Holleczek

PDV

Projekt Prozeßlenkung mit DV-Anlagen

Entwicklungsnotizen

PDV-E 89

Stufe 2 - ASME - PEARL - Compiliersystem
- Grundideen -

P. Holleczek
Physikalisches Institut der Universität
Erlangen

P4.2/3A;ER-FIE/1

Oktober 1976

GESELLSCHAFT FÜR KERNFORSCHUNG

ASME-INTERNE NOTIZ

NR: E-036
VERSION NR.: 1
DATUM: 30.7.75
27 Seiten

VERFASSER: Holleczek

TITEL: Gedanken zu Stufe 2 des ASME-PEARL-Compiliersystems

Zusammenfassung:

Es wird ein Vorschlag unterbreitet, auf welchem Wege die Portabilität von PEARL Stufe 2 verglichen mit Stufe 1 verbessert werden könnte. Im Anhang sind einige fragmentarische Vorstöße auf diesem Weg skizziert. Sie sollen zeigen, daß der Weg gangbar ist.

Gliederung:

I Vorbemerkung
II Grundideen
III Schnittstellen
III.1 CIMIC-B Hilevel-Prozeduren
III.2 CIMIC-B Lolevel-Prozeduren
III.3 BS-Anpassung
IV Voraussichtliche Wirksamkeit
IV.1 Portabilitätsgrad
IV.2 Effektivität
V Ausblick

Anhang I: Vergleich ORG 306 - ORG 330
Anhang II: Vergleich elementarer Eigenschaften einiger Rechner

Grundgedanken zu PEARL-Stufe 2

Will man PEARL-Stufe 1 verbessern, kann man folgende Gesichtspunkte ins Auge fassen:

1.) Sprachumfang
2.) Transportabilität
3.) Sprachqualität
4.) Codeeffektivität / Zeitverhalten
5.) Compilereffektivität
6.) Bedienungskomfort,

die von Benutzern und Implementatoren verschieden stark gewichtet werden dürften.

"Der" ASME Implementator-und-Benutzer verbessert 6.) im Rahmen eines eigenen Test-und-Bediensystems, hat es gelernt, die sich oft ausschließenden Folgerungen aus 4.) und 5.) gegeneinander abzuwägen; weiß inzwischen, daß Forderungen nach 3.) leicht in 1.) ausarten, muß sich aber als "nicht-produkt-gebunden" in eigenem Interesse um 2.) kümmern.

Bei einer Erhöhung der Transportabilität müssen das Problem des Erstellungswerkzeugs und die Frage des Angriffspunktes diskutiert werden.

Das in Stufe 1 benutzte "Portabilitätswerkzeug" FORTRAN hat sich bewährt, ist aber voraussichtlich an seinen Grenzen angestoßen. Eine andere einheitliche "System"-Sprache steht den ASME-Mitgliedern (noch) nicht zur Verfügung. Als Ausweg bietet sich eine Sprache, die mit abschätzbarem vernünftigem Aufwand auf die Zielmaschinen zu bringen ist: CIMIC (s. Kap. I).

In Angriff genommen und gelöst wurde in Stufe 1 die Transportabilität des oberen Compilerteils. Für Stufe 2 stehen also das Laufzeitpaket (und evtl. das Betriebssystem) im Brennpunkt. Da jedoch portable Betriebssysteme im Augenblick allgemein noch Gegenstand von Spekulationen sind, konzentrieren sich jetzt die Überlegungen auf ein 'portableres' Laufzeitpaket (s. Kap. II).

Auf dem Wege zu dem genannten Ziel sind sicher noch mehr Schnittstellenprobleme zu lösen als bisher (s. Kap. III). Ob der Erfolg (s. Kap. IV) die Bemühungen rechtfertigen wird, muß sich zeigen. Jedenfalls bleibt der Weg für zukünftige Entwicklungen offen (s. Kap. V).

Vorbemerkung - Aufteilung CIMIC A/B

Cimic für ASME Stufe 1 [1] zerfällt bei näherem Hinsehen grob in zwei Teile:

Der eine Teil der Cimic-Anweisungen läßt sich mit Hilfe eines Übersetzers ziemlich direkt auf Zielassembler-Codefolgen abbilden, der andere nur auf Einsprünge in ein zielmaschinenspezifisches Laufzeitsystem, das dann seinerseits Aufrufe an das Ziel-BS enthalten kann.

Die erste Art der Cimic-Anweisungen entstammt im wesentlichen dem PEARL-Teilgebiet Algorithmik (Cimic-A), der zweite den Betriebssystemnahen PEARL-Teilgebieten (Cimic-B). *

Cimic-A kann man als eine bzw. die Untermenge von Cimic betrachten, die sich recht effektiv auf die Zielmaschinen bringen läßt und zur Formulierung algorithmischer Zusammenhänge geeignet ist. Umgekehrt kann ein genau abgegrenztes Cimic-A den Kern einer 'System'-Sprache bilden. Daß Cimic-A außerdem auf die Belange von Rechnern moderner Architektur eingehen muß, ohne die bisherigen Zielmaschinen zu vergessen, versteht sich von selbst (s. Anhang II: Vergleich elementarer Eigenschaften einiger Rechner).

---

*

Eine solchermaßen durch die Semantik geprägte Trennlinie läßt sich mehr oder weniger deutlich auf allen Ebenen des Compiliersystems (s. Abb. 1) feststellen. Sie verläuft aber sicher nicht überall an der gleichen Stelle, da immer auch algorithmische Sprachelemente benötigt werden, um betriebssystemnahe Funktionen zu formulieren.

II Grundideen

Ein nicht unbeträchtlicher Anteil des Implementationsaufwandes an den Zielmaschinen war die Realisierung von Cimic-B (d.h. z.B. des Laufzeitpakets) [2, 3, 4, 5, 6] . Der Grund liegt darin, daß Cimic-B-Anweisungen in der Stufe 1 fast noch PEARL-Quell-Niveau hatten.

Gelingt es, die Realisierung von Cimic-B wenigstens zum Teil portabel zu gestalten, ist der Implementationsaufwand für eine einzelne Zielmaschine schon stark reduziert.

