklung im Mini- (und Mikroprozessor-) Bereich dem Aspekt der "Dezentralisierung" im weitesten Sinne steigende Bedeutung beizumessen ist. Man kann aus dieser Sicht insbesondere die Frage stellen, ob es nicht für den Hersteller und den Kunden sehr viel effizienter ist, die Entwicklung klassischer, universeller Betriebssysteme einzustellen und dafür "verteilte" Systeme zu entwickeln, die z. B. Dialogfunktionen prinzipiell und möglichst weitgehend auf dedizierten

Vorrechnern abwickeln, die mit einem "Batchrechner" im Hintergrund hinreichend schnell gekoppelt sind.

Die unsichere Form dieser Fragestellung läßt gleichermaßen die Komplexität des Problems und die "Instabilität" der augenblicklichen Situation erkennen, in der der Fortschritt der Hardware-Technologie von gestern die Softwarekonzepte von morgen überrollt.

Dieser Bericht macht deutlich, daß wir heute die Realisierung der "Idealkonfiguration" eines Rechenzentrums, die aus einem lokalen Netz besteht, das jedem Benutzer einen "eigenen" (physikalisch oder hypothetisch vorhandenen) Rechner mit für ihn passenden Eigenschaften und Schnittstellen zur Verfügung stellt, als Zielvorstellung ins Auge fassen können.

Offensichtlich ist der Weg zur Erreichung dieses Zieles jedoch immer noch lange und aufwendig.

Bedenkt man außerdem, daß bei der Auswahl eines Rechnersystems noch eine Vielzahl anderer, hier nicht diskutierter Erwägungen von Bedeutung ist, dann ist zu erwarten, daß ein Rechenzentrum zur Durchführung seiner Servicefunktionen noch geraume Zeit auf die Dienste eines Universalrechners angewiesen sein wird. Allerdings sollte dieser in ein Netz eingebettet werden, das sich schrittweise der Idealkonfiguration annähern kann.

In diesem Sinne ist die bestellte Siemensanlage 7.770 für die Universität des Saarlandes als sichere Ausgangsbasis anzusehen.

H. Scheidig

Inhalt
======

27. 6.80
We

27. 6.80
We

Vorwort

Die Idee zu diesem Bericht kam bei einem Vergleichstest zwischen der Siemens-Anlage 7.760 und der TR440 (erstes Beispiel), der einige recht unerwartete Resultate brachte. Bei genauerer Betrachtung dieser Ergebnisse schien es lohnenswert, durch weitere Tests Daten zu sammeln, die eine Interpretation dieses ersten kleinen Tests erlauben.

Der folgende Bericht beschreibt mehrere Rechner aus der Sicht eines Software-Entwicklers. Es ist daher klar, dass die hier dargestellten Ergebnisse und ihre Interpretation nicht alle Aspekte der einzelnen Rechner beschreiben; ausserdem musste wegen der Fuelle des zugrundeliegenden Materials eine gewisse Auswahl getroffen werden, die nach dem genannten Gesichtspunkt geschah. Daraus ergibt sich automatisch, dass die hier gemachten Feststellungen in keiner Weise den Anspruch erheben koennen, eine offizielle Stellungnahme des Saarbruecker Rechenzentrums zu bilden, doch schienen die gesammelten Ergebnisse wichtig genug, um sie einem groesseren Kreis verfuegbar zu machen.

Ich moechte allen danken, durch deren Mithilfe es mir erst moeglich wurde, das hier dargestellte Material in relativ kurzer Zeit zu sammeln. Hier sind die Mitarbeiter R. Berberich, H. Frick, B. Kett, J. Messerschmidt, A. Neisius und A. Schmitt des Rechenzentrums der Universitaet des Saarlandes und des Fachbereiches Angewandte Mathematik und Informatik zu nennen, die bei der Erprobung der Testbeispiele geholfen haben sowie durch Korrekturlesen und konstruktive Kritik dafuer gesorgt haben, dass unklare Stellen und Ungenauigkeiten beseitigt werden konnten. Die Verantwortung fuer eventuell noch vorhandene Fehler und Auslassungen, die beim Umfang des untersuchten Materials praktisch unvermeidbar sind - von denen ich aber hoffe, dass sie so gleichmaessig verteilt sind, dass sie die Ergebnisse nicht verschieben - liegt selbstverstaendlich nur bei mir selbst.

Besonderen Dank moechte ich den Herren G. Hamann und H. Wendel vom Institut fuer zerstoreungsfreie Pruefverfahren der Fraunhofer Gesellschaft und V. Gartner von der Firma Modcomp aussprechen, die es mir ermoeglicht haben, den Vergleich auch auf die Rechner VAX11/780 und Modcomp 7870 auszudehnen.

Abschliessend moechte ich meiner Hoffnung Ausdruck geben, dass dieser Bericht Probleme der derzeitigen Rechnersituation an deutschen Hochschulen aufweist und Denkanstoesse zu ihrer Loesung gibt. Ich wuerde mich freuen, so einen Beitrag zur Entwicklung leistungsfaehiger Hochschul-Rechenzentren der Zukunft zu leisten.

27. 6.80
We

Vorwort zur 2. Auflage

Die relativ grosse Nachfrage nach der vorliegenden Studie fuehrte dazu, dass die erste Auflage innerhalb sehr kurzer Zeit vergriffen war. Gleichzeitig erhoben sich einige Kontroversen um manche der Aussagen dieser Studie. Beide Aspekte machten die Herausgabe einer zweiten Auflage unter Beruecksichtigung neuer, in der Zwischenzeit verfuegbarer Information notwendig.

Die 2. Auflage wurde unter Beruecksichtigung der Ergebnisse von Gespraechen mit den Herstellern der verglichenen Rechner sowie unter Einbeziehung des gegenwaertigen Standes der Diskussion um die weitere Entwicklung des Saarbruecker Rechenzentrums erstellt. Bei dieser Diskussion wurden die in der 1. Auflage der Vergleichsstudie erarbeiteten Ergebnisse in Zusammenarbeit zwischen dem Fachbereich Angewandte Mathematik und Informatik und dem Rechenzentrum der Universitaet des Saarlandes daraufhin betrachtet, wie weit sie massgebliche Hinweise auf eine sinnvolle und realisierbare Weiterentwicklung des Saarbruecker Rechnernetzes geben koennen. Die diesbezueglichen Resultate wurden in die 2. Auflage des vorliegenden Berichts einbezogen.

Durch die Hinzunahme weiteren Materials und durch Ueberpruefung aller Aussagen der 1. Auflage ergaben sich in manchen Einzelheiten kleinere Aenderungen und Erweiterungen, doch keine wesentliche Verschiebung am fachlichen Gehalt des Berichts. Um eventuellen Fehlinterpretationen der hier getroffenen fachlichen Aussagen vorzubeugen, wurde der Bericht insbesondere um genauere Spezifikationen des verwendeten Vergleichsmassstabes ergaenzt. Insbesondere sei hier darauf hingewiesen, dass ein solcher Vergleich natuerlich keine Aussage darueber beinhaelt, ob einer der betrachteten Rechner in einem absoluten Sinn besser oder schlechter ist als ein anderer; das Ziel eines "Operationalen Vergleichs" ist eine Aussage darueber, wie gut die betrachteten Rechner einem bestimmten Aufgaben-Profil genuegen.

Abschliessend moechte ich allen Mitarbeitern des Fachbereichs Angewandte Mathematik und Informatik und des Rechenzentrums der Universitaet des Saarlandes sowie der Siemens AG fuer die Diskussionen danken, die den Inhalt des Berichts erweitern und praezisieren halfen.

18. 7.80
We

1. Einleitung

1.1. Zielsetzung

Fuer den waehrend der ersten Phase des Migrationsprojekts vom TR440 auf eine Siemens-BS2000-Anlage geplanten Parallelbetrieb beider Maschinen stellt sich das Problem, eine Abschaetzung der Leistungsfaehigkeit der neuen Anlage Siemens 7.760 sowie ihrer Einsatzmoeglichkeiten zu finden, um mit dieser Hilfe einen effizienten Parallelbetrieb organisieren zu koennen.

Hierzu wurde eine Reihe von Tests durchgefuehrt, die im wesentlichen die folgenden Parameter erfassen sollten:

- die Geschwindigkeit numerischer Berechnungen
- die Effizienz des von verschiedenen Compilern erzeugten Codes
- die Genauigkeit der Gleitkomma-Arithmetik
- die Durchlaessigkeit und Parametrisierbarkeit des E/A-Systems
- das Antwort-Verhalten im interaktiven Betrieb
- die Handhabbarkeit des Programmiersystems hinsichtlich der zur Unterstuetzung der interaktiven Programm-Entwicklung und der Implementierung bzw. Uebernahme fertiger Software-Pakete zur Verfuegung gestellten Hilfsmittel, insbesondere:
  - Text-Editor
  - Erzeugung und Verwaltung von Modul- und Programm-Bibliotheken
  - Testhilfen
  - Eingriffs-Moeglichkeiten
  - Kommando-Sprache (Umfang, Format, Interaktivitaet)
  - System-Meldungen (Umfang, Verstaendlichkeit)
- das ergonomische Verhalten des Systems, insbesondere:
  - Verstaendlichkeit
  - Erlernbarkeit
  - Arbeitsaufwand zur Steuerung (besonders Schreibaufwand)
  - Tolerierung von Benutzerfehlern
  - Reaktion auf Fehler des Benutzers
  - Uebersichtlichkeit
  - Einheitlichkeit der Benutzerschnittstellen
  - Vorhersehbarkeit des Systemverhaltens
  - Kontrollierbarkeit des Systems durch den Benutzer

Waehrend sich die zuerst genannten Groessen ohne ernste Schwierigkeiten quantitativ erfassen lassen, koennen die beiden letzten Problemkreise nur qualitativ beurteilt werden. Alle diese Groessen beschreiben im wesentlichen das Verhalten eines Rechners aus der Sicht des einzelnen Benutzers an einem interaktiven Eingabe-Geraet; der Schwerpunkt der Betrachtungsweise liegt dabei auf dem Einsatzgebiet

der Software-Entwicklung. Die Ergebnisse der vorliegenden Studie sind daher vor allem fuer einen bestimmten Anwenderkreis relevant, doch sind die meisten der hier getroffenen Aussagen im Prinzip fuer alle Benutzer eines Rechners von Bedeutung, wenn auch zum Teil nur am Rande.

Die in der vorliegenden Untersuchung betrachteten Faktoren beziehen sich sehr stark auf Software-Entwicklung in Forschung und Lehre; diese wird an Universitaeten weitgehend von Personal durchgefuehrt, das den Rechner als Hilfsmittel zur Erreichung eines bestimmten Zieles verwendet, nicht jedoch von hauptberuflichen Programmierern wie in kommerziellen Rechenzentren. Die Vertrautheit mit den Begriffen und Arbeitsweisen, die in der EDV ueblich sind, ist hier in sehr verschiedenem Masse vorhanden; waehrend ein Teil der Benutzer am besten mit einem "Turn-Key"-System bedient waere, das ausser einem Einschaltknopf keine weiteren Bedienungselemente haben sollte, gibt es auch Benutzergruppen, deren Kenntnisstand als ausgesprochen hoch zu bezeichnen ist. Entsprechend vielfaeltig sind auch die Anforderungen, die an einen Universitaetsrechner gestellt werden: Vom Laufenlassen eines einfachen Auswertungsprogramms bis hin zu Experimenten auf System-Ebene sollten alle Arbeitsweisen moeglichst optimal unterstuetzt werden.

Zu diesen Anforderungen kommt in den letzten Jahren in zunehmendem Masse die Belastung durch Programmierkurse, in denen einer grossen Anzahl von Studenten (zur Zeit etwa 600) innerhalb eines Semesters die Grundbegriffe der Programmierung in einer hoeheren Programmiersprache (in Saarbruecken insbesondere FORTRAN, COBOL und Pascal) beigebracht werden muessen. Wenn diese Kurse mit praktischen Uebungen, zweckmaessig im interaktiven Betrieb ueber Terminals, durchgefuehrt werden sollen, so stellt dies natuerlich noch einmal Anforderungen besonderer Art an einen Universitaetsrechner; auch diese Situation ist grundlegend verschieden von der Programmierer-Ausbildung in einem kommerziellen Rechenzentrum.

Neben diesen universitaetsspezifischen Aufgaben fuehrt ein solches Rechenzentrum ueblicherweise auch Aufgaben kommerziellen Charakters aus, die eher dem Benutzerprofil der kommerziellen EDV entsprechen.

Da es ausgesprochen schwierig ist, alle diese Arten der Arbeit mit einem Rechner unter einem gemeinsamen Gesichtspunkt zu betrachten, werden in den folgenden Abschnitten zumindest die kommerziellen Anwendungen, der Batch-Betrieb und alle routinemaessigen Produktionslaeufe bewusst aus der Betrachtung ausgeklammert, um zumindest fuer die interaktiven Anwendungen des Rechners zu sinnvollen und ueberschaubaren Aussagen zu kommen.

Aus diesen Gruenden werden insbesondere keine Aussagen ueber folgende Parameter gemacht:

- Zuverlaessigkeit
- Datenschutz
- Unterstuetzung grosser Batch-Anwendungen

- Unterstuetzung reiner Produktionslaeufe fertiger Programme

- Unterstuetzung von Transaktions-Betrieb

- globale System-Kontrolle

- Systemverhalten im Jobmix

- Ausbaumoeglichkeiten (besonders Hintergrundspeicher)

- Lastgrenze der Systeme

- Verfuegbarkeit von:
  - Programmiersprachen
  - Datenbanksystemen
  - Anschlussmoeglichkeiten fuer eigene/fremde Hardware
  - Hilfsmitteln zur Systemwartung (Hardware/Software)
  - Software-Pakete (des Herstellers/von Anwendern)

Die Ergebnisse der folgenden Untersuchungen sind daher unter Beruecksichtigung des Untersuchungszieles zu interpretieren, naemlich:
"Wie verhaelt sich das einzelne System einem Benutzer gegenueber, der von einem Terminal aus interaktive Programm-Entwicklung macht?"

Aussagen darueber, ob sich ein System, das sich im Hinblick auf diese Fragestellung gut verhaelt, auch insgesamt bewaehrt, lassen sich aus dieser Untersuchung wegen der Einschraenkung der Fragestellung nicht machen. Dagegen werden Maengel in Bezug auf diese spezielle Fragestellung allgemein den Gebrauchswert eines Systems negativ beeinflussen, unabhaengig davon, ob das betreffende System in anderer Hinsicht eventuell Vorzuege hat. Die Ergebnisse dieser Studie sind in diesem Sinn als Beschreibung von oberen Leistungsgrenzen fuer die betrachteten Rechner zu werten.

Es darf hier jedoch nicht uebersehen werden, dass in einem Universitaets-Rechenzentrum ein Benutzer-Profil vorherrschend ist, das sich sehr auf interaktive Programm-Entwicklung durch zum grossen Teil mit dem Rechner nur unvollstaendig oder gar nicht vertraute Personen konzentriert, im Gegensatz zu vielen kommerziellen Rechenzentren, bei denen Produktionslaeufe fertiger Programm-Pakete, betreut durch eine relativ geringe Anzahl von mit dem Rechner relativ gut vertrauten Leuten, vorherrschen. Dieses spezielle Benutzer-Profil einer Universitaet bringt es mit sich, dass gerade die nicht quantifizierbaren Aspekte eines Betriebssystems sehr stark die Effizienz der einzelnen Programmierer beeinflussen; sie haben daher letztlich auf die Verwendbarkeit und Leistungfaehigkeit einer Anlage in dieser Umgebung einen erheblich groesseren Einfluss als die reine Prozessorgeschwindigkeit oder die Speichergroesse. Aus diesem Grund werden die Qualitaet der vom System zur Verfuegung gestellten Dienstleistungen sowie ihre ergonomischen Charakteristika in der folgenden Analyse besondere Beachtung finden.

Bei der Interpretation der Resultate dieses Vergleichs ist natuerlich zu bedenken, dass damit keine Aussage getroffen wird, ob ein

bestimmter Rechner gut oder schlecht bzw. besser als ein anderer in einem allgemeinen Sinn ist. Eine solche Aussage waere auch sinnlos, da die Strukturen und auch die intendierten Anwendungsgebiete der einzelnen Rechner und Betriebssysteme so verschieden voneinander sind, dass ein allgemeiner Vergleich keine absolute Aussage als Ergebnis haben kann. Durch Einschraenkung der Betrachtungsweise lassen sich jedoch sehr wohl Aussagen darueber machen, ob ein bestimmter Rechner fuer eine definierte Aufgabe gut oder schlecht geeignet ist, und genau eine solche Aussage ist das Ziel dieser Untersuchung: die Frage, ob und in welchem Umfang die betrachteten Rechner fuer einen Einsatz als Timesharing-Maschinen fuer interaktive Programm-Entwicklung unter den Randbedingungen eines Universitaets-Rechenzentrums geeignet sind. Eine solche eingeschraenkte Aussage hat dennoch als Entscheidungshilfe keinen geringen Wert, da es relativ sinnlos ist, einen Rechner fuer eine Anwendung einzusetzen, fuer die er nicht geeignet ist, waehrend andererseits eine anwendungsbezogene Rechnerauswahl zu insgesamt effizientem und oekonomischem Rechnereinsatz fuehrt.

Um zu verhindern, dass die nachfolgende Analyse zu einseitig und voreingenommen wird, indem sie sich nur auf die beiden Rechner TR440 und Siemens 7.760 bezieht und die dort vorgefundenen Verhaeltnisse verabsolutiert, wurden einige der Tests auch auf weiteren Rechnern, auf die Zugriff bestand, durchgefuehrt. Die Auswahl dieser Rechner geschah dabei allein unter dem Aspekt der Verfuegbarkeit, so dass auch ihre Einbeziehung zu keinen absoluten Ergebnissen fuehrt; dennoch erlaubt es ihre Einbeziehung, die fuer die Rechner des Migrations-Projektes (*) gewonnenen Ergebnisse in einen etwas groesseren Rahmen zu stellen und so zu Aussagen groesserer Allgemeinheit zu kommen. Weil in alle der betrachteten Rechner ausser dem TR440 eine explizite Einarbeitung erforderlich war, war es moeglich, den Kenntnis- und Gewohnheitsvorsprung der Arbeit mit dem TR440 abzuschaetzen und so aus den Betrachtungen qualitativer und ergonomischer Art zu eliminieren, so dass diese einen hoeheren Grad an Objektivitaet und Allgemeingueltigkeit bekommen, als dies bei einem reinen Vergleich zwischen nur einem bekannten und einem neuen Rechner moeglich waere.

Schliesslich muss hier noch betont werden, dass der vorliegende Vergleich nur existierende Software beruecksichtigt; Produkt-Ankuendigungen einzelner Hersteller wurden ignoriert, sofern diese Produkte nicht in auslieferbarer Form vorliegen. Hier haette im Prinzip nur die Alternative bestanden, die Produkt-Philosophie aller der beteiligten Hersteller miteinander zu vergleichen, was aber dieser Studie einen voellig anderen Inhalt und eine andere Zielsetzung gegeben haette. Da jedoch hier versucht werden sollte, Informationen ueber den Ist-Stand zu erhalten, mussten noch nicht oder erst auf dem Papier existierende Produkte unberuecksichtigt bleiben. Ein Vergleich geplanter Produkte eines Herstellers mit existierenden Produkten eines anderen Herstellers haette ausserdem eine schiefe Betrachtungsweise zur Folge gehabt; er ist aus Gruenden der intellektuellen Redlichkeit und der wissenschaftlichen Korrektheit strikt abzulehnen.

---

* : Das Migrationsprojekt ist ein vom BMFT unter der Nr. APM-2-081 2252 gefoerdertes Vorhaben des GRZ Berlin und der Rechenzentren der Universitaeten Duesseldorf, Kaiserlautern und Saarbruecken.

1.2. Untersuchte Rechner

1.2.1. Uebersicht

Hier sollen die wichtigsten Charakteristika der wesentlichsten der in diesem Vergleich untersuchten Rechner kurz dargestellt werden. Dabei wird auf die im ersten Beispiel des Vergleichs auch erwaehnten Rechner AEG80/20 und AEG80/40 (16- bzw. 32-bit-Prozessrechner) nicht weiter eingegangen, da die Ergebnisse fuer diese beiden Rechner so schlecht waren, dass eine weitere Beruecksichtigung im Vergleich nicht sinnvoll erschien. Ebenso eruebrigt sich eine genauere Beschreibung der Systeme Tektronix 4051 und SMP80, zweier Mikroprozessor-Systeme auf der Basis der Prozessoren Motorola 6800 bzw. Intel 8080, die nur interessehalber in den ersten Vergleich aufgenommen wurden. Auch auf eine genauere Beschreibung der Siemens-Anlage 7.541 kann hier verzichtet werden, da es sich dabei um eine kleinere und langsamere Version der 7.760 handelt.

Die Tabelle der beiden folgenden Seiten soll dabei nur einen groben Ueberblick ueber die wesentlichsten Eigenschaften der betrachteten Maschinen vermitteln. Eine vollstaendige Charakterisierung der betrachteten Rechner auf so kleinem Raum ist natuerlich nicht moeglich; ein solcher Vergleich wird zusaetzlich durch Unterschiede in den Architekturen der einzelnen Rechner, die in der Tabelle nicht beruecksichtigt werden konnten, sehr schwierig. Als weitere Komplikation kommt hinzu, dass aufgrund von Verschiedenheiten der Architektur manche Werte der Tabelle bei den einzelnen Rechnern leicht unterschiedliche Bedeutungen haben. Insgesamt ergibt sich das in der folgenden Tabelle dargestellte Bild.

Anmerkungen zur Tabelle auf den folgenden Seiten:

(*) keine Daten verfuegbar

(a) 2 Bit Typenkennung

(b) zusaetzlich nicht-privilegierter Befehlsvorrat der pdp11

(c) jedoch auch Zweitbefehle moeglich

(d) I: integer, R: real, E: extended precision;
I*3/6 bzw. R*6/12 werden bei der TR440 als I*2/4 bzw. R*4/8 bezeichnet

(e) organisiert als 256 k bzw. 512 k Worte mit je 52 Bit

(f) zusaetzlicher Speicher als Multiport-Memory bei Mehrprozessor-Systemen

(g) errechnete Werte

(h) Ausstattung der Testsysteme des Vergleichs

(i) historischer Preis

	TR440	Si 7.760	VAX11/780	Modcomp
Verarbeitungs-breite [bit]	48+2(a)	32	32	16..80
Instruktionen	236	169	248(b)	367
Befehlsformate	1(c)	5	variabel	3
Befehlslaenge[bit]	24	16/32/48	8..296	16/32
Datentypen (d)	I*3/6 R*6/12 Pack.Dec. Numeric String Bitfeld	I*2/4 R*4/8/16 Pack.Dec. Numeric String Bitfeld	I*1/2/4/8 R*4/8/E Pack.Dec. Numeric String Bitfeld	I*2/4 R*4/6/8 String Bitfeld
Register fest allgemein verw.	11 -	34 51	64 + 5 12	(*) 16 * 15
Cache [kByte]	-	32	8	-
Hauptspeicher[MB] maximal:	1.5 (e) 3 (e)	4 8	1 8(+4)(f)	0.5 2
Zykluszeit [ns] Hauptspeicher Cache effektiv	 900 - 125 (g)	 1600(g) 200 270(g)	 1800 200 290	 (*) - 450
Datenrate [MB/s]	6	6	13.3	8
Plattenspeicher [MB] (h)	8 * 30 9 * 200	4 * 300 5 * 400	2 * 176	1 * 80
Kanalrate [MB/s] (Schnellkanal)	3	3.3	2	1
Schnellkanaele max	4	6	4	16

27. 6.80
We

	TR440	Si 7.760	VAX11/780	Modcomp
Log. Adressraum[B]	192k/12M	16 M	4096 M	4 M
Adressierung	virtuell	virtuell	virtuell	virtuell
Adressraumverw.	paging	paging	paging	paging
Speicherverwaltung	real	virtuell	virtuell	real
Prog.-Verdraengung	swapping	(*)	swapping	swapping
Front-End-Proz.	TR86S AEG80/20 DUET	DUET	-	-
Service-Proz.	-	-	LSI11	-
Jobs (max.)	> 200	128	64	(*)
Technologie	ECL I	ECL,TTL, Schottky	Schottky-TTL	Schottky-TTL
Speicher	Ferritkern	MOS	MOS	Kern/MOS
Preis [Mio DM]	13 (i)	5	0.65	0.3

Die hier angegebenen Preise sind die der Testsysteme im Vergleich; sie enthalten insbesondere auch die durchaus nicht zu vernachlaessigenden Preise fuer die Peripherie. So wuerde etwa eine VAX11 mit einem Ausbau der Plattenperipherie auf 3 Gigabyte etwas ueber 1.5 Mio DM kosten, waehrend umgekehrt eine Siemens 7.541, die vom Hersteller als das zur VAX11 vergleichbare Modell betrachtet wird, mit Floating-Point-Prozessor, 2 Megabyte Hauptspeicher und 350 Megabyte Plattenkapazitaet etwa 0.8 Mio DM kosten wuerde.

27. 6.80
We

1.2.2. Kurz-Charakterisierung

Um einen allgemeinen Eindruck von den wichtigsten Eigenschaften der Rechner des Vergleichs zu geben, werden hier schlagwortartig einige wesentliche Charakteristika der Architekturen dieser Rechner genannt, ohne dass diese Aufstellung in irgendeiner Weise den Anspruch erhebt, vollstaendig oder repraesentativ zu sein.

- TR440: Es handelt sich um einen Grossrechner mit einer relativ exotischen Architektur, gekennzeichnet durch:

  - Verwendung einer Typenkennung zur Unterscheidung verschiedener Datentypen

  - ungewoehnliche Adressierungsstruktur mit Adressierung z.T. ueber Zweitbefehle im Adressteil und Modifikatoren

  - Verwendung von Registern mit fester Bedeutung, aufgeteilt in Rechenwerks- und Befehlswerks-Register

  - Verwendung einer im Prinzip beliebig grossen Anzahl von Index-Registern, aufgeteilt in Registerbloecke; die Register sind im Hauptspeicher abgespeichert, doch werden die 4 aktuellsten Register in einem Assoziativspeicher gehalten

  - generell einfache Befehle, doch stehen auch sehr komplexe Sonderbefehle zur Verfuegung

  Das verwendete Betriebssystem BS3 ist ein System fuer gemischten Batch- und Timesharing-Betrieb.

- Siemens 7.760: Es handelt sich um einen mikroprogrammierbaren Grossrechner mit einer Architektur, die aehnlich der des IBM-Systems /370 ist, gekennzeichnet durch:

  - meist einfache Maschinenbefehle in mehreren vorgegebenen Formaten; diese spezifizieren hauptsaechlich den Adress-Aufbau

  - 16 allgemeine Register (im Grundzustand), zusaetzlich 4 doppelt lange Gleitpunkt-Register

  - hohe Prozessorgeschwindigkeit

  - grosser Cache-Speicher

  - Verwendung mehrerer Maschinenzustaende mit jeweils eigenen Registersaetzen

  Das verwendete Betriebssystem BS2000 ist als Vielzweck-System fuer Batch, Transaktions-Betrieb und Timesharing konzipiert.

- VAX11/780: Es handelt sich um einen mikroprogrammierbaren Rechner der "Super-Mini"- (oder "Midi"-) Klasse mit einer folgendermassen zu kennzeichnenden Architektur:

  - sehr komplexe Befehle mit meist mehreren Adressteilen; die Befehle entsprechen zum Teil Sprachkonstruktionen hoeherer Programmiersprachen

  - Adressierung ist unabhaengig vom Befehl; fuer jeden Adressteil koennen separat bis zu 38 verschiedene Adressierungsmodi spezifiziert werden

  - keine E/A-Befehle; E/A-Geraete werden ueber ihre Register angesprochen, die Adressen im Adressraum der Maschine haben

  - 16 allgemeine Register, davon 4 softwaremaessig mit Sonderbedeutungen belegt

  - assoziativer write-through Cache-Speicher

  - "intelligente" Konsole mit Watchdog-Prozessor und Anschluss fuer Ferndiagnose

  Das verwendete Betriebssystem VAX/VMS ist ein kombiniertes Real-Time-/Timesharing-Betriebssystem mit Moeglichkeiten fuer Batch-Betrieb.

- Modcomp 7870 (CLASSIC): Es handelt sich um einen Rechner der "Super-Mini"-Klasse, dessen Architektur folgendermassen zu kennzeichnen ist:

  - sehr umfangreicher Befehlsvorrat

  - mehrere parallele Registersaetze mit je 15 allgemeinen Registern

  - extrem schneller Prozesswechsel

  - weitgehende Parallelisierung der Datenwege (Multibus-Architektur)

  - schneller Hauptspeicher

  Das verwendete Betriebssystem MAX IV ist ein prioritaetsgesteuertes Real-Time-System mit Timesharing-, Transaktions- und Batch-Moeglichkeiten.

Fuer ausfuehrlichere Information ueber die einzelnen Rechner muss auf die Unterlagen der Hersteller verwiesen werden.

27. 6.80
We

2. Umfang und Durchfuehrung der Tests

2.1. Quantitative System-Aspekte

Vergleichswerte fuer die Geschwindigkeit der betrachteten Rechner - sowohl absolut als auch im Hinblick auf die Effizienz des von verschiedenen Compilern erzeugten Codes - wurden durch Betrachtung der Zeiten gewonnen, die fuer die Durchfuehrung einfacher, rechenintensiver Programme benoetigt wurden; gleichzeitig ergaben sich hier auch Aussagen ueber die Genauigkeit der Gleitpunkt-Operationen der einzelnen Rechner. Dabei wurden die betrachteten Programmiersprachen hauptsaechlich unter dem Gesichtspunkt der Verfuegbarkeit der Compiler auf den Testmaschinen ausgewaehlt; weitergehende Schluesse irgendwelcher Art sind aus der getroffenen Auswahl nicht zu ziehen. COBOL wurde generell nicht betrachtet, so dass aus den Tests keine Aussagen ueber das Verhalten der Maschinen in einem rein kommerziellen Jobmix zu entnehmen sind; das Schwergewicht der Untersuchung liegt dabei einwandfrei auf Anwendungen technisch-wissenschaftlichen Charakters mit FORTRAN als wichtigster Programmiersprache.

Es braucht nicht eigens betont zu werden, dass sich aus einzelnen Programmen keine allgemeine Beurteilung der Leistungsfaehigkeit und der Geschwindigkeit der betrachteten Rechner ableiten laesst, zumal diese Programme aller Wahrscheinlichkeit nach fuer den von den Anlagen zu bearbeitenden Jobmix nicht repraesentativ sind. Dies gilt insbesondere, da saemtliche der betrachteten Programme nur die reine Rechenleistung erfassen; E/A-Geschwindigkeiten wurden hier nicht betrachtet. Dennoch lassen sich bei etwas genauerer Betrachtung und Analyse der Vergleichsergebnisse sinnvolle Aussagen ueber die Groessenordnung der von den einzelnen Maschinen zu erbringenden Rechenleistung und moegliche Engpaesse machen; insbesondere geben die Messergebnisse Hinweise darauf, an welchen Stellen weitergehende Untersuchungen quantitativer Art zweckmaessig waeren.

Das erste der verwendeten Testbeispiele berechnet den Wert pi**2/3 (= 3.289868134) als Summe ueber 1/n**2 unter Variierung folgender Randbedingungen:

- verwendete Programmiersprache
- spezifizierte Rechengenauigkeit
- Anzahl der Iterationen (vorgegeben bzw. ueber Abbruchkriterium)

Der betrachtete Algorithmus lautet (in FORTRAN-Notation):

```
      Y=0.
      DO 1 I=1,1000000
      Z=Y
      X=FLOAT(I)
      Y=Y+2./(X*X)
      IF(Y.EQ.Z) GO TO 3
    1 CONTINUE
    3 WRITE(4,2) Y,I
    2 FORMAT(1X,E25.12,I10)
      END
```

Dabei wurden zum Teil kleine Aenderungen vorgenommen, um die verschiedenen Abbruchkriterien zu spezifizieren. Speziell wurde die Abfrage auf Y = Z in manchen Testlaeufen entfernt, um die Schleife bis zum Ende zu durchlaufen, und der Endwert der Schleife wurde zum Teil auf verschiedene Werte gesetzt. Die eigentliche Rechenschleife blieb von diesen Aenderungen jedoch unberuehrt. Weiterhin wurden bei manchen der Testlaeufe die Genauigkeitsattribute der Variablen X, Y und Z geaendert, um zu Aussagen einerseits ueber die Rechengenauigkeit und andererseits ueber die Geschwindigkeit mehrfach genauer Arithmetik im Vergleich mit einfach genauer Arithmetik zu kommen.

