Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 79 (1988)

Heft: 17

Artikel: Speicherorganisation in Mikroprozessorsystemen

Autor: Götte, A.

DOI: https://doi.org/10.5169/seals-904076

Nutzungsbedingungen

Die ETH-Bibliothek ist die Anbieterin der digitalisierten Zeitschriften auf E-Periodica. Sie besitzt keine Urheberrechte an den Zeitschriften und ist nicht verantwortlich für deren Inhalte. Die Rechte liegen in der Regel bei den Herausgebern beziehungsweise den externen Rechteinhabern. Das Veröffentlichen von Bildern in Print- und Online-Publikationen sowie auf Social Media-Kanälen oder Webseiten ist nur mit vorheriger Genehmigung der Rechteinhaber erlaubt. Mehr erfahren

Conditions d'utilisation

L'ETH Library est le fournisseur des revues numérisées. Elle ne détient aucun droit d'auteur sur les revues et n'est pas responsable de leur contenu. En règle générale, les droits sont détenus par les éditeurs ou les détenteurs de droits externes. La reproduction d'images dans des publications imprimées ou en ligne ainsi que sur des canaux de médias sociaux ou des sites web n'est autorisée qu'avec l'accord préalable des détenteurs des droits. En savoir plus

Terms of use

The ETH Library is the provider of the digitised journals. It does not own any copyrights to the journals and is not responsible for their content. The rights usually lie with the publishers or the external rights holders. Publishing images in print and online publications, as well as on social media channels or websites, is only permitted with the prior consent of the rights holders. Find out more

Download PDF: 21.01.2026

ETH-Bibliothek Zürich, E-Periodica, https://www.e-periodica.ch

Speicherorganisation in Mikroprozessorsystemen

A. Götte

Die Wortlänge und damit der Adressumfang von Mikrocomputern ist nicht immer optimal der Speichergrösse angepasst. Dieser Beitrag beschreibt die heute gebräuchlichen Verfahren zur Behebung dieser Schwierigkeit. Wichtige Begriffe wie logischer und physikalischer Adressbereich, virtueller Speicher, Memory Management, Segmentierung und Paging werden ausführlich behandelt.

La longueur de mot et par là le volume d'adresses d'un microordinateur ne sont pas toujours adaptés de manière optimale à la capacité de mémoire. Le présent article décrit les méthodes actuellement en usage pour aplanir les difficultés. Des termes importants tels que domaine logique et physique d'adresse, mémoire virtuelle, management de mémoire, segmentation et pagination sont traités en détail.

Überarbeitete Fassung eines Beitrags in Syslogic Mitteilungen Nr. 7/88.

Adresse des Autors
Andreas Götte, El.-Ing. HTL,
Bucher + Suter AG, Worblaufenstr. 163,
3048 Worblaufen.

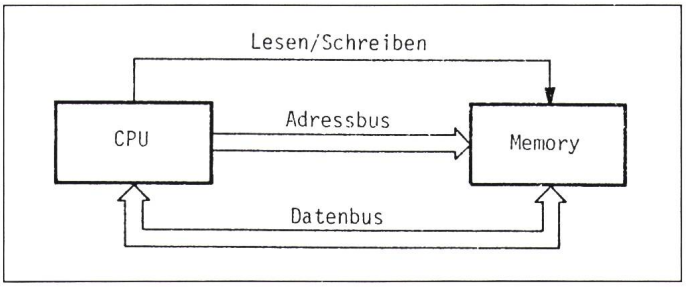
Neben der zentralen Prozessoreinheit (CPU) ist der Speicher (Memory) eine der wesentlichen Komponenten eines Mikroprozessorsystems. Er enthält sowohl den Programmcode (Instruktionen für den Mikroprozessor) wie auch die Programmdaten. Durch Anlegen einer Adresse auf den Adressbus (Fig. 1) wird eine gewünschte Speicherzelle angewählt und eine Information über den Datenbus – entsprechend der Instruktion auf der Steuerleitung – aus dem Memory in die CPU gelesen oder aus der CPU ins Memory geschrieben.