Grundideen des im folgenden für ASME Stufe 2 vorgeschlagenen Konzepts sind:

- Umformulieren von Cimic-B

  Die Cimic-Sprachelemente, die bisher dem Anteil -B- angehörten, werden nicht mehr in Befehlsform dargestellt, sondern in Form von (PEARL-Sprachelement nahen) Prozeduren: 'Cimic-B Hilevel-Prozeduren'.
  Dadurch wird eine bessere "Codegenerierbarkeit" erreicht, ganz einfach deswegen, weil viel mehr "über einen Leisten geschoren wird".

- Einführung einer neuen Schnittstelle

  Ein möglichst niedrig liegender und dennoch von den Ziel-BS'en unabhängiger Satz von Funktionen wird definiert. Diese Schnittstelle ist ebenfalls in Prozedurform konzipiert: 'Cimic-B Lolevel-Prozeduren'.

- Codierung in Cimic-A

  Die Codierung der Cimic-B Hilevel-Prozeduren erfolgt in Cimic-A und enthält Aufrufe von Cimic-B Lolevel-Prozeduren. Sie erfolgt nur einmal für Stufe 2.

- Stark reduzierte Zielmaschinenabhängigkeit

  Die Zielmaschinen- bzw. Ziel-BS-Abhängigkeit schlägt sich in der Codierung der Cimic-B Lolevel-Prozeduren nieder.

Abb. 1 zeigt die neu eingeführten Cimic-B Prozeduren und ihre Rolle bei der Erzeugung von ablauffähigen PEARL-Programmen.

Abb. 1 Wesentliche Vorgänge bei der Erzeugung eines ablauffähigen PEARL-Programms und des dazu nötigen Laufzeit-Systems auf einer Zielmaschine (mit oberem Compilerteil, Codegenerator, Binder, Assembler, Betriebssystem)

Legende: Codefolge, Prozedur-Aufruf, Prozedur-Definition, Übersetzung

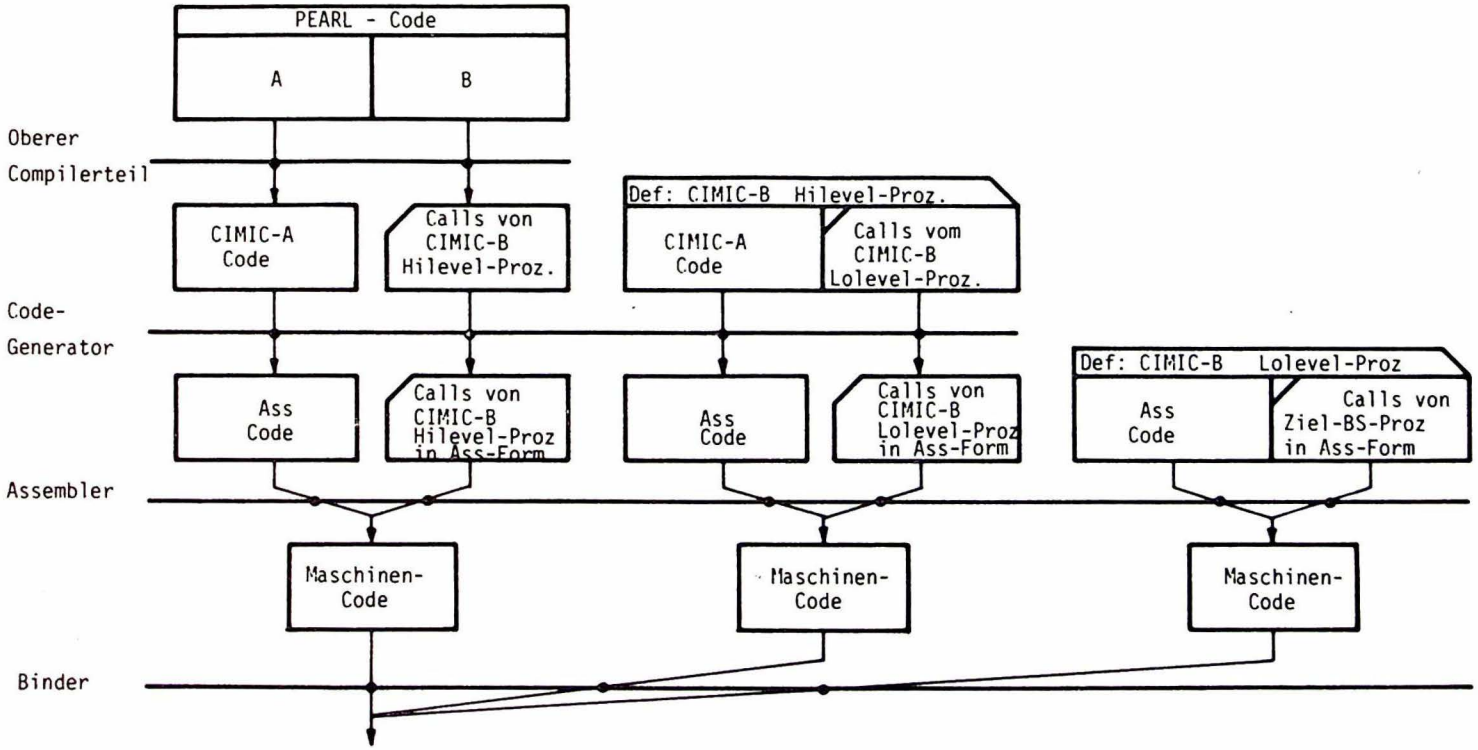


Abb. 1a Vorgänge bei der Übersetzung eines PEARL-Programms

Abb. 1b Vorgänge bei der Erstellung der zielmaschinenunabhängigen BS-nahen (CIMIC-B-Hilevel-)Prozeduren:
Abfassen der Def: 1x für Stufe 2
Übersetzungen: 1x für Zielmaschine

Abb. 1c Vorgänge bei der Erstellung der zielmaschinenabhängigen BS-Adaptions-(CIMIC-B Lolevel-) Prozeduren:
Abfassen der Def: 1x für Zielmaschine
Übersetzungen: 1x für Zielmaschine

III Schnittstellen

Will man Teile des bisherigen Laufzeitpaketes nicht in Assembler, sondern in Cimic-A schreiben, ergeben sich einige neue (Schnittstellen-) Probleme:

1. Cimic-B Hilevel-Prozeduren

   Betrachtet man das Laufzeitpaket grob als einen Satz von Prozeduren (was nach der bisherigen Praxis gerechtfertigt ist), dann stellt sich die Frage, wie diese Prozeduren an Cimic anzuschließen sind, bzw. was für eine Art Aufruf den anderweitig definierten Prozeduren gegenübersteht.