In einem weiteren Testbeispiel wurden die Werte des Sinus im ersten Quadranten im Abstand von einer Bogensekunde berechnet und die dafuer benoetigten Zeiten verglichen. Der verwendete Algorithmus lautet (in FORTRAN-Notation):

```
      RHO=ATAN(1.)/(45.*3600.)
      DO 1 I=1,90
      DO 1 J=1,60
      DO 1 K=1,60
      R=(3600*(I-1)+60*(J-1)+K-1)*RHO
    1 S=SIN(R)
      END
```

Da die Verhaeltnisse der gemessenen Rechenzeiten sehr gut den Ergebnissen des ersten Tests entsprechen, wurden die Tests nur einigen der beteiligten Maschinen durchgefuehrt, und auch dort nur in FORTRAN mit einfacher Genauigkeit. Von einer Erweiterung auf andere Sprachen, Maschinen oder Genauigkeiten sind fuer dieses Testbeispiel keine Erkenntnisse zu erwarten, die ueber die Ergebnisse des ersten Tests hinausgehen.

Zur genaueren Untersuchung der die reine Rechengeschwindigkeit beeinflussenden Faktoren wurde schliesslich auf den wichtigsten der hier untersuchten Rechner ein etwas umfangreicherer Benchmark implementer aus folgenden Einzeltests bestand:

```
Test 1 ( 40x) : Production of 1000 random numbers
Test 2 (300x) : Polynomial of order 10
Test 3 (400x) : Scalar product of two vectors
Test 4 (400x) : Sum of two vectors
```

Test 5 (100x) : Square root iterations
Test 6 (100x) : Find element of vector with maximum modulus
Test 7 (100x) : Convert FLOAT to FIXED
Test 8 (100x) : Count frequency
Test 9 (30x) : Bit test and summation
Test10 (30x) : Matrix-Addressing test

Ein Listing der Quelle des Benchmarks einschliesslich der Routinen zur Zeitmessung, jedoch ohne deren Anschluss an das Betriebssystem, befindet sich im Anhang.

Fuer die einzelnen Tests wurden die benoetigten Rechenzeiten bei vielfachem Durchlauf der Tests in Schleifen bestimmt; daraus wurde fuer jeden Test die Zeit fuer einen Schleifendurchlauf berechnet. Zusaetzlich wurde die Gesamtzeit fuer den Durchlauf des Benchmarks bestimmt, die ein durch die Anzahl der Schleifendurchlaeufe gewichtetes Mittel darstellt. Um die Ergebnisse fuer die einzelnen Tests miteinander vergleichen zu koennen, wurden die Schleifendurchlaufzeiten des jeweils schnellsten Rechners im Test gleich 1 gesetzt und die Zeiten fuer die anderen Rechner hierauf invers bezogen, um so ein Mass fuer die relativen Geschwindigkeiten zu erhalten. Die Benchmarks wurden separat fuer einfache Genauigkeit der meisten Gleitpunkt-Variablen und fuer doppelte Genauigkeit saemtlicher Gleitpunktvariablen durchgefuehrt.

Der allgemeine Aussagewert von Benchmarks fuer das Verhalten einer Maschine im tatsaechlichen Betrieb unter einem normalen Jobmix ist bekanntermassen recht eingeschraenkt, so dass Verallgemeinerungen der Ergebnisse dieses Benchmarks mit grosser Vorsicht zu betrachten sind.

Schliesslich muss noch darauf hingewiesen werden, dass hier nur die reinen Rechenzeiten betrachtet werden. Die Verweilzeiten der Jobs waren in allen Rechnern des Vergleichs bei leerer Maschine im wesentlichen mit den Rechenzeiten identisch; Verweilzeiten bei belasteter Maschine wurden nicht betrachtet.

2.2. Qualitative System-Aspekte

Waehrend die vorstehend beschriebenen Tests zwar zu Aussagen ueber Geschwindigkeit, Effizienz und Genauigkeit numerischer Berechnungen fuehren, erlauben sie kaum eine Beurteilung der Betriebssysteme hinsichtlich ihres Gebrauchswertes fuer Software-Erstellung im interaktiven Betrieb. Um hier zu sinnvollen Aussagen ueber die im ersten Abschnitt angegebenen qualitativen System-Aspekte zu kommen, wurde auf einigen der Testmaschinen ein groesseres kommerziell erhaeltliches Software-Paket, das Terminal-Control-System TCS der Tektronix-PLOT10-Software, soweit implementiert, dass es in der Lage war, seinen eigenen Selbsttest durchzufuehren.

Es handelt sich hier um die Erstellung einer Unterprogrammbibliothek aus etwas ueber 100 in FORTRAN geschriebenen Routinen, deren groesster Teil auf keine besonderen Sprachkonstruktionen Bezug nimmt, also im wesentlichen unabhaengig von der Programmumgebung ist. Die Anpassung an die Gegebenheiten eines speziellen Betriebs- bzw. Laufzeitsystems sind durch "patchen", also durch gezielte Anpassung eines System-Interfaces aus 6 Routinen vorzunehmen. Zur Durchfuehrung des PLOT10-Selbstests ist weiterhin eine Bibliothek aus einem Hauptprogramm und 9 Subroutines zu erzeugen und mit den beiden anderen Bibliotheken zu einem lauffaehigen Programm zusammenzubinden. Die einzelnen Teile dieses Systems haben folgenden Umfang:

- PLOT10-Routinen: 2698 Zeilen
- Interface-Routinen: ca. 130 Zeilen
- Selbsttest-Routinen: 746 Zeilen
- Preview-Routinen: 1428 Zeilen

Bei der Implementierung dieses Software-Pakets, die in Art und Umfang typisch fuer die Uebernahme nicht zu umfangreicher Fremdsoftware sein duerfte und die gleichzeitig Aufschluesse ueber das Systemverhalten bei Aenderungen vorhandener Software gibt, wurden die einzelnen Betriebssysteme, und zwar:

- BS3 (MV19) auf der TR440
- BS2000 (MV5.0) auf der Siemens 7.760
- VAX/VMS (MV1.6) auf der VAX11/780
- MAX IV auf der Modcomp 7870

hinsichtlich ihres Verhaltens und ihrer Bedienbarkeit beobachtet. Daraus lassen sich ohne grosse Schwierigkeiten Aussagen ueber den Gebrauchswert der einzelnen Systeme fuer den Einsatz als Time-Sharing-Maschinen ableiten, die zweckmaessig beim Einsatz der Siemens-Anlage zur Vermeidung von Ineffizienz und Engpaessen beruecksichtigt werden sollten.

3. Ergebnisse und Analyse

3.1. Quantitative System-Aspekte

3.1.1. Ueberblick

Zunaechst wurden auf den einzelnen Rechnern beim ersten Testprogramm jeweils 1000000 Iterationsschritte durchgefuehrt; variiert wurden die Implementierungssprache und die spezifizierte Rechengenauigkeit. Dabei ergaben sich folgende Rechenzeiten und erreichte Genauigkeiten:

Rechner	Sprache	Zeit[s]	Schritte	Ergebnis	Fehler
TR440	FTN-S	34.0	1000000	3.2898616	7.E-6
	FTN-D	197.9	1000000	3.28986613	2.E-6
	Pascal	44.9	1000000	3.28986	< 1.E-5
	ALG60	32.95	1000000	3.28986	< 1.E-5
	ALG60/OA	48.8	1000000	3.28986	< 1.E-5
	PL/I	56.5	1000000	3.2898616	7.E-6
	BASIC	39.1	1000000	3.2898616	7.E-6
	PS440	38.3	1000000	3.2898616	7.E-6
	Comskee	29.5	1000000	3.28986613	2.E-6
	TAS	26.9	1000000	3.2898589	1.E-5
Si 7.760	BGFOR-S	18.7	1000000	3.28786	2.E-3
	BGFOR-D	28.0	1000000	3.28986613	2.E-6
	FOR1-S	20.1	1000000	3.28786	2.E-3
	FOR1-D	30.96	1000000	3.28986613	2.E-6
	FOR1-Q	245.7	1000000	3.28986613	2.E-6
	ALGOL	47.2	1000000	3.28986	< 1.E-5
	Pascal	43.0	1000000	3.28986	< 1.E-5
	PL/I-S	39.8	1000000	3.28785	2.E-3
	PLI1-S	16.3	1000000	3.28786	2.E-3
	PLI1-Q	273.3	1000000	3.28986613	2.E-6
	BASIC	39.4	1000000	3.28786	2.E-3
Si 7.451 ohne MDW	FOR1-S	35.9	1000000	3.28786	2.E-3
	FOR1-D	73.8	1000000	3.28986613	2.E-6
Si 7.451 mit MDW	FOR1-S	26.9	1000000	3.28786	2.E-3
	FOR1-D	35.6	1000000	3.28986613	2.E-6
VAX11/780	FORTRAN-S	11.5	1000000	3.2894506	4.E-4
	FORTRAN-D	22.24	1000000	3.28986613	2.E-6
Modcomp 7870	FORTRAN-S	24.5	1000000	3.2890329	8.E-4
	FORTRAN-I	31.92	1000000	3.2898648	4.E-6
	FORTRAN-D	38.14	1000000	3.28986613	2.E-6

Rechner	Sprache	Zeit[s]	Schritte	Ergebnis	Fehler
AEG80/20	FORTRAN-S	780.0	1000000	3.28786	2.E-3
AEG80/40	FORTRAN-S	70.0	1000000	3.28786	2.E-3

Dabei sind die einzelnen Sprachen durch ihre Compiler bezeichnet; wo Genauigkeiten spezifiziert wurden, sind sie in der Tabelle mit folgenden Bedeutungen angegeben:

- S : einfache Genauigkeit (REAL*4, BINARY FLOAT(21))
- I : eineinhalbfache Genauigkeit (REAL*6)
- D : doppelte Genauigkeit (REAL*8)
- Q : vierfache Genauigkeit (REAL*16, BINARY FLOAT(109))

Die Bezeichnung MDW bei der Anlage Siemens 7.541 bedeutet "Multiplizier-Dividier-Werk"; gemeint ist damit das Vorhandensein bzw. Fehlen eines Floating-Point-Prozessors. Die Anlage VAX11/780 wurde mit Floating-Point-Accelerator (FPA) gemessen.

Auf den ersten Blick ueberraschen einige dieser Ergebnisse sowohl hinsichtlich der Rechenzeiten als auch hinsichtlich der erreichten Genauigkeit der Iteration.

27. 6.80
We

3.1.2. Rechenzeiten

Aus dem Vergleich der bei den einzelnen Sprachen fuer das erste Beispiel benoetigten Rechenzeiten lassen sich folgende Schluesse ziehen:

- TR440: Bei den einzelnen Programmiersprachen laesst sich fuer die Effizienz des erzeugten Codes folgende Reihenfolge angeben:

  TAS >> ALG60 > FTN > PS440 > BASIC >> Pascal >> PL/I

  Dabei ueberrascht zum einen die groessere Effizienz des ALGOL-Programms gegenueber der FORTRAN-Version, die durch die starke Optimierung des vom ALGOL-Compiler erzeugten Codes erreicht wird - bei Abschalten dieser Optimierung ergibt sich der erheblich schlechtere Wert der Zeile "ALG60/OA" - und zum anderen das relativ schlechte Abschneiden von PS440 und Pascal, was auf verbesserungswuerdige Stellen in den betreffenden Compilern hinweist. Voellig aus dem Rahmen faellt Comskee, eine in Saarbruecken entwickelte Sprache fuer linguistische Datenverarbeitung, das bei nur unwesentlich groesserer Laufzeit gegenueber TAS eine hoehere Genauigkeit erreicht; dieser Effekt ist noch nicht geklaert.
  Die bei FORTRAN-DOUBLE-PRECISION auftretende sehr hohe Rechenzeit ist darauf zurueckzufuehren, dass hier wegen des sehr eingeschraenkten Befehlsvorrats fuer doppelte Genauigkeit ein sehr ineffizienter Code erzeugt wird, insbesondere da die doppelt-genaue Division software-maessig durchgefuehrt werden muss.

- Siemens 7.760: Bei den einzelnen Programmiersprachen laesst sich fuer die Effizienz folgende Reihenfolge angeben:

  PLI1 > BGFOR > FOR1 >> BASIC > PL/I > Pascal > ALGOL

  Hier erstaunt besonders, dass sowohl der neue PLI1-Compiler als auch der alte BGFOR-Compiler schnellere Programme erzeugen als der neue FOR1-Compiler; um Aussagen ueber die Ursachen fuer das relativ schlechte Abschneiden dieses Compilers machen zu koennen, wurde der erzeugte Maschinencode betrachtet (siehe Abschnitt 3.1.3.).
  Ebenso ueberrascht die verhaeltnismaessig geringe Laufzeit-Effizienz von Pascal.
  Beim Rechnen mit doppelter Genauigkeit waechst die Rechenzeit um etwa die Haelfte an, was auf effiziente Hardware-Instruktionen fuer doppelte Genauigkeit und deren gute Ausnutzung durch die Compiler hindeutet. Beim Uebergang auf vierfache Genauigkeit waechst die Rechenzeit auf das Zwoelffache der Rechenzeit bei einfacher Genauigkeit an, da hier nicht mehr auf Hardware-Instruktionen dieser Genauigkeit zurueckgegriffen werden kann und Software-Loesungen notgedrungen langsam sind.

- Siemens 7.541: Hier wurde nur FORTRAN betrachtet, da es sich bei dieser Maschine im wesentlichen nur um eine langsamere Version der 7.760 handelt. Ein Sprachvergleich entfaellt daher. Der Floating-Point-Prozessor bewirkt fuer einfache Genauigkeit eine Beschleunigung um ein Drittel, fuer doppelte Genauigkeit um den Faktor 2.

- VAX11/780: Hier wurde nur FORTRAN betrachtet, so dass ein Sprachvergleich entfaellt. Beim Uebergang auf doppelte Genauigkeit verdoppelt sich die Rechenzeit, was auf gegenueber der Siemens-Anlage groesseren Aufwand fuer doppelt genaue Arithmetik deutet.

- Modcomp 7870: Hier wurde nur FORTRAN betrachtet, so dass ein Sprachvergleich entfaellt. Beim Uebergang auf doppelte Genauigkeit waechst die Rechenzeit auf das Eineinhalbfache, was auf sehr effiziente doppelt-genaue Arithmetik hindeutet. Als Besonderheit ist hier die Moeglichkeit der Verwendung anderthalbfacher Genauigkeit zu nennen, die auch hinsichtlich der Rechenzeiten zwischen einfacher und doppelter Genauigkeit liegt.

- AEG 80/20 und AEG 80/40: Auch hier wurde nur FORTRAN betrachtet, so dass ein Sprachvergleich entfaellt.

Vergleicht man die Rechenzeiten der einzelnen Anlagen fuer FORTRAN bei einfacher und doppelter Genauigkeit, so kommt man zu folgenden Ergebnissen:

Rechner	Sprache	Zeit[s]	Schritte	Ergebnis	Fehler
TR440	FTN-S	34.0	1000000	3.2898616	7.E-6
	FTN-D	197.9	1000000	3.28986613	2.E-6
Si 7.760	BGFOR-S	18.7	1000000	3.28786	2.E-3
	BGFOR-D	28.0	1000000	3.28986613	2.E-6
	FOR1-S	20.1	1000000	3.28786	2.E-3
	FOR1-D	30.96	1000000	3.28986613	2.E-6
Si 7.451 ohne MDW	FOR1-S	35.9	1000000	3.28786	2.E-3
	FOR1-D	73.8	1000000	3.28986613	2.E-6
Si 7.451 mit MDW	FOR1-S	26.9	1000000	3.28786	2.E-3
	FOR1-D	35.6	1000000	3.28986613	2.E-6
VAX11/780	FORTRAN-S	11.5	1000000	3.2894506	4.E-4
	FORTRAN-D	22.24	1000000	3.28986613	2.E-6
Modcomp 7870	FORTRAN-S	24.5	1000000	3.2890329	8.E-4
	FORTRAN-D	38.14	1000000	3.28986613	2.E-6
AEG80/20	FORTRAN-S	780.0	1000000	3.28786	2.E-3
AEG80/40	FORTRAN-S	70.0	1000000	3.28786	2.E-3

Daraus ergibt sich, dass die in diesem Test mit Abstand schnellste Maschine die VAX11/780 ist. Setzt man fuer deren Geschwindigkeit den relativen Wert 1 ein, so ergeben sich die folgenden Relativgeschwindigkeiten der einzelnen Maschinen fuer einfache Genauigkeit (bei Siemens bzgl. des FOR1-Compilers; fuer die Anlage 7.541 jeweils ohne und mit Floating-Point-Prozessor):

27. 6.80
We

TR440	Si 7.760	Si 7.541		VAX11/780	Modcomp	AEG80/20	AEG80/40
0.34	0.57	0.32	0.43	1.00	0.47	0.01	0.16

und fuer doppelte Genauigkeit:

TR440	Si 7.760	Si 7.541		VAX11/780	Modcomp
0.11	0.72	0.30	0.62	1.00	0.58

Diese Ergebnisse erstaunen zunaechst etwas, da die schnellste Maschine in diesem Test eben nicht eine der Grossrechenanlagen, sondern ein Prozessrechner mittlerer Groesse ist, der von den Anschaffungskosten um eine Groessenordnung niedriger liegt als die langsameren Grossrechner. Bei doppelter Genauigkeit liegt die Relativgeschwindigkeit der Siemens-Anlagen hoeher, was auf die oben erwaehnte effiziente doppeltgenaue Arithmetik zurueckzufuehren ist, waehrend umgekehrt die TR440 hier sehr schlecht abschneidet, da sie nur uber eingeschraenkte Doppelwort-Arithmetik verfuegt. Die beiden AEG-Maschinen liegen in ihrer Relativgeschwindigkeit ausserordentlich niedrig, was auf schwerwiegende Maengel im Betriebssystem bzw. in der FORTRAN-Implementierung dieser Maschinen hindeutet.

Das zweite Testbeispiel erbrachte fuer einfache Genauigkeit die folgenden Resultate:

Rechner	Sprache	Zeit[s]
TR440	FTN-S	41.14
Si 7.760	BGFOR-S	21.4
	FOR1-S	26.0
VAX11/780	FORTRAN-S	15.22

Daraus ergeben sich die folgenden Relativgeschwindigkeiten fuer die einzelnen Rechner:

TR440	Si 7.760	VAX11/780
0.37	0.59	1.00

Die Abweichungen zum ersten Beispiel sind nicht signifikant, so dass eine weitere Verfolgung dieses zweiten Beispiels unterblieb.

27. 6.80
We

3.1.3. Rechengeschwindigkeiten

Die ueber die Rechenzeiten ermittelten Relativgeschwindigkeiten der einzelnen Rechner stimmen absolut nicht mit dem ueberein, was man zunaechst erwarten wuerde. Um eine Erklaerung fuer diese Beobachtung zu finden, wurde der fuer einfache Genauigkeit erzeugte Maschinencode fuer den Programmteil des ersten Testbeispiels untersucht, der der 1000000-mal zu durchlaufenden Schleife (ohne Abfrage auf das Abbruchkriterium) entspricht, und zwar sowohl mit als auch ohne die Instruktionen zur Anfangs- und Endebehandlung der Schleife. (Dieser Code ist fuer die TR440, die Siemens-Anlagen und die VAX11 zum Vergleich in einer Tabelle auf der naechsten Seite aufgelistet.) Die Maschineninstruktionen wurden ausserdem noch daraufhin betrachtet, ob sie auf Operanden im Speicher zugreifen oder nur auf Maschinenregister. Fuer die Anzahl der aus den 5 FORTRAN-Statements erzeugten Maschineninstruktionen ergaben sich folgende Werte:

Rechner	Befehle		Speicherzugriffe	
Schleifeninitialisierung	mit	ohne	mit	ohne
TR 440	18	16	13	11
Siemens 7.760/7.541	22	17	16	12
VAX 11/780	10	6	4	1
Modcomp 7870	17	15	8	8

Aus den Rechenzeiten einerseits und der Anzahl der Maschineninstruktionen der Schleife (ohne Anfangs- und Endebehandlung) ergeben sich folgende mittlere Ausfuehrungszeiten fuer eine Maschineninstruktion in diesem Beispiel (in Mikrosekunden, ohne besondere Beruecksichtigung der Speicherzugriffe):

TR440	Si 7.760	Si 7.541	VAX11/780	Modcomp
2.13	1.18	1.58	1.92	1.63

Dieses Ergebnis stimmt schon recht gut mit den relativen Hardware-Geschwindigkeiten dieser Rechner ueberein, obwohl es sich hier nur um eine sehr grobe Abschaetzung handelt. Genauere Abschaetzungen unter Beruecksichtigung aller Effekte hoeherer Ordnung, wie Speicherzugriff, virtuelle Adressierung, Zugriff ueber Cache, Zugriffskollision bei Parallelzugriff des zweiten TR440-Prozessors usw., duerften zwar diese Werte noch geringfuegig aendern, kaum aber zu grundverschiedenen Ergebnissen fuehren.

TR 440	Si 7.760/7.541	VAX11/780
**********	DO 1 I=1,1000000 **********	**********
B (1),	LA 15,1(0,0) 1	MOVL #1,R9;
C I,	ST 15,268(0,13) I	
	L 10,244(0,12) 1E6	
	L 1,268(0,13) I	
**********	Z=Y **********	**********
L = B Y,	LE 2,264(0,13) %L3:Y	L$IAJD: MOVL R10,Z(R11);
C Z,	STE 2,272(0,13 Z	
**********	X=FLOAT(I) **********	**********
BQB I,	LD 4,232(0,12)	CVTLF R9,R12;
SH Z 2,	STE 4,104(0,13)	
NRM FG,	ST 1,108(0,13) I	
AA 12,	XI 108(13),128	
C X,	SD 4,104(0,13)	
**********	Y=Y+2./(X*X) **********	**********
R GML A,	LER 6,4 X	MULF3 R12,R12,R0;
GDVI (2.),	STE 4,276(0,13) X	DIVF3 R0,#^X10,R0;
GA Y,	MER 6,4 X	ADDF2 R0,R10;
C Y,	LE 4,248(0,12) 2.	
	DER 4,6 %T1	
	AER 2,4 Y,%T2	
	STE 2,264(0,13) Y	
**********	1 CONTINUE **********	**********
B I,	AR 1,15 I,1	AOBLEQ #1E6,R9,L$IAJD;
AA 1,	LA 15,1(0,0) 1	
C I,	BCT 10,68(0,11) %L3	
SB (1E6),		
SKG0 L,		
	ST 1,268(0,13) I	MOVL R9,I(R11);
		MOVL R10,Y(R11);
		MOVL R12,X(R11);

Anmerkung: Die dargestellten Codes entsprechen dem Assembly-Listing der jeweiligen FORTRAN-Compiler unter Verwendung der Code-Optimierung. Zur Erhoehung der Lesbarkeit wurde an einzelnen Stellen der Code der Compiler durch aequivalente Darstellungen ersetzt, z.B. bedeutet "1E6" generell das Literal 1000000. (Diese Ueberarbeitung war insbesondere fuer die TR440 erforderlich, deren Assembly-Listing fuer den Uneingeweihten voellig unverstaendlich ist.) Die in der Schleife durchlaufenen Instruktionen befinden sich zwischen der FORTRAN-Zeile "Z=Y" und der gestrichelten Line; die restlichen Instruktionen werden vor bzw. nach der Schleife nur einmal ausgefuehrt.

27. 6.80
We

Die Inkonsistenz zwischen Rechengeschwindigkeit und Rechenzeit beruht somit auf der sehr unterschiedlichen Anzahl der pro Schleifendurchlauf auszufuehrenden Maschineninstruktionen; sie wird also wesentlich bestimmt durch die Komplexitaet der einzelnen Maschineninstruktionen und deren Ausnutzung durch den Compiler.

Im Einzelnen sieht man, dass von der Anzahl der Maschineninstruktionen her die Siemens-Anlage bei diesem Beispiel geringfuegig schlechter abschneidet als die TR440; die Ursache hierfuer ist der etwas groessere Aufwand fuer die Konvertierung von Integer nach Real und fuer die Real-Arithmetik. Unangenehmer ist schon, dass auch die Anzahl der Speicherzugriffe hoeher ist als bei der TR440, obwohl von der Maschinenarchitektur her das Gegenteil der Fall sein muesste: Die Siemens-Anlage muesste in der Lage sein, bei Schleifen dieses Typs fast oder sogar ganz ohne Speicherzugriffe auszukommen, da sie als Mehr-Register-Maschine ueber ausreichend viele Mehrzweck-Register verfuegt, im Gegensatz zur TR440 mit ihren 4 Rechenwerks-Registern. Man sieht hier deutlich, dass der Cache-Speicher der Siemens-Maschine ein sehr wichtiger Faktor fuer deren Effizienz ist, da nur durch ihn die Zugriffe auf den langsameren Hauptspeicher reduziert werden.

Umgekehrt werden durch die Verwendung von sehr komplexen Maschineninstruktionen, die zum Teil direkt den FORTRAN-Statements entsprechen, und durch gute Ausnutzung der Mehrzweck-Register die erheblich besseren Werte der VAX11 erreicht; die Cache-Speicher dieser Maschine fuehren erst in zweiter Linie zu einer Beschleunigung des Gesamtablaufes.

Dieser Vergleich zeigt recht deutlich, dass die hohe Geschwindigkeit der Siemens-Hardware insgesamt weniger zu kurzen Rechenzeiten fuehrt als die Verlagerung von Operationsteilen von der Assembler-Ebene in die Mikroprogramm-Ebene, wie dies fuer die VAX-Architektur gilt. Speziell laesst sich durch Verlagerung der Operationen, die fuer die eigentliche Problem-Bearbeitung irrelevant sind, auf die Mikroprogramm-Ebene wesentlich mehr an Effizienz gewinnen als durch rein quantitative Massnahmen zur Steigerung der Geschwindigkeit einer vorgegebenen Architektur, die bei der Siemens-Anlage durch ihren Bezug auf die schon 15 Jahre alte Architektur des IBM-Systems /360 recht altertuemlich ist. Operationen, die sich zum Beispiel gut auf die Mikroprogramm-Ebene verlagern lassen, sind Erzeugen und Laden von Operanden-Adressen und Bereitstellen von Operanden fuer nachfolgende Operationen - was letztlich zu Maschineninstruktionen auf relativ hoher Ebene und zu Operationen mit vielen und variablen Adressteilen fuehrt. Man sieht hier, dass diese Massnahme zu erheblicher Effizienzsteigerung bei relativ geringem Aufwand fuehrt, wenn man erst einmal einen mikroprogrammierbaren Rechner hat. Es ist unverstaendlich, wieso Siemens diese Vorteile der Mikroprogrammierung nicht ausnutzt, obwohl dies ohne weiteres bei relativ niedrigen Kosten moeglich waere. Diese Beobachtung liefert wieder einmal ein Argument dafuer, dass durch effizientere Strukturen mehr gewonnen werden kann als durch nur quantitative Verbesserungen historischer Rechner-Architekturen, zumal diese Kosten im wesentlichen nur ein einziges Mal - naemlich beim Entwurf - entstehen, waehrend die Kosten fuer schnellere Hardware jedes Exemplar eines Rechners verteuern.

3.1.4. Einfluss der Optimierung

Um den Einfluss der Optimierung des von den FORTRAN-Compilern erzeugten Objektcodes abschaetzen zu koennen, wurden fuer das erste Testbeispiel und fuer das Benchmark-Programm die Laufzeiten mit und ohne Optimierung miteinander verglichen. Dieser Vergleich wurde fuer die TR440, die Siemens 7.760 und die VAX11/780 durchgefuehrt. Der FORTRAN-Compiler der Siemens-Anlage bietet einen zusaetzlichen, vom Hersteller als "eventuell riskant" bezeichneten Optimierungsmodus, der auch bedingt ausfuehrbare Schleifenteile beeinflusst; dieser Modus wurde ebenfalls in den Vergleich einbezogen; negative Auswirkungen, die nach Angabe des Herstellers bei manchen Programmen auftreten koennen, wurden nicht festgestellt.

Die FORTRAN-Compiler der einzelnen Rechner fuehren dabei folgende Optimierungsverfahren durch:

- TR440:
  - Berechnung konstanter Index-Ausdruecke bei der Uebersetzung
  - Optimierung von logischen Ausdruecken
  - Erkennung gemeinsamer Teilausdruecke
  - Optimierung der Indexrechnung
  - Schleifenoptimierung
  - Verwendung von Index-Fortschaltung
  - Begrenzung der Anzahl verwendeter Index-Register
  - Globale Registerzuordnung
  - Vermeidung von Spruengen als Sprungziele
  - Auswahl geeigneter Maschinenbefehle

- Siemens 7.760:
  - Berechnung konstanter Ausdruecke bei der Uebersetzung
  - Optimierung von logischen Ausdruecken
  - Erkennung gemeinsamer Teilausdruecke
  - Optimierung der FORMAT-Abarbeitung
  - Optimierung der Indexrechnung
  - Schleifenoptimierung
  - Globale Registerzuordnung

- VAX11/780:
  - Berechnung konstanter (Teil-)Ausdruecke bei der Uebersetzung
  - Zusammenlegung gleicher Konstanten und Parameter-Listen
  - Optimierung von logischen Ausdruecken und Spruengen
  - Erkennung gemeinsamer Teilausdruecke
  - Optimierung der Indexrechnung
  - Schleifenoptimierung
  - Entfernung nicht erreichbaren Codes
  - Globale Registerzuordnung
  - Auswahl geeigneter Maschinenbefehle

Fuer das erste Beispiel ergaben sich die folgenden Rechenzeiten ohne Optimierung:

TR440	Si 7.760	VAX11/780
35.79	20.277	14.71

Mit Optimierung ergeben sich die folgenden Werte (fuer die Siemens-Anlage sind beide Modi aufgefuehrt):

TR440	Si 7.760		VAX11/780
35.57	20.600	20.652	11.70

Dies entspricht Verkuerzungen der Rechenzeit um folgende Bruchteile der urspruenglichen Rechenzeit:

TR440	Si 7.760		VAX11/780
.01	-.02	-.02	.20

Analyse der Assemblies ergab, dass bei der TR440 kein Unterschied im erzeugten Code beider Versionen bestand, ebenso bei der Siemens-Anlage fuer die beiden optimierten Versionen, so dass die geringen Unterschiede in den Laufzeiten als Messungenauigkeiten zu werten sind. Gegenueber der nicht-optimierten Version erforderten die optimierten Programme auf der Siemens-Anlage 2 zusaetzliche Befehle und 2 zusaetzliche Speicherzugriffe in der Anfangsbehandlung der Schleife, doch wurden die Speicherzugriffe in der Schleife um 3 reduziert - was vermutlich deshalb nicht zu einer Beschleunigung des Programms fuehrte, weil saemtliche Werte im Cache standen. Bei der VAX11 brachte die Optimierung ebenfalls 3 zusaetzliche Befehle und 2 zusaetzliche Speicherzugriffe in der Endebehandlung der Schleife, doch wurde dadurch die Anzahl der Speicherzugriffe innerhalb der Schleife um 7 (von 8 auf 1) drastisch verringert, was die Beschleunigung des Programms erklaeren duerfte.

Fuer den Benchmark ergaben sich die Werte der Tabellen der naechsten Seiten. Die geringen Rechenzeit-Unterschiede zu den Tabellen aus Abschnitt 3.1.6. folgen aus den unterschiedlichen Rechnerbelastungen zu den einzelnen Testzeiten; sie sind als Messungeauigkeiten zu werten.

Rohzeiten [s]:

Ergebnisse ohne Optimierung:

Test	TR440	Si 7.760	VAX11/780
1	11.776	2.299	3.780
2	7.330	3.342	2.870
3	11.201	3.961	3.380
4	11.456	3.649	2.950
5	29.872	2.699	2.500
6	2.385	0.798	0.900
7	4.085	1.818	1.110
8	3.004	0.955	0.950
9	1.126	0.536	1.000
10	9.442	3.064	3.670
Mittel	91.676	23.121	23.110

Ergebnisse mit Optimierung:

Test	TR440	Si 7.760		VAX11/780
1	10.468	2.024	2.004	3.470
2	3.212	2.314	2.292	1.980
3	2.986	2.385	2.387	2.400
4	2.837	1.632	1.623	2.170
5	27.112	2.170	2.158	2.100
6	1.319	0.600	0.597	0.730
7	2.245	1.424	1.430	0.920
8	2.159	0.600	0.622	0.640
9	0.773	0.462	0.412	0.870
10	7.584	1.579	1.579	2.680
Mittel	60.694	15.189	15.104	17.960

Rechenzeitverkuerzung durch die Optimierung:

Test	TR440	Si 7.760		VAX11/780
1	.11	.12	.13	.08
2	.56	.31	.31	.31
3	.73	.40	.40	.29
4	.75	.55	.56	.26
5	.09	.20	.20	.16
6	.45	.25	.25	.19
7	.45	.22	.21	.17
8	.28	.37	.35	.33
9	.31	.14	.23	.13
10	.20	.48	.48	.27
Mittel	.34	.34	.35	.22

Man stellt dabei fest, dass Test 1 generell, Test 5 bei der TR440 und Test 9 bei der Siemens-Anlage in Optimierungs-Stufe 1 nur gering durch die Optimierung beschleunigt werden; es handelt sich hier um Tests, die vornehmlich Arithmetik durchfuehren. Gut werden dagegen die Tests 3 und 4 auf der TR440 und der Siemens-Anlage und Test 10 auf der Siemens-Anlage beschleunigt. Bei diesen Tests handelt es sich um lineare Abarbeitung grosser Felder, die bei der TR440 durch Fortschaltbefehle fuer die Zugriffs-Operationen und bei der Siemens-Anlage durch Vorziehen und Vereinfachen der Index-Berechnung optimiert werden, wie sich aus den Assembly-Listings ergibt. Test 9 ist der einzige Test, bei dem die Optimierungs-Stufe 2 der Siemens-Anlage eine nennenswerte Verkuerzung der Rechenzeit gegenueber Stufe 1 bringt, und zwar durch Optimierung der Index-Berechnung durch Vorziehen eines bedingten Schleifenteils. Bei der VAX11 profitieren dagegen die Tests 2 und 8 am meisten von der Optimierung, und zwar wieder durch Abspeichern aller fuer die Schleifen relevanten Variablen in Register. Da hier diese Optimierung generell einen starken Einfluss auf die Rechengeschwindigkeit hat, waehrend andererseits durch die hohe Korrespondenz (fast 1:1) zwischen FORTRAN-Statements und Maschinen-Befehlen auch nicht eigens optimierter Code schon sehr effizient sein kann, zeigt diese Maschine bezueglich der Optimierung ein etwas anderes Verhalten als die beiden anderen Rechner und generell geringere Beschleunigung der Programme durch die Optimierung.