Die Eigenschaften eines Mikrocomputersystemspeichers sind im wesentlichen durch zwei Zahlen bestimmt, nämlich durch die Anzahl *Adressbit* und die Anzahl *Datenbit*. (Die Zugriffszeit hat für diese Betrachtung keine Bedeutung.) Die Anzahl der Adressbit (Breite des Adressbusses) bestimmt die *maximale Grösse* des Speichers. Die heute gebräuchlichen Adressbusbreiten sind in der Tabelle I angegeben. Die Anzahl der Datenbit (Breite des Datenbusses) ergibt die durch einen einzelnen Speicherzugriff ladbare Anzahl Bit. Bei heutigen Mikroprozessoren werden Datenbusbreiten von 8, 16 und 32 Bit verwendet.

Insbesondere bei Mikroprozessoren mit kleinen Wortlängen (8 Bit, 16 Bit) führt der beschränkte Adressierumfang (Tab. I) sehr schnell zu unliebsamen Einschränkungen. Bei Mikroprozessoren mit grosser Wortlänge (32 Bit) anderseits kann die Software mehr Speicherzellen direkt adressieren als unter Umständen vorhanden sind. Wie man die damit verbundenen Probleme meistern kann, wird im folgenden dargelegt. Dabei spielen die Begriffe des logischen, physikalischen und virtuellen Speichers eine wichtige Rolle. Unter *logischem Speicher* versteht man das Speicherabbild, mit dem der Programmcode, also die Software, kommuniziert. Der *physikalische Speicher* dagegen ist der wirklich vorhandene, hardwaremässig zugreifbare Speicher. Bei der einfachsten Mikroprozessorarchitektur gibt es diese Unterscheidung nicht; der logische Speicher ist ein identisches Abbild des physikalischen Speichers, und jede logische Adresse im Programmcode entspricht exakt einer physikalischen Adresse in der Hardware, d.h. einer ganz bestimmten Speicherzelle.

Anzahl Adressbit	Anzahl Speicherzellen
16	65 536 = 64 Kilo
20	1 048 576 = 1 Mega
24	16 777 216 = 16 Mega
32	4 294 967 296 = 4 Giga

