1. Systemstruktur

1. Das System.016 ist ein hierarchisch strukturiertes universelles Multiprozessor/Multicomputersystem mit heterogenen verteilten Speichermitteln. Im Rahmen des System.016 kann eine kompatible Familie von Hoch- und Höchstleistungsrechnern geschaffen werden. Es ist für numerische und nichtnumerische Anwendungen vorgesehen, im besonderen für wissensbasierte transaktionsorientierte Anwendungssysteme. Eine typische Ausprägung ist das bedienerfreie 1-Schrank-Rechenzentrum, das z. B. als Zentrale oder als Server in Netzwerk- Konfigurationen eingesetzt wird.
(Mit S.016 soll die Idee der kompatiblen Rechnerfamilie wieder aufgegriffen werden, wie sie Mitte der 60er Jahre mit dem System/360 verwirklicht wurde, und zwar für die 90er Jahre, beruhend auf den zwischenzeitlich gesammelten Erfahrungen und gezielt für die dann verfügbaren Technologien. Die S.016-Architektur wird von Grund auf neu ausgearbeitet; ohne Rücksicht auf "Kompatibilität", aber mit dem Ziel der Nutzbarkeit eingeführterx~~eingeführter~~ Verarbeitungsmodelle, Programmiersprachen, vorhandener Datenbestände, Programme usw. Leistungsmäßig muß die kleinste S.016- Konfiguration technologisch vergleichbare Personalcomputer und Multimikroprozessorsysteme deutlich übertreffen.)

2. Überblick über die Struktur. Bild 1 zeigt die grundsätzliche Struktur des S.016. Die Komponenten werden nachfolgend erläutert:

- INPUT-OUTPUT-SUBSYSTEM IOS. Damit ist das S.016 mit der Außenwelt verbunden. Das betrifft im wesentlichen Netzwerke und Terminals, aber auch Interfaces zur direkten Prozeßkopplung, zu Fremdsystemen und zu speziellen Peripheriegeräten.