3.1.5. Rechengenauigkeit

Um festzustellen, bei welchen Werten die Iteration fuer die einzelner Real-Darstellungen abgebrochen wird, d.h. die weiteren Summanden 0 werden, wurde die Iteration in den folgenden Beispielen bei Erfuellung dieses Kriteriums abgebrochen. Hier ergaben sich die folgenden Werte fuer Rechenzeit und erreichte Genauigkeit:

Rechner	Sprache	Zeit[s]	Schritte	Ergebnis	Fehler
TR440	FTN-S	10.7	262145	3.2898616	7.E-6
	BASIC	11.7	262145	3.2898616	7.E-6
	PS440	11.6	262145	3.2898616	7.E-6
	TAS	7.1	262145	3.2898589	1.E-5
Si 7.760	BGFOR-S	-	1449	3.28786	2.E-3
	BGFOR-D	-	1449	3.28849	1.4E-3
	BGFOR-D	480.21	16113921	3.28986800	1.E-7
	FOR1-S	-	1449	3.287858	2.E-3
	FOR1-D	-	1449	3.288488	1.4E-3
	FOR1-D	3019.66	94906272	3.289868103	3.E-8
	PLI1-Q	28381.8	100000000	3.28986811	2.E-8
	BASIC	-	1449	3.28786	2.E-3
Si 7.541	FOR1-S	-	1449	3.28786	2.E-3
VAX11/780	FORTRAN-S	0.09	4097	3.2894506	4.E-4
	FORTRAN-D	227.37	10000000	3.289867933	1.E-6
Modcomp	FORTRAN-S	0.09	2048	3.2890329	8.E-4
SMP80	F80-S	45.0	4097	3.2894506	4.E-4
	F80-D	-	60000	3.289835	3.E-5
AEG80/40	FORTRAN-S	-	1449	3.28786	2.E-3
Tek 4051	BASIC	2100.0	59618	3.289835	3.E-5
	BASIC	8220.0	207974	3.289851	1.7E-5

In diese Tabelle wurden auch der Vollstaendigkeit halber einige Werte aufgenommen, bei denen die Maximalzahl von Iterationsschritten nicht erreicht wurde (Siemens: PLI1-Q, VAX: FORTRAN-D, SMP80: F80-D und die Werte fuer das Tek 4051-Geraet). Es zeigt sich, dass aufgrund der kuerzeren Wortlaenge bei 32-bit-Arithmetik die Iteration schon relativ frueh abgebrochen wird, wobei hier das Format der Real-Darstellung einen unvermutet grossen Einfluss hat. Bei doppelter Genauigkeit stellt dagegen die Anzahl der Iterationschritte den bestimmenden Faktor fuer die erreichte Genauigkeit dar.

Hinsichtlich der bei den einzelnen Iterationen erreichten Rechengenauigkeiten zeigt sich, dass die TR440 aufgrund ihrer Wortlaenge von

48 bit erwartungsgemaess bei einfacher Genauigkeit erheblich naeher an dass richtige Ergebnis herankommt (Fehler: 7.E-6) als die 32-bit-Maschinen, die auf Fehler von 2.E-3 (!) bei Verwendung der Real-Darstellung zur Basis 16 bzw. von 4.E-4 bei Darstellung zur Basis 2 kommen. Der beim IBM-Real-Format sehr grosse Fehler von 2.E-3 (auf den Siemens- und den AEG-Anlagen) deutet zwar primaer nur auf die Empfindlichkeit des verwendeten Algorithmus gegenueber Rundungsfehlern hin; generell muss man jedoch aus diesem Ergebnis die Konsequenz ziehen, dass ein Teil der Programme, die auf der TR440 noch mit einfacher Genauigkeit auskommen, auf den Siemens-Anlagen mit doppelter Genauigkeit laufen muessen, was fuer diese Programme den Geschwindigkeitsvorsprung dieser Anlagen wieder weitgehend zunichte macht. Real-Darstellung zur Basis 2, wie sie bei der VAX11 und bei dem SMP80-Mikroprozessorsystem verwendet wird, fuehrt zur fuenffachen Genauigkeit, doch ist der darstellbare Zahlenbereich etwas kleiner. Die Modcomp-Anlage dagegen verwendet ein Real-Format, das hinsichtlich Genauigkeit und Zahlenbereich zwischen den beiden anderen 32-bit-Formaten liegt.

Bei der Uebernahme von Programmen auf die Siemens-Anlage, die mit einfacher Genauigkeit Iterationen durchfuehren, ist somit besondere Vorsicht anzuraten und gegebenenfalls auf doppelte Genauigkeit auszuweichen.

Einigermassen ueberraschend ist das unterschiedliche Ergebnis fuer das Erreichen des Abbruch-Kriteriums fuer den BGFOR-Compiler und den FOR1-Compiler bei doppelter Genauigkeit auf der Siemens-Anlage; hier haette man Uebereinstimmung erwartet, da beide Compiler dieselbe Arithmetik benutzen sollten. Diese Compiler-abhaengige Genauigkeit koennte fuer numerische Verfahren, bei denen die erreichte Genauigkeit abzuschaetzen ist, moeglicherweise Probleme aufwerfen.

3.1.6. Ergebnisse des Benchmarks

Die einzelnen Tests des Benchmarks ueberpruefen die fuer folgende Operationen erforderlichen Rechenzeiten:

Test 1: Production of 1000 random numbers:
Double Precision-Addition, -Subtraktion, -Multiplikation, Bildung des Divisionsrestes

Test 2: Polynomial of order 10:
Hornerschema (Gleitpunkt-Multiplikation und -Addition)

Test 3: Scalar product of two vectors:
Gleitpunkt-Multiplikation und -Addition

Test 4: Sum of two vectors:
Gleitpunkt-Addition

Test 5: Square root iterations:
Gleitpunkt-Addition und -Division

Test 6: Find element of vector with maximum modulus:
Bildung des Absolutwertes, Gleitpunkt-Vergleich

Test 7: Convert FLOAT to FIXED:
Gleitpunkt-Multiplikation und -Addition, Konversion in Festpunktzahl

Test 8: Count frequency:
Festpunkt-Vergleich und -Addition

Test 9: Bit test and summation:
Festpunkt-Subtraktion, -Multiplikation, -Division, -Vergleich

Test10: Matrix-Addressing test:
Gleitpunkt-Addition, Adressierung grosser Felder

Saemtliche Tests arbeiten auf Feldern von 4k Byte Laenge und mehr, so dass speziell bei Rechnern mit Cache der Zugriff auf den Hauptspeicher erzwungen wird, da in jedem Test wenigstens einmal jedes Feldelement geladen wird.

Die Ergebnisse des Benchmarks sind in den folgenden Tabellen zusammengestellt.

Rohzeiten [s]:

Ergebnisse fuer einfache Genauigkeit:

Test	TR440	Si 7.760	Si 7.541	VAX11/780	Modcomp
1	11.135	1.994	2.289	3.470	3.528
2	3.446	2.280	3.511	1.930	6.707
3	3.041	2.366	3.335	2.320	8.706
4	3.079	1.617	2.475	2.110	6.985
5	29.072	2.151	2.665	2.090	3.733
6	1.504	0.592	0.899	0.710	1.676
7	2.469	1.412	1.900	0.900	1.908
8	2.423	0.591	0.972	0.670	1.987
9	0.894	0.461	0.677	0.830	1.040
10	8.662	1.568	2.744	2.660	12.555
Mittel	65.725	15.031	21.467	17.690	48.822

Ergebnisse fuer doppelte Genauigkeit:

Test	TR440	Si 7.760	Si 7.541	VAX11/780	Modcomp
1	11.710	2.016	2.308	3.500	3.430
2	9.510	3.242	4.428	3.300	8.265
3	14.232	3.565	4.700	3.670	10.320
4	7.931	2.334	3.300	2.920	8.120
5	27.633	1.953	2.327	1.720	3.730
6	2.967	0.679	0.963	0.930	2.025
7	5.264	1.475	1.905	1.510	2.480
8	2.479	0.592	0.972	0.610	2.070
9	1.083	0.517	0.746	0.900	1.136
10	12.415	2.019	3.187	6.750	13.420
Mittel	95.223	18.391	24.865	25.930	54.996

Zeiten fuer einen Schleifendurchlauf [ms]:

Ergebnisse fuer einfache Genauigkeit:

Test	TR440	Si 7.760	Si 7.541	VAX11/780	Modcomp
1	278.38	49.85	57.23	86.75	88.20
2	11.49	7.60	11.70	6.43	22.36
3	7.60	5.92	8.34	5.80	21.76
4	7.70	4.04	6.19	5.28	17.46
5	290.72	21.51	26.65	20.90	37.33
6	15.04	5.92	8.99	7.10	16.76
7	24.69	14.12	19.00	9.00	19.08
8	24.24	5.91	9.72	6.70	19.87
9	29.81	15.37	22.57	27.67	34.67
10	288.73	52.27	91.47	88.67	418.50
Mittel	41.08	9.39	13.42	11.06	30.51

Ergebnisse fuer doppelte Genauigkeit:

Test	TR440	Si 7.760	Si 7.541	VAX11/780	Modcomp
1	292.74	50.40	57.70	87.50	85.50
2	31.70	10.81	14.76	8.25	27.55
3	35.58	8.91	11.75	9.18	25.90
4	19.83	5.84	8.25	7.30	20.20
5	276.33	19.53	23.27	17.20	37.30
6	29.67	6.79	9.63	9.30	19.30
7	52.64	14.75	19.05	15.10	24.70
8	24.79	5.92	9.72	6.10	19.85
9	36.09	17.23	24.87	30.00	36.85
10	413.84	67.30	106.23	225.00	455.00
Mittel	59.51	11.49	15.54	16.21	34.37

Relative Rechengeschwindigkeiten, bezogen auf die
jeweils schnellste Maschine:

Ergebnisse fuer einfache Genauigkeit:

Test	TR440	Si 7.760	Si 7.541	VAX11/780	Modcomp
1	.18	1.00	.87	.57	.57
2	.56	.85	.55	1.00	.29
3	.76	.98	.70	1.00	.27
4	.52	1.00	.65	.77	.23
5	.07	.97	.78	1.00	.56
6	.39	1.00	.66	.83	.35
7	.36	.64	.47	1.00	.47
8	.24	1.00	.61	.88	.30
9	.52	1.00	.68	.56	.44
10	.18	1.00	.57	.59	.12
Mittel	.23	1.00	.70	.85	.31

Ergebnisse fuer doppelte Genauigkeit:

Test	TR440	Si 7.760	Si 7.541	VAX11/780	Modcomp
1	.17	1.00	.87	.58	.59
2	.26	.76	.56	1.00	.30
3	.25	1.00	.76	.97	.35
4	.29	1.00	.71	.80	.29
5	.06	.88	.74	1.00	.46
6	.23	1.00	.71	.73	.34
7	.28	1.00	.77	.98	.59
8	.24	1.00	.61	.97	.29
9	.48	1.00	.69	.57	.46
10	.16	1.00	.63	.30	.15
Mittel	.19	1.00	.74	.71	.33

Es zeigt sich, dass bei einfacher Genauigkeit die Siemens 7.760 und die VAX11 bei den meisten Tests vergleichbar schnell sind, waehrend die Geschwindigkeit der Siemens 7.541 bei etwa zwei Drittel, die der beiden anderen Anlagen bei etwa einem Drittel dieses Wertes liegen. Bei doppelter Genauigkeit liegen, wie aus den anderen Beispielen zu erwarten, die Siemens-Anlagen relativ besser gegenueber den anderen Maschinen, die untereinander etwa dasselbe Verhaeltnis zeigen wie bei einfacher Genauigkeit.

Aus den einzelnen Testwerten lassen sich folgende Schluesse ziehen:

- Die relative Geschwindigkeit der einzelnen Rechner schwankt in nicht kohaerenter Weise zwischen den einzelnen Tests, zum Teil in sehr weiten Grenzen. Es ist daher nicht sonderlich sinnvoll, aus dem Benchmark absolute Rechnergeschwindigkeiten entnehmen zu wollen. Je nach dem Spektrum der an die Rechner gestellten Anforderungen kann die relative Leistung um den Faktor 8 schwanken.

- Die Leistungseinbrueche beim TR440 lassen sich durch vergleichsweise langsame Gleitpunkt-Division sowie durch den schon genannten eingeschraenkten Vorrat an Befehlen fuer doppelte Genauigkeit erklaeren.

- Bei einfacher Genauigkeit schneiden die Siemens-Anlagen bei der Konversion von Gleitpunkt-Darstellung in Festpunkt-Darstellung relativ schlecht ab; dies ist auf die software-maessig vorgenommene Konvertierung zurueckzufuehren.

- Wie schon im ersten Beispiel festgestellt, ist die doppelt-genaue Arithmetik der VAX11 vergleichsweise langsam. Test10 des Benchmarks zeigt weiterhin einen Leistungsabfall beim Zugriff auf grosse Felder; die Ursache hierfuer ist, dass diese Felder nicht mehr in den Cache-Speicher hineinpassen und dann der langsamen Hauptspeicher zu einem Engpass wird. Eben dieser Effekt ist auch dafuer verantwortlich, dass die relative Leistung der VAX11 bei diesem Beispiel um etwa ein Drittel niedriger liegt als beim ersten Beispiel.

- Die Modcomp-Anlage bietet insgesamt das homogenste Bild; auch bei Uebergang auf doppelte Genauigkeit waechst die Rechenzeit nicht allzustark an, was auf effizient implementierte doppelt-genaue Arithmetik hindeutet.

Betrachtet man nicht nur die reinen Rechengeschwindigkeiten, sondern das Preis-Leistungs-Verhaeltnis der einzelnen Rechner, so sieht das Bild allerdings ganz anders aus: Die drei Minicomputer haben ein aehnliches Preis-Leistungs-Verhaeltnis, das erheblich guenstiger ist als dasjenige der Grossrechner - selbst wenn man die staerkere Ausstattung der Peripherie und die generellen Ausbaumoeglichkeiten des Grossrechners beruecksichtigt. Dies bedeutet in letzter Konsequenz, dass ein Einsatz von Grossrechnern heute aus Wirtschaftlichkeitsgruenden nur noch fuer solche Anwendungen zu vertreten ist, wo ein Einsatz eines oder mehrerer Kleinrechner technisch nicht zweck-

maessig ist - etwa bei der Bedienung extrem grosser On-Line-Datenbestaende. Erwaegungen dieser Art legen insbesondere fuer den Einsatz an Universitaeten die Verwendung von Rechnern der "Super-Mini"-Klasse aus Wirtschaftlichkeitsgruenden nahe, doch ist dies selbstverstaendlich nur dann sinnvoll, wenn diese Rechner dem Benutzer wenigstens den von Grossrechnern gewohnten Komfort bieten. Abschnitt 3.2. dieser Studie vergleicht daher die einzelnen Rechner im Hinblick auf dieses Kriterium.

3.1.7. Implementierung der PLOT10-Software

Die Routinen der PLOT10-Software liessen sich auf allen Maschinen ohne nennenswerte Anpassungsschwierigkeiten uebersetzen, was in diesem Fall allerdings weniger auf die Qualitaet der einzelnen FORTRAN-Compiler als vielmehr auf den geringen ausgenuetzten Sprachumfang der PLOT10-Routinen selbst zurueckzufuehren ist. Die Uebersetzungszeiten fuer die einzelnen Programm-Pakete des Tests lagen auf allen Maschinen in der gleichen Groessenordnung, bei der TR440 und der Modcomp allerdings ueber dem Durchschnitt aller Maschinen, waehrend sie bei der VAX11 am niedrigsten waren, doch sind die Unterschiede nicht als gravierend zu betrachten.

Uebersetzungszeiten (Echtzeit bei leerer Maschine [m:s]):

Routinen	TR440	Si 7.760	VAX11/780	Modcomp
Test	>< 0:33	0:24	0:38	1:11
PLOT10	4:00	2:44	2:18	4:00
Interface	>< 0:06	0:12	0:08	0:23
Preview	>< 0:43	0:41	0:38	*
Gesamt	>< 5:22	4:01	3:42	> 5:34

* : keine Daten; >< : hochgerechnete Werte

Bei der Siemens 7.760 wurde jedoch ein eigenartiges Verhalten beobachtet: Damit der FORTRAN-Compiler vertretbare Uebersetzungszeiten liefert, muessen 7 System-Jobs permanent vorhanden sein, die offenbar ein Pagen des Compilers verhindern oder zumindest stark verringern sollen. Fehlen diese 7 Jobs, so steigen die Uebersetzungszeiten um mehr als den Faktor 10 an (von 4 auf 45 Minuten fuer die Uebersetzung von rund 5000 Zeilen FORTRAN Quelle bei leerer Maschine)! Dieser Effekt deutet auf schwerwiegende Maengel in der Paging-Strategie des Betriebssystems oder in der internen Struktur des Compilers.

Groessere Unterschiede ergaben sich in dem Aufwand, der insgesamt zur Implementierung des Systems erforderlich war:

- TR440: Hier mussten Routinen zur Umcodierung der ein-/auszugebenden Daten zwischen ASCII-Code und dem Interncode ZC1 des Rechners an das PLOT10-System angeschlossen werden. Da diese Routinen gluecklicherweise schon vorhanden waren, lag sich der Gesamtaufwand fuer die Implementierung bei einigen Stunden bei vollem Rechner (bei leerem Rechner waeren etwa zwei Stunden erforderlich gewesen). Eine separate Implementierung der Umcodierungsroutinen in Assembler haette einen zusaetzlichen Arbeitsaufwand in der Groessenordnung von ein bis zwei Tagen bedeutet. Eine Implementierung dieser Routinen in FORTRAN waere nur schwer durchzufuehren und darueberhinaus ineffi-

zient und daher keinesfalls sinnvoll.

- Siemens 7.760: Hier mussten Routinen zur Umcodierung der ein-/auszugebenden Daten zwischen ASCII-Code und EBCDI-Code an das PLOT10-System angeschlossen werden. Ein Vorschlag von Tektronix zum Aufbau dieser Routinen lag vor, doch stellte sich bei dessen Implementierung heraus, dass die von Siemens angegebenen EBCDIC-Tabellen saemtlich falsch sind, so dass erst Versuche zum Herausfinden des wirklich implementierten Codes angestellt werden mussten. Durch weitere den Betrieb der Siemens-Anlage behindernden Schwierigkeiten, die allerdings nur zum Teil Siemens zur Last zu legen sind, dauerten die Implementierungsversuche zwei Wochen; ohne diese externen Schwierigkeiten waere dasselbe Ergebnis schon nach etwa einer Woche erzielt gewesen - dass naemlich PLOT10 mit dem zur Zeit dieses Tests vorhandenen FORTRAN-Laufzeit-System ueberhaupt nicht implementierbar war, da bei der Daten-Ein- und -Ausgabe zum Teil Zeichen umgeschluesselt bzw. unterdrueckt wurden, die zur Steuerung des Graphik-Systems erforderlich sind. Die Implementierung wurde daher ergebnislos abgebrochen. Inzwischen wurde vom Hersteller eine Aenderung zum Laufzeit-System ausgeliefert, die diesen Fehler nicht mehr enthaelt, so dass eine Implementierung der Umcodierungsroutinen in FORTRAN moeglich ist.

- VAX11/780: Da diese Maschine als Interncode ASCII hat, waren keine Umcodierungsroutinen erforderlich. Schwierigkeiten traten beim Arbeiten mit FORTRAN-A1-Format auf, das hier einen etwas anderen Aufbau hat als bei den anderen Maschinen, doch liess sich dieses Problem einfach loesen. Die gesamte Implementierung nahm nur etwas ueber eine Stunde in Anspruch.

- Modcomp 7870: Da diese Maschine als Interncode ASCII hat, waren keine Umcodierungsroutinen erforderlich. Es traten nur geringe Schwierigkeiten auf, die sich auf die Anordnung einiger Statements in der FORTRAN-Quelle sowie auf Leerzeilen in Quelltext bezogen, aber ohne Muehe zu loesen waren. Die gesamte Implementierung nahm knapp zwei Stunden in Anspruch.

Die Implementierung liess sich also bei allen Maschinen ausser der Siemens-Anlage ohne Muehe und ohne grossen Aufwand durchfuehren, auf diesem Rechner dagegen zum Zeitpunkt des Tests trotz Muehe und Aufwand ueberhaupt nicht.

Die bei dieser Implementierung und auch beim sonstigen Arbeiten mit den genannten Rechnern gewonnenen Erfahrungen sollen nun in den folgenden Abschnitten erlaeutert und interpretiert werden.

3.2. Qualitative System-Aspekte

Die unter anderem mit der Implementierung der PLOT10-Software gewonnenen Erfahrungen zeigen, dass erhebliche Unterschiede in der Bedienbarkeit und in den Benutzerschnittstellen der betrachteten Systeme bestehen. Diese Erfahrungen werden in den folgenden Abschnitten unter den verschiedenen eingangs genannten Aspekten betrachtet.

Da die hier beschriebenen Benutzerschnittstellen im wesentlichen Eigenschaften der Betriebssyteme der betrachteten Rechner darstellen, werden die folgenden Angaben auf diese Betriebssysteme und nicht auf die Rechner bezogen. Die Aussagen gelten im allgemeinen fuer alle Rechner, auf denen das betrachtete Betriebssystem laeuft; allerdings bedeutet dies fuer BS3 und (zur Zeit) fuer VAX/VMS keine Erweiterung des Gueltigkeitsbereiches der Aussagen, da beide Systeme nur auf je einem Rechnertyp laufen. Die Aussagen ueber BS2000 und MAX IV sind dagegen fuer die entsprechenden Rechnerfamilien der Hersteller (4004, 7.5xx, 7.7xx bzw. MODCOMP IV, CLASSIC 7860/7870) global gueltig.

3.2.1. Interaktivitaet

Bei allen vier der betrachteten Systeme besteht die Moeglichkeit, mit Hilfe einer Kommandosprache die Erstellung und den Betrieb eigener Programme von Bildschirm-Terminals aus im Dialogbetrieb vorzunehmen. Die angeschlossenen Terminals sind saemtlich Voll-Duplex-Geraete, die den eingetippten Text zum Rechner senden, auf dem Bildschirm jedoch nur Text darstellen, der vom Rechner gesendet wird. Dieser Typ von Terminal wird am besten als eine Kombination von zwei separaten Geraeten betrachtet: einer Tastatur, die Zeichen zum Rechner sendet, und einem Darstellungsmedium (Bildschirm oder Papier), auf dem vom Rechner kommende Informationen sichtbar gemacht werden. Wesentlich fuer das Verstaendnis der Arbeitsweise solcher Geraete ist die Tatsache, dass zwischen Tastatur und Darstellungsmedium keine direkte Verbindung und damit auch keine direkte Korrespondenz besteht, sondern jede Zuordnung ueber den angeschlossenen Rechner und die Terminal-Treiber erfolgt. Im Gegensatz dazu sehen sowohl TR440 als auch Siemens standardmaessig einen Bildschirm-Betrieb im Blockmodus mit lokaler Datenhaltung des Bildschirm-Inhalts im Terminal (also ohne Trennung zwischen Tastatur und Darstellungsmedium) vor, was einige Anforderungen an die zu verwendenden Terminals stellt und zu Einschraenkungen in der Bedienbarkeit einfacher Asynchron-Bildschirme fuehrt.

Die in jeder Richtung als eine Einheit uebertragenen Textmengen sind bei den vier Systemen verschieden; zusaetzlich werden die Betriebsbedingungen im Saarbruecker Rechnernetz (gueltig fuer TR440 und Siemens 7.760) separat angegeben, da diese teilweise von den vom Hersteller vorgesehenen Bedingungen abweichen:

- TR440 (unter den Randbedingungen des Saarbruecker Netzes, das zu den vom Hersteller vorgesehenen Bildschirmschnittstellen vergleichbare bzw. diesen ueberlegene Leistungen anbietet): Text wird zeichenweise eingegeben und vom Rechner reflektiert. Innerhalb einer Textzeile ist es moeglich, durch BS (Backspace) auf schon geschriebene Zeichen zurueckzupositionieren und so einzelne fehlerhaft eingegebene Textstellen zu korrigieren. Ausserdem ist ein Sprung auf den Anfang des Textes moeglich, so dass durch erneutes Vorwaertsfahren in diesem Text auch Korrekturen ueber mehrere Zeilen hinweg, wenn auch mit einigem Aufwand, moeglich sind. Nach Eingabe mehrerer Zeilen - das Maximum ergibt sich aus der Anzahl eingegebener Zeichen, die 512 nicht ueberschreiten darf - kann der gesamte Text an den Rechner abgeschickt werden; erst bei Eingabe des Abschickzeichens (Line Feed) wird der Text wirklich an den Hauptrechner uebertragen und kann anschliessend nicht mehr korrigiert werden.

  Eingegebene Texte werden vom System in spitze Klammern eingeschlossen; diese Zeichen sind syntaktisch bedeutungslos, koennen jedoch das Druckbild einer Ausgabe auf einem druckenden oder graphischen Terminal zerstoeren. Es ist nicht moeglich, die Ausgabe dieser Zeichen zu unterdruecken.

  Nach der Eingabe eines solchen Textblocks sind keine weiteren Eingaben (mit Ausnahme einer begrenzten Menge von Vermittler-Kommandos) moeglich, bis vom Hauptrechner eine Ausgabe erfolgt ist, die das Terminal wieder eingabeberechtigt macht.

  Es ist moeglich, durch die Eingabe spezieller Kommandos laufende Ausgaben zu steuern (abbrechen, wiederholen) und laufende Benutzerprogramme anzuhalten oder abzubrechen.

  Laengere Ausgaben vom Rechner erfolgen blockweise; jeder Block ist vom Benutzer durch eine leere Eingabe zu quittieren, damit der naechste Block vom Rechner gesendet wird. Der Bildschirm wird erst nach dem letzten Block wieder eingabeberechtigt.

  Ein zeichenweiser Dialog sowie ein echter Voll-Duplex-Betrieb ist mit diesem System nicht moeglich, da der Bildschirm zwangsweise das eingegebene Zeichen als Reflexion erhaelt und da die eigentliche Kommunikation mit dem Rechner im Halb-Duplex-Betrieb auf der Ebene groesserer Informationsbloecke geschieht.

  Druckausgaben von Dateien koennen als eigene Auftraege dem Output-Spooler uebergeben werden; zusaetzlich besteht ueber spezielle Kommandos die Moeglichkeit, den am Terminal gefuehrten Dialog mitprotokollieren zu lassen und dieses Protokoll am Jobende ausgedruckt zu erhalten. Das Protokoll enthaelt alle Ausgaben auf den Bildschirm sowie alle ausgefuehrten Kommandos. Eine genaue Kontrolle der auszudruckenden Textmengen ist problemlos moeglich.

- BS3 (Herstellerstandard): Hier werden nur Blockmodus-Bildschirme bestimmter Typen (SIG50/51) und Fernschreiber ueber das Satellitensystem des TR86S unterstuetzt. Waehrend die Kommunikation ueber die

Blockmodus-Bildschirme in aehnlicher Weise ablaeuft wie im Saarbruecker Netz, werden Fernschreiber nur aeusserst unzulaenglich unterstuetzt, da alle lokalen Editiervorgaenge einer noch nicht abgesendeten Eingabe ueber spezielle Kommandos und nicht ueber Editier-Characters abgewickelt werden muessen. Die fernschreiberaehnlich bedienten alten SIG100-Bildschirme nehmen eine Zwischenstellung zwischen diesen beiden Moeglichkeiten ein; wegen ihrer hohen Kosten und des hohen Aufwandes zu ihrer Ansteuerung sind sie heute nur noch als historische Kuriositaeten zu werten.

Die allgemeine Form des Dialogs mit dem Rechner und die gebotenen Eingriffsmoeglichkeiten unterscheiden sich nur unwesentlich von den im Saarbruecker Netz verfuegbaren.

- Siemens 7.760 (unter den Randbedingungen des Saarbruecker Netzes):
Text-Eingabe erfolgt wie zum TR440 und unterliegt deshalb im wesentlichen den fuer die TR440-Kommunikation geltenden Randbedingungen. Als schwerwiegende Einschraenkung gilt hier, dass Eingaben von Kommandos und Eingaben von Steueranweisungen an Dienstprogramme einzeln erfolgen muessen; es ist nicht moeglich, mehrere Kommandos und/oder Steueranweisungen zusammen (etwa getrennt durch Zeilenwechsel) einzugeben.

Eingaben an den Rechner werden vom System durch eine schliessende spitze Klammer quittiert; dieses Zeichen laesst sich nicht unterdruecken, was wie bei der TR440 unangenehme Folgen haben kann.

Nach der Eingabe eines solchen Textblocks, der in fast allen Faellen nur aus einer Zeile bestehen darf, sind keine nicht-leeren Eingaben moeglich, bis vom Hauptrechner eine Ausgabe erfolgt ist, die das Terminal wieder eingabeberechtigt macht. Eine leere Eingabe unterbricht das gerade laufende Programm; durch ein spezielles Kommando lassen sich Benutzer- und Dienstprogramme nach einem solchen Abbruch fortsetzen, doch kann durch eine solche Unterbrechung das Laufverhalten des Programms beeinflusst werden. Durch Kommandos gestartete Aktivitaeten des Organisationsprogramms werden durch eine leere Eingabe unwiderruflich abgebrochen; eventuelle Ausgaben werden dabei zerstoert. Nach Information des Herstellers wird diese Konvention ab der naechsten Systemversion geaendert; leere Eingabe unterbricht dann das laufende Programm nicht mehr, sondern dazu wird die Eingabe eines Sonderzeichens erfordert, dessen ASCII-Wert noch festzustellen ist (zur Zeit nur als nicht zuzuordnende Taste der Siemens-Bildschirme bekannt).

Laengere Textausgaben koennen auf drei verschiedene Arten ausgegeben werden (waehlbar durch ein Kommando):
- als Menge einzelner Zeilen mit Pausen im Zehntelsekundenbereich zwischen den Teilausgaben;
- desgleichen, doch wird nach jeweils etwa 10 Zeilen eine Kunstpause von etwa 3 Sekunden eingelegt;
- ebenso, doch wird nach jeweils etwa 10 Zeilen vom Benutzer eine nicht(!)-leere Eingabe zur Anforderung des naechsten Blocks erwartet.

Die hier genannten Formen der Ausgabe werden jedoch nicht von allen

Dienstprogrammen beruecksichtigt; ob ein Programm sich an die Einstellung der Ausgabeparameter haelt, ist im wesentlichen Gluecks-sache. Wiederholung einer Ausgabe ist nicht moeglich, Abbruch nur durch Abbruch des ausgebenden Programms. Unterbrechung einer laufenden Ausgabe konnte beim ersten der Ausgabemodi nicht zuverlaessig erzwungen werden.

Ein zeichenweiser Dialog sowie ein echter Voll-Duplex-Betrieb ist mit diesem System ebenfalls nicht moeglich, da der Bildschirm zwangsweise das eingegebene Zeichen als Reflexion erhaelt und da die eigentliche Kommunikation mit dem Rechner im Halb-Duplex-Betrieb auf der Ebene einzelner Zeilen geschieht.

Druckausgaben von Dateien koennen als eigene Auftraege dem Output-Spooler uebergeben werden; zusaetzlich besteht ueber spezielle Kommandos die Moeglichkeit, den am Terminal gefuehrten Dialog mitprotokollieren zu lassen und dieses Protokoll am Jobende ausgedruckt zu erhalten. Das Protokoll enthaelt alle Ausgaben auf den Bildschirm sowie alle Eingaben vom Bildschirm. Eine explizite Steuerung der ausgedruckten Textmengen ist nur schwer bis ueberhaupt nicht moeglich, da die einzelnen Dienstprogramme zusaetzlich Output in nur schwer beeinflussbarer Form erzeugen.