Tabelle 1 Adressierbarer Speicher

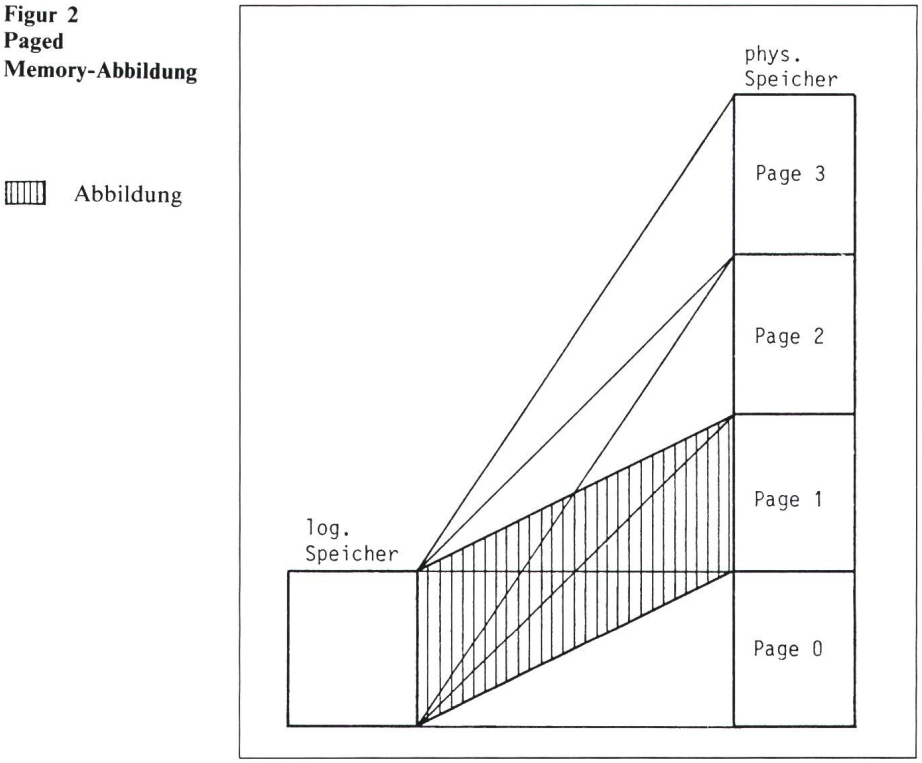


Figur 1 Vereinfachtes Mikroprozessor-Schema

Zur *Erweiterung* des durch die Architektur (genauer durch den Programmcode oder den Mikroprozessor-Chip) beschränkten Speichers werden nun Verfahren angewendet, die den logischen, d.h. adressierbaren Speicher auf einen bestimmten, ausgewählten Teil eines grösseren physikalischen Speichers abbilden. Daraus ergibt sich eine der Haupteigenschaften solcher Systeme, nämlich die, dass zu einem bestimmten Zeitpunkt einer Programmabarbeitung für die Software nicht der ganze physikalische Speicher sichtbar ist, sondern nur der zurzeit abgebildete Speicherbereich.

Im Laufe der Zeit wurden von den verschiedenen Mikrocomputer-Herstellern verschiedene Speicherabbildungsverfahren entwickelt, deren wichtigste im folgenden näher erklärt werden.

Figur 2
Paged Memory-Abbildung



Paged-Memory-Verfahren (Paging)

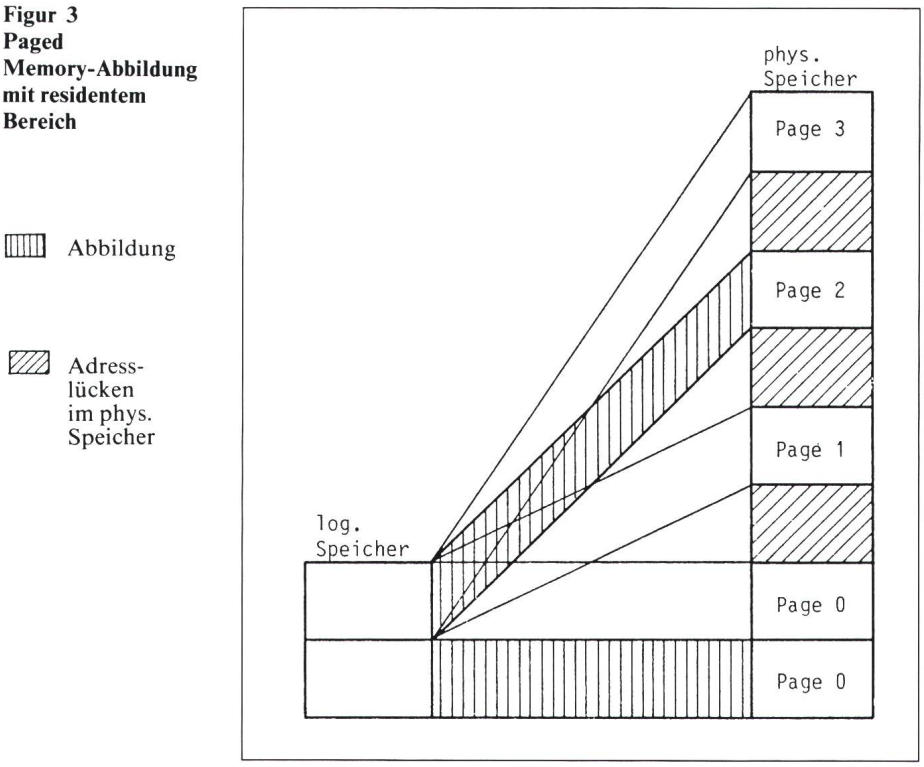
Beim Paging (Aufteilen in Seiten) werden die durch die CPU beschränkten Adressbit [1] durch eine normalerweise ausserhalb der CPU liegende Logik ergänzt, welche durch E/A-Operationen mit der Software verbunden ist. Der physikalisch adressierbare Speicher wird entsprechend der Anzahl der Erweiterungsbit vergrössert: Es entstehen Speicherseiten (Memory Pages), welche durch die Erweiterungsbit angewählt werden. Die verschiedenen Pages (Speicherseiten) können allerdings nicht gleichzeitig aktiv sein, sondern es bedarf jedesmal einer Umschaltung über die Adresserweiterungsbit. Der Struktur eines derartigen erweiterten Speichers sind deshalb gewisse Randbedingungen vorgegeben.