   Die in Frage kommenden Sprachmittel von Cimic-1 hatten bisher leider keinerlei Ähnlichkeit mit Prozeduraufrufen, eher waren sie Befehle, z.B.:

   ```
   GET␣device - type␣ref - 7 $
   FORMAT␣form␣ref - 6 $
   CREAD␣char (length)␣ref - 2 $
   GEND␣␣␣␣␣$
   ```

   Nichtsdesto weniger konnten diese Befehle bisher entweder Aufrufen von (Assembler-) Prozeduren (GET, CREAD, GEND) oder Parametern (Format, etc.) von Prozeduren entsprechen.

   Eine Entscheidung zwischen Aufrufen und Parametern wurde bisher vom Codegenerator* getroffen. Um hier zu mehr Durchsichtigkeit zu kommen, (was für transportable Software notwendig ist,) könnte man z.B. die aus einem PEARL-B-Statement entstandenen Cimic-B-Befehle entsprechend der Art ihrer Weiterverwendung als Prozeduraufruf oder als Parameter schreiben. Diese Methode hat jedoch den Nachteil, daß sie implementationsabhängige Ergebnisse liefert.

---

* auf Grund von Schnittstellenbeschreibungen

Ein Vorschlag, der mehr Einheitlichkeit böte, wäre, die von einem PEARL-B-Statement herrührenden Cimic-Befehle als nur einen Aufruf einer Cimic-Prozedur mit entsprechend vielen Parametern zu konzipieren.

Eine solche Art von Prozeduren existiert in Cimic bisher nicht. Cimic müßte deshalb um einen Mode CBHP (Cimic-B Hilevel-Prozedur), erweitert werden.

Der Aufruf einer solchen Prozedur hätte dann z.B. folgende Gestalt:

```
CALL ␣ PROC ␣ CBHP ␣ GET ... $
ARGIS ␣ devt ␣ ref-7 $
ARGIS ␣ FORM ␣ ref-6 $
ARGIS ␣ CHAR ␣ ...
CEND ␣ PROC ␣ CBHP ␣ GET ... $
```

Ihm könnnte eine Definition folgender Art gegenüberstehen:

```
BEGIN ␣ PROC ␣ CBHP ␣ GET ... $
! Formalparameter
LINK ␣ PROC ␣ CBHP ␣ GET ... $
! Code
RETURN␣ PROC ␣ CBHP ␣ GET ... $
! Parameterrückgabe
END ␣ PROC ␣ CBHP ␣ GET ... $
```

In Erweiterung der Cimic-I Referenzarten ref-1, ref-2, ... können die Namen GET, PUT, ... z.B. unter "ref-H" zusammengefaßt werden; ref-H (durch CBHP gekennzeichnet) enthält dann die betriebssytemorientierten Prozedurnamen auf dem aus Cimic-1 bekannten Level.
Die durch LINK ... eingeleitete Parameterübergabe der CIMIC-B Hilevel-Prozeduren kann naturgemäß auf einige Prüfungen verzichten, die bei PEARL-Prozeduren noch notwendig sind.

Ein typisches Problem innerhalb dieser Prozeduren ist z.B., die zum Teil komplexen (PEARL-)-Datentypen (z.B. Formate) in elementarere (z.B. Characterstrings) umzuwandeln.

2. Cimic-B Lolevel-Prozeduren

Die Codierung der Cimic-B Hilevel-Prozeduren erfolgt durch den bereits angeführten algorithmischen Subset von Cimic (Cimic-A) und führt zu dem Aufruf weiterer Prozeduren. Letztere gehören dann einem Mode CBLP (Cimic-B Lolevel-Prozeduren) an *. Die Namen dieser Prozeduren kann man sich unter der Referenzart ref-L zusammengefaßt denken. CBLP-Calls können als Aufrufe an ein Cimic - Lo-Betriebssystem verstanden werden. Diese Aufrufe sollten im Einklang mit den im Augenblick laufenden Entwicklungen neutraler BS-Elementarfunktionen konzipiert werden und nach Möglichkeit knapp auf ASME-Ziel-BSe abbildbar sein (s. Kap. IV.1).

Die Parameterübergabe in den genannten CBLPs ist wiederum weniger umfangreich als bei den CBHPs, da die Typenvielfalt hier noch stärker eingeschränkt sein dürfte.

Die Konzipierung der Cimic-B Lolevel-Prozeduraufrufe ist wohl die Hauptaufgabe beim Vorantreiben der Portabilität bei PEARL Stufe 2.

3. Abbildung der Cimic-B Lolevel-Prozeduren auf Ziel-BSe

Die Codierung der CBLPs (zum "Erreichen" eines Ziel-Betriebssystems) erfolgt z.B. durch Ziel-Assembler-Codefolgen (s. auch Kap. V.). Diese Codierung stellt die übrig bleibende 'BS-Adaption' dar. Der Umfang dieser Adaptionsarbeit dürfte sich - je nach "Höhe" der CBLPs - im Vergleich mit Stufe 1 deutlich verringern lassen.

---

* Eventuell erfolgt die bis auf CBLP-Aufrufe führende Codierung über einige Zwischenstufen. Es ist zu untersuchen, ob hierfür bei der Cimic-Konzipierung Vorkehrungen getroffen werden müssen (Einführen einer geeigneten Prozedurklasse o.ä.).

IV Voraussichtliche Wirksamkeit

Die Wirksamkeit der vorgeschlagenen Methoden hängt davon ab, welcher Portabilitätsgrad und welche Effektivität erreicht werden können:

- Der erreichbare Portabilitätsgrad, d.h. der portable Anteil am Gesamtaufwand für Systemerstellung und -adaption hängt wiederum davon ab, wie gut die eingeführten CIMIC-B Lolevel-Prozeduren auf Aufrufe der Ziel-Betriebssysteme abgebildet werden können.

- Die Rolle von CIMIC als Zwischensprache auf dem Weg zur Code-Erzeugung aus PEARL-Programmen verlangte schon in Stufe 1 der ASME-Implementation eine leichte Abbildbarkeit auf Ziel-Maschinen-Codefolgen (Code-Effektivität). Der vorliegende Vorschlag für ASME Stufe 2 sieht darüber hinaus CIMIC als Erstellungshilfsmittel für das Laufzeitsystem vor. Einer effektiven Code-Erzeugung kommt deshalb eine mehrfache Bedeutung zu.