- BS2000 (Herstellerstandard): Ohne besondere Vorkehrungen sind nur Siemens-Bildschirme zu betreiben. Diese Bildschirme arbeiten in einem Pseudo-Blockmodus, der das Absenden einzelner Textzeilen und deren Korrektur vor dem eigentlichen Absenden ermoeglicht. Die Textzeilen werden durch Eingabe von ETX an den Rechner uebertragen. Zusaetzlich ist es moeglich, schon abgeschickte Zeilen, sofern sie noch auf dem Bildschirm sichtbar sind, erneut an den Rechner abzusenden, wobei allerdings der Anfang des Abschickbereiches nicht klar definiert ist. Bildschirme fremder Hersteller, speziell Teletype-kompatible und graphische Bildschirme sind zwar bei geeigneter Generierung des Satellitensystems im Prinzip anschliessbar; sie werden jedoch im wesentlichen nicht unterstuetzt.

  Die allgemeine Form des Dialogs mit dem Rechner und die gebotenen Eingriffsmoeglichkeiten unterscheiden sich nur unwesentlich von den im Saarbruecker Netz verfuegbaren, doch koennen bei Verwendung von Bildschirmen des Herstellers deren Blockmodus-Eigenschaften (etwa fuer Screen-Editing) eingesetzt werden.

- VAX/VMS: Text wird zeichenweise eingegeben und unter der Kontrolle des einlesenden Programms reflektiert. Innerhalb einer Textzeile ist es moeglich, durch Rubout schon geschriebene Zeichen zu loeschen und so einzelne fehlerhaft eingegebene Textstellen zu korrigieren, ehe sie endgueltig an den Rechner uebertragen werden. Bei Eingabe von Carriage Return wird die Zeile an den Rechner uebertragen und ausgefuehrt; eine nachtraegliche Korrektur ihres Textes ist dann nicht mehr moeglich. Bei Bedarf, z.B. beim Arbeiten mit Bildschirm-Editoren, ist auch ein Dialog auf der Basis einzelner Zeichen moeglich, sowie ein Steuern des Bildschirms als zweidimensionales Ausgabegeraet.

Ein- und Ausgaben werden nicht durch vom Rechner dazwischengesetzte Zeichen voneinander getrennt; das Terminal-Protokoll enthaelt exakt die Zeichen, die unter Kontrolle des laufenden Programms dorthin ausgegeben werden sollen.

Die Terminals sind immer eingabeberechtigt; unmittelbar nach dem Absenden einer Zeile kann weiterer Text eingegeben werden, auch wenn der Rechner noch keine Antwort auf den zuletzt eingegebenen Text gegeben hat oder selbst wenn eine Ausgabe laueft.

Durch Eingabe von Steuerzeichen koennen laufende Programme jederzeit unterbrochen werden; sie koennen dann durch Kommandos fortgesetzt oder abgebrochen werden.

Ausgaben erfolgen kontinuierlich mit maximaler Geschwindigkeit; durch Eingabe von Steuerzeichen koennen sie an beliebiger Stelle angehalten, fortgesetzt, unterdrueckt und abgebrochen werden. Eine Wiederholung von Ausgaben ist nicht moeglich. Standardmaessig wird eine laufende Ausgabe durch Eingabe von XOFF (Control-S) angehalten und durch Eingabe von XON (Control-Q) fortgesetzt; dieses Protokoll wird von vielen externen Geraeten verschiedener Hersteller hardwaremaessig unterstuetzt, so dass fuer diese Geraete keine eigene Synchronisation erforderlich ist.

Der Dialog erfolgt durch die voellige Entkoppelung von Ein- und Ausgabe auch auf logischer Ebene im Voll-Duplex-Betrieb; es ist ohne weiteres moeglich, einen Dialog auf der Basis einzelner Zeichen zu fuehren.

Als Besonderheit ist hier noch zu vermerken, dass die Terminal-Schnittstelle parametrisiert ist, so dass es als Voreinstellung fuer bestimmte Benutzer/Terminals oder per Kommando bzw. vom Programm aus im interaktiven Betrieb moeglich ist, die aktuellen bzw. gewuenschten Terminal-Eigenschaften dem Rechner zu spezifizieren, z.B.:

- Unterdrueckung von Meldungen des Operateurs
- Echo der eingegebenen Zeichen vom Rechner oder nicht
- Umschaltung Scroll-/Page-Modus
- Synchronisation ueber XON/XOFF-Protokoll vom Rechner/Terminal
- Umschaltung Teilnehmer-/Teilhaber-Betrieb
- Verfuegbarkeit von Kleinschreibung/Umsetzen in Grossschreibung
- Transparent-Modus
- Uebertragungsgeschwindigkeit (Baud-Rate) (!)
- Automatische Behandlung von Tabulatorzeichen
- Abschalten der Moeglichkeit freier Eingabe
- Einstellung von Seiten-/Zeilenlaenge
- Automatische Erzeugung von Zeilenwechsel bei ueberlangen Zeilen

Druckausgaben von Dateien koennen als eigene Auftraege dem Output-Spooler uebergeben werden; zusaetzlich besteht ueber spezielle Kommandos die Moeglichkeit, Drucker als exklusive Geraete zu betreiben und ungepuffert zu bedienen. Eine Protokollierung des Bildschirm-Dialoges ist nicht vorgesehen. Eine genaue Kontrolle der auszudruckenden Textmengen ist problemlos moeglich.

- MAX IV: Text wird zeichenweise eingegeben und vom Rechner reflektiert. Innerhalb einer Zeile ist es moeglich, durch Rubout schon geschriebene Zeichen zu loeschen und so einzelne fehlerhaft eingegebene Zeichen zu korrigieren. Bei Eingabe eines Carriage Return wird der Text an den Rechner abgeschickt und kann dann nicht mehr nachtraeglich korrigiert werden.

  Ein- und ausgegebener Text werden nicht durch vom Rechner dazwischengesetzte Zeichen getrennt; das Terminal-Protokoll enthaelt exakt die ein- und ausgegebenen Texte.

  Nach der Eingabe eines solchen Textblocks sind in demselben Dialog keine weiteren Eingaben moeglich, bis der Rechner auf diese Eingabe - im allgemeinen mit einer Ausgabe - reagiert hat. Es ist jedoch moeglich, mehrere Dialoge auf demselben Terminal parallel zu fuehren und Eingaben an einen weiteren Dialog zu machen, ehe der erste Dialog fortgesetzt werden kann.

  Durch Eingabe von Break koennen laufende Programme jederzeit abgebrochen werden; durch Eingabe eines Kommandos laesst sich ein so abgebrochenes Programm fortsetzen.

  Ausgaben vom Rechner erfolgen kontinuierlich mit maximaler Geschwindigkeit; durch Eingabe von Steuerzeichen koennen sie jederzeit angehalten, fortgesetzt oder abgebrochen werden. Zusaetzlich ist es moeglich, laufende Ausgaben an beliebiger Stelle auf andere Geraete (z.B. vom Bildschirm auf Drucker oder Magnetband) umzulenken, ohne dass dazu das ausgebende Programm neu gestartet werden muesste.

  Ein zeichenweiser Dialog sowie ein echter Voll-Duplex-Betrieb ist standardmaessig hier nicht vorgesehen, da der Bildschirm zwangsweise das eingegebene Zeichen als Reflexion erhaelt und da die eigentliche Kommunikation mit dem Rechner im Halb-Duplex-Betrieb auf der Ebene einzelner Zeilen geschieht. Durch die Moeglichkeit paralleler Dialoge sowie durch die vom System gebotenen Werkzeuge zur Konfiguration eigener Umgebungen lassen sich diese Einschraenkungen bei dem Modcomp-System unschwer umgehen.

  Druckausgaben von Dateien koennen als eigene Auftraege an eine als Output-Spooler laufende Symbiont-Task uebergeben werden; zusaetzlich besteht ueber spezielle Kommandos die Moeglichkeit, Drucker exklusiv zu betreiben und ungepuffert zu bedienen. Eine Protokollierung des Bildschirm-Dialoges ist nicht vorgesehen. Eine genaue Kontrolle der auszudruckenden Textmengen ist problemlos moeglich.

Zusammenfassend laesst sich dazu sagen, dass der Benutzer bei der VAX11 volle Kontrolle ueber die von ihm ausgeloesten System-Aktivitaeten hat; speziell die Steuerung der Ausgabe ist bequemer als bei den anderen Systemen. Bei Verwendung von "Smooth-Scroll"-Bildschirmen sind Bedienungskomfort und Steuerungsmoeglichkeiten als nahezu optimal zu bezeichnen. Die Eingriffsmoeglichkeiten des Benutzers

sind bei der Modcomp-Anlage etwas geringer, doch laesst sich auch hier ziemlich weitgehender Komfort bei Bedarf zur Verfuegung stellen. Bei der TR440 ist der Komfort deutlich geringer als bei den beiden Prozessrechnern, was zum Teil durch die Groesse und Struktur des zu bedienenden Terminalnetzes bedingt ist. Es gibt jedoch auch hier Moeglichkeiten fuer den Benutzer, seine Programme und Ausgaben interaktiv zu steuern, wenn auch mit einigem Schreibaufwand und einem bei laengeren Ausgaben unangenehmen Nachfordernmuessen der einzelnen Teile. Die Siemens-Anlage bietet dagegen so gut wie ueberhaupt keine Interaktivitaet; der Benutzer steht dem System ziemlich hilflos gegenueber, weil ihm kaum Eingriffsmoeglichkeiten geboten werden und weil die wenigen vorhandenen Moeglichkeiten sehr fehleranfaellig sind - sowohl auf der Seite des Benutzers als auch auf der des Systems. Der Benutzer hat in vielen Faellen nur die Moeglichkeit, das laufende Programm abzubrechen - was bei den zur Zeit verwendeten Konventionen oft auch unabsichtlich geschieht, da hierzu nur eine leere Eingabe erforderlich ist. Durch Verwendung (relativ teurer) Blockmodus-Bildschirme des Herstellers laesst sich der Bedienungs-Komfort zwar etwas steigern, doch werden diese Bildschirme bei einem eventuellen Rechnerwechsel praktisch wertlos; auch sind sie zum Anschluss an Fremdrechner nur bei Implementierung entsprechender Treiberroutinen verwendbar.

Fuehrt man die Maschinen in der Reihenfolge der gebotenen Interaktionsmoeglichkeiten auf, so ergibt sich das folgende Bild:

VAX11/780 > Modcomp 7870 >> TR440 >> Siemens 7.760

Das interaktive Verhalten der einzelnen Systeme wird natuerlich ausser durch die gebotenen Moeglichkeiten auch sehr stark durch ihre Reaktionszeiten auf Eingaben des Benutzers am Bildschirm beeinflusst. Hier besteht ebenfalls ein deutlicher Unterschied zwischen den beiden Grossrechnern einerseits und den beiden Prozessrechnern andererseits: Waehrend die Reaktionszeiten der Grossrechner im Sekundenbereich liegen und bei staerkerer Systembelastung auf 10 Sekunden und mehr anwachsen, liegen die typischen Reaktionszeiten der Prozessrechner im Zehntelsekundenbereich und damit in der Groessenordnung der menschlichen Reaktionszeit. Bei der VAX11 wird auch durch die Moeglichkeit der kontinuierlichen Eingabe, ohne dass dazu eine Berechtigung des Bildschirms vorliegen muss, bei kurzfristigen Systemengpaessen die Illusion sofortiger Reaktion aufrechterhalten, was bei laengerer Arbeit am Bildschirm zu einer nicht unbetraechtlichen Entlastung des Benutzers fuehrt.

Generell laesst sich daher sagen, dass die VAX11 von den betrachteten Rechnern die bequemste Terminal-Schnittstelle bietet, was nicht zuletzt auf die Moeglichkeit jederzeitiger Eingabe zurueckzufuehren ist. (Ein Telephon mit der Moeglichkeit, dass beide Kommunikationspartner gleichzeitig sprechen koennen, ist in der Bedienung auch bequemer als eine Wechselsprechanlage, bei der nur jeweils einer zum Sprechen berechtigt ist - auch wenn von der Moeglichkeit gleichzeitigen Redens nicht immer Gebrauch gemacht werden muss. Man kann natuerlich auch mit der Wechselsprechanlage im Prinzip dieselben Informationen uebermitteln wie mit dem Telephon - dies sei hier unbestritten.)

27. 6.80
We

Setzt man diese Ergebnisse in Beziehung zu der fuer Timesharing-Systeme geforderten hohen Interaktivitaet bei Reaktionszeiten < 500 ms, so zeigt sich, dass die betrachteten Grossrechner aufgrund ihres hohen internen Verwaltungsaufwandes den Prozessrechnern hoffnungslos unterlegen sind und daher kaum mehr als Timesharing-Rechner bezeichnet werden duerfen, ein Ergebnis, das sich voll mit den Resultaten des von der Universitaet Erlangen durchgefuehrten Rechnervergleichs deckt [1]. Dazu kommt, dass der Benutzer in relativ grossem Abstand vom System gehalten wird, so dass seine Eingriffsmoeglichkeiten ziemlich gering sind, was speziell im E/A-Verhalten der Rechner zu unangenehmen und stoerenden Nebeneffekten fuehren kann. Auch hier sind die Prozessrechner deutlich ueberlegen, wobei die betrachteten Systeme dennoch einen voelligen Schutz der einzelnen Benutzer voreinander bieten, also keinesfalls unsicherer in ihrem Verhalten sind als die Betriebssysteme der Grossrechner.

3.2.2. Hilfsmittel zur Programm-Entwicklung im Dialog

3.2.2.1. Text-Editoren

Die meisten der betrachteten Rechner bieten mehrere Text-Editoren zur Auswahl an, wobei Bedienung, Komfort und Anwendungsgebiete dieser Editoren oft sehr verschieden sind.

- BS3: Hier stehen in Saarbruecken vier verschiedene Editoren zur Verfuegung:

  - PS&TEXTHALT (Kommandos #TEINTRAGE, #TZLOESCHE, #TKOPIERE usw.): Dieses Programm stellt bis heute den einzigen vom Hersteller gelieferten Editor dar. Es handelt sich um ein relativ primitives Paket von Kommandos, mit dem lokale Textveraenderungen sowie einfache Manipulationen an Texthaltungsdateien vorgenommen werden koennen. Die Bedienung ist einfach zu erlernen, aber ziemlich umstaendlich und schreibaufwendig.

  - KORRIGIERE (Kommando #KORRIGIERE): Ein (wegen des Fehlens eines verwendbaren Editors des Herstellers) auf der Basis einer Vorform des Berliner Editors in Saarbruecken entwickelter Zeileneditor fuer lokale Textveraenderungen mit Moeglichkeiten der Stringsuche. Dieser Editor ist sehr einfach zu erlernen, verfuegt jedoch nicht ueber Moeglichkeiten zur automatischen Textersetzung und zum Kopieren von Textstuecken.

  - KONTEX (Kommando #TAUFBEREITE): Ein von der GMD erhaltener Zeileneditor mit Moeglichkeiten zur Stringsuche und -ersetzung. Die Bedienung dieses Editors ist wegen seiner etwas umstaendlichen Syntax nicht ganz so einfach; lokale Textaenderungen lassen sich nur mit Muehe vornehmen.

  - SB&EDIT (Kommando #EDIT): Eine in Saarbruecken vorgenommene Implementierung des programmierbaren Character-Editors TECO der pdp10. Dieser Editor bietet sehr weitgehende Moeglichkeiten lokaler und globaler Textmanipulation, erfordert jedoch eine lange Eingewoehnungszeit, besonders fuer Benutzer, die nur Zeileneditoren gewoehnt sind.

  Waehrend die drei ersten dieser Editoren direkt die zu editierenden Dateien veraendern, arbeitet der vierte Editor mit einem Editierpuffer und automatischem Back-up der zu aendernden Texte; das Back-up umfasst die aktuelle und die vorhergehende Version des Textes.

- BS2000: Hier stehen zwei Editoren des Herstellers und ein Fremdeditor zur Verfuegung, von denen einer der Hersteller-Editoren jedoch den Anschluss bestimmter Siemens-Bildschirme voraussetzt und daher in Saarbruecken nicht einsetzbar ist.

- EDT: Ein Batch-orientierter Zeileneditor des Herstellers mit Moeglichkeiten zur automatischen Textmanipulation und zur Arbeit mit Editierprozeduren; lokale Textaenderungen werden nicht unterstuetzt. Die Bedienung ist wegen ihrer etwas umstaendlichen und teilweise inkonsistenten Syntax nicht ganz einfach; Fehlbedienungen, auch mit schwerwiegenden Folgen, koennen leicht unabsichtlich geschehen.

- EDOR: Ein Hardware-abhaengiger Bildschirmeditor, der nicht an die Randbedingungen eines heterogenen Netzes anpassbar ist und daher hier nicht eingesetzt werden kann.

- EDIERE: Ein programmierbarer Zeileneditor des GRZ Berlin mit Moeglichkeiten kontextabhaengiger Textmanipulation und lokaler mustergesteuerter Textersetzung. Dieser Editor bietet einen grossen Funktionsumfang und Anpassung an spezielle Umgebungen (Programmiersprachen, Datenbanksysteme usw.), erfordert aber entsprechenden Lernaufwand zu seiner Beherrschung. Ein Einsatz und damit eine Bewertung dieses Editors ist wegen Anpassungsschwierigkeiten an die Randbedingungen des Saarbruecker Netzes zur Zeit noch nicht moeglich. Da dieser Editor auch auf einer Reihe von TR440-Anlagen installiert ist, duerfte sein Einsatz fuer die entsprechenden Rechenzentren bei einer eventuellen Umstellung auf BS2000 zweckmaessig sein.

- KORRIGIERE: Eine (wegen des Fehlens eines verwendbaren und bequemen Editors des Herstellers) in Saarbruecken entwickelte Parallelform des TR440-Editors KORRIGIERE mit vergleichbaren Leistungen.

EDT kann wahlweise direkt auf den zu aendernden Dateien oder mit einem Editierpuffer arbeiten; im letzteren Fall ist kein Back-up vorgesehen. Daher fuehrt bei beiden Betriebsweisen ein unabsichtliches Zerstoeren der zu editierenden Textmenge zu einer nicht zu reparierenden Zerstoerung des Datei-Inhalts. Dasselbe gilt auch fuer EDOR, der ebenfalls direkt auf den zu aendernden Dateien arbeitet.

- VAX/VMS: Hier stehen fuenf sehr unterschiedliche Texteditoren zur Verfuegung.

  - SOS: Ein Zeileneditor des Herstellers mit Moeglichkeiten zur automatischen, manuellen und prozeduralen Textmanipulation und einer eingebauten Help-Funktion. Die Bedienung ist mit nicht zu grossem Aufwand erlernbar. Der Editor ist nur bei zeichenweisem Voll-Duplex-Betrieb einsetzbar, stellt jedoch keine besonderen Anforderungen an die Eigenschaften der zu betreibenden Terminals.

  - SLP: Ein sequentieller, Batch-orientierter Zeileneditor mit der Moeglichkeit, Zeilen, auch kontextabhaengig, auszutauschen. Die Bedienung ist relativ einfach, entsprechend dem eingeschraenktem Leistungsumfang dieses Editors.

  - EDT: Ein Zeileneditor mit Steuerung ueber eine Kommandosprache;

zusaetzlich steht ein eingeschraenkter Bildschirm-Editiermodus zur Verfuegung, der an alle Bildschirme mit Vollduplex-Anschluss und Cursorsteuerung angepasst werden kann. Die Bedienung ist mit nicht zu grossem Aufwand zu erlernen.

- TECO: Dieser Editor bietet sehr weitgehende Moeglichkeiten lokaler und globaler Textmanipulation durch extrem kurze Eingaben des Benutzers und die Moeglichkeit der Definition von Editiermakros und -programmen, ist jedoch relativ schwer zu erlernen.

- VTECO: Eine Erweiterung von TECO um einen Bildschirmeditor, der im Mischbetrieb mit der kommandogesteuerten Editierung von TECO benutzt werden kann. Dieser Editor stellt ziemlich den zur Zeit hinsichtlich Leistungsfaehigkeit und Benutzerkomfort optimalen Editor dar, erfordert aber ebenfalls echten Voll-Duplex-Betrieb zur Steuerung der Bildschirme und ist nur an Bildschirme mit Cursor-Adressierung anpassbar.

Alle Editoren arbeiten mit Editierpuffern und automatischem Back-up, das beliebig viele Versionen des zu editierenden Textes umfasst Um die Anzahl der dabei anfallenden Datei-Versionen reduzieren zu koennen, steht ein spezielles Kommando (PURGE) zur Verfuegung.

- MAX IV: Hier stehen drei Editoren zur Verfuegung, von denen jedoch nur einer untersucht werden konnte.

  - SED: Ein Zeileneditor mit Moeglichkeiten zur manuellen und automatischen Textmanipulation. Die Bedienung ist ohne sonderliche Schwierigkeiten zu erlernen.

Die zur Textmanipulation gebotenen Moeglichkeiten umfassen bei den meisten der betrachteten Rechner ein relativ breites Spektrum von einfach zu bedienenden, aber leistungsschwachen Primitiv-Editoren bis hin zu sehr leistungfaehigen, aber entsprechenden Lernaufwand erfordernden Programmen. Als Kompromiss werden von Siemens und DEC Bildschirmeditoren angeboten, die jedoch gewisse Anforderungen an die Terminal-Hardware stellen. Diese Anforderungen sind bei Siemens jedoch wesentlich hoeher als bei DEC, da der Siemens-Editor EDOR nur bestimmte Bildschirme genau dieses einen Herstellers unterstuetzt, waehrend die DEC-Bildschirmeditoren mit beliebigen Asynchron-Bildschirmen arbeiten koennen, sofern diese Vollduplex-Betrieb zulassen und ueber eine Moeglichkeit der Cursorsteuerung verfuegen. Stehen nur Teletype-aehnliche Bildschirme zur Verfuegung, so bieten ausser Siemens alle Rechner noch ausreichende Editiermoeglichkeiten. Bei Siemens steht als Editor dann nur EDT zur Verfuegung, und dieser Editor bietet keine Moeglichkeiten zur lokalen mustergesteuerten Textmanipulation; ausserdem ist seine Bedienung dank der uneinsichtigen, nicht orthogonalen und dazu noch fehlerhaft implementierten Syntax seiner Kommandos nicht ganz einfach. Dazu kommt noch, dass dieser Editor keine Back-up-Kopien fuehrt, so dass Benutzerfehler zur Zerstoerung der Original-Text-Dateien fuehren. Beim Zurueckschreiben des geaenderten Textes wird, wenn man keine besonderen Vorkehrungen trifft, der alte Text durch den neuen ueberschrieben; ein automati-

sches Retten des alten Textes - wenigstens bis zum naechsten Editor-Lauf - erfolgt nicht, obwohl dies unter Verwendung von Datei-Generationen oder qualifizierten Dateinamen im Prinzip moeglich waere. Das Fehlen eines automatischen Back-up-Mechanismus ist zwar auch bei den drei ersten der fuer BS3 aufgefuehrten Editoren zu bemaengeln, doch spielt es hier - mit Ausnahme des kaum benutzten Editors der GMD - keine so grosse Rolle, da diese Editoren nur lokale Textaenderungen durchfuehren und daher die Gefahr einer Zerstoerung der Dateien geringer ist.

Fuer die Programm-Entwicklung groesserer Systeme, bei der oft ein Rueckgriff - etwa zu Dokumentationszwecken - auf fruehere Versionen erforderlich ist, duerfte ein Back-up-System ueber beliebig viele Versionen am besten geeignet sein; zweckmaessig waere hier aber eine Moeglichkeit der Unterscheidung in transiente und permanente Systemversionen. Ein solches Programm-Entwicklungs-System waere unter Verwendung vorhandener System-Schnittstellen unter VAX/VMS sehr einfach implementierbar, auf allen anderen Maschinen nur mit wesentlich hoeherem Aufwand.

3.2.2.2. Datei-System

Saemtliche der betrachteten Rechner stellen alle gaengigen Datei-Typen sowie eine Datei-Verwaltung ueber Kataloge zur Verfuegung; hier wurden nur die Systeme der TR440, der Siemens und der VAX11 naeher betrachtet.

- BS3: (Langfristige) Dateien werden zu Gruppen, den sogenannten Benutzerkennzeichen BKZ, zusammengefasst; dabei koennen jedem Benutzer bis zu 6 dieser Gruppen zugeordnet werden, und Zugriff auf die erste dieser Gruppen kann ohne Angabe eines Zugiffspfades geschehen. Die Zuordnung der BKZs ist statisch; sie kann nur vom Systemverwalter geaendert werden.

  In Bearbeitung befindliche Dateien koennen zu frei definierten Gruppen, sogenannten Datenbasen, zusammengefasst werden; doch bestehen hierbei unter Umstaenden Wechselwirkungen mit den BKZs. Dateien auf externen Traegern werden durch zusaetzliche Angabe eines Datentraegerkennzeichens (EXDKZ) identifiziert. Zugriff auf Magnetbaender ist nur im Batch-Betrieb moeglich.

  Zugriff auf eigene Dateien ist unbeschraenkt moeglich; Zugriff auf Dateien eines nicht eingetragenen BKZ nur dann, wenn sie von ihrem Eigentuemer ausdruecklich als Gemeinschaftsdateien deklariert wurden. Zusaetzlicher Zugriffsschutz ist durch Angabe von Passwoertern moeglich; dabei wird zwischen lesendem und schreibendem Zugriff unterschieden. Die Auflistung von Katalog-Eintraegen privater Dateien wird fuer Fremdbenutzer unterdrueckt, damit diese nicht einmal von der Existenz solcher privater Dateien erfahren.

  Dateien werden durch einfache, unstrukturierte Namen innerhalb ihrer Datenbasis identifiziert; diese Namen sind um ein Paar aus Generations- und Versions-Nummer erweitert.

  Als Besonderheit ist hier zu erwaehnen, dass in jedem Job Dateien erzeugt werden koennen, die bei Job-Ende automatisch geloescht werden (Scratch-Dateien). Auf diese Dateien kann von anderen Jobs (auch desselben Benutzers) ueberhaupt nicht zugegriffen werden.

  Es stehen physikalische, sequentielle und drei Typen von index-sequentiellen Dateien zur Verfuegung. Bei sequentiellen und index-sequentiellen Dateien werden festes, beschraenktes und variables Satzformat unterstuetzt. Das Datei-System unterscheidet zwischen Text- und Binaerdateien.

- BS2000: Die Dateinamen sind eine Konkatenation von Knotennamen in einem Baum:

  - Jeder Knoten wird durch die Konkatenation der Knotennamen bezeichnet, die einen Weg von der Wurzel bis dorthin bilden.

- Jeder Nicht-End-Knoten bezeichnet den ganzen daran haengenden Subbaum.

- Jeder End-Knoten bezeichnet eine Datei.

- Die Knoten der ersten Ebene bezeichnen die einzelnen Benutzer; bei Zugriff auf eigene Dateien kann der Name des Knotens der ersten Ebene weggelassen werden.

- Der Dateiname kann an seinem Ende um eine Generations- und eine Versions-Nummer erweitert werden, wobei die verschiedenen Versionen einer Datei zu einer "Datei-Generations-Gruppe" zusammengefasst werden koennen.

Zugriff auf eigene Dateien ist unbeschraenkt moeglich; Zugriff auf Dateien eines anderen Benutzers nur dann, wenn sie von diesem ausdruecklich als Gemeinschaftsdateien deklariert wurden. Zusaetzlicher Zugriffsschutz ist durch Angabe von Passwoertern moeglich; dabei wird zwischen lesendem, schreibendem und ausfuehrendem Zugriff unterschieden. Es ist nicht vorgesehen, mehreren Benutzern durch Definition einer Benutzergruppe untereinander groessere Zugriffsrechte zu geben als der Allgemeinheit. Die Auflistung von Katalog-Eintraegen privater Dateien wird fuer Fremdbenutzer unterdrueckt, damit diese nicht einmal von der Existenz solcher privater Dateien erfahren sollen, doch kann diese Information ohne weiteres ueber ein PRINT-Kommando erhalten werden, zusammen mit der Angabe, welche Zugriffsformen durch Passwort geschuetzt sind. Da ausserdem der Zugriff mit Passwort ueber Definition prozesseigener Passwortlisten, bei denen nur ueberprueft wird, ob das benoetigte Passwort ueberhaupt in der Liste eingetragen ist, und nicht ueber direkte Zuordnung einzelner Passwoerter zu einzelnen Dateien geschieht, kann ein erfahrener Programmierer relativ leicht Teile des Datenschutzes umgehen.

Es stehen physikalische, sequentielle und indexsequentielle Dateien zur Verfuegung, sowie spezielle Magnetband- und Plattenzugriffsmethoden; bei sequentiellen und indexsequentiellen Dateien werden Saetze variabler und fester Laenge unterstuetzt, letztere sind jedoch relativ umstaendlich zu bedienen, so dass ihre Verwendung kompliziert und fehleranfaellig ist. Das Datei-System unterscheidet nicht zwischen Text- und Binaerdateien, falls diese derselben Zugriffsmethode unterliegen.

Als Kuriosum ist anzumerken, dass nach der Dokumentation der derzeitigen Systemversion 5.0 keine Magnetbaender mit mehr als einer Datei unterstuetzt werden; erst ab der in Kuerze ausgelieferten Version 6 stehen Multifile-Reels offiziell zur Verfuegung.

- <u>VAX/VMS:</u> Dateien werden durch strukturierte Namen identifiziert; die einzelnen Teile des Dateinamens spezifizieren einen Suchweg zur Lokalisierung der Datei, bestehend aus:

  - Rechner, zu dem die Datei gehoert

- Geraet, auf dem sich der Datentraeger befindet, der die Datei enthaelt

- Katalog, der die Datei enthaelt; hier kann ein beliebiger Baum zur Spezifikation einer Hierarchie von Katalogen eingesetzt werden

- Dateiname

- Erweiterung des Dateinamens; enthaelt Information ueber die semantische Bedeutung des Datei-Inhalts (z.B. als FORTRAN-Quelle, als uebersetzte Routine oder ausfuehrbares Programm)

- Versionsnummer

Die Struktur der einzelnen Komponenten des Dateinamens erlaubt die Angabe teilqualifizierter Namen unter Verwendung statischer oder dynamischer Voreinstellungen fuer die weggelassenen Teile bzw. zur Bezeichnung von Dateimengen, die nach bestimmten Kriterien als zusammengehoerig aufgefasst werden.

Der Zugriff auf alle Dateien (auch die eigenen) wird ueber ein Berechtigungs-Profil gesteuert, das einerseits zwischen
- Eigentuemer der Datei
- Projektgruppe des Eigentuemers
- Systemverwalter
- alle anderen Benutzer

und andererseits zwischen
- lesendem
- schreibendem
- ausfuehrendem
- loeschendem

Zugriff unterscheidet.

Es stehen sequentielle, indexsequentielle und direkt adressierte Dateien zur Verfuegung; dabei werden bei sequentiellen und indexsequentiellen Dateien feste und variable Satzformate unterstuetzt, bei direkten Dateien nur feste Satzformate. Das System unterscheidet nicht zwischen Text- und Binaerdateien.

Als Besonderheit ist hier zu nennen, dass jeder Benutzer sich beliebige Hierarchien von Unterkatalogen dynamisch aufbauen kann und dass Zugriffspfade durch frei vereinbarte logische Namen vordefiniert werden koennen, so dass die zugreifbare Dateimenge sehr flexibel und dynamisch voreingestellt werden kann.

Waehrend bei dem Zugriff auf die Daten in einer Datei relativ geringe Unterschiede zwischen den einzelnen Maschinen bestehen, ist die Organisation des gesamten Datei-Systems, speziell der Zugriff auf die Dateien als eigenstaendige Objekte, sowohl in seiner Leistungsfaehigkeit als auch in der Form der Bedienung sehr verschieden.

So tendiert das Siemens-System - wie uebrigens schon seine Vorbilder

bei IBM - zu sehr langen, oft in weiten Teilen aehnlichen Dateinamen, was wegen des damit verbundenen hohen Schreibaufwandes und der Gefahr, durch Eingabefehler falsche Namen zu spezifizieren, im interaktiven Betrieb sehr unpraktisch ist. Hier waere es sinnvoll, durch dynamische Definition von Voreinstellungen fuer die Namen beliebiger Knoten im Datei-Baum abgekuerzte Dateinamen verwenden zu koennen. Weiterhin ist es unverstaendlich, wieso die Moeglichkeit der Definition von Unterkatalogen, verbunden mit einer Baumsuche in einer solchen Hierarchie von Dateinamen, nicht ausgenutzt wird, obwohl sich dies bei der Baumstruktur der Dateinamen geradezu anbietet, sondern die Katalogeintraege eines Benutzers sequentiell durchsucht werden - was bei umfangreicheren Katalogen zu langsamen Datei-Zugriffen fuehrt. Hier wurden nur die Nachteile eines Dateibaumes implementiert; auf die Ausnutzung seiner Vorteile wurde weitgehend verzichtet. Zusammenfassend muss daher das Datei-System der Siemens-Anlage als das schlechteste und leistungsschwaechste der Systeme in diesem Vergleich bezeichnet werden.