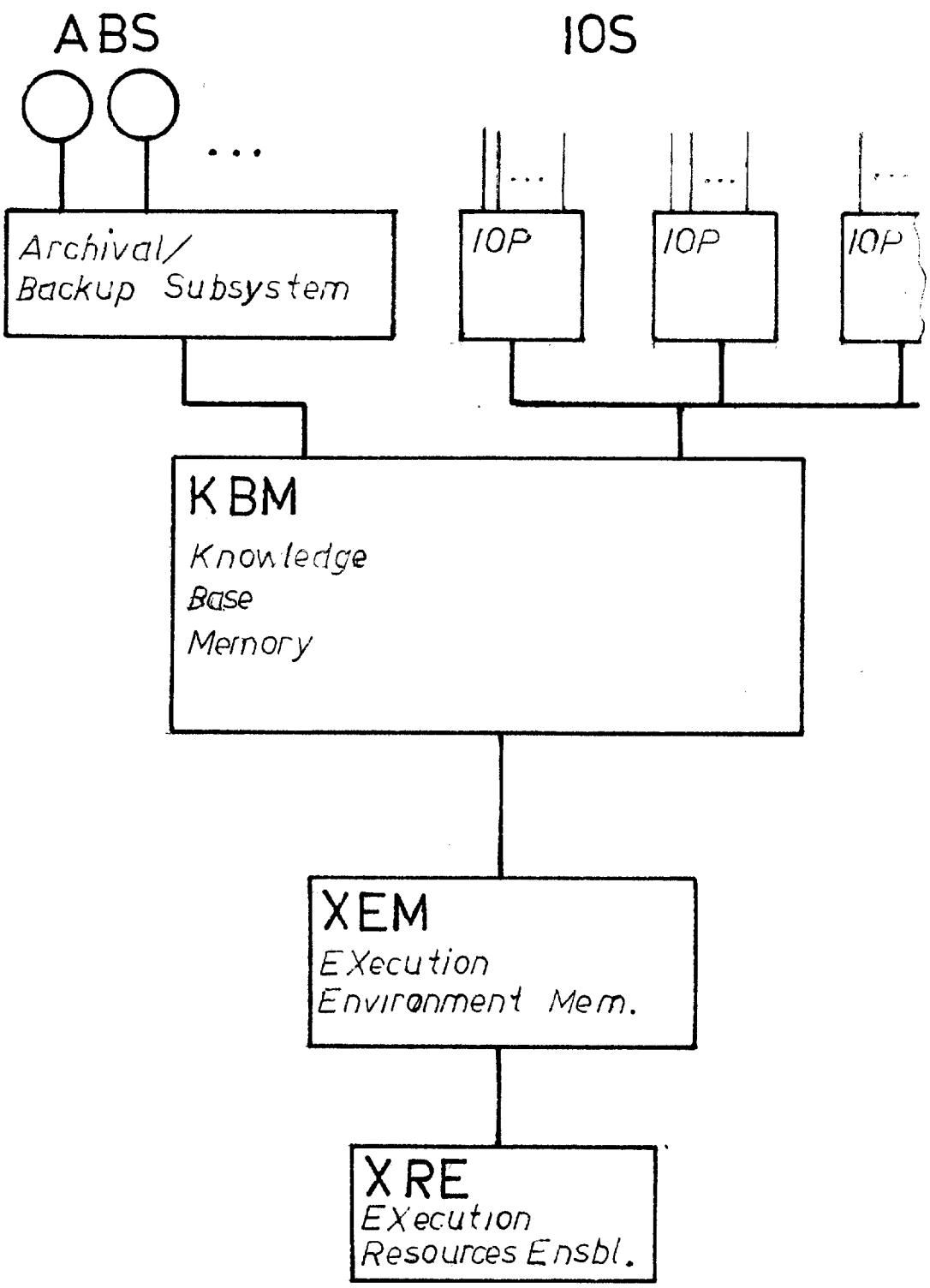


Bild 1.1 System.016 overview

- ARCHIVAL/BACKUP-SUBSYSTEM ABS. Dieses gewährleistet den Erhalt der Datenbestände, auch bei Außerbetriebsetzung des Systems, sowie die Wiederherstellung von Datenbeständen nach Fehlersituationen. Dazu sind direkt angeschlossene Massenspeicher vorgesehen (Magnetplatten, optische Platten, Magnetbandsubsysteme). Es wird ein vollständig bedienerfreies "Handling" angestrebt. Dazu muß das ABS Fehlertoleranzeigenschaften besitzen.

- KNOWLEDGE BASE MEMORY KBM. Dieses Subsystem stellt die Datenbestände zur Nutzung bereit und gewährleistet den sicheren Zugriff.

- EXECUTION ENVIRONMENT MEMORY XEM. Dieser Speicherkomplex hält die Daten, die aktuell zur Verarbeitung bestimmt sind, und nimmt Programme und Resultate auf.

- EXECUTION RESOURCES ENSEMBLE XRE. Dieses Subsystem enthält Verarbeitungsschaltungen, Steuermittel und Schnellspeicher.

Alle Subsysteme verfügen über eigene Prozessoren und zusätzliche lokale Speichermittel. Alle Subsysteme sind über reguläre Interfaces miteinander verbunden.
Das IOS besteht aus standardisierten I/O- Prozessoren, die an Multimaster- Bussysteme angeschlossen und mit jeweils spezifischen Interface- Koppelstufen beschaltet sind.
Das ABS enthält einen zentralen Verwaltungsprozessor sowie spezifische Steuermittel für die einzelnen Massenspeicher.
Das KBM- Subsystem enthält den zentralen Supervisor- Prozessor sowie lokale Prozessoren in den einzelnen Speichermoduln.
Für das gesamte System ist ein Diagnoseprozessor vorgesehen.
Ein XRE- Subsystem umfaßt praktisch 4 Prozessoren, die in verschiedenen Konfigurationen betrieben werden können.

3. Der S.016- Architektur liegen folgende Prinzipien zugrunde:

- das objektorientierte Verarbeitungsmodell
- das Ressourcen- Paradigma
- das Prinzip der kontrollierten Kardinalität
- das Prinzip der Vergegenständlichten Abstraktionen.

4. Objektorientiertes Verarbeitungsmodell. Alle Informationsstrukturen des S.016 werden als Objekte betrachtet. Objekte sind gleichsam die abstrakten,aus logischer Sicht gesehenen Behälter der Information. Die Gesamtheit aller Objekte wird als geordnete Menge angesehen. Somit kann jedes Objekt eindeutig durch seine Ordinalzahl bezeichnet werden. Auf Adressenangaben wird weitgehend verzichtet. Ordinalzahlen werden nur dann in Adressen umgewandelt, wenn tatsächlich zum betreffenden Objekt zugegriffen wird. Diese Wandlungen werden durch technische Mittel unterstützt. Das S.016 stellt so einpe einheitliche technische Grundlage zur Verfügung, mit der verschiedene Sprachkonzepte bzw. Verarbeitungsmodelle implementiert werden können. So hat man beispielsweise die Wahl, jede einzelne Variable als ein Objekt aufzufassen oder die Vorkehrungen der Objektverwaltung lediglich dazu auszunutzen, um die Gesamtheit der Variablen eines Programms zur Laufzeit zugänglich zu machen, wobei die einzelne Variable dann auf herkömmliche Weise adressiert wird.
Neben der Objektorientierung ist das Verarbeitungsmodell gekennzeichnet durch die Trennung von funktioneller Zuweisung und Zuordnung sowie durch das Prinzip der Wertübergabe ("call by value") .

5. Ressourcen- Paradigma. Die gesamte Hardware des S.016 wird als Sammlung von Ressourcen aufgefaßt. Die Ressourcen an sich (Verarbeitungsschaltungen, Datenwege, Speichermittel usw.) bestimmen nach Anzahl und Auslegung unmittelbar die Maximalleistung eines Systems. Die S.016- Architektur ist darauf ausgerichtet, die Ressourcen in jedem Zeitpunkt so gut wie möglich mit nützlicher, also zur Lösung von Anwendungsaufgaben unmittelbar beitragender Arbeit auszulasten.