Inwieweit diese Aspekte erfüllt werden (können), wird sich letztlich erst nach Fertigstellung der Stufe 2 herausstellen. Nichtsdestoweniger müssen sie zu Beginn der Arbeiten bekannt sein, da sie Konsequenzen auf die Konzeption des gesamten Programmiersystems haben (sollten). Insbesondere bedingt ihre Berücksichtigung eine vergleichende Betrachtung der Eigenschaften von möglicherweise einsetzbaren Maschinen als Arbeitsgrundlage. Dies wurde stichpunktartig vorab getan:

Die Abschätzung des Portabilitätsgrades an Hand zweier verschiedener Maschinen zeigt einen relativ erfolgversprechenden Weg. Die Untersuchung des Effektivitätskriteriums (zum Teil an Hand stark unterschiedlicher Maschinen) skizziert die zu lösenden Probleme.

1. Portabilitätsgrad

Ein hoher Grad an Portabilität verlangt eine (aus der Sicht der System-Hierarchie) möglichst 'niedrige' Vereinheitlichung bzw. Zusammenfassung der BS-Aufrufe des ASME-Maschinenspektrums, die zur Realisierung von PEARL-Merkmalen nötig sind. Mit einer solchen Vereinheitlichung betritt man nicht unbedingt Neuland. Ein weiter gestecktes Ziel, nämlich eine allgemeingültige Darstellung von BS(-Kern)-Funktionen wurde bereits angegangen [z.B. 7,8] . Auch unter besonderer Berücksichtigung einer modernen Maschine wurde dies bereits versucht [9] . Im vorliegenden Fall jedoch gilt es, speziell die für einen PEARL-Subset [10] nötigen BS-Aufrufe eines gewissen Maschinenspektrums einheitlich darzustellen. Dies scheint lösbar: Daß sich BS-Aufrufe für Maschinen stark unterschiedlicher Architektur (z.B. Siemens 306 und 330) nicht grundlegend unterscheiden müssen, zeigt ein flüchtiger Vergleich von

- Struktur von BS-Aufrufen
- Tasking & Timing
- E/A - Aufrufe
- Betriebsmittelverwaltung

der Betriebssysteme ORG 306 [11] und ORG 330 [12] (s. auch Anhang I). Ausgehend von diesem Vergleich und den Erfahrungen aus der Stufe 1 der ASME-Implementation läßt sich extrapolieren:

Wie weit die Portabilität vorangetrieben werden kann, wird stark von den Semantikgebieten (Tasking, Timing und Synchronisierung; Algorithmik; E/A) abhängen. Bei Tasking, Timing und Synchronisation dürfte sich nur bei ähnlichen Task-Kontrollblöcken ihr Aufbau, Eintragungen hierin und Schedule-Auflösungen vereinheitlichen lassen, sowie ihre Abbildung auf auch für die Synchronisation geeignete elementare Funktionen.

Bezieht man die komplexere Algorithmik in die Portabilitätsüberlegungen mit ein, so lassen sich vielleicht zusammengesetzte Modes in einfache auflösen. Ebenfalls die Standardfunktionen ließen sich auf diese Weise formulieren.

Bei der Prozeß-E/A wird es wohl wenig Spielraum für portable Entwicklungen geben, es sei denn, man hätte nur CAMAC-Peripherie zu versorgen und hätte es überdies nur mit einer Art von Controller zu tun. Bei der Standard- und der Graphischen E/A ist dagegen zu erwarten, CIMIC-B Hilevel-Prozeduren auf die Übergabe geeigneter (alphamerischer, binärer etc.) Puffer zurückführen zu können.

Alle diese Überlegungen sind zusammenfassend in den Abbildungen 2a und 2b dargestellt. Sie zeigen einen Ausschnitt aus einem möglichen portablen Laufzeitpaket und dazu eine geeignete Klassifizierung der durch die CIMIC-B Prozeduren zu leistenden Aufgaben. Bei der Angabe von in Frage kommenden Ziel-BS-Aufrufen wurde in Abb. 2a eine S 306 und in Abb. 2b eine S 330 als Zielmaschine berücksichtigt.

2. Effektivität

Wie (code-)effektiv das Laufzeitsystem mit Hilfe der vorgeschlagenen Methode implementiert werden kann, hängt davon ab,

- wie gut die elementare Architektur der CIMIC-Maschine zu derjenigen der aktuellen Zielmaschine paßt
- wie der CIMIC-Sprachvorrat die Systemerstellung unterstützt

Elementare Architektur

CIMIC-1 ergab sich aus der Abstraktion der elementaren Architektur der für Stufe 1 der ASME maßgebenden Zielmaschinen Siemens 306 und AEG 6050. Inzwischen müssen generell Maschinen mit qualitativ und quantitativ fortentwickelten Merkmalen