Bei der TR440 ergeben sich relativ einfache Dateibezeichnungen im Standardfall; Zugriffe auf fremde Dateien sowie auf Dateien auf wechselbaren Traegern sind oft fehlerprovozierend und nicht sonderlich benutzerfreundlich, was auf die Verwendung mehrerer, einander ueberlappender und schlecht aufeinander abgestimmter Konzepte zurueckzufuehren ist. Besonders unangenehm ist hier, dass ein interaktiver Benutzer keinen Zugriff auf Magnetbaender hat, so dass hierfuer jeweils ein Batch-Job gestartet werden muss.

Von den betrachteten Maschinen hat die VAX11 das flexibelste und leistungsfaehigste Datei-System; durch die dynamische Definition von Unterkatalogen, logischen Geraete- und Dateinamen sowie die Verwendung teilqualifizierter Namen bietet dieses System das breiteste Leistungsspektrum bei gleichzeitig hoher Benutzerfreundlichkeit.

3.2.2.3. Compiler und Verwaltung von Programm-Moduln

Saemtliche der betrachteten Rechner verfuegen ueber ein breites bis sehr breites Angebot an Programmiersprachen. Die Uebersetzungszeiten der drei moderneren Maschinen sind etwa gleich (fuer FORTRAN) und zum Teil kleiner als bei der TR440; die Bindezeiten sind sogar um ein Vielfaches kleiner als auf der TR440, was auf den relativ kompliziert aufgebauten Binaer-Output der TR440-Compiler zurueckzufuehren ist. Der angebotene Sprach-Umfang ist fuer viele Sprachen (besonders FORTRAN) auf den neueren Maschinen groesser als auf der TR440; den groessten Sprachumfang bei FORTRAN findet man auf der VAX11. Ein Einsatz von FORTRAN als Systemprogrammiersprache zur Programmierung eigener Anwendungssysteme und Programmier-Umgebungen ist ohne Einschraenkungen nur bei den beiden Prozessrechnern moeglich, da sowohl bei der TR440 als auch bei der Siemens der Sprachumfang zu restriktiv ausgelegt wurde, um einen vollen Durchgriff auf alle benoetigten Resourcen zu gestatten. Auch bei diesem Teil des Vergleichs sollen nur die drei Maschinen TR440, Siemens 7.760 und VAX11/780 respektive ihre Betriebssysteme BS3, BS2000 und VAX/VMS genauer betrachtet werden.

Zum Austesten fertiger Programme steht auf allen der betrachteten Rechner ein System von Testhilfen mit Moeglichkeiten zum definerten Eingriff in Programme zur Verfuegung. Es koennen sowohl auf der Ebene der Maschinensprache als auch auf der der Quellsprache fuer die wichtigsten der verwendeten Programmiersprachen
- Haltepunkte (Breakpoints) definiert
- Speicherbereiche ueberprueft
- Werte von Variablen ausgegeben/gesetzt
- Befehlsfolgen/Zugriff auf Speicherbereiche ueberwacht

werden. Die Arbeit mit den Testhilfe-Systemen geschieht auf einer der jeweiligen Kommandosprache aequivalenten Ebene und braucht daher hier nicht weiter analysiert zu werden.

Zur Bedienung der Compiler und zur Verwaltung von uebersetzten und von lauffaehigen Programmen ist festzustellen:

- BS3: Alle Compiler werden ueber eine einheitliche Schnittstelle (Kommando #UEBERSETZE) bedient. Es ist in jedem Fall moeglich, vom Compiler erzeugte Listen auf das eigene Terminal, auf den Schnelldrucker oder in Dateien ausgeben zu lassen, bzw. sie ganz zu unterdruecken.

  Der vom Compiler erzeugte sogenannte Montagecode wird in speziellen Scratch-Dateien, die bei Auftragsbeginn automatisch eingerichtet werden, unter dem Namen des uebersetzten (Unter-)Programms abgelegt. Ubersetzen eines gleichnamigen Programms ueberschreibt die alte Version automatisch.

  Es ist in gewissem Umfang moeglich, Programme aus Routinen aus verschiedenen Programmiersprachen aufzubauen, falls diese Sprachen dieselben Unterprogrammanschlusskonventionen verwenden; doch gilt

dies nicht allgemein fuer die Kombination beliebiger Programmiersprachen.

Der Aufruf von Betriebssystem-Leistungen (Systemdiensten) ist nur vom Assembler aus direkt moeglich; Aufruf von den meisten hoeheren Programmiersprachen aus ist nur unter Zwischenschaltung einer Assembler-Routine moeglich, von der aus der eigentliche Aufruf der Systemdienste relativ einfach und bequem ist.

Mittels eines Linkage Editors kann aus einer Menge uebersetzter Programme/Routinen ein lauffaehiges Programm erzeugt werden; die Menge aller lauffaehigen Programme wird ebenfalls in speziellen Speicherbereichen abgelegt. Auch lauffaehige Programme werden durch ihre Namen identifiziert und beim Ablegen neuer Programme gleichen Namens durch diese ueberschrieben.

Sowohl uebersetzte als auch lauffaehige Programme koennen in Bibliotheken organisiert werden; hierzu stehen zwei verschiedene Bibliotheksverwaltungssysteme zur Verfuegung, die - mit gewissen Einschraenkungen - auf denselben Bibliotheken arbeiten koennen. Beim Aufbau lauffaehiger Programme unter Verwendung von Bibliotheken koennen diese so angeordnet werden, dass sie in bestimmter Reihenfolge zur Absaettigung von Aussenbezuegen durchsucht werden. Als negativ muss hier angemerkt werden, dass die Bedienung der Bibliotheksverwaltungsprogramme relativ umstaendlich ist, was speziell bei ungeuebten Benutzern leicht zur Zerstoerung der Modulbibliotheken fuehrt. Hier waere ein standardmaessiger Linkage-Editor und Librarian, wie er auf moderneren Maschinen verfuegbar ist, zur Erhoehung der Leistungsfaehigkeit und Bedienbarkeit unbedingt erforderlich.

- BS2000: Es gibt keine einheitliche Schnittstelle zur Bedienung der Compiler; jeder Compiler erhaelt seine Steuerinformation auf eine andere Art und Weise, und diese Steuerinformation ist fuer keine zwei Compiler gleich aufgebaut. Dazu kommt noch, dass ein Teil der Compiler manche Steuerinformationen gar nicht oder falsch auswertet und dass die Steuerung der Compiler-Ausgaben nur unvollkommen moeglich ist. Es ist nicht bei jedem Compiler moeglich, die Ausgabe der erzeugten Listen wahlweise auf den eigenen Bildschirm, den Drucker oder in eine Datei zu lenken; speziell Ausgaben auf den eigenen Bildschirm sind oft nicht vorgesehen. Die neueren Compiler sollten zwar nach Planung des Herstellers auf einheitliche und konsistente Weise versorgt werden, doch wurde dies leider nicht realisiert, sondern weiteres Durcheinander in der Bedienung geschaffen, indem hier Wechselwirkungen zwischen Steuerung ueber Kommandosprache und Steueranweisungen nicht verhindert wurden.

  Der vom Compiler erzeugte Binaer-Output wird sequentiell in eine spezielle, automatisch erzeugte Datei geschrieben. Mehrfaches Uebersetzen eines Programmes fuehrt nicht zum Ueberschreiben der alten Version, sondern zur Ablage der neuen Version hinter der alten. Da sowohl Linking Loader als auch Linkage Editor die Bindemoduldatei ebenfalls sequentiell von vorn durchsuchen, ist es nicht moeglich, in dieser Datei einzelne Moduln zu ersetzen; man kann nur die Datei

als Ganzes loeschen und neu erzeugen.

Es ist in gewissem Umfang moeglich, Programme aus Routinen aus verschiedenen Programmiersprachen aufzubauen, falls diese Sprachen dieselben Unterprogrammanschlusskonventionen verwenden; doch gilt dies nicht allgemein fuer die Kombination beliebiger Programmiersprachen.

Der Aufruf von Betriebssystem-Leistungen (SVCs, Makros) ist nur vom Assembler aus direkt moeglich; Aufruf von den meisten hoeheren Programmiersprachen aus ist nur unter Zwischenschaltung einer Assembler-Routine moeglich, von der aus der eigentliche Aufruf der Systemdienste relativ einfach und bequem ist. Zum Anschluss der Assembler-Routine an die hoehere Sprache ist jedoch unter Umstaenden die Zwischenschaltung spezieller Anpassungs-Makros erfoderlich, was die Uebersetzung dieser Routine und den anschliessenden Bindevorgang etwas umstaendlich gestaltet.

Mittels eines Linking Loader kann eine Menge uebersetzter Programm-Moduln in einem Arbeitsgang gebunden, geladen und gestartet werden, was fuer einmalige Tests recht praktisch ist, fuer groessere, mehrfach zu verwendende Programme jedoch zu aufwendig ist.

Hierzu steht ein Linkage Editor zur Verfuegung, der es ermoeglicht, aus der Bindemoduldatei und aus beliebigen Bibliotheken lauffaehige Programme zusammenzubinden und in jeweils einer Datei abzulegen. Die Steuerung des Linkage Editors geschieht durch eine Folge von Anweisungen, durch die der Bindevorgang beschrieben werden kann.

Die Verwaltung von Bibliotheken uebersetzter Programme geschieht analog durch ein Dienstprogramm. Bibliotheken lauffaehiger Programme sind nicht vorgesehen und auch wegen der verwendeten Ablageform (einzeln in jeweils eigenen Dateien) nicht notwendig.

- VAX/VMS: Alle Compiler werden ueber eine einheitliche Schnittstelle bedient; der Aufruf eines Compilers geschieht durch ein Kommando, das aus dem Namen der zu uebersetzenden Sprache besteht. Es ist in jedem Fall moeglich, vom Compiler erzeugte Listen auf das eigene Terminal, auf den Schnelldrucker oder in Dateien ausgeben zu lassen, bzw. sie ganz zu unterdruecken.

  Der von den Compilern erzeugte Binaer-Output wird in automatisch erzeugte Dateien geschrieben, die mit den Quell-Programmen ueber die Namenskonventionen des Datei-Systems verknuepft sind. Neues Uebersetzen einer Datei gleichen Namens erzeugt eine Binaer-Datei mit um 1 erhoehter Versions-Nummer, so dass zwar standardmaessig auf die neue Version des Programms zugegriffen wird, aber dennoch der Zugriff auf die alte Version nicht verwehrt ist.

  Da Unterprogramm-Anschluss und Parameter-Uebergabe bei dieser Maschine Hardware-unterstuetzt nach einheitlichem Schema geschehen, ist es moeglich, Moduln aus beliebigen Programmiersprachen ohne irgendwelche Einschraenkungen zu einem lauffaehigen Programm zusammenzubinden.

Der Aufruf von Betriebssystem-Leistungen (Systemdiensten) geschieht ueber dieselbe Hardware-unterstuetzte Unterprogramm-Anschluss-Schnittstelle, so dass die Systemdienste von beliebigen Programmiersprachen aus direkt aufgerufen werden koennen.

Eine Menge uebersetzter Moduln kann mittels eines Linkage Editors gebunden und in eine Datei abgelegt werden, die ebenfalls mit dem Namen des Quell-Programms ueber die Namenskonventionen des Datei-Systems verknuepft ist. Neues Binden eines Programms erhoeht ebenfalls die Versions-Nummer dieser Datei, so dass auch hier beide Versionen verfuegbar bleiben und standardmaessig auf die neuere der beiden zugegriffen wird.

Besondere Erwaehnung verdient hier die Tatsache, dass Systembibliotheken prinzipiell nicht an Programme angebunden werden, sondern ueber feste, vom System vorgegebene Adressen im virtuellen Adressraum angesprochen werden, so dass bei Aenderungen einer Systembibliothek die betroffenen Programme nicht neu gebunden werden muessen. Wegen der Verwendung einer einheitlichen Unterprogramm-Schnittstelle kann das System ueberdies mit einer einzigen Laufzeitbibliothek fuer alle Programmiersprachen arbeiten; diese Bibliothek umfasst im wesentlichen alle Funktionen, die in irgendeiner der Sprachen benoetigt werden - bis hin zu den zentralen Teilen eines Scanners und eines Parsers -, stellt diese Funktionen jedoch ohne Einschraenkungen allen Sprachen zur Verfuegung.

Mittels eines speziellen Kommandos (LIBRARY) ist die Erzeugung und Manipulation von Modul-Bibliotheken in aehnlicher Weise wie auf der TR440 moeglich. Bibliotheken lauffaehiger Programme sind nicht vorgesehen und auch wegen deren Ablage einzeln in Dateien nicht notwendig.

Die angebotenen Dienstleistungen sind bei den drei Rechnern weitgehend vergleichbar, doch ist die Bedienung des Programmiersystems wegen der uneinheitlichen Compiler-Steuerung bei der Siemens-Anlage deutlich unbequemer und anfaelliger gegen Benutzerfehler als bei den beiden anderen Maschinen. Diese Fehleranfaelligkeit wird noch durch die Abarbeitung der Bindmoduldatei in der falschen Richtung deutlich erhoeht. Dagegen zeigt die Verwaltung der lauffaehigen Programme bei der Siemens und bei der VAX11, dass fuer diese die Verwendung spezieller Bibliotheken - wie bei der TR440 - entbehrlich ist; generell ist die Bibliotheksverwaltung eine Schwachstelle im BS3.

3.2.2.4. Kommandosprache

Die betrachteten vier Rechner verfuegen ueber Kommandosprachen (Job Control Languages), die sich in Aufbau und Leistungsfaehigkeit zum Teil sehr stark unterscheiden; lediglich die Kommandosprachen der TR440 und der VAX11, die die flexibelsten und leistungsfaehigsten der hier betrachteten Sprachen sind, haben strukturell und auch von den angebotenen Leistungen einige Aehnlichkeit.

- BS3: Die Kommandosprache hat eine einheitliche, flexible und uebersichtliche Syntax. Alle Kommandos sind aufgebaut aus:

  - einem Steuerzeichen, das einen Text als Kommando identifiziert ("Fluchtsymbol")

  - dem Kommandonamen

  - einer (moeglicherweise leeren) Kommando-spezifischen Liste von Parametern; jeder Parameter wird identifiziert durch:
    - entweder seine Stellung in der Liste
    - oder einen vor den Parameter-Wert gesetzten Parameter-Namen, gefolgt von einem Gleichheitszeichen;

    beide Formen der Parameter koennen in einem Kommando gemischt verwendet werden.

  Kommando- und Parameter-Namen koennen beliebig abgekuerzt werden, solange ihre Eindeutigkeit gewahrt bleibt. Die einzelnen Teile eines Kommandos werden durch Kommata voneinander getrennt, Listen von Parameter-Werten auf einer Parameter-Position durch Apostroph.

  Parameter werden in obligate und optionale unterschieden; optionale Parameter koennen weggelassen werden, waehrend weggelassene obligate Parameter im Dialog nachgefragt werden. Fuer beide Typen von Parametern koennen statisch oder dynamisch Voreinstellungen (Default Values) definiert werden. Dadurch und durch die Moeglichkeit der Abkuerzung von Kommando- und Parameter-Namen koennen Kommando-Eingaben oft sehr kurz sein, was im interaktiven Betrieb wichtig ist.

  Die einzelnen Kommandos sind nach einer einheitlichen Struktur aufgebaut, die sich weitgehend bis in die Form der Parameter-Listen erstreckt. Die Bedeutung der Kommandos und ihrer Parameter ergibt sich weitgehend schon aus ihren Namen. Zusaetzliche Hilfen werden dem Benutzer durch ein Informations-System geboten (Kommando #INFORMIERE), das die Erfragung des syntaktischen Aufbaus der Kommandos, ihrer Parameter mit deren Voreinstellungen sowie fuer viele Kommandos eine Beschreibung ihrer Anwendung und Wirkung ermoeglicht.

  Die Laenge der einzelnen Kommandos ist im Prinzip unbeschraenkt. Eine Eingabe an den Rechner kann (fast) beliebig viele Kommandos enthalten, da die einzelnen Kommandos durch die Fluchtsymbole von-

einander getrennt sind. Ausserdem ist die Eingabe von Kommandos auch dann moeglich, wenn ein Benutzerprogramm Daten erwartet; die Umschaltung in den Kommando-Modus erfolgt automatisch, sofern sie nicht vom Benutzer-Programm selbst unterdrueckt wird.

Es ist ohne weiteres moeglich, eigene Kommandos und Kommando-Prozeduren zu definieren; ausserdem koennen Mengen solcher Kommandos und Prozeduren (auch Definitionen von Kommandos und Voreinstellungen) in Dateien eingetragen und diese dann als indirekte Kommandos ausgefuehrt werden (Kommando #TUE). Hierbei sind Spruenge, Abfragen von Fehlersituationen und vom Benutzer gesetzten Wahlschaltern zulaessig.

Das Niveau der Kommandosprache entspricht damit etwa dem einer hoeheren Programmiersprache; dabei sind natuerlich durch die spezielle Bedeutung dieser Sprache einige Aenderungen und Einschraenkungen gegenueber einer normalen Programmiersprache gegeben.

Die Menge der zur Verfuegung stehenden Kommandos ist fuer alle ueblicherweise benoetigten Operationen ausreichend; da sie fuer Spezial-Anwendungen dynamisch erweitert werden kann, lassen sich alle Anforderungen durch die Kommandosprache abdecken. Negativ ist dagegen zu vermerken, dass die Menge der Kommandos nicht orthogonal aufgebaut ist: Fuer gleiche oder aehnliche Operationen, die sich nur in einzelnen Parametern unterscheiden, koennen oder muessen oft verschiedene und verschieden aufgebaute Kommandos verwendet werden, was den Umfang der Kommandosprache unnoetig vergroessert.

- <u>BS2000:</u> Die Kommandosprache hat eine uneinheitliche, unflexible und starre Syntax. Kommandos sind ueblicherweise aufgebaut aus:

  - einem Steuerzeichen, das einen Text als Kommando identifiziert; im Dialogbetrieb wird dieses Steuerzeichen vom Rechner ausgegeben und braucht dann nicht mehr geschrieben zu werden

  - einem frei waehlbaren Namen (optional), der als Sprungziel verwendet werden kann

  - dem Operationsnamen

  - einer (moeglicherweise leeren) Liste von Operanden; dabei geschieht die Identifizierung der Operanden
    - fuer bestimmte Operanden durch ihre Stellung in der Liste
    - fuer andere Operanden durch ein vor den Wert gesetztes Schluesselwort, das ihn identfiziert, gefolgt von einem Gleichheitszeichen
    - fuer einen dritten Operanden-Typ durch einen im System definierten Operandennamen, dem keine Angabe eines Wertes folgen darf

  - Kommentaren (optional)

  Kommando- und Operandennamen duerfen nicht abgekuerzt werden; fuer manche Kommandos existiert jedoch ein zweiter, kuerzerer Name. Die

einzelnen Teile eines Kommandos werden durch Zwischenraeume (Blanks) voneinander getrennt; die Operanden der Operanden-Liste muessen dagegen durch Kommata voneinander getrennt werden. Der frei waehlbare Name muss mit einem Punkt beginnen; Kommentare muessen in Anfuehrungszeichen eingeschlossen sein. Kommandos duerfen sich ueber bis zu 11 Zeilen erstrecken; dabei muessen alle Zeilen ausser der letzten mit einem Minuszeichen abgeschlossen werden, das im Dialogbetrieb in der letzten Schreibposition der Zeile, im Stapelbetrieb dagegen in Spalte 72 stehen muss, und alle Zeilen muessen mit dem Steuerzeichen beginnen.

Operanden werden in obligate und optionale unterschieden; bei weggelassenen durch ihre Stellung identifizierten Operanden muessen die entsprechenden Kommata geschrieben werden. Fuer einen Teil der Operanden sind statisch Voreinstellungen definiert, fuer andere nicht, selbst wenn sie obligat sind und nur einen Wert erlauben. Hierdurch sind k/mmando-Eingaben oft sehr lang und durch ihre komplizierte Struktur fehleranfaellig. Bei Fehlern werden Kommandos im allgemeinen einfach als syntaktisch falsch zurueckgewiesen, ohne dass ein Hinweis auf die Art oder den Ort des Fehlers gegeben wird.

Die Struktur der Kommandos ist uneinheitlich; aus dem Aufbau eines Kommandos lassen sich keinerlei Schluesse auf den irgendeines anderen Kommandos ziehen. Selbst Operanden exakt gleicher Bedeutung haben zum Teil in verschiedenen Kommandos verschiedenen Aufbau. Die Namen der Kommandos und ihrer Operanden bestehen in der ueberwiegenden Anzahl der Faelle aus unverstaendlichen Abkuerzungen, und selbst vom Text her verstaendliche Kommandos tuen oft etwas ganz anderes, als ihr Name suggeriert. Es stehen keine Hilfsmittel zur Verfuegung, um im Dialog Bedeutung, Form und Aufbau von Kommandos zu erfragen.

Eine Eingabe an den Rechner darf nur genau ein (Teil-)Kommando enthalten; im Dialogbetrieb ist Kommando-Eingabe prinzipiell nur dann moeglich, wenn der Rechner durch Ausgabe eines Steuerzeichens seine Bereitschaft dazu angezeigt hat.

Es ist nicht moeglich, eigene Kommandos zu definieren, doch kann man Kommando-Prozeduren schreiben und ausfuehren lassen sowie deren Parameter mit Voreinstellungen versehen. Die Syntax des Aufrufs von Kommando-Prozeduren deckt sich nicht mit der Syntax normaler Kommandos, so dass hier weitere Fehlbedienungen provoziert werden.

Das Niveau der Kommandosprache entspricht etwa dem eines nicht zu komfortablen Makro-Assemblers; diese Aequivalenz geht hin bis zu syntaktischen Einzelheiten. Speziell die Kommandos des Datenverwaltungssystems entsprechen bis in die letzten Einzelheiten den aequivalenten Makro-Aufrufen des Assemblers; sie muessen auch genauso ausfuehrlich angegeben werden und werden so starr und unbeholfen abgearbeitet, dass der Verdacht naheliegt, dass hier ueberhaupt kein Kommandoentschluessler vorhanden ist, sondern dass die Kommandos direkt einem Assembler mit angeschlossenem Interpreter zur Bearbeitung uebergeben werden.

Die Menge der zur Verfuegung stehenden Kommandos ist weder ortho-

gonal noch ausreichend; fuer die meisten der im normalen Betrieb benoetigten Operationen, zum Beispiel:
- Ausgabe einer Datei auf den Bildschirm
- Editieren von Texten
- Compilieren
- Binden von Programmen
- Verwalten von Modul-Bibliotheken

stehen ueberhaupt keine Kommandos zur Verfuegung, waehrend andererseits fuer aehnliche Operationen verschiedene Versionen eines Kommandos bzw. verschiedene Kommandos verwendet werden. Die Situation wird noch verschlimmert dadurch, dass die fehlenden Kommandos durch das Starten von Dienstprogrammen ersetzt werden, wobei jedes dieser Dienstprogramme wieder ueber eine eigene Steuersprache bedient wird, die sowohl zur Kommandosprache als auch zu allen anderen Steuersprachen inkompatibel ist. Dazu kommt noch, dass an vielen Stellen Wechselwirkungen zwischen Steuersprachen und Kommandosprache bestehen, die zu unangenehmen Nebeneffekten fuehren koennen. (So hebt zum Beispiel die Steueranweisung "*COMOPT NOLISTFILE=(DIAG)" an den FORTRAN-Compiler die Wirkung des Kommandos "/PARAM ERRFIL=YES" wieder auf.

Um ungeuebten Benutzern einen Einstieg in die Systembedienung zu ermoeglichen, wird seit Kurzem ein spezieller Kommando-Interpreter (EASP - "Einfache Anweisungs-Sprache") angeboten. Hier haben Kommando- und Spezifikationsnamen einen einheitlichen Aufbau; beide Arten von Namen koennen frei abgekuerzt werden. Die Kommandos sind in Gruppen eingeteilt; als Kommandonamen muss man die Konkatenation von Gruppen- und Einzelnamen spezifizieren. Die Arbeit mit diesem Kommando-Interpreter erfolgt in Form eines gefuehrten Dialogs, bei dem durch Eingabe vollstaendiger Kommandos auch einzelne Schritte uebersprungen werden koennen. Laut Angabe des Herstellers gibt es hier auch die Moeglichkeit der Definition eines eigenen Gedaechtnisses, doch ist es auch hiermit nicht moeglich, die einzelnen Benutzergruppen zugaenglichen Dienstleistungen einzuschraenken, da die Standard-Kommandos neben EASP weiter benutzbar bleiben.

Da dieses Software-Produkt erst kurze Zeit zur Verfuegung steht, koennen hier nur erste Eindruecke geschildert werden:

- Die Bedienung ist wesentlich einheitlicher, flexibler und einfacher als die der normalen Kommandosprache.

- Die Einteilung der Kommandos in Gruppen und die Spezifikation der Kommandonamen als Konkatenation von Gruppen- und Einzelnamen erscheint gekuenstelt und uneinsichtig; die Namen selbst weisen auf die Funktion der Kommandos/Spezifikationen oft nicht hin. Eine Ueberarbeitung des verwendeten Vokabulars waere daher dringend erforderlich.

- Es werden relativ gute Informationsdienste geboten, die ein Arbeiten mit dem System ohne Verwendung von Manuals ermoeglichen. Wenn man jedoch vergessen hat, wie man den Interpreter wieder verlaesst, kann das Beenden des eigenen Jobs eine relativ schwierige Angelegenheit werden, da genau zum Aufruf der diesbezueglichen Informationsdienste die Kenntnis der Anwei-

sungen zum Abbruch der Kommando-Eingabe erforderlich ist.

- Auch hier wurde als Trennzeichen zwischen Kommando und Spezifikationen Blank, als Trennung zwischen den Spezifikationen Komma gewaehlt, so dass die Kommandosyntax wieder uneinheitlich ist; eine Vereinheitlichung der Syntax waere zweckmaessig.

- Erste Versuche mit dem System scheiterten zum Teil daran, dass der Interpreter bestimmte Annahmen ueber den Zugriff auf Dienstprogramme macht, die fuer die Saarbruecker Systemkonfiguration teilweise falsch sind. Inwieweit hier Parametrisierungen moeglich oder vorgesehen sind, ist noch nicht bekannt; notwendig sind sie in jedem Fall.

- Der Interpreter scheint relativ aufwendig zu sein und daher unverhaeltnismaessig viel Rechenzeit zu benoetigen; es waere zu ueberpruefen, wieweit dieser Effekt im Dauerbetrieb den Rechner belastet.

- Wie bei den IBM-Timesharing-Systemen entsteht auch hier die Schwierigkeit, dass im Endeffekt zwei verschiedene Bedienungsweisen der Maschine nebeneinander bestehen. Es waere daher zu ueberpruefen, ob nach einer Ueberarbeitung dieses Produkts die alte Kommandosprache voellig aufgegeben werden koennte.

- VAX/VMS: Die Kommandosprache hat eine einheitliche, flexible und uebersichtliche Syntax. Alle Kommandos sind aufgebaut aus:

  - dem Zeichen "$", das einen Text als Kommando identifiziert; im Dialog wird dieses Zeichen vom Rechner ausgegeben und braucht dann nicht mehr geschrieben zu werden

  - einer Sprungmarke (optional), gefolgt von einem Doppelpunkt

  - dem Kommandonamen

  - einer (moeglicherweise leeren) Kommando-spezifischen Liste von Parametern in beliebiger Reihenfolge; diese Parameter koennen bestehen aus:
    - einem Dateinamen oder einer durch Kommata oder "+" getrennten Liste solcher Namen
    - einem Qualifikator, der durch das Zeichen "/" eingeleitet wird; einige Qualifikatoren koennen durch Voranstellen von "NO" in ihr Gegenteil verkehrt werden, anderen koennen durch "=" Werte zugewiesen werden

  - durch "!" eingeleiteten Kommentaren (optional)

  Kommando-, Qualifikator- und Parameter-Namen koennen beliebig abgekuerzt werden, solange ihre Eindeutigkeit gewahrt bleibt; die Eindeutigkeit einer Abkuerzung auf vier Zeichen wird garantiert. Die einzelnen Teile eines Kommandos werden durch Zwischenraum oder Tabulator voneinander getrennt, Listen von Parameter-Werten auf einer Parameter-Position durch Kommata.

Parameter werden in obligate und optionale unterschieden; optionale Parameter koennen weggelassen werden, waehrend weggelassene obligate Parameter im Dialog nachgefragt werden. Fuer beide Typen von Parametern koennen durch Ueberlagerung mit logischen Namen Voreinstellungen definiert werden. Dadurch und durch die Moeglichkeit der Abkuerzung von Kommando- und Parameter-Namen sind Kommando-Eingaben im allgemeinen sehr kurz, was im interaktiven Betrieb wichtig ist.

Die einzelnen Kommandos sind nach einer einheitlichen Struktur aufgebaut, die sich weitgehend bis in die Form der Parameter-Listen erstreckt. Die Bedeutung der Kommandos und ihrer Parameter ergibt sich weitgehend schon aus ihren Namen. Zusaetzliche Hilfen werden dem Benutzer durch ein Informations-System geboten (Kommando HELP), das die Erfragung der Bedeutung der Kommandos, ihrer Parameter mit deren Voreinstellungen sowie eine Beschreibung ihrer Anwendung und Wirkung ermoeglicht.

Kommandos duerfen sich ueber mehrere Zeilen erstrecken; dabei muessen alle Zeilen ausser der letzten mit einem Minuszeichen abgeschlossen werden.

Es ist ohne weiteres moeglich, eigene Kommandos und Kommando-Prozeduren zu definieren; ausserdem koennen Mengen von Kommandos, Parameter-Werten, Qualifikatoren und Prozeduren (auch Definitionen von Kommandos und Voreinstellungen) in Dateien eingetragen und diese dann als indirekte Kommandos bzw. Parameter ausgefuehrt werden. Hierbei sind Spruenge, Abfragen von Fehlersituationen und vom Benutzer gesetzten Wahlschaltern zulaessig. Es ist sogar moeglich, in der Kommando-Sprache eigene numerische und String-Variable zu erzeugen, Arithmetik zu treiben und aus Dateien zu lesen bzw. text in Dateien zu schreiben.

Das Niveau der Kommandosprache entspricht damit dem einer hoeheren Programmiersprache; die Struktur dieser Sprache ist stark an FORTRAN angelehnt.

Die Menge der zur Verfuegung stehenden Kommandos ist fuer alle ueblicherweise benoetigten Operationen ausreichend; da sie fuer Spezial-Anwendungen dynamisch erweitert werden kann, lassen sich alle Anforderungen durch die Kommandosprache abdecken. Da die Sprache orthogonal aufgebaut ist, d.h. fuer jeden Operationstyp genau ein Kommando zur Verfuegung steht, dessen Anwendung auf verschiedene Situationen durch Parametrisierung gesteuert wird, konnte diese Allgemeinheit bei gleichzeitig minimalem Sprachumfang erzielt werden, was das Erlernen dieser Kommandosprache sehr erleichtert.

- MAX IV: Die Kommandosprache ist eine typische Prozessrechnersprache, die die Steuerung des Dialogs durch kurze, einfache Kommandos ermoeglicht. Kompliziertere Operationen lassen sich durch Aufbau von Kommando-Prozeduren auf frei definierte Kommandos abbilden, so dass die urspruengliche Kommandosprache nur als Geruest zum Aufbau einer allgemeineren Sprache, die leicht auf die Beduerfnisse

eines speziellen Anwendungsgebietes abzustimmen ist, aufgefasst werden kann. Fuer allgemeinen Timesharing-Betrieb steht ein solches System zur Verfuegung, das jedoch hier nicht untersucht werden konnte. Auf eine weitergehende Betrachtung der Kommandosprache der Modcomp wird aus diesem Grund hier verzichtet.

Zusammenfassend laesst sich zum Thema der Kommandosprachen feststellen, dass hier TR440 und VAX11 vergleichbare Leistungen anbieten, gegen die Siemens in geradezu grotesker Weise abfaellt - bedingt durch die Orientierung an veralteten IBM-Systemen, deren konzeptuelle Grundlagen in die fruehen 60er Jahre fallen, also in die Zeit vor der Einfuehrung der ersten Timesharing-Systeme. Eine moegliche Hife koennte hier - nach entsprechender Ueberarbeitung - die Anweisungssprache EASP sein, doch muesste hier zunaechst in einem Feldversuch ueberprueft werden, wie sich dieses neue Produkt bewaehrt. Es waere wuenschenswert, wenn diese Ueberarbeitung EASP vereinfachen wuerde; speziell sollten alle Informationen, die dem System im Prinzip bekannt sein muessten, automatisch eingesetzt und nicht vom Benutzer angefordert werden. Als vorbildhaft sind hier nach wie vor die klassischen Timesharing-Betriebssysteme TOPS-10 [2],[3], TENEX [4] und UNIX [5] zu nennen; die Qualitaet der dort gebotenen Kommandosprachen und deren Bedienungsfreundlichkeit wird von keinem der hier betrachteten Rechner erreicht, wenn auch TR440 und VAX11 diesem Ziel relativ nahe kommen.