Da gleichzeitig immer nur ein physikalischer Bereich (Page) auf den logischen Bereich abgebildet ist, ergeben sich für die Software gewisse Probleme. Würde man eine Speicherstruktur, wie z.B. in Figur 2 dargestellt, ohne weitere Massnahmen realisieren, so müsste man bei der Programmierung beachten, dass nach einer Veränderung der Adresserweiterungsbit (Seitenwechsel) der Programmfluss unterbrochen und auf einer anderen Seite weitergeführt würde. Allein die Vorstellung dieses Vorganges erregt bei Programmierern Schwindel. Soll dieses Problem auch nur einigermassen sauber und klar gelöst werden, muss ein bestimmter Teil des physikalischen Speichers jederzeit und unabhängig von den Adresserweiterungsbits auf den logischen Speicher abgebildet sein (Fig. 3). Dieser Speicherteil wird Grundbereich oder *residenter Bereich* genannt.

In diesen residenten Speicherbereich müssen nun folgende kritische Softwareteile gelegt werden:

- Routinen zum Umschalten der Adresserweiterung (Paging-Mechanismus),
- Codes, die von verschiedenen Programmteilen gemeinsam gebraucht werden (Subroutinen) sowie gemeinsame Daten,

Figur 3
Paged Memory-Abbildung mit residentem Bereich



ELIBA 2

Devisieren. Offerieren. Abrechnen. Alles auf dem Personal Computer.

AN DER SWISSDATA,
AM STAND NR. 434, HALLE 214,
6.-10. SEPT. IN BASEL.

ELIBA 2 DAS 1. PROGRAMM DER ELEKTROBRANCHE.

IBACOM SOFTWARE AG
Ringstrasse 34
CH-7000 Chur
Telefon 081 25 11 55
Telefax 081 24 35 34

Chur · St. Moritz
Vaduz · Geroldswil/ZH

IBACOM
für Computer.

ELIBA 2.
Devisieren. Offerieren. Abrechnen. Alles auf dem Personal Computer.