6. Kontrollierte Kardinalität. Die Ressourcen werden als geordnete endliche Mengen aufgefaßt. Die maximale Kardinalität einer jeden solchen Menge wird nach bestimmten Kriterien festgelegt. Manche Kardinalitäten werden im Rahmen der Architekturdefinition als Festwerte bestimmt.
Damit sollen optimierenden Compilern feste Ziele über das Modellspektrum und die Lebensdauer der S.016- Architektur geboten werden.
(In herkömmlichen Architekturen sind nur wenige Ressourcen so festgelegt, z. B. die Registersätze oder die Länge der Speicheradresse. Hier sollen darüber hinaus alle Speicheranordnungen einschließlich der Schnellspeicher, die Anzahl und Verbindungsstrukturen von Verarbeitungswerken bzw. Prozessoren usw. bis hin zur Steuerwirkung in jedem Maschinenzyklus eindeutig festgelegt werden.)

7. Vergegenständlichte Abstraktionen. Die Architektur des S.016 beruht auf grundlegenden Konzepten der Informationsverarbeitung, die in leistungsfähigen Schaltmitteln direkt vergegenständlicht sind, also auf wesentlichen, leistungsbestimmenden Abstraktionen (Datentypen + Operationen + Ablaufprinzipien), die die Tiefenstrukturen der jeweiligen Verarbeitungsprozesse widerspiegeln.
(Meßtechnische Erkenntnisse werden vorwiegend für die Festlegung von Kardinalitäten herangezogen, z. B. für zweckgerechte Bemessung von Speichern. Für die Gestaltung von Befehlen und Informationsstrukturen wird hingegen kaum auf meßtechnische Erkenntnisse zurückgegriffen, da diese sehr stark von konventionellen Architekturen und Programmiergepflogenheiten geprägt sind. Demgegenüber geht die S.016-Architektur von allgemein anerkannten Operationen und Informationsstrukturen aus, die aus einschlägigen Grundlagen-Wissenschaften stammen (z. B. numerischer Mathematik, Linguistik usw.). Daneben sind ebenfalls allgemein anerkannte elementare Operationen und Datenstrukturen vorgesehen, wobei als Erfahrungsgrundlage auf eingeführte Maschinenarchitekturen, Programmiersprachen und Betriebssystemkonzepte zurückgegirffen wird.)

8. Die Skalierung der S.016- Architektur für ein preis- und leistungsmäßig abgestuftes Modellspektrum wird durch folgende Maßnahmen erreicht:

- verschiedene Technologien
- verschiedene Speicherausstattung (Speicherkapazität, Zykluszeit)
- Multiprozessorkonfigurationen.

(Über verschieden breite Verarbeitungsschaltungen und Datenwege ist zu entscheiden. Vorteil: Aufwandsverringerung. Nachteile: 1. mehrfache Aufwendungen für Entwicklung und Verifizierung, 2. kein echter Hardware- Baukasten; nicht alle Moduln sind freizügig untereinander kombinierbar.)

9. Das Grundschema der Multiprozessorkonfiguration ist der binäre Baum (Bild 1.2). An ein XEM können 2 XRE angeschlossen werden, an ein KBM 2 XEM. Das ergibt zunächst ein einzelnes System aus 2 XEM und 4 XRE, womit 16 Verarbeitungsprozessoren verfügbar sind.
Noch größere Komplexe sind durch Zusammenfassungegn mit übergeordneten KBM- Moduln aufzubauen. Dabei sind die Wissensbasisspeicher, E/A- Mittel usw. zwar technisch in Baumform verschaltet, sie werden aber logisch als eine Ebene betrachtet.
Eine Konfiguration aus 3 KBM, 4 XEM und 8 XRE (32 Verarbeitungsprozessoren) wird noch als sinnfällig angesehen; noch größere Systeme werden dann sicherlich besser als Netzwerk-Konfigurationen aufgebaut. Bild 1.3

10. Grundlegende Kardinalitäten der Speicherausstattung:

- jedes KBM- Subsystem enthält ein Aktives Memory Array (AMA) aus 16 Moduln. Jeder Modul hat einen eigenen Prozessor, alle Moduln sind über Crossbar- Netzwerk untereinander verbunden.

- jedes XRE- Subsystem enthält 16 Moduln. Zu jedem Modul sind wahlfreie Zugriffe möglich.