3.2.2.5. Ein-/Ausgabe-System

Fuer die Anwendbarkeit und Leistungsfaehigkeit eines Programmiersystems sind die den Benutzer-Programmen gebotenen Ein- und Ausgabe-Schnittstellen, sowohl hin zu Dateien als auch Geraeten, ebenso wichtig wie die reine Rechenleistung der Hardware. Allerdings ist die Beurteilung eines E/A-Systems naturgemaess schwieriger als die rein numerische Ueberpruefung der Rechengeschwindigkeit - wenn auch dies schon problematisch ist, wie Abschnitt 3.1. gezeigt hat.

- BS3: Zugriff auf Dateien geschieht ueber eine einheitliche Schnittstelle, den sogenannten Achtelseiten-Transport. Die einzelnen Datenstroeme der Benutzerprogramme werden durch interne, automatisch vergebene Stromnummern identifiziert. Bei der Erzeugung eines logischen Kanals wird dieser mit einer Datei assoziiert und seine Stromnummer festgestellt. Alle weiteren Operationen verwenden dann nur noch diese Stromnummer zur Bezeichnung des Kanals, und beim Schliessen des Kanals wird die Stromnummer geloescht, womit dann kein weiterer Zugriff auf diesem Weg erfolgen kann.

  Zugriff auf die Standard-Ein-/Ausgabe-Medien der Programme Bildschirm, Drucker, Eingabestrom (z.B. vom Kartenleser) geschieht zwar auf unterer Ebene innerhalb des Systems ueber denselben Mechanismus, doch werden vom Betriebssystem andere Systemdienste zur Ein- und Ausgabe auf diesen Geraeten angeboten, was zu einer gewissen Sonderstellung dieser Geraete fuehrt.

  Auf allen Ebenen oberhalb dieser recht sauberen System-Schnittstelle herrscht eine bunte Vielfalt von E/A-Konzepten, die sich von einer Programmiersprache zur anderen in beliebigem Masse unterscheiden koennen. Dennoch ist es moeglich, Daten, die von einem beliebigen Programm nach definierten Konventionen irgendwo abgelegt wurden, von ebenso beliebigen anderen Programmen, die sich an diese Konventionen halten, wieder richtig einzulesen. Zugriff auf sequentielle und indexsequentielle Dateien ist praktisch unbeschraenkt moeglich, Zugriff auf physikalische Dateien mit Einschraenkungen.

  Die Zuordnung von logischen Kanaelen zu Dateien und/oder irgendwelchen der Standardgeraete geschieht ueblicherweise beim Start eines Programms; dabei sind Geraete und Dateien austauschbar, solange ihre physikalischen Charakteristika dies erlauben. Dynamischer Kanalwechsel in laufenden Programmen wird nicht unterstuetzt, ist jedoch ueber Assembler-Routinen moeglich.

- BS2000: Zugriff auf Dateien geschieht ueber sogenannte Link-Namen, die zwischen dem zugreifenden Programm und dem Datei-System vereinbart werden. Dazu wird durch ein spezielles Kommando der durch ihren Namen identifizierten Datei der Link-Name als Charakteristikum zugewiesen, und ueber diesen Namen kann dann das Programm auf die Datei zugreifen. Der Hersteller haelt diesen Mechanismus fuer sehr komfortabel, doch zeigen die bisherigen Erfahrungen mit EDV-

fremden Benutzern, dass es sich hier um ein relativ uneinsichtiges Konzept handelt, dessen Verstehen und Beherrschung unverhaeltnismaessig hohen Lernaufwand erfordern. Die Verwendung logischer Dateinamen als Synonyme zur Steuerung des Zugriffs waere begrifflich einfacher und gleichzeitig leistungsfaehiger und bequemer, da sie alle Funktionen der Link-Namen nachzubilden gestattet, aber nur dort wirklich eingesetzt werden muss, wo sich ein direkter Zugriff ueber den echten Dateinamen aus irgendwelchen Gruenden verbietet.

Zugriff auf die Standard-E/A-Medien geschieht ueber spezielle Dateinamen, die fest im System bekannt sind. Diese Dateien koennen nicht ueber den Mechanismus der Link-Namen angesprochen werden, so dass die Pseudo-Dateien des Systems nicht mit normalen Dateien und zum Teil auch nicht untereinander austauschbar sind. Bei Zugriff auf E/A-Geraete sind deren Charakteristika dem System in allen Einzelheiten bekannt, was unter anderem zur Folge hat, dass das E/A-System nur bestimmte Siemens-Bildschirme kennt und gewoehnliche Teletype-aehnliche Asynchron-Terminals mit einstellbarer Uebertragungsgeschwindigkeit nur sehr eingeschraenkt bedienen kann.

Zugriff auf Dateien unterliegt den fuer die einzelnen Datei-Strukturen vorgesehenen Zugriffsmethoden und laeuft daher je nach Datei-Typ ueber verschieden anzusteuernde Basis-Software. Dies hat zur Folge, dass Datei-Typen einerseits und Dateien und Geraete andererseits nicht gegeneinander ausgetauscht werden koennen. Eine weitere Folge ist die sehr eingeschraenkte Datenuebertragungsmoeglichkeit zwischen Programmen, die in verschiedenen Programmiersprachen geschrieben wurden: Es gibt nur dann die Moeglichkeit der Datenuebertragung, wenn es eine Zugriffsmethode gibt, die von beiden Sprachen aus ansteuerbar ist - was in diesem System alles andere als eine Selbstverstaendlichkeit ist -, es sei denn, man beschraenkt sich auf Daten im Lochkartenformat und Zugriff ueber die entsprechenden Pseudo-Dateien.

Die Zuordnung von logischen Kanaelen zu Dateien und/oder irgendwelchen der Standardgeraete geschieht entweder vor dem Start eines Programms durch Definition von Link-Namen oder Umdefinition der Pseudo-Dateien oder waehrend des Programmlaufes durch vom Programm selbst verursachte Definition eines Kanals. Dabei sind wegen der oben genannten Einschraenkungen Geraete und Dateien im wesentlichen bis auf Ersetzungen durch identische Objekte nicht austauschbar.

Man stellt bei dieser Struktur des E/A-Systems fest, dass keine Trennung zwischen logischer und physikalischer E/A vorliegt, da beide im Konzept der Zugriffsmethoden miteinander vermischt sind. Diese Konstruktion hat mehrere Konsequenzen:

- Es gibt keine einheitliche, fuer alle E/A-Vorgaenge als gemeinsame Basis verwendbare Schnittstelle; dies behindert die dynamische Austauschbarkeit der E/A-Kanaele eines Programms in sehr hohem Masse bzw. schraenkt ihre Moeglichkeit stark ein.

- Die Laufzeitbibliotheken aller Programmiersprachen muessen im Prinzip jeweils alle Zugriffsmethoden enthalten, wenn voller Datenaustausch moeglich sein soll; dies fuehrt zu hoher Redun-

danz im Systemaufbau mit den negativen Folgen grossen Umfangs und schlechter Wartbarbkeit.

- In Fehlersituationen koennen Wechselwirkungen zwischen logischen und physikalischen Fehlern bzw. deren Behandlung auftreten, was die Lokalisierung der Fehlerursache sehr erschweren und ueberdies zu sekundaeren Fehlern fuehren kann.

- Die Schnittstellen des E/A-Systems werden durch die Vermischung einander fremder Konzepte unsauber, schlecht handhabbar und schwer verstaendlich.

Zusammenfassend muss daher gesagt werden, dass der Aufbau eines E/A-Systems unter Verwendung von Zugriffsmethoden weder dem heutigen Kenntnisstand noch den damit realisierbaren Dienstleistungen entspricht [6].

- VAX/VMS: Zugriff auf Dateien und Geraete (auch Fremdrechner) geschieht ueber eine einheitliche Schnittstelle, die sogenannten QIO-Systemdienste. Logische Kanaele werden eroeffnet, indem ihr Name mit einem Geraet und gegebenenfalls mit einer Datei auf diesem Geraet assoziiert wird.

  Zugriff auf die Standard-E/A-Medien geschieht ueber spezielle Dateinamen, die fest definiert und dem System bekannt sind. Diese Pseudo-Dateien werden ueber genau dieselbe Schnittstelle wie alle anderen Geraete und Dateien angesprochen und koennen frei gegen diese ausgetauscht werden, sofern ihre physikalischen Eigenschaften dies zulassen.

  Zugriff auf Dateien geschieht ueber dieselbe einheitliche Schnittstelle, so dass von der Programmierung her Dateien und Geraete voellig gleich behandelt werden. Zusaetzlich stehen fuer die Arbeit mit Dateien mit einer internen logischen Struktur (z.B. indexsequentielle Dateien) sogenannte Record Management Services zur Verfuegung, die die Verwaltung dieser Struktur uebernehmen koennen, wenn ein Benutzerprogramm dies nicht selbst tun will.

  Die Zuordnung von logischen Kanaelen zu Dateien und/oder Geraeten geschieht entweder vor dem Start eines Programms durch Zuweisung dieser Kanaele oder waehrend des Programmlaufes durch eine vom Programm selbst verursachte Definition eines Kanals. Dabei sind Dateien und Geraete frei austauschbar, solange ihre physikalischen Charakteristika einem solchen Austausch nicht entgegenstehen.

- MAX IV: Zugriff auf Dateien und Geraete (auch Fremdrechner) geschieht ueber eine einheitliche Schnittstelle, das sogenannte Basic-Input-Output-System BIOS. Logische Kanaele werden eroeffnet, indem ihr Name mit einem Geraet und gegebenenfalls mit einer Datei auf diesem Geraet assoziiert wird.

  Zugriff auf die Standard-E/A-Medien geschieht ueber spezielle Dateinamen, die fest definiert und dem System bekannt sind. Diese

Pseudo-Dateien werden ueber genau dieselbe Schnittstelle wie alle anderen Geraete und Dateien angesprochen und koennen frei gegen diese ausgetauscht werden, sogar waehrend eines Programmlaufes, sofern ihre physikalischen Eigenschaften dies zulassen.

Zugriff auf Dateien geschieht ueber dieselbe einheitliche Schnittstelle, so dass von der Programmierung her Dateien und Geraete voellig gleich behandelt werden. Zusaetzlich stehen fuer die Arbeit mit Dateien mit einer internen logischen Struktur ein sogenanntes File Management System zur Verfuegung, das die Verwaltung dieser Struktur uebernehmen kann. Dabei ist es hier moeglich, auf demselben Datentraeger verschiedene physikalische Strukturen einander zu ueberlagern, um bei Bedarf Zugriffscharakteristiken zu optimieren.

Die Zuordnung von logischen Kanaelen zu Dateien und/oder Geraeten geschieht entweder vor dem Start eines Programms durch Zuweisung dieser Kanaele oder waehrend des Programmlaufes durch eine vom Programm selbst verursachte Definition eines Kanals bzw. durch Benutzerkommandos, die existierende - selbst gerade benutzte - Kanaele umlenken. Dabei sind Dateien und Geraete unter Beachtung ihrer physikalischen Charakteristika frei austauschbar.

Dieser Vergleich zeigt, dass die beiden Prozessrechner ueber sauber strukturierte, flexible und leistungsfaehige E/A-Systeme verfuegen, die diese Vorzuege durch modularen Aufbau und Verwendung einheitlicher Schnittstellen erreichen. Das E/A-System der TR440 ist zwar im Kern aehnlich gut aufgebaut, doch wird dieser Kern leider von aelteren Konzepten im Programmiersystem ueberwuchert, so dass seine guten Eigenschaften nur sehr unvollstaendig an den Benutzer hoeherer Programmiersprachen weitergegeben werden. Die Siemens-Anlage verwendet dagegen mehrere, zueinander inkompatible Zugriffsmethoden und vermischt logische und physikalische E/A miteinander; dies fuehrt zu hoher Redundanz bei geringer Flexibilitaet und schlechter Strukturierung - eine Tatsache, die im wesentlichen darauf zurueckzufuehren ist, dass dieses E/A-System nach Konzepten der fruehen 60er Jahre aufgebaut wurde und dass selbst diese veralteten Konzepte ohne die geringste Koordinierung implementiert wurden.

3.2.3. Kompatibilitaet mit Fremdrechnern

Bei der Uebernahme fremder Software sind im allgemeinen eine Reihe von Anpassungen vorzunehmen, die je nach dem Aufbau des eigenen Rechners und den von diesem befolgten Konventionen mehr oder weniger umfangreich sein koennen. Hier ergibt sich ein Bild, das erheblich von dem der vorigen Abschnitte abweicht.

- BS3: Dieser Rechner ist als Exot zu betrachten, da er in allen wesentlichen Eigenschaften von allen anderen auf dem Markt befindlichen Rechnern abweicht, hauptsaechlich wegen folgender Eigenschaften:
  - Wortlaenge 48 bit
  - Typenkennung zur Unterscheidung von:
    - Gleitpunktzahlen
    - Festpunktzahlen
    - Befehlen
    - Text (Character-Daten)
  - Adressierungsstruktur
  - eigener Character-Code

  Dadurch stimmen die verwendeten Datenformate mit keinem Fremdrechner ueberein, und alle Texte (speziell auch Quell-Programme) muessen zur Uebertragung umgeschluesselt werden, was bei Sonderzeichen eine Fehlerquelle sein kann. Der Aufbau von TR440-Magnetbaendern ist zusaetzlich durch die Abspeicherung einer verklammerten Datei-Struktur so kompliziert, dass es fuer Fremdrechner so gut wie nicht moeglich ist, TR440-Baender zu erzeugen oder zu lesen. Es ist jedoch ohne Schwierigkeiten moeglich, IBM- und ANSI-Baender zu erzeugen und zu lesen, so dass hier zumindest eine Uebertragung von Quelltext durchfuehrbar ist.

  Dank der speziellen Wortstruktur und der Verwendung einer Typenkennung treten zum Teil bei der Uebernahme fremder Programm-Systeme erhebliche Schwierigkeiten auf, die sich nur durch aufwendige Anpassungen, im allgemeinen durch Assembler-Routinen, loesen lassen.

  Die Struktur des TR440-Assemblers ist so radikal von der anderer Assembler verschieden, dass es hoffnungslos ist, fremde Assembler-Programme umzustellen, gleich von welcher Maschine; es ist in praktisch jedem Fall einfacher, diese Programme voellig neu zu schreiben.

- BS2000: Hier besteht weitgehend die vom Hersteller herausgestellte Kompatibilitaet zu IBM, bis hin zur Wortstruktur und zum Befehlsvorrat der Maschine. Da standardmaessig EBCDI-Code gefahren wird, lassen sich Programme von IBM relativ leicht uebernehmen. Problematisch ist hier nur, dass die Sonderzeichen im Code zum Teil falsch abgebildet werden, wobei diese Fehler von der Dokumentation teilweise ignoriert bzw. durch andere Fehler ersetzt werden.

- VAX/VMS: Die VAX11/780 ist Hardware-maessig kompatibel zur pdp11, so dass alle dort vorhandene Software ohne Aenderungen uebernommen werden kann, bis hin zu unter RSX-11M/S uebersetzten und gebundenen Programmen. Als Interncode wird ASCII verwendet, so dass auch von dieser Seite keine Schwierigkeiten kommen. Generell ist Daten- und Programm-Austausch mit anderen DEC-Maschinen problemlos moeglich, waehrend Austausch mit IBM aufwendiger sein kann, da die Datenformate verschieden sind. Besonders unangenehm ist hier, dass Character-Daten in anderer Reihenfolge als bei den meisten anderen Rechnern abgespeichert werden, was unter Umstaenden zu Schwierigkeiten verursachen kann.

Waehrend also Siemens und VAX11 eine gewisse Kompatibilitaet zu anderen Rechnern zu verzeichnen haben, steht die TR440 in dieser Beziehung ziemlich allein.

3.3. Ergonomische System-Aspekte

In den folgenden Abschnitten sollen sowohl die Ergebnisse der vorangegangenen qualitativen Betrachtungen als auch weitere Beobachtungen im Umgang mit den verglichenen Systemen im Hinblick darauf untersucht werden, welchen Aufwand die interaktive Programm-Entwicklung auf den einzelnen Systemen dem Benutzer abverlangt und wie diese Systeme ihm bei seiner Arbeit helfen bzw. ihn behindern. Dabei sollen hier nur TR440, Siemens 7.760 und VAX11/780 bzw. ihre Betriebssysteme BS3, BS2000 und VAX/VMS betrachtet werden, da das Modcomp-System, das betrachtet wurde, eher als Roh-System zu werten ist, auf dem mehr oder weniger benutzerfreundliche Endsysteme aufgebaut werden koennen.

3.3.1. Verstaendlichkeit und Erlernbarkeit

Verstaendlichkeit einer Benutzerschnittstelle und Erlernbarkeit ihrer Bedienung haengen ihm wesentlichen von folgenden Faktoren ab:

- Klarheit und Einfachheit der verwendeten Systemkonzepte
- innere Konsistenz des Systemverhaltens
- struktureller Aufbau des Dialogs mit dem Rechner
- Verstaendlichkeit der Rueckmeldungen des Systems

Hier bestehen zwischen den betrachteten Maschinen zum Teil erhebliche Unterschiede.

- BS3: Die verwendeten Konzepte
  - Job (Auftrag: Abschnitt bzw. Gespraech)
  - Programm ("Operator")
  - Datei
  - Modul ("Montage-Objekt")

  sind strukturell klar und sauber aufgebaut, und dieser Aufbau spiegelt sich auch in ihrer Bedienung, speziell also der Kommandosprache, auf uebersichtliche Art wieder. Leider wird ein Teil dieser guenstigen Eigenschaften wieder durch die relativ umstaendliche Verwaltung von Programm- und Modul-Bibliotheken sowie die verschiedenen Datei-Identifizierungs-Mechanismen dem Benutzer gegenueber verschleiert, so dass der Einstieg in das System schwerer faellt als dies von den verwendeten Konzepten her eigentlich notwendig waere.

  Das System bietet dem Benutzer relativ wenig unangenehme Ueberraschungen; die meisten Reaktionen erfolgen auf im Prinzip vorhersehbare Art und Weise, insbesondere weil die dem System gegebenen Kommandos in den meisten Faellen voneinander unabhaengig sind, so dass ungewuenschte Wechselwirkungen zwischen Kommandos und Nach-

wirkungen frueherer Kommandos auf ein Minimum reduziert sind.

Der Dialog mit dem Rechner laeuft wegen des verwendeten Protokolls als Wechselgespraech, bei dem jeder Partner im Prinzip beliebige Informationsmengen an den anderen uebergeben kann. Treten hierbei Fehler des Benutzers auf, so versucht das System, die Auswirkungen dieser Fehler auf ein Minimum zu reduzieren, indem die weitere Abarbeitung der eingegebenen Information unterbrochen und ein Korrekturdialog eingeschachtelt wird. Im allgemeinen koennen hierdurch Fehler relativ einfach und schnell behoben werden, doch entstehen, wenn waehrend der Korrekturphase weitere Fehler auftreten, manchmal recht verzwickte Situationen, die den ungeuebten Benutzer ziemlich verwirren koennen.

Fehlermeldungen des Systems erfolgen in verschiedener Ausfuehrlichkeit, in Abhaengigkeit davon, ob Testmodus eingestellt ist oder nicht. Die kurze Form gibt im allgemeinen auf einfache und meist gut verstaendliche Weise Auskunft ueber die Art des Fehlers, waehrend die laengere Form zusaetzlich auf die Ursache des Fehlers hinweist und bei seiner Lokalisierung hilft. Die Meldungen erfolgen in Deutsch.

- BS2000: Es gibt keine klare Trennung der verschiedenen System-Konzepte, weder in den einzelnen System-Beschreibungen noch in der Implementierung der Konzepte. So wird zum Beispiel nicht klar unterschieden zwischen lauffaehigen Programmen und den Dateien, die diese Programme enthalten, oder zwischen Binde-Moduln, den sie enthaltenden Dateien und den aus ihnen aufgebauten Programmen. Dieses Durcheinander herrscht sowohl in der Beschreibung des Systems als auch in der Kommandosprache, wo oft Programm-, Modul- und Dateinamen bunt gemixt auftreten, wobei dasselbe Kommando unter Umstaenden voellig verschiedene - oft falsche, ungewollte oder sinnlose - Reaktionen hervorruft, wenn man ihm das falsche Objekt als Parameter uebergibt. (Ein Musterbeispiel hierfuer ist das allgegenwaertige EXECUTE-Kommando.) Andererseits werden oft gleiche Objekte durch kuenstliche Unterscheidungen zu verschiedenen, verschieden aufgebauten und verschieden zu bedienenden Objekten gemacht. Hiervon ist, wie schon im vorherigen Abschnitt erlaeutert, insbesondere das E/A-System betroffen, das durch diese Unklarheit der verwendeten Konzepte in etwa ein halbes Dutzend zueinander inkompatibler Teilsysteme zerfaellt. Es ist schwer bis unmoeglich, einem (nicht an das bei IBM auf diesem Gebiet herrschende Durcheinander gewoehnten) Benutzer Verstaendnis fuer solche schlecht aufeinander abgestimmten und schlecht voneinander abgegrenzten Konzepte beizubringen; dies fuehrt natuerlich zu erheblichen Problemen bei der Einfuehrung neuer Benutzer in diese Anlage: Man muss hierzu einen recht hohen Personalaufwand ansetzen, was den Betrieb dieser Anlage gerade fuer ein Universitaets-Rechenzentrum nicht unwesentlich verteuert oder ineffizient macht. Erste Erfahrungen im Einsatz der Maschine im Lehrbetrieb zeigten jedenfalls, dass die Betreuung der Studenten einen unverhaeltnismaessig hohen Aufwand erfordert, waehrend Lernende und Betreuer von der Arbeit mit dem System frustriert sind.

Das Systemverhalten im interaktiven Betrieb bringt jeden Tag neue Ueberraschungen; es ist selten, dass eine vorher noch nicht erprobte Operation das erwartete Ergebnis bringt. Dies liegt zum Teil an falscher, missverstaendlicher oder fehlender Dokumentation (siehe Abschnitt 3.3.3.), zu einem sehr grossen Teil jedoch einfach auch an Implementierungsfehlern, die von falschen Scan-Algorithmen fuer die Entschluesselung von Kommandos und Steueranweisungen ueber Fehler in den Compilern oder ihren Laufzeitsystemen bis hin zu falsch oder gar nicht behandelten Hardware-Fehlern (besonders im E/A-System) reichen. Dazu kommt noch die generelle Unverstaendlichkeit der Kommandosprache, aus deren Schluesselwoertern nur zu oft nicht zu erkennen ist, was der Implementator damit gemeint hat. (Was versteht man z.B. unter "BCNTRL"?) Es ist daher kaum moeglich, ohne staendige Konsultierung mehrerer Manuals mit diesem System zu arbeiten.

Eine weitere Ursache unangenehmer Ueberraschungen ist das Fehlen sauberer Schnittstellen innerhalb des Systems, was oft zu unkontrollierter Fortpflanzung von Fehlern fuehrt, bis hin zur Zerstoerung des Mikroprogramms der Zentraleinheit, wenn ein Drucker im unrechten Augenblick ausgeschaltet wird. Auch das Konzept der Pseudo-Dateien als Stellvertreter der Standard-E/A-Medien ist eine solche Quelle haeufiger Fehler, da bei falscher Zuordnung dieser Pseudo-Dateien, etwa weil vergessen wurde, eine temporaere Zuordnung wieder aufzuheben, Datenstroeme in unkontrollierter Weise umgelenkt werden, was zum Beispiel dazu fuehrt, dass der Linkage Editor als Steueranweisungen ein Quellprogramm einliest.

Da in den meisten Faellen bei einem solchen Fehler keine Reaktion auf System-Ebene, sondern hoechstens die Ausgabe einer Meldung erfolgt, treten oft auch noch Folgefehler in unueberschaubarem Ausmass auf. So werden zum Beispiel beim Uebersetzen syntaktisch falscher Programme von einigen Sprachprozessoren dennoch Binde-Moduln erzeugt, die sich binden, laden und starten lassen - natuerlich im allgemeinen ohne sinnvoll zu laufen. Auf dieser Ebene liegt auch das schon in Abschnitt 3.2.2.3. besprochene sequentielle Abspeichern der Binde-Moduln in einer Datei, die dann vom Binder in derselben Reihenfolge durchsucht wird, so dass ein Ersetzen einzelner Module durch neuere Versionen ohne Zuhilfenahme der Bibliotheksdienste nicht moeglich ist.

Der Dialog mit dem Rechner laeuft auch hier als Wechselgespraech, wobei die zum Rechner zu uebertragende Informationsmenge jedoch ausser bei der Steuerung spezieller Programme nur aus einer einzigen Anweisung bestehen darf. Treten hierbei Fehler des Benutzers auf, so wird ueblicherweise ein eingegebene Kommando als Ganzes zurueckgewiesen, ohne dass das System Korrekturversuche vornimmt.

Fehlermeldungen erfolgen in verschiedener Ausfuehrlichkeit, in Abhaengigkeit von der durch ein Kommando waehlbaren Art der Protokollfuehrung. Die von einzelnen Dienstprogrammen ausgegebenen Meldungen werden hiervon nicht beeinflusst; sie sind zum Teil separat zu steuern, zum Teil auch ueberhaupt nicht. Die Meldungen erfolgen in mehr oder weniger schlechtem Englisch, das von orthographischen und grammatischen bis hin zu sachlichen Fehlern voll ist; dies hat

zur Folge, dass die Meldungen im allgemeinen nur dem Eingeweihten Hinweise auf die Fehlerursache geben, da sie oft nur nach woertlicher Rueckuebersetzung ins Deutsche unter Ignorierung jeglicher grammatischer Regeln einen erkennbaren Text ergeben bzw. da sie auch nicht zu selten das Gegenteil von dem ausdruecken, was gemeint ist.

Eine moegliche Verbesserung an der Benutzerschnittstelle koennte der Einsatz des Kommando-Interpreters "EASP" (siehe Abschnitt 3.2.2.4.) bringen, wenn dessen Anfangsschwierigkeiten ueberwunden sind. Man darf jedoch den positiven Effekt einer alleinigen Verbesserung der Kommandosprache nicht ueberschaetzen, denn die durch unklare Systemkonzepte und unsaubere interne Schnittstellen, die zum Teil von den eigenen Dienstprogrammen nicht eingehalten werden, herkommenden Probleme lassen sich leider nicht so einfach loesen. Es erscheint fraglich, ob der Aufwand zur sauberen Durchstrukturierung des Betriebssystems, wie der Hersteller sie fuer die naechsten fuenf Jahre verspricht, geringer als der fuer Entwurf und Implementierung eines voellig neuen Systems unter Beruecksichtigung des heutigen Kenntnisstandes ist.

- VAX/VMS: Die verwendeten Konzepte
  - Prozess
  - Programm
  - Datei
  - Modul

  sind strukturell klar und sauber aufgebaut, und dieser Aufbau wird in ihrer Bedienung, speziell in der Kommandosprache, dem Benennungs-Schema fuer Dateien und in den System-Schnittstellen exakt wiedergegeben.

  Das System verhaelt sich dem Benutzer gegenueber in vorhersehbarer Weise; die Kommandos tun das, was man von ihnen erwartet und zeigen keine unerwuenschten Abhaengigkeiten voneinander.

  Der Dialog mit dem Rechner kann ohne Einhaltung eines Protokolls gefuehrt werden, da die Bildschirme, die fuer Programm-Entwicklung freigegeben sind, jederzeit Eingaben akzeptieren. Bei fehlerhafter Eingabe von Kommandos werden diese mit einer Erklaerung des Fehlers zurueckgewiesen, was wegen der Kuerze der Kommandos im allgemeinen keine uebermaessige zusaetzliche Arbeit verursacht. Unvollstaendig eingegebene Kommandos werden im Dialog mit dem Benutzer vervollstaendigt.

  Fehlermeldungen des Systems erfolgen in ausfuehrlicher Form in gut verstaendlichem Englisch; bei Bedarf koennen jederzeit ueber die eingebaute HELP-Funktion zusaetzliche Informationen angefordert werden, so dass in vielen Faellen ohne schriftliche Unterlagen mit dem System gearbeitet werden kann, da es sich weitgehend selbst erklaert.

Der vorangegangene Vergleich zeigt, dass sich die VAX11 einem Dialogbenutzer gegenueber am entgegenkommendsten verhaelt, was insofern

auch nicht verwundert, da diese Maschine auf Timesharing-Einsatz hin entworfen und optimiert wurde. Dieses Bild deckt sich auch gut mit dem schon in Abschnitt 3.2.1. besprochenen Interaktions-Verhalten der beiden Prozessrechner. Waehrend die TR440 trotz einiger Inkonsistenzen dem Benutzer immer noch gute Bedienungsmoeglichkeiten bietet, versagt die Siemens-Anlage auf diesem Gebiet voellig. Der Benutzer sieht sich einem grossen, unstrukturierten Haufen von Programmen gegenueber, die ihn bei seiner Arbeit mehr behindern als ihm helfen. Da diese Situation - wie bei Siemens-Anwendern an Hochschulen allgemein bekannt - unzumutbar ist, wurden von den verschiedenen Anwendern und auch von Siemens selbst eine Reihe von Hilfsmitteln, vor allem auf der Basis von Kommandoprozeduren, geschaffen, die jedoch im wesentlichen nur das herrschende Durcheinander um weitere Nuancen bereichern. Ein Einsatz dieses Systems zur Programm-Entwicklung durch typische Benutzer eines Universitaets-Rechenzentrums ist daher mit einem solchen Aufwand verbunden, dass er in keiner Weise dem entspricht, was heute selbst billigste Minicomputer und Mikroprozessorsysteme bieten. Hier muss daher in vollem Umfang das diesbezuegliche Ergebnis der Erlanger Studie [1] bestaetigt werden, dass naemlich "der von Kleinrechnern gebotene Bedienungskomfort spuerbar bis erheblich besser" ist "als jener, den die Grossrechner anzubieten vermoegen". Speziell erfuellt die Siemens-Anlage in keiner Weise die Forderung an ein interaktives System, dass es fuer einfache Anwendungen und Routine-Arbeiten moeglich sein muss, voellig ohne Zuhilfenahme von Manuals zu arbeiten; das System verhaelt sich in dieser Hinsicht weder "well-behaved" noch "humanized" oder gar "friendly" [7]. Vom Hersteller angebotene "Teachware" kann hier zwar in Einzelfaellen helfen, doch ist ihr genereller Einsatz wegen des dazu erforderlichen Aufwandes - auch in finanzieller Hinsicht - fuer ein Universitaets-Rechenzentrum nicht moeglich; sie ersetzt also keinesfalls eine einfachere und saubere Systemstruktur.

3.3.2. Bedienungskomfort und Kontrollierbarkeit des Systems

Waehrend fuer einen neuen oder gelegentlichen Benutzer einer Rechenanlage Erlernbarkeit und Verstaendlichkeit des Systems diejenigen Faktoren sind, die die Effizienz des Arbeitens mit dieser Maschine am staerksten bestimmen, sind es fuer einen erfahrenen und an das System gewoehnten Benutzer hauptsaechlich der Bedienungskomfort und das Ausmass der Kontrolle, die er ueber seine Programmier-Umgebung ausueben kann. Diese Faktoren stehen zwar in einigem Zusammenhang mit den vorher besprochenen, doch gibt es auch Unterschiede, die eine gesonderte Behandlung rechtfertigen. Die wesentlichsten Eigenschaften eines Systems, die Komfort und Kontrollierbarkeit beeinflussen, sind die folgenden:

- Arbeitsaufwand zur Steuerung

- Umfang und Bedienbarkeit der System-Schnittstellen

- Eingriffsmoeglichkeiten

- Wahrscheinlichkeit eines Fehlverhaltens (des Benutzers oder des Systems)

- Reaktion auf auftretende Fehler

- Moeglichkeiten zur Fehlerlokalisierung und -behebung

Auch hier bestehen erhebliche Unterschiede zwischen den betrachteten Systemen.

- BS3: Die vom Betriebssystem gebotenen Moeglichkeiten zur Steuerung des Arbeitsablaufes sind im wesentlichen ueber die Kommandosprache anzusprechen. Der hierzu noetige Schreibaufwand haelt sich in vernuenftigen Grenzen, da Kommandos und Parameternamen abgekuerzt werden koennen. Da jedoch andererseits in vielen Faellen Parameter spezifiziert werden muessen, die das System eigentlich aus dem Kontext des eigenen Jobs oder durch geeignet gewaehlte Voreinstellungen (Default Values) von Werten automatisch ableiten koennte, waeren noch eine Reihe von Verbesserungen zur Arbeitsersparnis denkbar. Durch Definition eigener Kommandos, Kommando-Prozeduren und Voreinstellungen lassen sich hier fuer den einzelnen Benutzer und auch fuer die Gesamtheit aller Benutzer Arbeitserleichterungen schaffen.