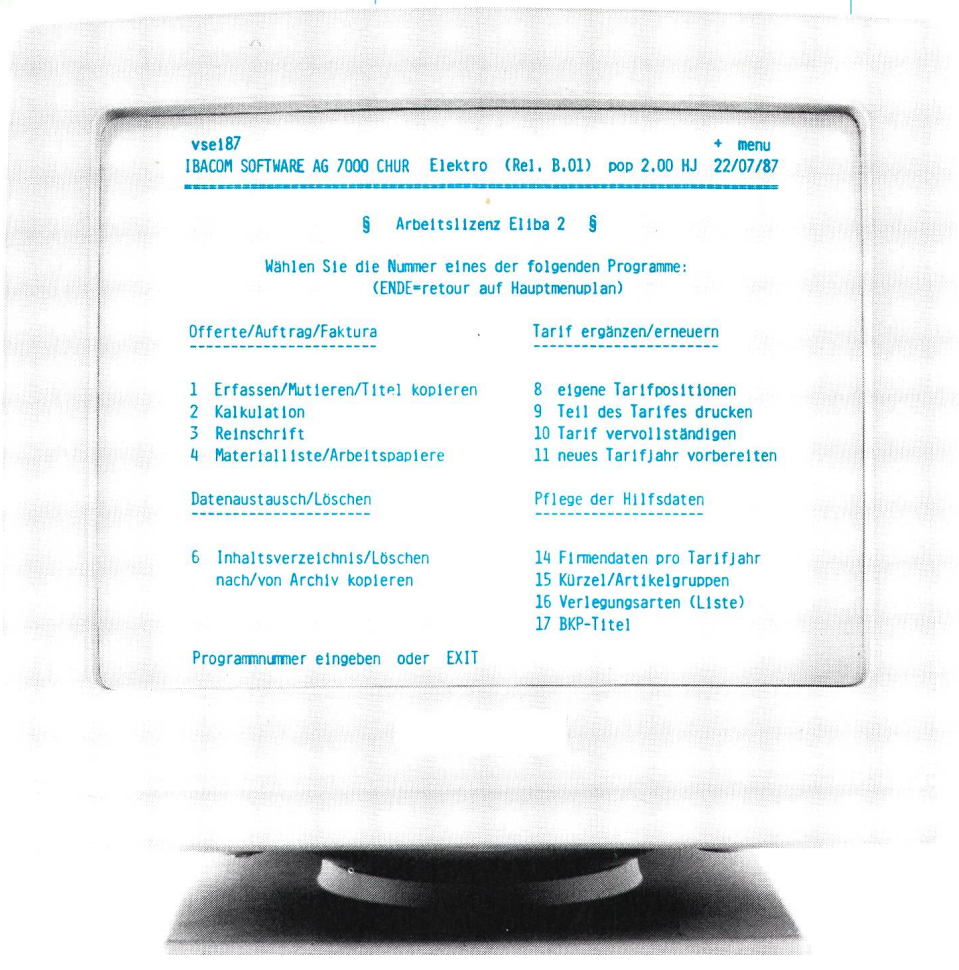
Wenn heutzutage der Meister spätabends sein Tagwerk geleistet hat, geht für ihn die Arbeit oft erst richtig los. Dann muss er noch Berichte schreiben, Offerten und Rechnungen erstellen, Eintragungen in das Original-Devi machen, das Kunden-Buch nachführen und was der zeitraubenden Arbeiten mehr sind. Und weil das für einen Menschen, der eigentlich einen ganz anderen Beruf hat, oft sehr mühsam ist, überlassen immer mehr Elektriker die lästige Büroarbeit einem Computer-System und unserem Software-Programm ELIBA 2.
Dieses verschafft einen schnellen Überblick über alle Leistungen, Zahlungen, Mahnungen und erledigt rasch und präzise alle anfallenden administrativen Arbeiten. Mit dem Resultat, dass man selber etwas weniger erledigt ist.

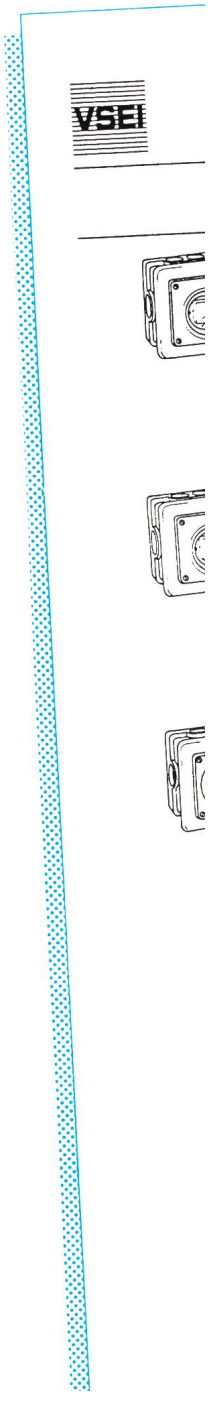
Datenbank:

VSEI-Tarif, Kürzel, eigene Artikel, Materialgruppen, Rabattgruppen, Verlegungsarten, Archivservice, Baukostenplan, Firmenstamm usw.

Funktionsumfang:

Erfassen von Offerten/Al
nungen, mutieren in best
Offerte/Rechnung, autom
sches Duplizieren, Kalkul
nach verschiedenen Fein
(Spezialstundensätze, Ra
gruppen, Rabattsätze usv
Rabattvergabe pro Titel,
Umwandlung in Regie-Off
Drucken von Arbeitspapi
und noch viele weitere M
keiten.