System.016
Binary Tree
Multiprocessor

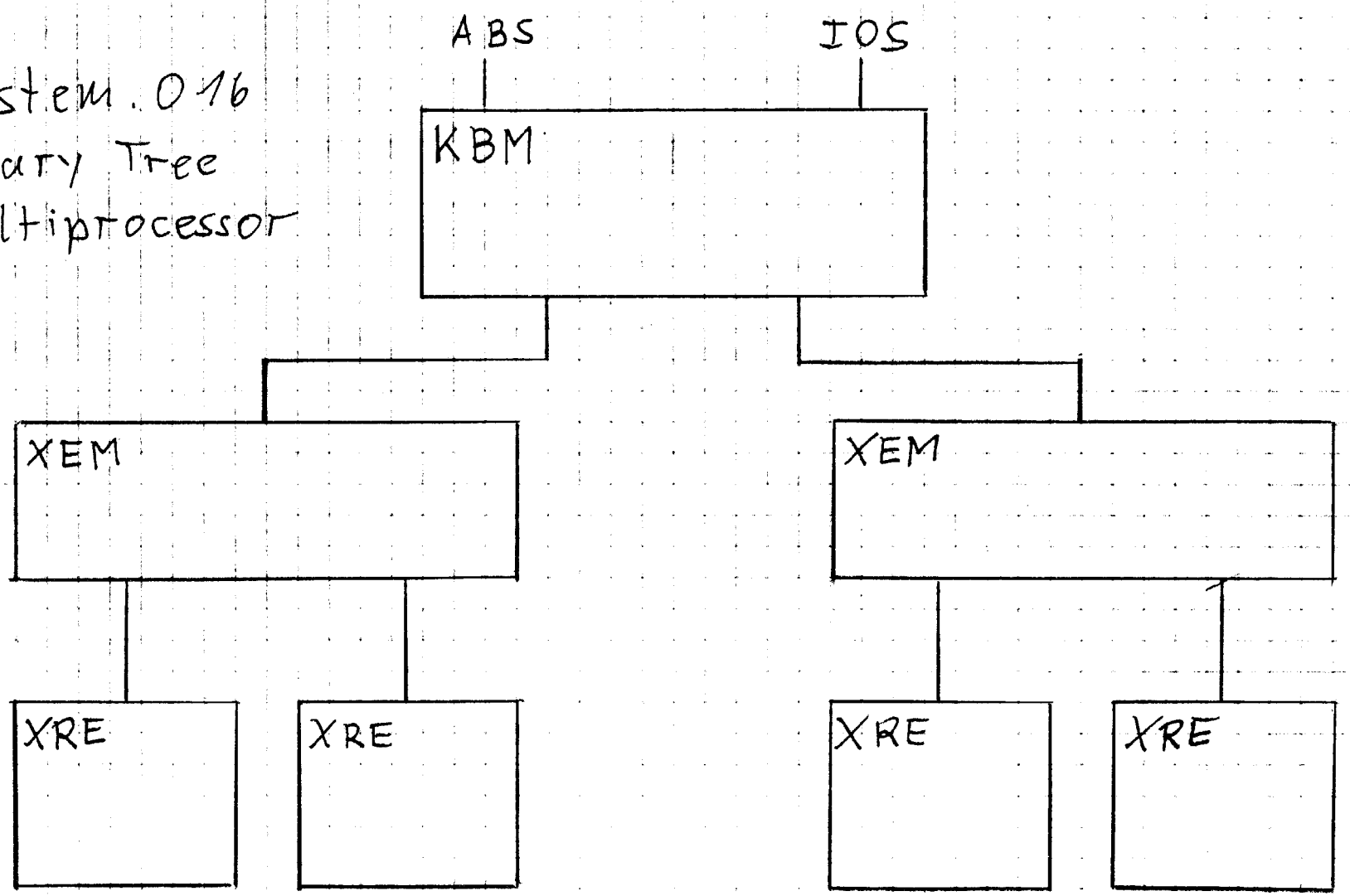
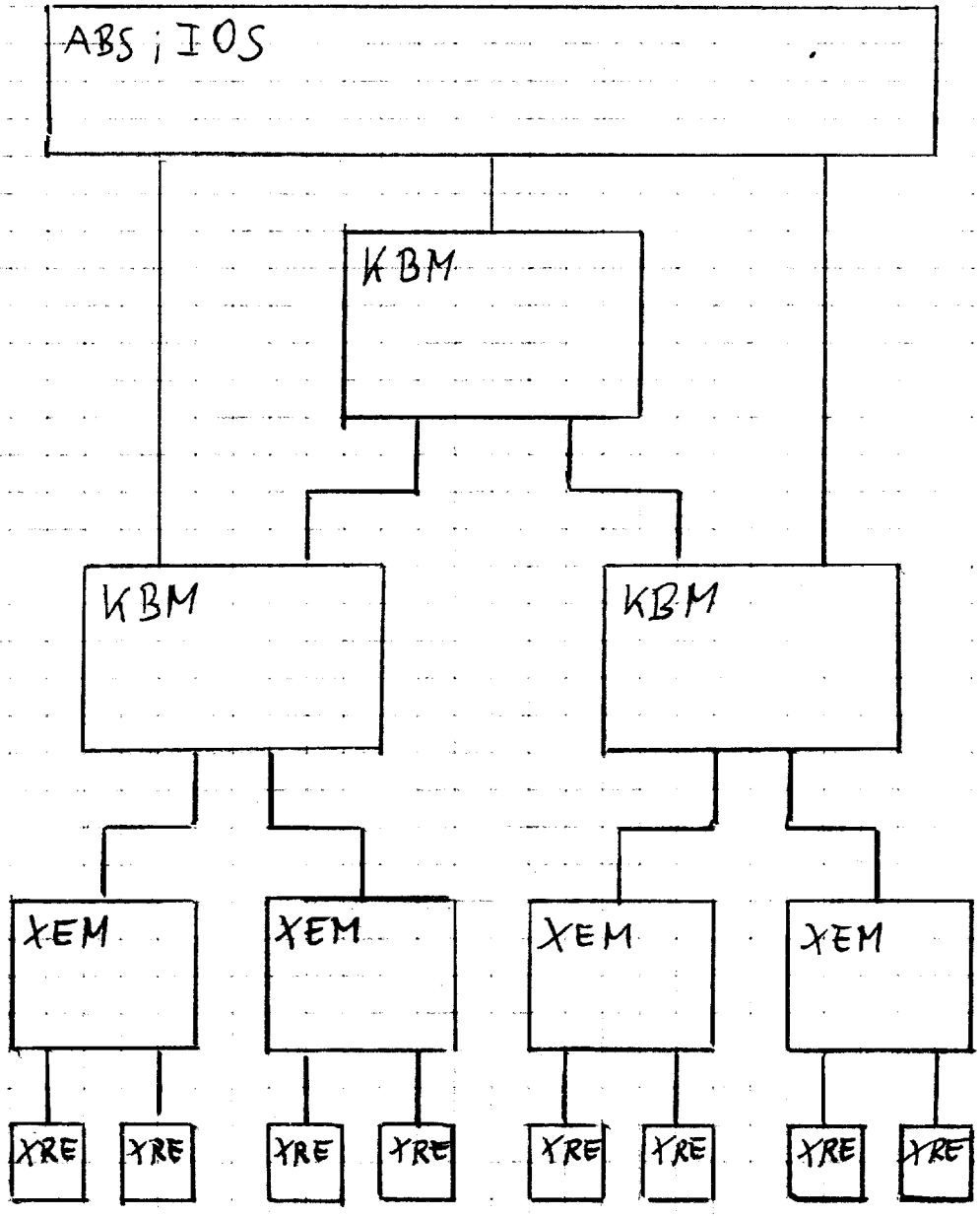