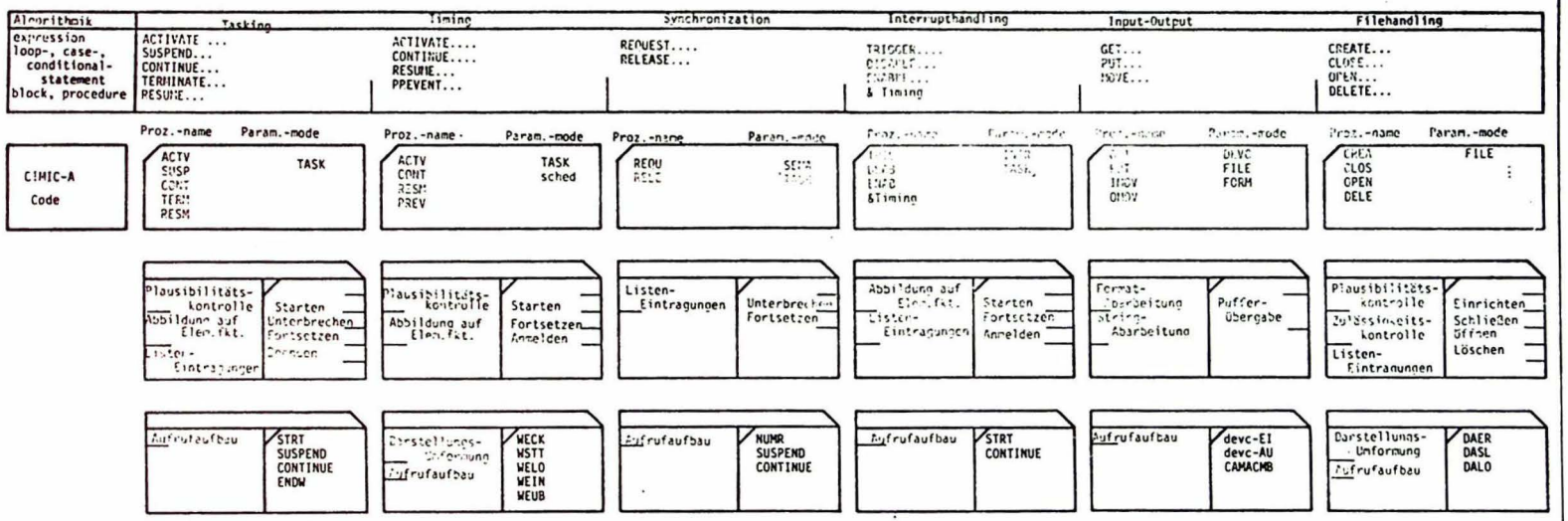


Abb. 2a Schematische Zusammenstellung möglicher Schreibweise und denkbarer Aufgabenklassen der bereits eingeführten CIMIC-B Hi- bzw. Lolevel-Prozeduren. Die Zusammenstellung ist unterteilt nach :

(horizontal:) Semantikgebieten (Algorithmik/Tasking, Timing, Synchronization, etc.)
(vertikal:) Zugehörigkeit zu verschiedenen Generierungsvorgängen (s. Abb. 1, hier am rechten Rand wieder aufgeführt)
Die 'Calls von Ziel-BS-Proz. in Ass-Form' beziehen sich auf eine Zielmaschine S 306 (s. Anhang I)

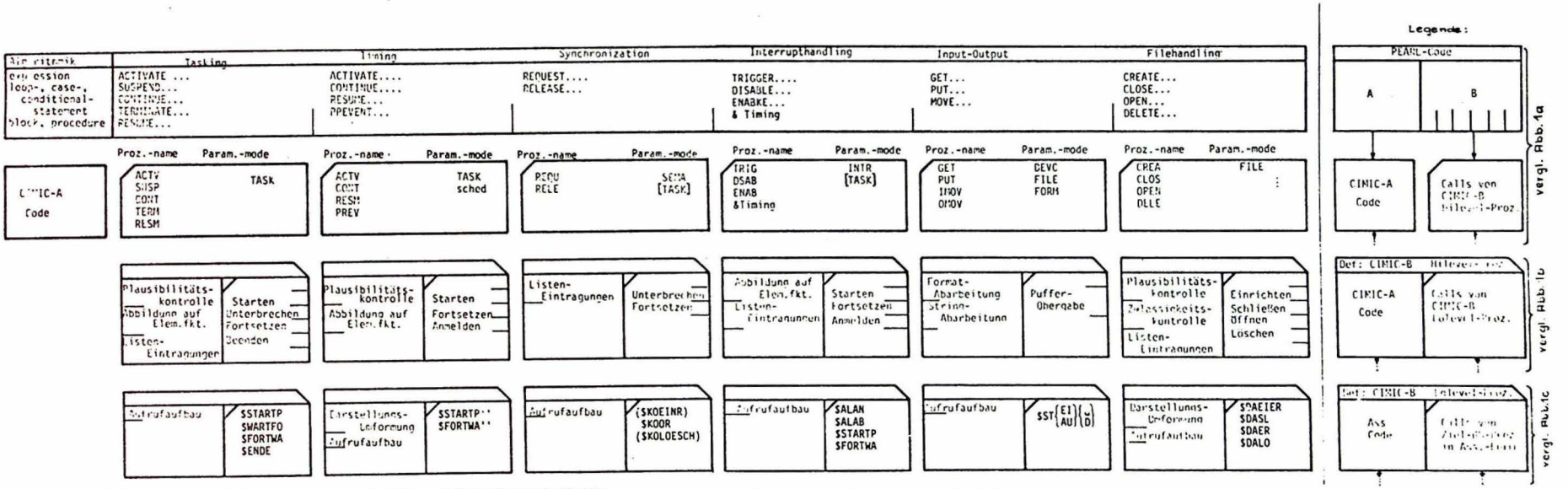


Abb. 2b Schematische Zusammenstellung möglicher Schreibweise und denkbarer Aufgabenklassen der bereits eingeführten CIMIC-B Hi- bzw. Lolevel-Prozeduren. Die Zusammenstellung ist unterteilt nach :

(horizontal:) Semantikgebieten (Algorithmik/Tasking, Timing, Synchronization, etc.)
(vertikal:) Zugehörigkeit zu verschiedenen Generierungsvarianten (s. Abb. 1, hier am rechten Rand wieder aufgeführt;
Die 'Calls von Ziel-ASS-Proz. in Ass-Form' beziehen sich auf eine Zielmaschine S 330 (s. Anhang I)

(Register, Operanden, Adressierung) berücksichtigt werden. Insbesondere dürfen die sich weit verbreitenden Kleinstrechner mit ihrer meist modernen Struktur nicht außer Acht gelassen werden. Ein universelles, 'alte' wie 'neue' Maschinen gleichermaßen effektiver abdeckendes CIMIC wird es dementsprechend nicht geben können. Zur Beleuchtung und Abgrenzung dieses Problems wurden deshalb einige Maschinen verglichen, die (in Erlangen) im Augenblick leicht verfügbar sind, sich durch interessante Fähigkeiten auszeichnen oder evtl. eine weite Verbreitung versprechen (Stand Juli 1975). Diese Auswahl ist recht weit gespannt und legt mit Sicherheit nicht das ASME-Maschinenspektrum fest. Die in Anhang II gezeigte Zusammenstellung berücksichtigt die Aspekte Register(-arten), Operanden(-zahl, -art) und Adressierungsmodi im Vergleich mit der CIMIC-1 Maschine und wirft die Frage nach ihrer Verwirklichung in der CIMIC-2 Maschine auf.