  Die vom System gebotenen Schnittstellen zur Erzeugung eigener Programmier-Umgebungen sind fuer die meisten Aufgaben ausreichend, doch ist ein relativ grosser Teil dieser Dienstleistungen nur ueber den Assembler ansprechbar, was ihre Bedienbarkeit beeintraechtigt. Einen Engpass stellt hier eigentlich nur das E/A-System dar, das fuer spezielle Anwendungen (z.B. interaktive Graphik) unter Umstaenden zu restriktiv sein kann.

Wie in Abschnitt 3.2.1. festgestellt wurde, sind die Eingriffsmoeglichkeiten des Benutzers nicht als optimal zu bezeichnen, doch reichen sie fuer die meisten Anwendungen aus.

Das Fehlerverhalten der Maschine ist dagegen als gut zu bezeichnen. Die Bedienung des Systems provoziert im allgemeinen keine besonderen Fehler, und aufgetretene Fehler bleiben normalerweise in ihren Auswirkungen beschraenkt und gut lokalisierbar.

- BS2000: Die dem Benutzer gebotenen Moeglichkeiten zur Steuerung des Arbeitsablaufes sind ueber verschiedene System-Komponenten:
  - Organisationsprogramm
  - Datenverwaltungssystem
  - Zugriffsmethoden
  - Kommunikationszugriffssystem
  - Dienstprogramme (Compiler, Editoren, Linkage Editor usw.)

  verteilt, deren Arbeit vom Benutzer gesteuert und koordiniert werden muss. Der hierzu noetige Schreibaufwand ist sehr hoch, da einerseits sehr viele Kommandos und Steueranweisungen gegeben werden muessen, andererseits diese Eingaben im Schnitt sehr lang sind und nur in den seltensten Faellen abkuerzbar sind. Eine gewisse Erleichterung kann hier durch die Definition von Kommando-Prozeduren geschaffen werden, doch fuehrt dies zu einer starken Vermehrung der Dateien und damit zu langen Katalog-Zugriffen.

  Die vom System gebotenen Schnittstellen zur Erzeugung eigener Programmier-Umgebungen sind fuer einfache Aufgaben ausreichend, doch ist ein relativ grosser Teil dieser Dienstleistungen nur ueber den Assembler ansprechbar, was ihre Bedienbarkeit beeintraechtigt. Fuer kompliziertere Aufgaben, die insbesondere Arbeiten mit Dateien und dem E/A-System beinhalten, sind die gebotenen Schnittstellen unzureichend, weil sie bei geringer Leistungsfaehigkeit innere Inkompatibilitaeten aufweisen und zu ihrer Steuerung einen hoehen Aufwand erfordern - falls sie die zu erbringende Leistung ueberhaupt realisieren.

  Wie in Abschnitt 3.2.1. festgestellt wurde, sind die Eingriffsmoeglichkeiten des Benutzers sehr eingeschraenkt und dazu noch umstaendlich in ihrer Bedienung.

  Auch das Fehlerverhalten des Systems muss als schlecht bezeichnet werden; die Bedienung provoziert wegen ihrer Umstaendlichkeit Fehler des Benutzers, und bei solchen Fehlern werden wenige oder gar keine Hilfen zu ihrer Lokalisierung geboten - und wegen der im vorangegangenen Abschnitt beschriebenen Fehlerfortpflanzung kann sich eine solche Fehlerlokalisierung und -korrektur zu einer umfangreichen und komplizierten Aufgabe auswachsen. Zu diesen vom Benutzer selbst verursachten Fehlern kommen noch die zahlreichen Inkonsistenzen und Fehler im System und in den Dienstprogrammen selbst, so dass die Arbeit mit BS2000 zusaetzlich erheblich belastet wird.

- VAX/VMS: Die vom Betriebssystem gebotenen Moeglichkeiten zur Steue-

rung des Arbeitsablaufes sind im wesentlichen ueber die Kommandosprache anzusprechen. Der hierzu noetige Schreibaufwand ist wegen der Kuerze der Kommandos, der Einfachheit und Leistungsfaehigkeit der Kommandosprache, den zahlreichen Abkuerzungsmoeglichkeiten und zweckmaessig eingestellten Default Values sehr gering. Durch die Moeglichkeiten der

- Definition eigener
  - Kommandos
  - Kommando-Prozeduren
  - logischer Namen
- indirekten Parametrisierung von Kommandos
- Erzeugung und Voreinstellung von Datei-Unterkatalogen

lassen sich beliebige, einem Problem angepasste Programmier-Umgebungen erzeugen und mit minimalem Arbeitsaufwand bedienen.

Da alle Systemdienste ueber dieselbe, einheitlich festgelegte Unterprogramm-Schnittstelle bedient werden, hat jeder (dazu berechtigte) Benutzer vollen Zugang auf alle Leistungen des Systems von beliebigen Programmiersprachen aus.

In Abschnitt 3.2.1. wurde schon festgestellt, dass die dem Benutzer gebotenen Interaktionsmoeglichkeiten wesentlich groesser und besser bedienbar sind als bei den beiden Gross-Rechnern; der Benutzer hat volle Kontrolle ueber die von ihm ausgeloesten Aktivitaeten.

Das Fehlerverhalten der Maschine ist durchweg als gut zu bezeichnen. Die Bedienung des Systems provoziert im allgemeinen keine besonderen Fehler, und aufgetretene Fehler bleiben normalerweise in ihren Auswirkungen beschraenkt und gut lokalisierbar.

Auch der erfahrene Benutzer hat somit an der VAX11 die besten Arbeitsbedingungen, was sich wieder aus der Ausrichtung dieser Maschine auf Timesharing-Betrieb erklaert. Die TR440 ist in ihrer Bedienung etwas umstaendlicher, doch faellt dieser Unterschied nicht allzusehr ins Gewicht. Aus den Ergebnissen fuer die Siemens-Anlage laesst sich zwar nicht ableiten, dass die Maschine fuer interaktive Programm-Entwicklung durch erfahrene Benutzer voellig ungeeignet waere, da man diesem Benutzerkreis erheblich mehr zumuten kann als dem im vorigen Abschnitt angesprochenen; Die geschilderten Maengel bewirken jedoch, dass Programm-Entwicklung auf dieser Maschine erheblich ineffizienter und unter groesseren Schwierigkeiten ablaeuft als auf den anderen Maschinen dieses Vergleichs - was letztlich zur Folge hat, dass die Herstellung von Software auf der Siemens-Anlage unverhaeltnismaessig teuer und langwierig ist. Der Hersteller ist hier der Auffassung, dass durch die Verwendung von Kommando-Prozeduren sowohl dem ungeuebten Benutzer als auch dem erfahrenen Programmierer die Arbeit erleichtert und effizienter gemacht werden kann. Dies ist zweifelsohne richtig, doch bestehen bei den anderen hier betrachteten Systemen dieselben Moeglichkeiten - aber als zusaetzliche Leistung, die verwendet werden kann, ohne dass man auf sie in der Weise angewiesen waere wie im BS2000.

3.3.3. Dokumentation

Fuer die Arbeit mit einem Rechner ist weiterhin von Bedeutung, welche Arten von Dokumentation den Benutzern zur Verfuegung stehen. Hier sind im Prinzip vier verschiedene Ebenen der Dokumentation zu betrachten:

- einfuehrendes Material und Uebersichten
- einfache Beschreibungen fuer den gelegentlichen Benutzer
- ausfuehrliche Beschreibungen fuer den erfahrenen Programmierer
- Systemdokumentation fuer das fuer den Betriebsablauf verantwortliche Personal und Systemquellen zur Wartung der Systemsoftware

Auch hier bestehen erhebliche Unterschiede in Qualitaet und Quantitaet des verfuegbaren Materials.

BS3: Waehrend einfuehrendes Material und einfache Beschreibungen oft mangelhaft oder veraltet sind - da es sich hier um eine sterbende Maschine handelt -, sind die ausfuehrlichen Beschreibungen und die Systemdokumentation im allgemeinen sehr brauchbar und vollstaendig. Da saemtliche Systemquellen zur Verfuegung stehen, ist es, falls geeignetes Personal zur Verfuegung steht, ohne weiteres moeglich, Aenderungen, Erweiterungen und Korrekturen am System vorzunehmen. Als positiv ist ausserdem zu vermerken, dass Benutzern aller Art ein Kommando-Taschenbuch zur Verfuegung steht, das alle wesentlichen Informationen zur Bedienung des Programmiersystems durch die Kommandosprache in handlicher und uebersichtlicher Form enthaelt. Die Beschreibungen der einzelnen Programmiersprachen sind dagegen ziemlich unlesbar und unverstaendlich, und das einzige Buch, das alle fuer den Assembler-Programmierer wesentlichen Informationen enthaelt, wird seit 1971 nicht mehr aufgelegt. Es wurde durch ein Manual ersetzt, das den groessten Teil der benoetigten Information ueberhaupt nicht enthaelt; diese Information ist auch sonst nirgends mehr in einem der aktuellen Manuals aufgefuehrt.

- BS2000: Einfuehrendes Material besteht hier im wesentlichen aus fuer den Benutzer unbrauchbaren Reklameschriften. Die eigentlichen Unterlagen fuer den Benutzer bestehen, soweit vorhanden, aus einer fast unuebersehbaren Anzahl von Manuals, die einerseits in buntem Durcheinander Informationen fuer Anfaenger und Fortgeschrittene gemischt enthalten, wobei andererseits aber zusammen benoetigte Informationen oft auf mehrere Baende verteilt sind. So wird zum Beispiel in einem Band erklaert, wie irgendwelche Programme/Kommandos zu formulieren sind, waehrend die Information, wie diese Objekte dann zu bedienen sind bzw. was sie bedeuten, in einem anderen Band steht. Ein "Manualbaum" (eine graphische Uebersicht ueber die wichtigsten der verfuegbaren Manuals), der in vielen der Handbuecher enthalten ist, soll hier dem Benutzer die Orientierung erleichtern; da jedoch nur die Titel der Manuals angege-

ben sind, bietet der Manualbaum dem, der Information ueber ein bestimmtes Problem sucht, wenig Hilfe, zumal diese Titel oft in verkuerzter Form angegeben sind und daher die betreffenden Manuals nicht eindeutig identifizieren.

Ebenso stoerend ist die Tatsache, dass zur Bedienung erforderliche Informationen und Einzelheiten der Implementation, die eigentlich nur den Implementator selbst oder die Software-Wartung etwas angehen, in bunter Reihenfolge durcheinander stehen. Dieser Aufbau der Dokumentation hat leider zur Folge, dass man selbst fuer einfache Fragen mehrere Manuals gleichzeitig konsultieren und nach der relevanten Information unverhaeltnismaessig lange suchen muss, was insbesondere durch das Fehlen von Sachregistern in den meisten der Handbuecher noch unterstuetzt wird. Der Hersteller hat fuer die Manuals ab der folgenden Systemversion die Einfuehrung solcher Register angekuendigt, doch erhebt sich dabei natuerlich die Frage, warum zunaechst unvollstaendige Dokumentation ausgeliefert wird, statt die Register direkt beim erstmaligen Aufbau der Manuals mit zu erzeugen.

Es gibt zwar seit einigen Monaten ein Kommando-Taschenbuch mit Uebersichts-Tabellen, doch ist es allein nicht verwendbar, da fuer die Operanden der Befehle nur ihre Form, nicht aber ihre Bedeutung angegeben ist. Dabei waere gerade fuer Universitaets-Betrieb eine solche Kurz-Dokumentation dringend erforderlich, da die vollstaendigen Manuals so teuer sind, dass ihre Anschaffung in groeeseren Stueckzahlen - etwa fuer den Lehrbetrieb und zur Verteilung an die einzelnen Institute - voellig ausgeschlossen ist. Aus demselben Grund scheidet auch ein Einsatz der angebotenen "Teachware" in nennenswertem Umfang aus. Man muss hier natuerlich auch beruecksichtigen, dass sich viele der beschriebenen Sachverhalte auch kaum auf engem Raum darstellen lassen, da sie durch die unklaren Systemstrukturen in sich schon unuebersichtlich und kompliziert sind.

Dagegen ist die Dokumentation fuer die Systemwartung oft von erschreckender Duerftigkeit; viele benoetigte Daten sind ueberhaupt nicht dokumentiert oder in den Manuals nicht auffindbar; auch hier wirkt sich das Fehlen von Registern in den meisten der Manuals aus. Systemquellen sind nach den bisherigen Erfahrungen oft nicht ohne Schwierigkeiten zu erhalten.

Ein weiteres Problem ist hier, dass an vielen Stellen zwischen den beschriebenen und den tatsaechlich realisierten Systemleistungen Unterschiede bestehen, so dass man sich durchaus nicht darauf verlassen kann, dass sich das System so verhaelt, wie es nach der Dokumentation zu erwarten waere; selbst angegebene Bedienungsbeispiele sind zum Teil falsch oder veraltet.

- VAX/VMS: Einfuehrendes Material und Uebersichten stehen in guter Qualitaet zur Verfuegung; sie sind auch als Unterlagen fuer einfache Arbeiten verwendbar. Kernstueck der Dokumentation fuer Benutzer aller Gruppen ist ein Satz von etwa 10 ausfuehrlichen Manuals, die ziemlich alle wesentlichen Informationen in uebersichtlicher und gut verstaendlicher Form enthalten. Als negativ ist jedoch zu ver-

merken, dass diese Manuals sich zu stark auf die Programmierung in Assembler beziehen, so dass vorhandene Schnittstellen von hoeheren Programmiersprachen aus zum Teil nur bedient werden koennen, wenn man die Beschreibung geeignet uebertraegt - was aber Beherrschung des Assemblers voraussetzt. Fuer Systemwartung und -verwaltung steht eine ausfuehrliche Systemdokumentation mit Quellen auf Mikrofilm zur Verfuegung. Laut Angabe des Herstellers ist zusaetzlich Dokumentation in Form von Uebersichtstabellen als Nachschlagewerk fuer den laufenden Betrieb verfuegbar, doch konnte dieses Material nicht betrachtet werden, da es an der Test-Installation nicht vorlag.

Waehrend die Dokumentation bei TR440 und VAX11 fuer alle ueblichen Zwecke ausreicht, an einigen Stellen jedoch bei beiden Maschinen erweitert (und bei der TR440 aktualisiert) werden koennte, ist sie bei der Siemens-Anlage in vielen Teilen unzureichend und verworren. Da diese Situation untragbar ist, wurde von Siemens eine neue und vollstaendige Dokumentation fuer die naechste Version des Betriebssystems versprochen, doch nuetzt dies den jetzigen Benutzern leider nichts. Hier ist ein Hinweis auf das Vorgehen anderer EDV-Hersteller angebracht, bei denen zum Beispiel die folgende Aussage gemacht wurde: "... has always operated on the theory that a software product is complete and ready for shipment only after the documentation, as well as the operating system is tested and proven successful." (Da hier nicht untersucht wird, inwieweit dieser Hersteller entsprechend seiner Aussage verfaehrt, soll sie hier ohne Angabe der Quelle stehen, gemeint lediglich als wuenschenswertes Ziel.)

4. Zusammenfassung

Der vorangegangene Vergleich ergibt grosse Unterschiede zwischen den betrachteten Rechnern, sowohl in Bezug auf die quantitativ erfassbare Rechengeschwindigkeit und die Effizienz des von den einzelnen Compilern erzeugten Codes als auch, was die Leistungsfaehigkeit und Bedienbarkeit der zur Verfuegung stehenden Programmiersysteme betrifft. Erstaunlich ist hierbei, dass diese Leistungsunterschiede nicht zwischen Grossrechnern auf der einen Seite und Prozessrechnern auf der anderen Seite bestehen, sondern dass gute und schlechte Eigenschaften bei Rechnern aus beiden Klassen zu finden sind. Die Ergebnisse der einzelnen Abschnitte lassen sich am ehesten in folgender Weise zu einem Ueberblick zusammenfuegen:

- Rechengeschwindigkeiten: Hier ist eine gewisse Ueberlegenheit der Grossrechner gegenueber den Prozessrechnern zu verzeichnen, doch ist diese Ueberlegenheit bei weitem nicht so gross, wie aus den Kosten der Rechner erwartet werden koennte. Waehrend die beiden Prozessrechner VAX11/780 und Modcomp 7870 sowie die Siemens 7.541 etwa das gleiche Preis-Leistungs-Verhaeltnis zeigen, ist dieses Verhaeltnis fuer die beiden Grossrechner voneinander verschieden - erklaerbar durch das relativ hohe Alter der TR440 - und wesentlich schlechter als fuer die Prozessrechner. Die Gruende fuer das relativ schlechte Abschneiden der beiden Grossrechner sind der hohe interne Verwaltungaufwand, eine Architektur mit Anschlussmoeglichkeiten fuer sehr umfangreiche Datenperipherie und die teilweise ziemlich primitiven Maschineninstruktionen, die einen Overhead bedingen, der die hohe Geschwindigkeit von Prozessor und Hauptspeicher aufwiegt - bedingt im wesentlichen durch System-Architekturen, die nicht mehr dem mit heutiger LSI-Technologie erreichbaren Optimum des Preis-/Leistungs-Verhaeltnisses entsprechen. Diese Tatsache wird auch durch die Siemens 7.541 illustriert, die durch Verwendung modernerer Technologie und kompakteren Aufbau ein erheblich guenstigeres Preis-Leistungs-Verhaeltnis als die 7.760 hat, wenn auch die Moeglichkeiten dieser Technologie nicht voll ausgeschoepft werden.

- Rechengenauigkeiten: Da die hier betrachteten Prozessrechner ueber Verarbeitungsbreiten von 32 und 64 Bit verfuegen, sind sie in ihrer Rechengenauigkeit der Siemens-Anlage vergleichbar (bzw. bei einfacher Genauigkeit wegen der Verwendung anderer Datenformate sogar leicht ueberlegen). Die TR440 mit ihrer Wortlaenge von 48 Bit ist natuerlich den 32-bit-Maschinen in dieser Beziehung ueberlegen, was unter Umstaenden den Geschwindigkeitsvorsprung der Siemens 7.760 und der VAX11/780 wieder zunichte machen kann.

- Interaktivitaet: In Bezug auf Kontrollierbarkeit des Systems und Eingriffsmoeglichkeiten des Benutzers sind die Prozessrechner wegen ihrer hauptsaechlichen Anwendungsgebiete den Grossrechnern bei weitem ueberlegen. Dies liegt zum einen an Benutzerschnittstellen, die auf derartige Interaktivitaet hin ausgelegt sind, zum anderen an der Optimierung der Hardware auf schnelle Reaktion auf eintreffende Signale, was sich natuerlich auch fuer die Implementierung von Timesharing-Systemen gut einsetzen laesst. Dabei ergab die Arbeit mit

dem System VAX/VMS, dass sich diese hohe Interaktivitaet durchaus mit voelligem Schutz des Systems gegenueber Fehlbedienungen durch den Benutzer verbinden laesst - sofern dieser keine Berechtigungen hat, die ihm den Abschuss des Systems erlauben.

- Programmiersystem: Die dem Benutzer gebotenen Moeglichkeiten zur Erstellung und zum Austesten von Software sind auf allen Maschinen gegeben, doch ist die Qualitaet der vorhandenen Schnittstellen sehr unterschiedlich, wobei auch hier die Prozessrechner wenigstens ebenso gut abschneiden wie die Grossrechner. Betrachtet man die Komponenten des Programmiersystems im Einzelnen, so stellt man sogar fest, dass die zur Verfuegung stehenden Mittel auf der VAX11 am zahlreichsten und am besten bedienbar sind, waehrend die Siemens-Anlage einen Wust schlecht strukturierter, umstaendlich zu bedienender und oft zueinander inkompatibler Subsysteme praesentiert und damit in der Qualitaet ihres Programmiersystems weit unter allen anderen Rechnern in diesem Vergleich liegt.

Diese Ergebnisse lassen sich dahingehend interpretieren, dass die Leistungsfaehigkeit moderner Prozessrechner der 32-bit-Klasse durchaus mit der von Grossrechenanlagen vergleichbar ist. Die Unterschiede in den Verarbeitungskapazitaeten zwischen diesen beiden Rechnergruppen koennen weitgehend durch das erheblich guenstigere Preis-Leistungs-Verhaeltnis der Prozessrechner aufgefangen werden, das bei gleichem finanziellen Aufwand Anschaffung und Betrieb eines Netzes solcher "Midi"-Rechner anstelle eines Grossrechners erlaubt. Der Einsatz von Grossrechnern ist daher eigentlich nur noch an den Stellen gerechtfertigt, an denen explizit ihre hohe Verarbeitungskapazitaet erforderlich ist, etwa zum Betrieb sehr grosser Datenbank-Systeme oder zur sehr schnellen Auswertung oder Bearbeitung grosser Datenbestaende. Bei dieser Argumentation wird die Anschlusskapazitaet fuer Peripherie nicht beruecksichtigt; diese ist bei den Grossrechnern im Maximalausbau hoeher als bei den Prozessrechnern, doch sind heute auch voll ausgebaute Prozessrechner-Systeme durchaus mit mittleren Grossrechnern vergleichbar.

Diese Ergbenisse sind durchaus nicht ueberraschend; so wurde schon 1975 von Stone [8] vorausgesagt, dass durch die zunehmende Verwendung hochintegrierter Schaltungen Minicomputer eine Leistungsfaehigkeit erreichen werden, die der der traditionellen Grossrechner weitgehend entspricht. Die Ergebnisse der hier vorliegenden Untersuchung sowie auch der schon mehrfach zitierten Studie [1] lassen den Schluss zu, dass mit der Verfuegbarkeit von 32-bit-Prozessrechnern die von Stone angekuendigte Entwicklung realisiert ist. Da diese Prozessrechner - bedingt durch ihr traditionelles Einsatzgebiet - gleichzeitig ueber ein sehr schnelles und flexibles Interrupt-System verfuegen muessen, bieten sie gleichzeitig besonders im E/A-System Dienstleistungen, die bei Grossrechnern nicht realisierbar sind.

Fuer die Verwendbarkeit der vorhandenen Benutzerschnittstellen und Programmiersysteme ist im wesentlichen nur die Modernitaet der zugrundeliegenden Betriebssyteme massgebend. Hier schneidet die TR440 mit BS3 recht gut ab, da in diesem Betriebssystem viele spaetere Entwicklungen schon vorweggenommen wurden, teilweise allerdings leider nur in Ansaetzen, die durch umstaendliche oder schlechte Implementie-

rungen wieder verschleiert wurden. Die Betriebssysteme der beiden Prozessrechner entsprechen dem heutigen technischen Standard; sie bieten dem Benutzer sehr weitgehende Arbeitsmoeglichkeiten und gute Bedienbarkeit. Bei der Entwicklung dieser Systeme wurden offenbar die in den letzten zehn bis zwoelf Jahren beim Entwurf moderner Betriebssysteme [9],[4],[10],[11],[5],[12] gemachten Erfahrungen verwertet und eingearbeitet. Dies laesst sich leider von BS2000, dem Betriebssystem von Siemens, keinesfalls sagen; durch die Fixierung auf IBM und das Festhalten an fast eineinhalb Jahrzehnte alten, ueberholten Systemkonzepten entspricht dieses System in keiner Weise den heutigen technischen Moeglichkeiten und weitgehend noch nicht einmal relativ bescheidenen Anforderungen einer Timesharing-Umgebung. Besondere Engpaesse struktureller Art stellen hier das Datei-System, das weder den Anforderungen noch den technischen Moeglichkeiten [13] entspricht, sowie die chaotische Bedienung durch die Kommandos und Steueranweisungen fuer Dienstprogramme dar. Zumindest hier ist es Siemens in vollem Umfang gelungen, zu IBM aequivalente Software anzubieten. "The best-known job control language (called JCL), which was introduced with IBM's System/360, is infamous for its almost total lack of pragmatic considerations. Untold hours of effort have been spent trying to teach JCL to users and devising means of permitting them to get their work done in spite of it." [14]

Fuer die praktische Arbeit mit den einzelnen Systemen zeigt sich, dass das Verhalten und die Verfuegbarkeit der Benutzerschnittstellen einen erheblich groesseren Einfluss auf die Verwendbarkeit der Rechner fuer Entwicklung von Software haben als alle anderen Faktoren zusammen. Selbst die oft gerne diskutierte Frage, ob ein System virtuell ist oder nicht, spielt in diesem Zusammenhang nur eine sehr untergeordnete Rolle, da fuer die Programm-Entwicklung im wesentlichen nur die Groesse des verfuegbaren Adressraumes und nicht dessen physikalische Realisierung [15] interessiert - und dieser ist fuer die TR440 bei Verwendung von 22-bit-Adressierung und fuer die Siemens-Anlage vergleichbar gross; hier faellt lediglich die VAX11 mit 4 Gigabyte Adressraum aus dem Rahmen.

Unter Beruecksichtigung aller Faktoren ergibt sich, dass die Entwicklung von Software am einfachsten und muehelosesten auf der VAX11 geschieht, waehrend die TR440 hier zum Teil schlechter abschneidet. Programm-Entwicklung unter BS2000 auf der Siemens-Anlage erfordert dagegen erheblichen Aufwand und ist nur sehr umstaendlich zu handhaben, so dass mit diesem Betriebssystem ausgeruestete Rechner als Universitaets-Timesharing-Rechner ausgesprochen schlecht geeignet sind - speziell im Vergleich zu den heute bei Kleinrechnern gebotenen Dienstleistungen. Dabei spielt es keine Rolle, dass viele Software-Probleme mit erheblicher Muehe dennoch auf dieser Maschine geloest werden koennen; diese Muehe ist Verschwendung von Geld, Zeit und Arbeit, wenn es System-Architekturen gibt, die es gestatten, auf sie zu verzichten. Es muss an dieser Stelle ganz deutlich klargestellt werden, dass es nicht die Aufgabe einer Universitaet ist, Software zu erzeugen, die auf dem Markt schon erhaeltlich ist, also das Rad noch einmal erfinden zu muessen, nur weil diese Software auf einem bestimmten Rechner nicht oder nur schlecht vorhanden ist; speziell ist es also fuer Universitaets-Betrieb unzumutbar, zur Verbesserung einer Benutzer-Schnittstelle gezwungen zu sein. Dies ist Aufgabe des Herstellers,

der personell und vom Arbeitsbereich dafuer viel besser geeignet ist.

Will man die hier gesammelten Daten zur Auswahl eines geeigneten Nachfolgerechners fuer die TR440 verwenden, so ist natuerlich zu bedenken, dass die hier betrachteten System-Aspekte nur einen speziellen Einsatzbereich abdecken, auf den sich die genannten Resultate beziehen. Ohne dass die Anwendungsspektren fuer die an den einzelnen Hochschulen installierten Rechner bekannt sind und ohne dass die wesentlichen auf dem Markt befindlichen Rechner auf diese Anwendungen hin untersucht wurden, lassen sich keine Aussagen ueber sinnvolle Nachfolge-Systeme fuer die TR440-Installationen machen. Dazu waere es zwingend notwendig, die hier beschriebene Untersuchung in erheblich groesserem Rahmen durchzufuehren, und zwar sowohl in Bezug auf die betrachteten Rechner-Typen als auch auf die untersuchten Faktoren, die hier rein aus der Sicht des Software-Entwicklers ausgewaehlt wurden. Erst eine solche vollstaendige Untersuchung koennte Aussagen darueber zulassen, welche Rechner in den naechsten Jahren sinvollerweise zu beschaffen waeren. Andererseits ist gerade bei der derzeitigen Dynamik des Rechnermarktes und in Anbetracht der durch 32-bit-Prozessrechner (wie Data General Eclipse/MV8000, DEC VAX11/780, Norsk Data Nord 500/100 und PRIME 750) repraesentierten neuen Rechner-Generation die Durchfuehrung eines solchen allgemeinen Rechnervergleichs dringend erforderlich, da fuer eine sachorientierte Beschaffung von Universitaetsrechnern keine sicheren Beurteilungskriterien vorhanden sind, so dass die Chancen fuer eine zufaellig getroffene optimale oder auch nur akzeptable Entscheidung sehr gering sind.

Aus der vorliegenden Untersuchung geht hervor, dass es fatal ist, sich ohne Kenntnis des derzeitigen technischen Standes auf einen Rechner a priori festzulegen, und dass Prozessrechner der oberen Leistungsklasse inzwischen durchaus in die Auswahl der betrachteten Rechner einbezogen werden muessen. Die aus diesen Ergebnissen zu ziehenden Konsequenzen - abgesehen von der Notwendigkeit einer ausfuehrlicheren Untersuchung - werden im abschliessenden Kapitel kurz dargestellt, zusammen mit einem Modell fuer die Belieferung eines Universitaets-Betriebes mit Rechenleistung unter Beruecksichtigung des derzeitigen technischen Standes.

5. Konsequenzen

Aus den Ergebnissen des hier beschriebenen Vergleichs einiger Rechner ergeben sich verschiedene Konsequenzen, je nachdem ob man eine Optimierung der momentanen Betriebssituation eines bestimmten Universitaets-Rechenzentrums oder eine allgemeine langfristige Strategie zur optimalen Bereitstellung von Rechenleistung im Universitaetsbereich im Auge hat.

Fuer den Parallelbetrieb von TR440 und Siemens 7.760 empfiehlt sich wegen der Ergebnisse des Vergleichs folgendes Vorgehen:

- Da die Leistungen und Schnittstellen des Siemens-Programmiersystems erheblich schlechter sind als die der TR440, sollte nur diejenige Software-Entwicklung auf der Siemens-Anlage durchgefuehrt werden, die unumgaenglich notwendig ist, und zwar weitgehend durch erfahrenes Personal. Dies hat zwar zur Folge, dass die Erstellung dieser Software erheblich teurer wird als auf anderen Rechnern, doch koennen die zusaetzlichen Kosten und die Blockierung der betroffenen Programmierer durch das System bei starker Einschraenkung der Software.Entwicklung auf der Siemens-Anlage moeglicherweise noch in vertretbarem Rahmen gehalten werden.

- Ein Einsatz der Siemens-Anlage im Lehrbetrieb und zu Softwaree-Entwicklung durch fachfremde oder Gelegenheits-Programmierer ist nicht empfehlenswert, da der Aufwand zum Erlernen der System-Bedienung in keinem Verhaeltnis zu dem zu erwartenden Ergebnis steht. Dieser Aufwand wird noch durch die schlechte Dokumentation erhoeht, da weder von einem EDV-Anfaenger noch von einem Nicht-Informatiker erwartet werden kann, dass er sich durch die Siemens-Dokumentation durchfindet bzw. zufaellig gefundene Daten versteht. Dies hat dann zur Folge, dass die Arbeitskapazitaet derjenigen, die sich mit dem System auskennen, durch andauernde Rueckfragen und Hilferufe von Benutzern blockiert wird - ein Effekt, der sich trotz des bisher erst geringen Einsatzes des BS2000-Systems schon bemerkbar gemacht hat. Aus den hier genannten Gruenden sollte daher dieser Teil des Rechenzentrums-Betriebes moeglichst solange auf der TR440 verbleiben, bis entweder BS2000 durch ein besseres und moderneres Betriebssystem abgeloest wird oder bis die Anschaffung eines echten Timesharing-Rechners fuer diese Aufgaben moeglich ist.

- Dagegen laesst sich der Betrieb auf dem TR440 dadurch entlasten, dass die Siemens-Anlage weitgehend alle groesseren Produktionslaeufe uebernimmt. Huer spielen die Maengel des Programmiersystems nur bei der Installation dieser Programme eine Rolle, waehrend sich beim eingentlichen Betrieb im wesentlichen nur die Charakteristike der Prod gramme selbst auswirken. Durch die gegenurber der TR440 hoehere Verarbeitungsgeschwindigkeit der Siemens-Anlage sowie den groesseren Speicheraufbau lassen sich diese Jobs auf der Siemens-Maschine effizienter durchfuehren, waehrend gleichzeitig die TR440 nicht unerheblich entlastet werden kann. Speziell groessere Software-Systeme wie etwa SPSS und rechenintensive Batch-Jobs (z.B. der Universitaets-Verwaltung und der Theoretischen Physik) sollten daher moeglichst

30. 6.80
We

bald auf die Siemens-Anlage uebernommen werden, waehrend interaktive Jobs und wenig oder nur einmal verwendete Programme besser auf der TR440 bleiben.

Diese Ergebnisse entsprechen durchaus dem, was man ganz naiv erwarten koennte: Die Siemens-Anlage als im Grunde kommerzieller Rechner ist am besten fuer solche Anwendungen einsetzbar, deren Aufgabenstruktur den Dienstleistungen eines kommerziellen Rechenzentrums am aehnlichsten ist, und dies sind genau die Anwendungen, bei denen einmal erstellte Programme sehr oft gerechnet werden, und dies moeglichst in der Form von Batch-Jobs. Ein Einsatz dieses Systems fuer Anwendungen hoher Interaktivitaet, einmaliger Programm-Benutzung und Lehrbetrieb wird zu Ineffizienz des Betriebs und Frustration der Rechenzentrums-Mitarbeiter fuehren.