Bild 1.2



System. 016 Extended Binary Tree Multiprocessor

Bild 1.3

11. Grundlage aller S.016- Datenstrukturen ist ein Maschinenwort von 96 bit zuzüglich 8 TAG- Bits (sowie weiteren Bits für Fehlererkennung bzw. -korrektur), im folgenden als Eimer bezeichnet.

12. Es gibt grundsätzlich 3 Typen von RAM- Strukturen bzw. -Schaltkreisen:

- Schnellspeicher zu 256, 1024, 4096, 16k bzw. 64 k Eimern, die im Rahmen des schnellsten Maschinenzyklus betreibbar sind (Richtwert: 25 ns).

- Wahlfrei adressierbare Speicher für 4M...16M Eimer, zu denen mit dem Doppelten des schnellsten Logikzyklus zugegriffen werden kann. Diese kommen im XEM sowie als Steuerspeicher in den übergeordneten Komponenten zum Einsatz. Im Verlauf der technologischen Entwicklung ist anzustreben, diese in der Zykluszeit auf die Größenordnung des schnellsten Maschinenzyklus zu bringen. Kapazitätserhöhung ist unwichtig.

- Speicher möglichst hoher Kapazität (64 Mbit/Schaltkreis), wobei die Zugriffszeit nicht besonders kritisch ist (Richtwert: das 4...8-fache des Maschinenzyklus). Diese werden im AMA des KBM eingesetzt.

Zum technischen Stand: 1989 sind angekündigt:

- ein statischer 1Mbit RAM mit 9 ns (Hitachi)

- ein 16 Mbit DRAM mit 55...60 ns (NEC).

Schltkreise mit über 10^6 Transistoren können mit einem Maschinenzyklus von 25 ns arbeiten (Intel 80860). Schnelle 4kbit- SRAM haben eine Zugriffszeit von 5 ns.

13. In den KBM und XEM ist jeder Speichermodul aus direkten Zusammenschaltungen von Speicherschaltkreisen mit Adressierungsschaltungen, Mitteln zur Schreib- und Lesesteuerung, zur Fehlerkontrolle usw. aufgebaut. Diese Schaltmittel werden als Adressen- und Datenweglogik (ADL) bezeichnet

14. Für die Realisierung des S.016 wird zwischen verschiedenen Ebenen unterschieden:

- Level 1- Maschinen beruhen auf 4 Mbit- DRAMs
- Level 2- Maschinen beruhen auf 16 Mbit DRAMs
- Level 3- Maschinen nutzen schnelle 16 Mbit DRAMs in den XEM und 64 Mbit DRAMs in den KBM
- Hypothetische Level 4- Maschinen sind durch 256 Mbit DRAMs in den KBM gekennzeichnet.

Für die kleinste Level 1- Maschine wird ein Zyklus von 100 ns angenommen. Außerdem sind KBM und XEM physisch identisch (XRE ist direkt an KBM angeschlossen, in den AMA- Moduln ist eine entsprechende Aufteilung vorgesehen, Bild 1.4). Ab Level 2 beträgt der Maschinenzyklus 25 ns. In jedem XRE können in 4 Werken 8 verkettete Gleitkommaoperationen ablaufen (MULTIPLY-ADD- Verkettung). Tafel 1.1 zeigt unter diesen Voraussetzungen verschiedene Aufwands- und Leistungsabstufungen.