Sind gewisse Merkmale von CIMIC-2 an einer Zielmaschine nicht vorhanden (z.B. Indexregister an 306), können sie z.B. vom Codegenerator simuliert werden. Hierzu zeigt die Erfahrung, daß es i.a. effektiver ist, 'mächtigere' Merkmale für eine 'einfache' Zielmaschine z.B. durch Simulation einzuführen, als für eine 'mächtige' Zielmaschine umständliche Lösungen anwenden zu müssen.

Systemerstellungsaspekte

Die zusätzliche Rolle von CIMIC als Systemerstellungshilfsmittel (d.h. quasi als 'Systemsprache') legt den Wunsch nahe, komplexe Datentypen (und hierfür geeignete Zugriffsmechanismen) zur Verfügung zu haben, z.B.

- mehrdimensionale Felder
- Strukturen allgemeinerer Art
- Zeiger-Variable (Ref. 2 - Größen)

Da CIMIC in erster Linie als Zwischensprache bei der PEARL-Übersetzung dient, enthält es zunächst höchstens solche Datentypen, die auch im zugehörigen PEARL-Subset auftreten.

Es gilt hier also, CIMIC ggfs. um solche 'systemsprachen-typische' Sprachelemente zu erweitern. Eine Erweiterung dieser Art sollte auf jeden Fall sprach-'konform' erfolgen, um die CIMIC-Architektur nicht zu verletzen. Es gibt hierzu zwei Möglichkeiten. Der erweiterte CIMIC-Sprachvorrat wird

- durch einen entsprechend erweiterten Codegenerator ablauffähig gemacht
- durch einen dem Codegenerator vorgeschalteten Prozessor in einfachere bereits existierende Sprachelemente aufgelöst

Bei der ersten Methode sollten sinnvollerweise etwaige durch Subsetausdehnungen bedingte CIMIC-Erweiterungen vorab berücksichtigt werden. Allerdings wird der erweiterte Codegenerator für alle Installationen obligatorisch. Die zweite Methode verlangt evtl. einen konzeptionellen Mehraufwand. Den zu ihrer Realisierung nötigen Preprozessor bedingt aber eine Systemerstellung auf CIMIC-Ebene ohnehin. Da er außerdem nur einmal erstellt werden muß, ist diese Methode wohl die elegantere.

V. Ausblick

Es darf nicht verwundern, daß die unterbreiteten Vorschläge sich sämtlich um die ASME-Zwischensprache CIMIC ranken: Die Einführung einer sauber definierbaren Compiler-Schnittstelle und ihre Verwirklichung in CIMIC erwies sich immerhin in ASME Stufe 1 als das Werkzeug, um mit einem Schritt einen nicht geringen Anteil eines Prozeßprogrammiersystems portabel gestalten zu können. Nichtsdestoweniger jedoch kann ein gutes und neuartiges Werkzeug noch universeller verwendet werden, was hiermit gezeigt werden sollte.

Dennoch seien im folgenden noch kurz die Methoden genannt, die gleichsam 'unterhalb' der bisher skizzierten CIMIC-Maschine bei der Portabilitätsverbesserung sich anbieten:

Ein portables Betriebssystem?

Die Formulierung der Anforderungen an ein Betriebssystem durch Prozeduraufrufe auf niedriger Ebene ('CBLP's') legt den Gedanken nahe, von hier aus keine BS-Adaption mehr zu betreiben, sondern wieder ebenenweise den Weg zu einem teilweise portablen (in ... geschriebenen) Betriebssystem zu beschreiten. Das bisher größte Hindernis, das Fehlen einer einheitlichen BS-Schnittstelle, wäre mit Hilfe des vorliegenden Vorschlags beseitigt.

Ein portabler Codegenerator?

Nicht nur für eine, sondern evtl. auch für mehrere der im Anhang II genannten oder ähnlicher Maschinen Codegeneratoren zu schreiben, wird Problem des Implementators werden. Dies verleitet zu dem Gedanken (gerade bei vergleichender Betrachtung mehrerer Zielmaschinen) Codegeneratoren zu parametrisieren und somit Arbeit zu sparen. Sollten solche Untersuchungen fruchtlos sein, könnten sie notfalls auch nur zu einer Sammlung von Erstellungsrichtlinien für ungeübte 'Codegenerierer' führen.

Literatur

[1] B. Eichenauer; CIMIC-1, Ges. f. Kernforschung m.b.H., PDV-Entwicklungsnotiz PDV-E 15

[2] R. Rössler; Programmstruktur und Laufzeitverwaltung für den PEARL-Subset (Stufe 1) der ASME an der Zielmaschine SIEMENS 306, Phys. Inst. d. Univ. Erlangen, ASME-INTERNE-NOTIZ E 019

[3] F.J. Prester; Standard-E/A für den Erlanger ASME-PEARL-Subset, Phys. Inst. d. Univ. Erlangen, ASME-INTERNE-NOTIZ E 021

[4] W. Lindstedt; Bedienung von CAMAC-Peripherie mit PEARL-I/O-Statements im ASME Subset 1, Phys. Inst. d. Univ. Erlangen, ASME-INTERNE-NOTIZ E 023

[5] F.J. Prester; Graphische E/A für den Erlanger ASME-PEARL-Subset, Phys. Inst. d. Univ. Erlangen, ASME-INTERNE-NOTIZ E 030

[6] P. Holleczek; Das Filehandling für den Erlanger ASME-PEARL-Subset, Phys. Inst. d. Univ. Erlangen, ASME-INTERNE-NOTIZ E 032

[7] R. Baumann e.a.; Funktionelle Beschreibung von Prozeßrechner-Betriebssystemen, VDI/VDE Ges. Meß- u. Regeltechnik, Entwurf VDI/VDE 3554

[8] J. Nehmer, O. Eggenberger; Hardwarenahe Elementarfunktionen der Ablaufsteuerung für Prozeßrechnerbetriebssysteme, Ges. f. Kernforschung m.b.H., PDV-Bericht KFK-PDV 49

[9] J. Ehrig, H. Hahn, H. Hotes, A. Ibarra, E. Skrebutis; COPF-Einsatz von Programmbausteinen für Realzeit-Betriebssysteme, Ges. f. Kernforschung m.b.H., PDV-Bericht KFK-PDV 40

[10] Programmieranleitung für das ASME-PEARL-Subset/1. Ges. f. Kernforschung m.b.H., PDV-Entwicklungsnotiz PDV-E 70

[11] Fa. Siemens AG; Organisationsprogramm-Beschreibung, Best. Nr. D13/3006

[12] Fa. Siemens AG; Organisationsprogramm 330 PPII, Best. Nr. P71100-B0014-X-X-35