Langfristig waere dagegen eine Rechenzentrums-Struktur anzustreben, die auch Programm-Entwicklung und Lehrbetrieb besser unterstuetzt, als dies unter Verwendung der bisherigen Systeme moeglich waere. Dazu waere anzustreben, dass moeglichst an allen Stellen im Universitaetsbereich, die Rechenleistung benoetigen, diese vor Ort zur Verfuegung gestellt werden koennte, wobei diese Rechenleistung eine weitgehend selbsterklaerende Form haben und auch fuer unerfahrene Benutzer ohne grossen Lernaufwand bedienbar sein sollte. Dieses Ziel waere durch den Einsatz von Timesharing-Systemen mit guten Benutzer-Schnittstellen und die Verwendung programmierter Instruktion fuer die Erlernung der Rechnerbedienung durchaus realisierbar und ist in manchen Systemen sogar schon realisiert [3].

Durch Einsatz solcher Systeme in den einzelnen Fachbereichen lassen sich deren spezielle Beduerfnisse weit besser und kostenguenstiger befriedigen, als dies mit einem zentralen Grossrechner moeglich ist, der alle Benutzer etwa gleich schlecht bedient. Dies kann auch durch Einsatz von Rechnern, die fuer ein bestimmtes Anwenderprofil besonders guenstig sind, in den Bereichen der Universitaet geschehen, deren Anwendungstruktur bekannt oder einfach zu definieren ist.

Fuer einen oekonomischen Betrieb eines derartigen Rechnernetzes sind auch noch einige andere Ueberlegungen von Wichtigkeit, die auch in der einen oder anderen Weise Auswirkungen auf den physikalische Aufbau des Netzes haben werden:

- Mehrere Argumente sprechen fuer eine zentrale Aufstellung der einzelnen Rechner, wenn auch die Verarbeitungsleistung dezentral angeboten wird:

  - Die Rechner, zumindest jedoch die Plattenspeicher, benoetigen klimatisierte Unterbringung, die oft durch eine zentrale Klimaanlage billiger und leistungsfaehiger zur Verfuegung gestellt werden kann als durch nachtraeglichen Umbau nicht fuer Rechnerbetrieb vorgesehener Raeume.

  - Operating und Wartung koennen gemeinsam von qualifiziertem Personal durchgefuehrt werden, das oft in den einzelnen Instituten fehlt und auch zu teuer ist. Bei moderneren Maschinen besteht das Operating oft nur noch aus dem Auflegen von Datentraegern und aus

Papierabreissen, das auch von Benutzern durchgefuehrt werden kann - sofern die Bestimmungen des Datenschutze keinen "Closed-Shop-" Betrieb erfordern -, so dass dieser Punkt in Zukunft eine geringere Rolle spielen duerfte als heute. Das Problem der Hardware- und auch Software-Wartung dagegen wird noch solange zentrale Aufstellung der Rechner zweckmaessig sein lassen, wie Fernwartung und Selbstkonfigurierung nicht zur Standard-Ausruestung der Rechner gehoeren.

- In diesem Zusammenhang ist auch die Erstellung und Anpassung spezieller Grund-Software und besonderer Anwender-Software zu nennen, die im allgemeinen nur von qualifizertem Personal vorgenommen werden kann, wenn es - wie nur zu oft der Fall - nicht moeglich ist, diese Software fertig fuer den geplanten Einsatzbereich vom Hersteller zu beziehen. Falls zur Erstellung dieser Software tiefe Eingriffe in das System vorgenommen werden muessen, was insbesondere bei der Anbindung an ein heterogenes Netz meist noetig ist, empfiehlt es sich, diese Aufgaben ebenfalls zentral durchzufuehren, da auch hierzu vielen Instituten das erforderliche Personal fehlt und eine Einstellung von Informatikern eigens zur Betreuung eines Institutsrechners meist weder moeglich noch vertretbar ist.

- Teure und empfindliche Geraete sollten ebenfalls zentral aufgestellt bleiben, um Bedienung und Wartung zu vereinfachen und die Gefahr einer Beschaedigung zu verringern. Falls diese Geraete in Form eines Geraete-Verbundes aus dem ganzen Netz bedient werden koennen, stellt diese Zentralisierung kaum eine Einbusse an Funtionalitaet dar, bietet jedoch eine erheblich kostenguenstigere Loesung als Mehrfachanschaffung und ungeschuetzte Aufstellung; gleichzeitig wird die Verfuegbarkeit und Auslastung dieser Geraete hoeher. Geraete, die hier zu nennen sind, waeren zum Beispiel Schnelldrucker (abgesehen von Billigstgeraeten), Mikrofilm-Ausgaben, teure Graphik-Geraete oder Peripherie-Speicher sehr hoher Kapazitaet.

- Gegen die zentrale Aufstellung der Dialogrechner spricht, dass man von diesen Rechnern zu jedem vor Ort angeschlossenen Geraet eigene Leitungen benoetigt, was nur bei geringer Leitungslaenge und niedrigen Uebertragungsgeschwindigkeiten ohne grossen Aufwand realisierbar ist. Innerhalb eines nicht zu ausgedehnten Universitaetsgelaendes koennen fuer Terminal-Anschluesse oft Linienstrom-Leitungen verwendet werden, doch kann dies bei schon vorhandenen Leitungen zu geringen Querschnitts eine Quelle von Stoerungen sein. Anschluesse hoeherer Geschwindigkeit koennen ueber Modems bedient werden, doch ist deren Anzahl aus Kostengruenden beschraenkt. Konzentratoren, die die Anzahl der Leitungen aus einem zentralen Rechenzentrum hinaus verringern, sind meist problematisch, da sie die Interaktivitaet der Terminal-Schnittstelle oft negativ beeinflussen.

Ein Vergleich dieser Argumente fuer und gegen eine zentrale Aufstellung der einzelnen Rechner und ihrer Peripherie zeigt, dass die fuer eine bestimmte Universitaet zu waehlende Verteilung der Rechenleistung in hohem Masse von den oertlichen Gegebenheiten abhaengt, so dass hier keine allgemeingueltigen Aussagen gemacht werden koennen.

Die logische Struktur eines solchen Rechnernetzes bleibt von diesen Ueberlegungen natuerlich unberuehrt; man erhaelt auch dann Rechenleistung "vor Ort", wenn der Rechner selbst weit entfernt ist und nur ueber eine leistungsfaehige Schnittstelle bedient wird.

Abgesehen von diesen rein praktischen Ueberlegungen, die eher fuer ein relativ heterogenes Netz mit hoher Verarbeitungsleistung "vor Ort" sprechen, gibt es noch ein allgemeineres Argument gegen die Konzentration zu grosser Benutzer-Anzahlen in einem einzigen Timesharing-System: Die Anzahl der Benutzer, die ein solches System ohne Gefahr der Ueberlast bedienen kann - der sogenannte Saettigungspunkt des Systems -, ist proportional zur CPU-Auslastung und zum Verhaeltnis zwischen dem mittleren zeitlichen Abstand zwischen zwei Eingaben eines Benutzers und der mittleren Rechenzeit fuer eine (kurze) Eingabe [16],[17]. Mit ueblichen Werten fuer diese Groessen erhaelt man Werte um etwa 30 fuer diesen Saettigungspunkt, die sich - bei gleichbleibendem Benutzerverhalten - nur durch Erhoehung der Rechengeschwindigkeit und effizientere Interruptbehandlung vergroesseren lassen. Diese Vorgehensweise ist sowohl nach oben begrenzt als auch ab einer bestimmten Rechnergeschwindigkeit wegen der rapide ansteigenden Hardware-Kosten unwirtschaftlich, da ab einer bestimmten Grenze, die im wesentlichen durch die verwendete Technologie bestimmt wird, die Kosten fuer eine Beschleunigung der Zentraleinheit ohne Aenderungen der Architektur sehr stark anwachsen [18],[19]. Grosse Anzahlen interaktiver Benutzer sind daher unter den derzeitigen technischen Gegebenheiten am effizientesten ueber mehrere mittelgrosse Timesharing-Systeme (ca. 30 Benutzer je Rechner mit etwa 1 bis 2 Megabyte Hauptspeicher) zu bedienen.

Wie der hier beschriebene System-Vergleich gezeigt hat, existieren gerade fuer Rechner in dieser Groessenordnung moderne Betriebssysteme, die sich in dieser Hinsicht erheblich benutzerfreundlicher verhalten als die traditionellen Grossrechner-Betriebssysteme. Um hier eine sinnvolle Auswahl treffen zu koennen, waere eine genaue Untersuchung der vorhandenen Anwenderprofile im Hinblick auf
- Anteil erfahrener/unerfahrener Benutzer
- Verhaeltnis interaktiver/stapelorientierter Last
- Verhaeltnis von Programm-Entwicklung und Produktionslaeufen
- gesamte zu erbringende Rechenleistung
- Umfang der online benoetigten Datenbestaende

durchzufuehren. Parallel dazu muessten die auf dem Markt befindlichen Systeme daraufhin untersucht werden, wieweit sie den Anforderungen der so parametrisierten Anwenderprofile Genuege tun. Ausserdem muessten Daten ueber den Aufwand zur Bedienung dieser Rechner auf der Ebene des Systemverwalters und des Operating sowie ueber die Zuverlaessigkeit ihrer Hardware und Software durchgefuehrt werden - Aspekte, die in der vorliegenden Untersuchung bewusst weitgehend ausgeklammert wurden.

Es ist zu erwarten, dass sich dabei fuer die einzelnen Hochschulen durchaus verschiedene Strukturen der optimal zu verwendenden Rechnersysteme ergeben werden, je nach der Form der vorhandenen Anwenderprofile. Diese Situation wird noch dadurch verkompliziert, dass die Anschaffung eines Systems, das bestimmte Anwendungsformen unterstuetzt, zu einer Zunahme eben dieser Anwendungen fuehren und damit das ur-

spruenglich vorhandene Profil veraendern kann, waehrend umgekehrt die Anschaffung eines Systems, das fuer bestimmte Arbeiten ungeeignet ist, ebenfalls zu Rueckwirkungen auf der Seite der Benutzer fuehren kann, bis hin zur Abwanderung oder zur Anschaffung eigener Systeme.

Hier ist insbesondere zu bedenken, dass gerade die rasante Entwicklung auf dem Gebiet der Mikroprozessor-Technologie eine Verschiebung dieser Grenze nach unten erwarten laesst. Die Entwicklung leistungsfaehiger Mikroprozessoren (wie etwa Intel 8088, Motorola 68000, Zilog Z8000) sowie das zunehmende Angebot an software-kompatiblen LSI-Versionen groesserer Prozessrechner wird in den naechsten Jahren zur Verfuegbarkeit sehr billiger und leistungsfaehiger Timesharing-Systeme kleiner Kapazitaet und von Einbenutzer-Systemen fuehren, die eine ideale Ergaenzung eines Rechnernetzes der hier beschriebenen Struktur sein koennen.

Fuer das im vorangehenden Bericht favorisierte Anwendungsprofil mit einem hohen Interaktivitaetsanteil und Einsatz des Rechners im Lehrbetrieb und fuer EDV-fremde Benutzer soll nun abschliessend ein Rechnersystem beschrieben werden, das - unter Beruecksichtigung der derzeitigen technischen Moeglichkeiten - die benoetigten Leistungen erbringen koennte.

Die Ergebnisse des Vergleichs zeigen deutlich, dass die geforderte hohe Interaktivitaet von den betrachteten Grossrechnern nicht erbracht wird. Darueberhinaus waere der dazu noetige Aufwand auch sehr unwirtschaftlich, da ein Grossteil der teuren Grossrechnerleistung fuer letztlich unproduktive interne Verwaltung des Rechners selbst verwendet werden muesste. Die Arbeit an der TR440 zeigt hier genau, dass bei zunehmender Last die Grenzen der Grossrechner sehr schnell erreicht werden, ohne dass dabei die Verarbeitungskapazitaet des Rechners wirklich ausgenutzt wird, was die weiter oben gemachten allgemeinen Aussagen genau bestaetigt.

Das traditionelle Vorgehen zur Entlastung des Grossrechners besteht hier im Einsatz billiger, aber primitiver Front-End-Rechner, die zumindest die physikalische Steuerung des Terminalnetzes und moeglicherweise auch einfachste Editierfunktionen uebernehmen koennen. Dafuer muss jedoch eine geringere Interaktivitaet in Kauf genommen werden, da der Benutzer mit dem eigentlichen Rechner nur noch ueber ein dazwischengeschaltetes Protokoll verkehrt, was zu niedrigeren Kommunikationsgeschwindigkeiten und eventuell sehr starken Einschraenkungen der Kommunikationsmoeglichkeiten fuehren kann. Die Entlastung des Hauptrechners bleibt jedoch gering, da die einzelnen interaktiven Jobs immer noch sehr oft ausgelagert werden muessen, was nicht unerheblichen Aufwand mit sich bringt.

Die verhaeltnismaessig hohe Leistung, die inzwischen bei 32-bit-Prozessrechnern erreicht ist bei gleichzeitig relativ guenstigen Preisen dieser Rechner, zeigt, dass inzwischen eine andere Vorgehensweise realisierbar und wirtschaftlich ist, die die genannten Maengel der Front-End-Loesung nicht aufweist. Diese Loesung besteht darin, den interaktiven Betrieb und speziell die Programm-Entwicklung von mehreren zu einem Verbund zusammengeschalteten Prozessrechnern durchfuehren zu lassen, von denen jeder eine bestimmte, jedoch nicht zu grosse Anzahl

von Terminals bedient. Die lokale Datenhaltung der Terminals eines dieser Teilnetze waere dann ebenfalls auf dem zugeordneten Rechner durchzufuehren, so dass der groesste Teil der Arbeit der einzelnen Benutzer ablaufen koennte, ohne dass dazu die einzelnen Rechner miteinander kommunizieren muessten. Die auf diesen lokalen Rechnern zur Verfuegung stehenden Moeglichkeiten muessten - und koennten - alle wesentlichen Komponenten zur Programmentwicklung und zur Datenhaltung nicht zu umfangreicher Datenbestaende (einige 100 Megabyte) umfassen. Speziell muessten alle notwendigen Compiler und Dienstprogramme auf den lokalen Rechnern verfuegbar sein.

Programm-Entwicklung und nicht zu umfangreiche Produktionslaeufe koennen in einem solchen System lokal durchgefuehrt werden, was bei geeigneter Auswahl der lokalen Rechner zu erheblicher Effizienzsteigerung der Software-Entwicklung gegenueber der Loesung mit Grossrechner und Front-End-Prozessor fuehrt. Die bei der Durchfuehrung des hier beschriebenen Vergleichs gemachten Erfahrungen lassen eine Effizienzsteigerung um den Faktor 2 bis 5 durchaus realistisch erscheinen, was einer Kostenreduktion der Software-Erstellung um denselben Faktor entspricht.

Rechenintensive Produktionslaeufe und Arbeit mit sehr grossen Datenbestaenden (im Gigabyte-Bereich) koennten in einem solchen Netz auf verschiedene Art eingebaut werden, wobei die jeweils wirtschaftlichste Form vom Bedarf an solchen Anwendungen abhaengt. Bei kleinem bis mittlerem Aufkommen an solchen Jobs duerfte es am guenstigsten sein, entweder ein besonders gut ausgebautes Prozessrechner-System ausschliesslich fuer diese Aufgaben zu reservieren und ueber eine RJE-Schnittstelle zu versorgen oder diese Aufgaben ueber eine Verbindung mit einem Grossrechner in ein fremdes Rechenzentrum auszulagern. Bei groesserem Aufkommen an solchen rechenintensiven Jobs kann es sich dagegen lohnen, hierfuer einen eigenen Grossrechner zu installieren, der dann aber nur mit solchen Jobs versorgt wird. Dieser Grossrechner waere dann zweckmaessigerweise ueber eine RJE-Schnittstelle als Back-End-Rechner an das Netz anzuschliessen und haette keine interaktive Belastung zu tragen, was eine Optimierung auf Batch-Betrieb hin und hohe Rechenleistung zulaesst. Daher kann ein solcher Back-End-Prozessor einfacher und billiger als ein Grossrechner, der zusaetzlich mehr schlecht als recht ein Terminalnetz treibt, realisiert werden, so dass die Gesamtkosten einer solchen Konfiguration durchaus nicht hoeher als die der traditionellen Front-End-Loesung sein muessen - bei wesentlich hoeherer Leistungsfaehigkeit.

Verwendet man verschiedene Rechnertypen als Netz- und Back-End-Rechner, so entsteht dann allerdings das Problem der Kompatibilitaet zwischen diesen Rechnertypen. In einer ersten Ausbaustufe muesste Programm-Entwicklung fuer den Back-End-Rechner unter Beruecksichtigung seines Programmiersystems auf den lokalen Rechnern des Netzes getrieben werden. Die eigentlichen Produktions-Jobs muessten dann jedoch auf dem Back-End-Rechner noch einmal erzeugt werden, was Schwierigkeiten mit sich bringen kann und daher keine Loesung von Dauer ist. In einer zweiten Ausbaustufe waere daher eine Angleichung von Netz- und Back-End-Rechnern zu erzielen, was durch Bereitstellung aequivalenter Compiler und Datei-Systeme auf beiden Maschinentypen geschehen kann. Als endgueltige Loesung waere natuerlich totale Kom-

patibilitaet anzustreben, so dass lokal erzeugte Programme direkt binaer auf den Back-End-Rechner uebertragen und dort ausgefuehrt werden koennen - entweder unter Verwendung von Cross-Compilation im Netz oder ueber totale Angleichung der Befehlssaetze beider Maschinentypen. Dabei ist es klar, dass dieses Kompatibilitaetsproblem noch erheblich schwieriger zu loesen ist, wenn das Rechner-Netz heterogen mit unterschiedlichen Optimierungszielen fuer einzelne Teilbereiche aufgebaut wurde; man muesste hier jedoch klaeren, wie hoch tatsaechlich der Bedarf an Dienstleistungen ist, die ueber die lokalen Standardanforderungen hinausgehen.

Abschliessend erhebt sich die Frage, welche Rechnertypen als Netz-Rechner und welche als Back-End-Rechner zu verwenden waeren. Die Ergebnisse dieses Berichtes zeigen nur, dass inzwischen als Netz-Rechner geeignete Typen zur Verfuegung stehen, dass also ein Netz der beschriebenen Struktur ueberhaupt realisierbar ist. Um zu Aussagen ueber eine leistungsfaehige und gleichzeitig wirtschaftliche Realisierung eines solchen Netzes zu kommen, muesste eine groessere Anzahl von Prozessrechnern betrachtet werden, die ueber folgende Eigenschaften verfuegen sollten:

- Verarbeitungsbreite wenigstens 32 Bit

- Gleitkomma-Arithmetik

- vektorisierte Mehrebenen-Interrupts

- Context-Switching im Mikrosekunden-Bereich

- leistungsfaehiges Programmiersystem mit:
  - programmierbarer Kommandosprache hoher Ebene
  - gut verstaendlichen System-Meldungen und Informationsdiensten
  - Editoren
  - Compiler
  - interaktiven Testhilfen
  - Datenverwaltung

- einheitliche Schnittstellen:
  - zum Betriebssystem
  - zu Datei-System und Geraete-Steuerung
  - zwischen Unterprogrammen aller Programmiersprachen

- parametrisierbare Terminalsteuerung mit:
  - Steuerung auf der Ebene einzelner Character
  - Moeglichkeit kontinuierlicher Eingabe
  - logischem Voll-Duplex-Betrieb
  - Anschluss-Moeglichkeiten fuer Asynchrongeraete beliebigen Typs
  - weitgehenden Eingriffsmoeglichkeiten in laufende Programme
  - flexibler Steuerung von Ausgaben ueber den Bildschirm (z.B. ueber XON/XOFF- oder ACK/ETX-Protokoll (die von einer Reihe von Terminals hardware-maessig unterstuetzt werden)

Rechner mit den genannten Charakteristika werden zur Zeit schon von etwa einem halben Dutzend verschiedener Hersteller angeboten, und der Preisverfall auf diesem speziellen Markt laesst erwarten, dass in den

naechsten zwei Jahren Rechner dieser Leistungsklasse weniger als eine halbe Million DM kosten werden, so dass ein Netz der oben beschriebenen Art billiger als ein heutiger Grossrechner sein koennte.

Als Back-End-Rechner waere eine Maschine wie die Siemens 7.760 - fuer geringeren Bedarf an Back-End-Leistung auch einer der Siemens-Kompaktrechner der Serie 7.500 - durchaus verwendbar, doch muesste dann ueberlegt werden, ob Aenderungen zur Erhoehung der Kompatibilitaet des Netzes im Netz oder in der Back-End-Maschine durchzufuehren waeren. Die Ergebnisse des Rechnervergleichs lassen hier eher eine Aenderung des Back-End-Systems fuer sinnvoll erscheinen.

Ein System der hier beschriebenen Art waere mit heutigen Mitteln ohne Schwierigkeiten realisierbar und koennte ein wirtschaftliches und leistungsfaehiges Nachfolge-System fuer die TR440 werden. Eine Definition einer Benutzer-Schnittstelle fuer ein solches System, die als Entwurfsgrundlage fuer einen Top-Down-Entwurf der oberen Ebenen des Betriebssystems eines Knotenrechners in einem solchen Universitaetsnetz dienen koennte, wurde in Folge dieses Rechnervergleichs erarbeitet; sie koennte auch als Grundlage fuer einen Katalog von Kriterien zur Begutachtung der Benutzerschnittstellen von Rechnern allgemein verwendet werden [20].

6. Literatur

Fuer die vorangegangene Untersuchung wurde, soweit vorhanden und erreichbar, Dokumentation zu den Betriebssystemen BS3, BS2000, VMS und MAX IV verwendet. Zusatezlich zu diesen Schriften der einzelnen Hersteller, deren Aufzaehlung sich hier eruebrigt, wurde noch auf die folgenden Artikel von allgemeinerer Bedeutung Bezug genommen:

[1] D. Schrammel, W. Bayer, H.-P. Gliemann, M. Heigl, P. Mertens: Ein Leistungs- und Wirtschaftlichkeitsvergleich zwischen Klein- und Grosscomputern; Arbeitsbericht des Instituts fuer Mathematische Maschinen und Datenverarbeitung (Informatik) der Friedrich Alexander Universitaet Erlangen Nuernberg, Band 13, Nr. 1, Erlangen, Januar 1980

[2] C. G. Bell, A. Kotok, Th. N. Hastings, R. Hill: The Evolution of the DEC-System-10; Comm. ACM, Vol. 21, No 1, January 1978, pp 44-63

[3] J. Palme: TOPSTEACH - A computer-aided introductory course to the DEC 10 system; Research Institute of National Defense, S-10450 Stockholm 80, October 1973

[4] D. G. Bobrow, J. D. Burchfiel, D. L. Murphy, R. S. Tomlinson: TENEX, a Paged Time Sharing System for the PDP-10; Comm. ACM, Vol. 15, No 3, March 1972, pp 135-143

[5] D. M. Ritchie, K. Thompson: The UNIX Time Sharing System; Comm. ACM, Vol. 17, No 7, July 1974, pp 365-375

[6] S. E. Madnick, J. J. Donovan: Operating Systems; McGraw-Hill, New York, 1974

[7] R. F. Ling: General Considerations on the Design of an Interactive System for Data Analysis; Comm. ACM, Vol. 23, No 3, March 1980, pp 147-154

[8] H. S. Stone (ed.): Introduction to Computer Architecture; Science Research Associates, Chicago, 1975

[9] E. W. Dijkstra: The Structure of the 'THE' Multiprogramming System; Comm. ACM, Vol. 11, No 5, May 1968, pp 341-346

[10] B. H. Liskov: The Design of the Venus Operating System; Comm. ACM, Vol. 15, No 3, March 1972, pp 144-149

[11] F. J. Corbato, J. H. Saltzer, C. T. Clingen: MULTICS - The First Seven Years; AFIPS, Proceedings, Spring Joint Computer Conference, 1972

[12] B. R. Borgerson, M. L. Hanson, P. A. Hartley: The Evolution of the Sperry Univac 1100 Series: A History, Analysis and Projection; Comm. ACM, Vol. 21, No 1, January 1978, pp 25-43

[13] S. E. Madnick, J. W. Alsop, II: A Modular Approach to File System Design; AFIPS, Proceedings, Spring Joint Computer Conference, 1972

[14] P. Freeman: Software Systems Principles: A Survey; Science Research Associates, Chicago, 1975

[15] P. J. Denning: Virtual Memory; ACM Comp. Surv., Vol. 2, No 3, September 1970, pp 153-189

[16] R. R. Muntz: Scheduling and Resource Allocation in Computer Systems; enthalten in [14]

[17] J. Rodriguez-Rosell, J. Dupuy: The Design, Implementation and Evaluation of a Working Set Dispatcher; Comm. ACM, Vol. 16, No 4, April 1973, pp 247-253

[17] C. G. Bell, J. C. Mudge, J. E. McNamara: Technology Progress in Logic and Memories; enthalten in: C. G. Bell, J. C. Mudge, J. E. McNamara: Computer Engineering; Digital Press, Bedford MA, 1978

[19] G. E. Moore: Microprocessors and Integrated Electronic Technology; Proc. IEEE, Vol. 64, No 6, June 1976, pp 837-841

[20] G. Weck: Benutzerschnittstelle fuer ein Timesharing-System; internes Arbeitspapier, Juni 1980

ANHANG
======

Benchmark-Programm DYNMIX

```
C
C *** DYNAMIC MIX PROGRAM 1 (DMP-1) ***
C
C     DOUBLE PRECISION A,B,XR (ONLY FOR "DOUBLE PRECISION" VERSION)
      DOUBLE PRECISION A,B,XR
      DIMENSION C(1000),V(109),L(1000),D(1000),N(33),TIME(11)
      DIMENSION NREP(10)
      COMMON/TIME/ATIME,PTIME,NUMBER,STIME,RTIME,TIME1,TIME2
      COMMON AA,AB(7000),AC(7000),AD
      DIMENSION AT(7)
      DATA AA,AB,AC,AD/14002*0.0/
C     EQUIVALENCE (C,AB),(L,AB(3000)),(D,AC)
C     DATA NREP/400,3000,4000,4000,1000,1000,1000,1000,300,300/
      DATA NREP/40,300,400,400,100,100,100,100,30,30/
C     DATA NREP /10*100/
C
      CALL TIMST('DYNAMIX ')
      CALL TTIME1
C     NULLSETZEN DER UHR
C
      NUMBER = 0
      TIME(11) = 0.0
      DO 99 I=1,33
99    N(I)=0
C
C
C     TP1: PRODUCTION OF 1000 RANDOM NUMBERS
C
      CALL TTIME1
      JJ1=NREP(1)
      DO 1 JJ=1,JJ1
      A=0.35127
      B=0.60849
      XR=0.73158
      DO 1 I=1,1000
      XR=XR*XR
      IF (XR.LT.B) XR=XR+A
C     D(I)=AMOD(10000.0*XR,1.0E0) (FOR "SINGLE PRECISION" VERSION)
C     D(I)=DMOD(10000.0*XR,1.0D0) (FOR "DOUBLE PRECISION" VERSION)
      D(I)=DMOD(10000.0*XR,1.0D0)
      C(I)=2.0*XR-1.0
1     CONTINUE
C
C
      CALL TTIME2
      TIME(1) = PTIME
C
C
```

```
C       TP2: POLYNOMIAL OF ORDER 10
C
        CALL TTIME1
        JJ1=NREP(2)
        DO 2 JJ=1,JJ1
        DO 2 K=1,100
        I1=K
        I2=K+10
        X=0.9
        POLY=0.0
        DO 2 I=I1,I2
        POLY=POLY*X+C(I)
2       CONTINUE
        CALL TTIME2
        TIME(2) = PTIME
C
C
C       TP3: SCALAR PRODUCT OF TWO VECTORS
C
        CALL TTIME1
        JJ1=NREP(3)
        DO 3 JJ=1,JJ1
        DO 3 K=1,10
        I1=K
        I2=I1+99
        SCP=0.0
        DO 3 I=I1,I2
        SCP=SCP+C(I)*C(I+500)
3       CONTINUE
        CALL TTIME2
        TIME(3) = PTIME
C
C
C       TP4: SUM OF TWO VECTORS
C
        CALL TTIME1
        JJ1=NREP(4)
        DO 4 JJ=1,JJ1
        DO 4 K=1,10
        I1=K
        I2=I1+99
        DO 4 I=I1,I2
        V(I)=C(I)+C(I+500)
4       CONTINUE
        CALL TTIME2
        TIME(4) = PTIME
C
C
C       TP5: SQUARE ROOT ITERATIONS
C
        CALL TTIME1
        JJ1=NREP(5)
        DO 5 JJ=1,JJ1
        DO 5 K=1,200
        B=C(K)
```

```
      R=1.0
      DO 5 I=1,5
      R=(R+B/R)/2.0
5     CONTINUE
      CALL TTIME2
      TIME(5) = PTIME
C
C
C     TP6: FIND ELEMENT OF VECTOR WITH MAXIMUM MODULS
C
      CALL TTIME1
      JJ1=NREP(6)
      DO 6 JJ=1,JJ1
      J=0
      Z=0.0
      DO 6 I=1,1000
      X=ABS(C(I))
      IF (X.GT.Z) GOTO 66
      GOTO 6
66    CONTINUE
      J=I
      Z=X
6     CONTINUE
      CALL TTIME2
      TIME(6) = PTIME
C
C
C     TP7: CONVERT FLOAT TO FIXED
C
      CALL TTIME1
      JJ1=NREP(7)
      DO 7 JJ=1,JJ1
      DO 7 I=1,1000
      L(I)=D(I)*32.0 +1.0
7     CONTINUE
      CALL TTIME2
      TIME(7) = PTIME
C
C
C     TP8: COUNT FREQUENCY
C
      CALL TTIME1
      JJ1=NREP(8)
      DO 8 JJ=1,JJ1
      DO 8 I=1,1000
      M=L(I)
      IF (M.GT.33) M=33
      N(M)=N(M)+1
8     CONTINUE
C
C
      CALL TTIME2
      TIME(8) = PTIME
C
C
```

```
C      TP9: BIT TEST AND SUMMATION
C
       CALL TTIME1
       JJ1=NREP(9)
       DO 9 JJ=1,JJ1
       SO=0.0
       S1=1.0
       DO 9 I=1,1000
       K=L(I)
       M=K/4-2*(K/8)
       IF (M) 101,101,102
101    SO=SO+D(I)
       GOTO 9
102    S1=S1+D(I)
9      CONTINUE
       CALL TTIME2
       TIME(9) = PTIME
C
C
C
C      TP10 : MATRIX-TEST
C
C
       CALL TTIME1
       NRPP=NREP(10)
       DO 10 JJK =1,NRPP
       DO 10 JJ=1,7000
       JK=7001-JJ
       AA=AC(JK)+AB(JJ)
       AD=AB(JK)+AC(JJ)
10     CONTINUE
       CALL TTIME2
       TIME(10)=PTIME
C
C      CALCULATION OF TOTAL TIME TP1-TP9
C
       DO 200 JJ=1,10
200    TIME(11)=TIME(11)+TIME(JJ)
C
C
C      OUTPUT RESULTS
C
C
C
       WRITE(4,1000) D(1000),POLY,SCP,R,J,Z,SO,S1,N
       WRITE(4,1001) TIME
1000   FORMAT (1H0,4E15.7,I5,3E15.7/3(1H 16I6/))
1001   FORMAT (9H TIME(J)=,11F8.3)
       CALL TIMENT('DYNAMIX ')
       STOP
       END
       SUBROUTINE TTIME1
       COMMON/TIME/ATIME
       CALL UTILIZ(ITIME)
       ATIME=ITIME/1.E5
```

```
      RETURN
      END
      SUBROUTINE TTIME2
      COMMON/TIME/ATIME,PTIME,NUMBER
      CALL UTILIZ(ITIME)
      ZTIME=ITIME/1.E5
      PTIME=ZTIME-ATIME
      NUMBER=NUMBER+1
      WRITE(4,200) NUMBER,PTIME
200   FORMAT (' ',I12,F10.4)
      RETURN
      END
      SUBROUTINE TIMST(ITEXT)
      INTEGER ITEXT(2)
      COMMON/TIME/ATIME,PTIME,NUMBER,STIME,RTIME,TIME1,TIME2
      WRITE(4,1) ITEXT
1     FORMAT('0****** START ',2A4,' ******'/1X)
      CALL UTILIZ(ITIME)
      STIME=ITIME/1.E5
      CALL ICLOCK(ITIME)
      TIME1=ITIME/1.E5
      RETURN
      END
      SUBROUTINE TIMENT(ITEXT)
      INTEGER ITEXT(2)
      COMMON/TIME/ATIME,PTIME,NUMBER,STIME,RTIME,TIME1,TIME2
      WRITE(4,1) ITEXT
1     FORMAT('0****** STOP ',2A4,' ******'/1X)
      CALL UTILIZ(ITIME)
      ZTIME=ITIME/1.E5
      RTIME=ZTIME-STIME
      CALL ICLOCK(ITIME)
      DTIME=ITIME/1.E5
      TIME2=DTIME-TIME1
      WRITE(4,2) RTIME,TIME2
2     FORMAT(' CPU-Time:     ',F10.4/' Elapsed Time:',F10.4)
      RETURN
      END
```