Lfd. Nr	Speicher-schaltkr.	Kapazität bei Interleaving-Rate 1	4	8
1	4M	64M	256M	—
2	16M	256M	1G	4G
3	64M	1G	4G	8G
4	256M	4G	16G	32G
Schaltkreisbedarf		2000	7000	13000

Tafel 1.1

Level	HBM-Nr.	HBM MBytes	XEM MBytes	XRE	Prozessoren	parall OP.	Zyklus	MFLOP	Schaltkr.
1.1	1	64M	—	1	4	8	100	80	3000
1.2	1	256M	64M	2	8	16	100	160	10 000
2.2	2	1G	64M	2	8	16	25	640	10 000
2.3	2	1G	2×64M	4	16	32	25	1280	13000
3.4	3	24G	4×256M	8	32	64	25	2560	50000

2. Die Ebenen der Abstraktion

1. Wissen. Im S.016 wird Wissen in Form permanent gespeicherter, in sich veränderbarer Informationsstrukturen einschließlich explizit codierter Strukturbeschreibungen, Bedeutungen und relationaler Beziehungen gespeichert bzw. verarbeitet. (Die Wissensbasis enthält selbstbeschreibende relationale Strukturen.)

2. Objekte. Dies sind die logischen Einheiten der Informationsstrukturen des S.016.

3. Behälter. Behälter sind die Einheiten der Abstraktion für die Verwaltung der technischen Speichermittel.

4. Vorkehrungen zur Wissensspeicherung und -verwaltung sind nicht unmittelbar vergegenständlicht. Die Architektur des S.016 beruht vielmehr ausschließlich auf dem objektorientierten Verarbeitungsmodell, und dessen Konzepte sind das Mittel, um die Wissensbasis zu implementieren.
In technischer Hinsicht ist das S.016 für relationale Operationen über halbgeordnete binär codierte Mengen besonders leistungsfähig ausgelegt (betrifft Vereinigung, Durchschnittsbildung usw.). Solche Operationen sind für die Wissensverarbeitung von entscheidender Bedeutung. (Obwohl regelbasierte Konzepte wie Prolog durchaus implementiert werden können, ist die Wissensverarbeitung vorzugsweise auf das Durchmustern von Mengen ausgerichtet: "memory based reasoning".)
Die relationalen Operationen werden für die interne Objekt- und Behälterverwaltung mitbenutzt (z. B. für die Typprüfung zur Laufzeit).
Objekt- und Behälterverwaltung sind völlig voneinander entkoppelt (die Objektverwaltung stellt einen Positions- Identifier von 44 bit zur Verfügung, der von der Behälterverwaltung entsprechend interpretiert wird).

5. Systemsprache. Die Vorzugssprache für die Systemimplementierung ist Ada. Die gesamte S.016- Architektur wird in Form von Ada- Packages beschrieben. Alle S.016- Systemfunktionen sind also faktisch Ada- Programme und können als solche in weitem Umfang modifiziert werden. Manche dieser packages sind vergegenständlicht.
Als Compilerziel gibt es einen einzigen Prozessortyp (den PK 1) in 2 Modifikationen: für die Quad- Anordnung im XRE und als Einzelprozessor in allen anderen Komplexen.
Somit gibt es 2 Optimierungsziele für den Compiler: die Nutzung des Einzelprozessors und die des QUAD im XRE (evtl. zusätzlich die Nutzung des XRE- TWIN an einem XEM und ggf. die Parallelisierung in Mehrprozessorkonfigurationen).

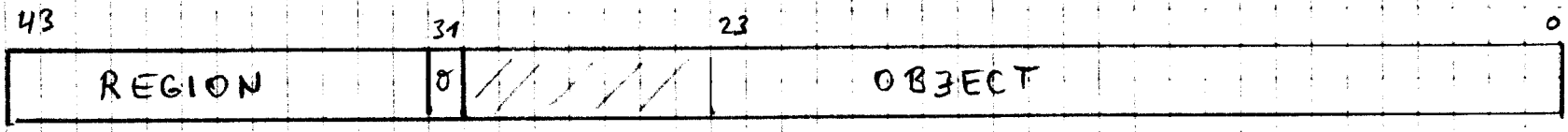
3. Die statische Objektstruktur

1. Das einzelne Objekt. Es ist gekennzeichnet durch seine Zugehörigkeit zu einer Menge von Objekten, durch seine Ordinalzahl in dieser Menge, seinen Typ und seinen Inhalt. Objekte können der Wissensbasis oder einer Laufzeitumgebung (Execution Environment) angehören. Es gibt temporäre und permanente Objekte. Objekte der Laufzeitumgebung sind stets temporär. Für jedes permanente Objekt wird ein einzigartiger Identifikator verwaltet (im Rahmen des ABS). Dieser Unique Object Identifier (UOID) ist nicht löschbar und nicht wiederverwendbar. Er bleibt über die Lebenszeit des Systems erhalten und kennzeichnet eindeutig ein bestimmtes Objekt in einem bestimmten System.
Temporäre Objekte können nach Gebrauch vollständig vernichtet werden. (Es ist aber möglich, temporäre Objekte im ABS zu retten.)

2. Objektidentifier. Jedes permanente Objekt in einem System wird durch einen 44 bit- Identifier bezeichnet. Die Menge aller permanenten Objekte eines Systems ist in 4096 Regionen eingeteilt. Jede Region kann bei Maschinen bis Level 3 bis zu 16 M Objekte enthalten. (Grundsätzlich ist eine 31 bit- Ordinalzahl vorgesehen, mit Bit 32 =0 als "escape"-Vorkehrung.)
In einer Laufzeitumgebung können 64 k Objekte vorgesehen sein. Es sind bis zu 4096 Laufzeitumgebungen definierbar. (Erweiterung auf 1 M Objekte vorgesehen.)
Bild 3.1 gibt einen Überblick über die Objektidentifier.