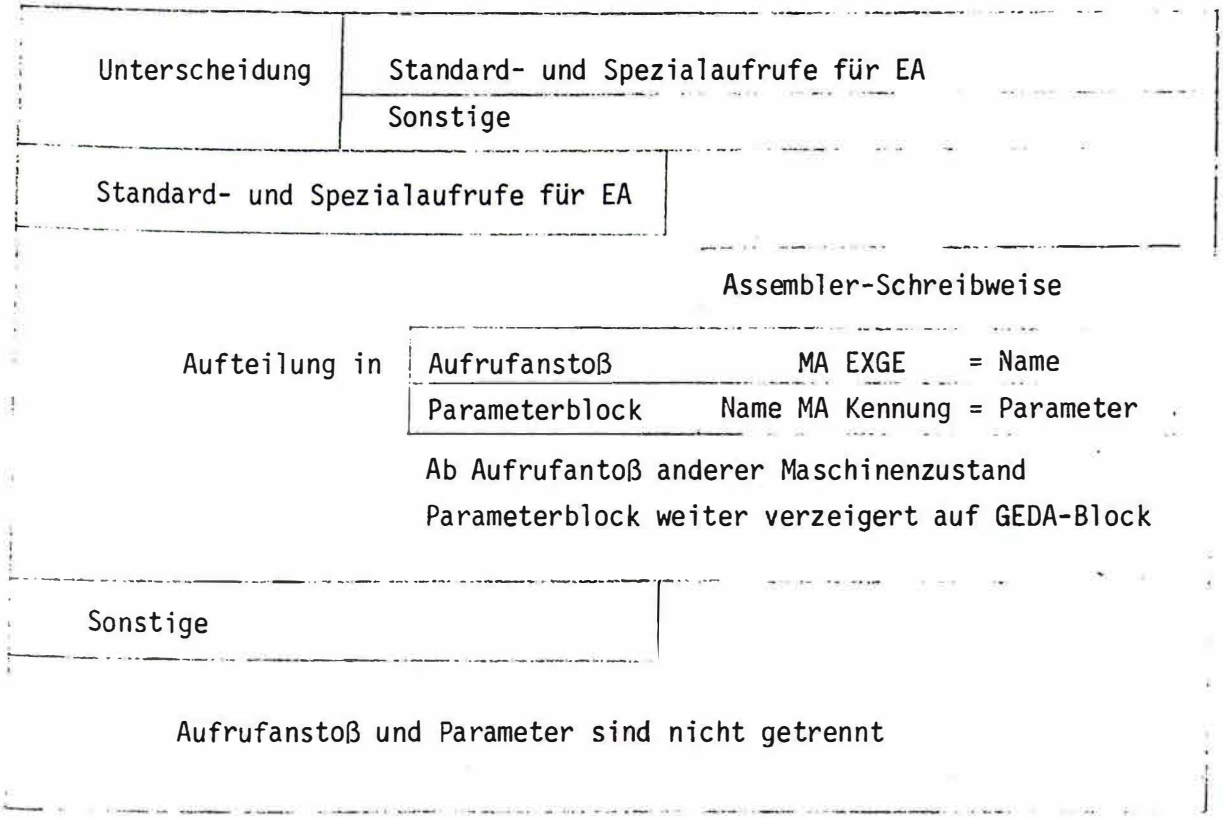
Anhang I

Vergleich ORG 306 - ORG 330 bezüglich

- BS-Aufrufstruktur
- Tasking
- E/A
- Betriebsmittelverwaltung

Vergleichbare Merkmale der beiden Betriebssysteme sind auf jeweils zwei aufeinander folgenden Seiten untergebracht.

306 - BS - Aufrufstruktur



330 - BS - Aufrufstruktur

		Assembler-Schreibweise		
Aufteilung in	Aufrufanstoß		§RUFORG"	Name
	Parameterblock	Name	§Kennung"	Parameter

Ab Aufrufanstoß anderer Maschinenzustand

Parameterblock evtl. weiter verzeigert auf GEDA-Block
(bei E/A-Aufrufen)

Aufrufanstoß im invarianten Programmteil

Parameterblock im varianten Programmteil

306 - Tasking

paralleles Objekt:
sequentielles Objekt: 'Programm' : Identifikation über Nummer
statisches Objekt:

Priorität: für 'Programm' : frei wählbar zwischen 1 und 48

Verknüpfung Objekt-Priorität: statisch (bei Laden)

Operationen:		
	Starten	Programm
	Beenden	eigenes Programm
	Beenden	fremdes Programm
	Anhalten*	Programm
	Fortsetzen*	Programm

Synchronisierung:** 'Belegen' / 'Freigeben' } einer 'Belegzelle'
mit Wirkung: { Eintrag in Warteschlange / Austrag aus Warteschlange

Timing: Tasking-Operationen lassen sich { sofort / zeitlich verzögert / periodisch / zu beliebiger Zeit } ausführen

*) Erweiterung des ORG 306 [2]

**) Bei ASME Stufe 1 PEARL-Implementation nicht verwendet [2]

330 - Tasking

paralleles Objekt: 'Programm': Identifikation über Name bzw. Nummer

sequentielles Objekt: { 'Aufgaben' innerhalb eines 'Programms' } Identifikation über Nummer

statisches Objekt: 'Common Code'

Priorität: für 'Programm': frei wählbar zwischen 1 und 255

für 'Aufgabe': best. durch Identifikationsnummer innerhalb 'Programm'

Verknüpfung Objekt - Priorität: dynamisch

Operationen:

Starten	Programm, Aufgabe	Parameter:2 Registerinhalte
Beenden eigenes	Programm, Aufgabe	
Beenden fremdes	Programm	
Anhalten	Programm	
Fortsetzen	Programm	

Synchronisation: { 'Erhöhen' 'Erniedrigen' } eines 'Koordinierungszählers'

mit Wirkung - je nach Stand des Koordinierungszählers:

{ Austrag aus Warteschlange, keine Operation, Eintrag in Warteschlange
Eintrag in Warteschlange, keine Operation, Austrag aus Warteschlange

---

Timing: Tasking-Operationen lassen sich { sofort, zeitlich verzögert, periodisch, zu fester Zeit } ausführen