3. Systemobjekte. Region 0 ist grundsätzlich die Systemregion. Sie enthält alle Objekte, die für den Betrieb des Systems notwendig sind (einschließlich Diagnose, Wartung, Dokumentation usw.). Dazu gehören die 4096 Regionenbeschreibungen und die Objekte der technischen Speicherverwaltung.

Identifier eines Objekts der Wissensbasis



Identifier eines Objekts der Laufzeitumgebung

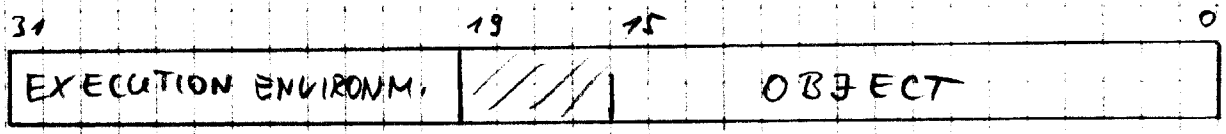


Bild 3.1

Weitere wichtige Objekte:

0 - Maschinenbeschreibung
1 - UOID- Generator
2 - Uhrzeit
3 - Logbuch
4 - Kaltstart.

4. Objektdescriptoren. Für jedes Objekt ist ein Objektdescriptor vorgesehen, der einen Eimer belegt. Alle Objektdeskriptoren einer Region sind in einer Object Reference Table ORT zusammengefaßt, die ihrerseits ein Objekt der Systemregion ist. Der Objektdescriptor enthält eine Angabe über die Position des Objekts in den technischen Speichermitteln, über seine Größe und über seinen Typ.

5. Typen. Bestimmte elementare Objekttypen sind im S.016 vergegenständlicht bzw. vordefiniert. Sie heißen Fundamentaltypen. Darüber hinaus können beliebige anwendungsbezogene Typen deklariert werden. Jeder solche Typ wird in einem Typbeschreibungsobjekt TDO beschrieben. Fundamentaltypen sind teils in den TAG- Bits der Eimer und teils in den Descriptoren codiert.

6. Nutzung der TAG- Bits. 2 Bits dienen zu Monitorzwecken. Sie werden vom Serviceprozessor verwaltet (normale Belegung: 0). 6 Bits liefern eine erste Codierung des Fundamentaltyps. Grundsätzlich wird unterschieden zwischen:

- Daten (static knowledge): 32 Fundamentaltypen
- Programmen (dynamic knowledge): 16 Fundamentaltypen
- strukturell-descriptive Angaben: 16 Fundamentaltypen.

In den Tafeln 3.1 und 3.2 sind die Fundamentaltypen von Datenobjekten und strukturell-descriptiven Angaben aufgezählt.

7. Ein Objekt von einem Fundamentaltyp, das selbst nur einen Eimer belegt, wird direkt in der jeweiligen ORT gespeichert (anstelle eines Descriptors).
Höher aggregierte zusammengesetzte Objekte enthalten einen eigenen Beschreibungsteil. Dieser ist in einen primären und einen sekundären Bereich aufgeteilt. Die Größe des primären Beschreibungsteils ist im Objektdescriptor angegeben, so daß eine Speicherverwaltung unmittelbar erkennen kann, in welchem Umfang Speicherplatz zugeordnet werden muß.
Für manche zusammengesetzte Objekte bestimmter Fundamentaltypen ist die vollständige Objektbeschreibung bereits im Objektdescriptor codiert. Diese Fundamentaltypen sind in Tafel 3.3 aufgezählt. Die Objektdescriptoren zeigt Bild 3.2.

8. Wichtige Kardinalitäten. Ein zusammengesetztes Objekt kann maximal (Beschreibung + Inhalt) 16 M Eimer belegen. (Es sind Code- Reserven vorgesehen, um ab Level 3 auf 2 G Eimer gehen zu können. Grundsätzlich ist aber ein Objekt einer Laufzeitumgebung auf 16 M Eimer beschränkt.)
Ein Programm- Objekt darf maximal 1 M Eimer belegen (dafür sind keine "escape"- Möglichkeiten vorgesehen).
Verbundobjekte dürfen aus maximal 16 M Objekten bestehen.

9. Programmobjekte. Diese sind zusammengesetzte Objekte, die aus einer Zugriffstabelle (Access Reference Table ART) und einem Codeteil bestehen. Die ART dient als Muster für den Aufbau des Activation Record zur Laufzeit. Für jedes Objekt, das vom Programm genutzt wird, ist ein Eimer vorgesehen, der wahlweise (je nach Parameterversorgung) enthalten kann:

- einen Zugriffsdescriptor
- einen Wert
- einen Objektdescriptor
- eine unmittelbare Adressenangabe.

Die ART kann bis zu 4096 Eimer umfassen. Sie kann in einen öffentlichen und einen privaten Teil aufgeteilt werden.
In Programmen, die alle Parameter vorgeprüft übergeben bekommen, ist die ART nicht explizit codiert.

10. Zugriffsdescriptoren. Diese enthalten den jeweiligen Objektidentifier sowie Zugriffsrechte. Weiterhin können bestimmte Selektorangaben unmittelbar codiert werden. Die Formate sind in Bild 3.3 dargestellt.

4. Die technische Speicherverwaltung

1. Behälterverwaltung. Für die Objekte der Wissensbasis ist ein Vorrat von Behältern verschiedener Größe vorgesehen. Insgesamt sind 4096 Behältertypen definierbar, in Level 1 und 2 64. Jeder Behältertyp wird durch einen Descriptor beschrieben. Alle Descriptoren sind in der Behältertyptabelle zusammengefaßt, die ihrerseits ein Systemobjekt ist.
Für jeden Behältertyp gibt es eine Behältertabelle, in der alle Behälter verzeichnet sind, sowie eine Freiliste für die freien Behälter.
Ein Behälter umfaßt mindestens 2 und höchstens 16 M Eimer (ab Level 3 ggf. mehr). Behälter von 2 bzw. 4 Eimern sind unmittelbar in der Behältertabelle zusammengefaßt. Für alle größeren Behälter enthält die Behältertabelle einen Zeiger (die Tabelle hat für jeden Behälter 4 Eimer, so daß die am häufigsten benutzten Angaben der primären Objektbeschreibung unmittelbar zugänglich sind).
Jedes Objekt wird in einem ausreichenden Behälter untergebracht.
Die gesamte Behälterverwaltung ist programmseitig steuerbar, so daß sich eine dynamische Anpassung an die Anwendungserfordernisse verwirklichen läßt.

2. Laufzeitumgebung. Die Speichermittel der Laufzeitumgebung werden im wesentlichen als Stack verwaltet. Es ist gewährleistet, daß aus 2 Objekten von 16 M Eimern ein drittes Objekt gleicher Größe erzeugt werden kann (der Top of Stack steht immer zur Aufnahme von 48 M Eimern zur Verfügung).
Des weiteren können temporäre Objekte statisch im XEM gehalten werden. Das ist Angelegenheit des Laufzeitsystems; eine Behälterverwaltung wird nicht durch Vergegenständlichung unterstützt.

3. Direktadressierung. Zum direkten Zugriff auf alle Speichermittel des Systems ist eine 64 bit- Eimeradresse vorgesehen.
(Normalerweise wird diese Zugriffsform nur von Systemroutinen verwendet. Es ist aber grundsätzlich möglich, zur Laufzeit das objektorientierte Verarbeitungsmodell zu verlassen und die direkte Eimeradressierung zu verwenden, z. B. bei Nutzung der S.016- Hardware als "number cruncher".)

4. Backup und Archivierung. Für den Wiederanlauf bleiben Objekte in ihrem "alten" Zustand auf Hintergrundspeichern des ABS erhalten. Das Zurückschreiben wird durch die Ereignissteuerung beeinflußt (in dem Ereignis, das den betreffenden Verarbeitungsauftrag auslöst, sind die Einzelheiten des Zurückschreibens angegeben).
Das Archivieren von Datenbeständen wird durch Systemroutinen gesteuert, die im Rahmen des ABS vorgesehen sind.
Für die Datenspeicherung im ABS ist eine besondere Behälterverwaltung vorgesehen. Dabei werden kleinere Behälter in größeren Behälterformaten der Externspeicher zusammengefaßt.
(Diese Behälterformate sind auf die Gegebenheiten der Massenspeicher abgestimmt, z. B. auf die Speicherkapazität einer Magnetplatten- Spur.)

5. Die dynamische Objektorganisation

1. Auslösung. Im S.016 wird ein Programmlauf durch ein Ereignis ausgelöst. Dies wird von physischen Routinen in den I/O- Prozessoren bzw. vom Supervisor- Prozessor veranlaßt (Ursachen sind z. B. externe physische Ereignisse, wie Meldungen von Terminals, oder auch Signale des internen Zeitgebers). Ein einmal laufendes Programm kann seinerseits Ereignisse auslösen.
Das Ereignis ist gekennzeichnet durch ein besonderes Objekt (Event Control Object ECO), das den Identifier des zu startenden Programms, den aufbereiteten aktuellen Activation Record (mit aktuellen Parametern) sowie Angaben zur Ressourcennutzung, zur Priorität und zur Objektverwaltung enthält.

2. Programmstart. Wird ein Programm gestartet, so wird zunächst der aktuelle Activation Record auf dem Laufzeitstack (des XRE bzw. des jeweiligen Prozessors) aufgebaut. Bei Ereignisauslösung wird er aus dem ECO kopiert. Ansonsten (bei Aufrufen) wird die ART kopiert und dann mit den aktuellen Parametern gefüllt. Dann wird der erste Befehl abgearbeitet.

3. Berechtigungs- und Typprüfungen. Die Berechtigungsprüfungen des Anwendungssystems obliegen dem Zugriffsystem der Wissensbasis. Dazu werden besondere relationale Strukturen aufgebaut, permanent gehalten und im Nutzungsfall abgefragt.
In der ART ist eine einfache Typprüfung angebbar (Typvergleich beim Holen der aktuellen Parameter). Komplizierte Typprüfungen zur Laufzeit, wie sie z. B. bei LISP oder Smalltalk notwendig sind, müssen programmiert werden. Dazu können die vorhandenen relationalen Operationen ausgenutzt werden (Aufbau von Entscheidungstabellen, die sinngemäß durchmustert werden; Type Control Object TCO).