306 - E/A

Aufteilung in { Standard- } aufrufe
{ Spezial- }

für { vorzugsweise 'magnetische' Geräte, z.Tl. auch 'Papier'-Geräte }
{ vorzugsweise 'Papier'-Geräte, z.Tl. auch 'magnetische' Geräte }

Standardaufrufe: Durchführung vom aktuellen Gerät unabhängig

Spezialaufrufe: Durchführung vom aktuellen Gerät abhängig

Parameterblock* (nur Standardaufrufe):
enthält zusätzlich Verweis auf GEDA-Block
der das aktuelle Gerät beschreibt

*) s. BS-Aufrufstruktur

330 - E/A

Aufteilung in { Standard- / Spezial- } aufrufe für { Standard- / Prozess-u.a. } Peripherie

Standardaufrufe: Aufteilung in { alphamerischen / binären } Transfer

Durchführung vom aktuellen Gerät unabhängig

Spezialaufrufe: Steuerung durch ein Ablaufmodell:
Es enthält in Einzelphasen unterteilte BS-Abläufe
und Platz für Eintragungen durch den Programmierer

Parameterblock*: enthält zusätzlich Verweis auf GEDA-Block,
der das aktuelle Gerät beschreibt.

*) s. BS-Aufrufstruktur

306 - Betriebsmittel (Geräte-) Verwaltung

Verwaltung Geräte global: vor jedem Gerät Warteschlange zusammen für E+A Eintrag in Warteschlange nach FIFO
Verwaltung Geräte nach 'Programmen': nur für Magnetbandgeräte möglich
Synchronisierung Geräte: Warten auf Transferende { explizit überhaupt nicht
Verhalten am Ende eines 'Programmes': offene Transfers in undefiniertem Zustand

330 - Betriebsmittel (Geräte-) Verwaltung

Verwaltung Geräte global: vor jedem Gerät Warteschlange { getrennt / zusammen } für E+A

Eintrag in Warteschlange nach { Priorität / FIFO }

Gerätereservierung nach 'Programmen': Aufruf { explizit / implizit }

Strategie { exklusiv: für eiges Programm / geteilt: mit anderen Programmen / mit Vorrang }

Synchronisierung Geräte: Warten auf Transferende { explizit / implizit / überhaupt nicht }

Verhalten am Ende eines 'Programmes': offene Transfers werden abgebrochen

Anhang II

Vergleich einiger elementarer Eigenschaften der Rechner

- Siemens 306
- Siemens 330
- DEC-PDP 11
- MOTOROLA 6800
- INTEL 8080
- [CIMIC-1]

	S 306	S 330	PDP 11	M 6800	I 8080	Cimic-1	Cimic-2
Register							
Mehrzweck-		8 + 8	7		6*		
Arithmetik-	2			2	1	1	
Basisadreß-	2						
Index-				1		1	
Stack-				1	1		
Operanden							
-zahl	1	< 2	≤ 2	< 2	< 2	1	
-art	wort	wort, byte, bit	wort ≙ byte			PEARL -typ	

◯: Adressierungsmodus vorhanden
(...): Herstellerbezeichnung für Adressierungsmodus

Adressierungsmodi+)							
ausgewähltes Register: enthält Operand		◯	(a)	(b)	(a)		
ausgewähltes Register: zeigt auf Operand in-& dekrementiert Zeiger		◯	(c)				
ausgewähltes Register: und Befehlsfolgewort ergeben Opnd.-Adresse		◯	(d)	(d)			
ausgewähltes Register: enthält Opnd.-Adresse		◯	(e)		(f)		
ausgewähltes Register: zeigt auf Opnd.-Adresse in-& dekrementiert Zeiger		◯	(g)				
ausgewähltes Register: und Befehlsfolgewort zeigt auf Opnd.-Adresse		◯	(h)				
Befehls(-Folge)Wort enthält Operand	◯			(ii)	(i)	◯	
Befehls(-Folge)Wort enthält Opnd.-Adresse	(j)			(j)	(j)	◯	
Bef.Zähler und Befehls(-Folge) -Wort ergeben Opnd.-Adresse				(k)			
ausgewähltes Wort enthält Opnd.-Adresse	(l)						

*) nur "temporär"

+) nicht notwendig vollständig

a) "register"
b) "implied"
c) "autoinc."
d) "indexed"
e) "register deferred"
f) "register indirect"
g) "autoinc. deferred"
h) "index deferred"
i) "immediate"
j) "direct"
k) "relative"
l) "substituiert"

	S 306	S 330	PDP 11	M 6800	I 8080	Cimic-1	Cimic-2
Register							
Mehrzweck-		8 + 8	7		6*		
Arithmetik-	2			2	1	1	
Basisadreß-	2						
Index-				1		1	
Stack-				1	1		
Operanden							
-zahl	1	< 2	≤ 2	< 2	< 2	1	
-art	wort	wort, byte, bit	wort ≙ byte			PEARL -typ	

Adressierungsmodi+)		S 306	S 330	PDP 11	M 6800	I 8080	Cimic-1	Cimic-2
ausgewähltes Register	enthält Operand		○	ⓐ	ⓑ	ⓐ		
	zeigt auf Operand in-& dekrementiert Zeiger		○	ⓒ				
	und Befehlsfolgewort ergeben Opnd.-Adresse		○	ⓓ	ⓓ			
	enthält Opnd.-Adresse		○	ⓔ		ⓕ		
	zeigt auf Opnd.-Adresse in-& dekrementiert Zeiger		○	ⓖ				
	und Befehlsfolgewort zeigt auf Opnd.-Adresse		○	ⓗ				
Befehls(-Folge)Wort enthält Operand		○			ⓘ	ⓘ	○	
Befehls(-Folge)Wort enthält Opnd.-Adresse		ⓙ			ⓙ	ⓙ	○	
Bef.Zähler und Befehls(-Folge) -Wort ergeben Opnd.-Adresse					ⓚ			
ausgewähltes Wort enthält Opnd.-Adresse		ⓛ						

○: Adressierungsmodus vorhanden
(...): Herstellerbezeichnung für Adressierungsmodus

*) nur "temporär"

+) nicht notwendig vollständig

a) "register"
b) "implied"
c) "autoinc."
d) "indexed"
e) "register deferred"
f) "register indirect"
g) "autoinc. deferred"
h) "index deferred"
i) "immediate"
j) "direct"
k) "relative"
l) "substituiert"