IBACOM
für Computer.

IBACOM AG, CHUR

Schmied Hans, Architekt, 7000 Chur

Badzimmer-Umbau

O F F E R T E

23 E L E K T R O A N L A G E N
****

231 ZENTRALE STARKSTROMANLAGEN
==========

231.2.1 Lichtinstallation

1	KRF 11		1051	40	10	5.40	
2	DRAHT T (4)	1,5	30120	60	48	1.30	
3	CU-DRAHT T	1,5	3012	70	9	2.25	
4	UP-DRUCKSCH. INKL.ANSCHLUESSE	3/1L 10/250	42241	30	1	49.95	
5	UP-STECKD. GR.1 T13 INKL.ANSCHLUESSE	10/250	46111	30	1	44.90	
6	Anschluss an Spiegelkasten		4	N	1	75.00	
7	Anschluss an Spiegelkasten		3	B	1	70.00	
8	Diverse Anpassungen in Regie		1	T			
9	MONTEUR (VSEI-REGIEANSATZ, PER STD.)		90161	09	2	52.00	1
10	Richtpreis Demontage des alten Spiegelkasten		2	R	1	200.00	2

231.2.1 Lichtinstallation 10.00% auf 410.50 ------- 63
==========

231 ZENTRALE STARKSTROMANLAGEN ------- 63
==========

23 E L E K T R O A N L A G E N
****

Gesamttotal ========= 639

639
****

AP-DRUCKSCHALT

VSEI-Pos. Nr. / Einheits-Nr.	VSEI-Positi / Artikeltext
4025	AP-DRU
40251	... IN
40252	... I
4023	AP-DF
40231	...
40232	...
4026	AP-I
40261	...
40262	...
4027	AP
40271	
40272	
4035	A L
40351	
40352	

4036	AP-L-DRUCKSCHALTER ORIENTIERUNGSLAMPE	10/250 6/1P	27,60	41,40	10	0,128	47,95	0,[illegible]	[illegible]	[illegible]
				47,20 *	15	0,154	49,25	0,193	51,25	59,90
					20	0,168	50,00	0,213	52,30	61,25
40361	... INKL. ANSCHLUESSE T		27,60	41,40	10	0,214	52,25	0,271	55,10	65,00
				47,20 *	15	0,240	53,55	0,303	56,75	67,10
					20	0,254	54,30	0,323	57,80	68,45
40362	... INKL. ANSCHLUESSE TT		27,60	41,40	10	0,270	55,15	0,337	58,60	69,45
				47,20 *	15	0,296	56,45	0,369	60,25	71,55

Kleinfakturierung

IBACOM
für Computer.

Dieses Programm ist speziell für die schnelle und bequeme Erstellung von kleinen Rechnungen ausgelegt. Am Bildschirm geben Sie die VSEI-Nummer, den Kürzel oder Ihren eigenen Text ein. Nachdem Sie Rabatt und Skonto eingegeben haben, erhalten Sie per Knopfdruck eine saubere Kleinfaktura.

Kleinfakturierung Elektro

Mai 87/

Beispiel einer Faktura:

Herrn
Walter Mei
Florhofstr
8005 Züric

Zürich, der

Rechnung Nr. 87005

Kundennr.:

1. Materialbezug
---

lt. Bestellung vom .... geliefert
an Ihr Domizil (freier Text)

Kühlschrank Sibir FX 567	1	1200
Abdeckblende	1	80.
Rabatt		10
Total Materialbezug		

2. Installation

Anschluss Ihres Saunaofens an Tableau

Saunaofen XZ-123			
KIR			
Anschluss Tableau	1009	1	680.0
	90405	8	7.0
		2	48.0
Total Installation			

Gesamttotal 30 Tage netto

10 Tage 2% Skonto Fr. 1944.30

Besten Dank für Ihren Auftrag.

IBACOM AG, CHUR

Schmied Hans, Architekt, 7000 Chur

Badzimmer-Umbau

R E C H N U N G

23	E L E K T R O A N L A G E N					
	**********					
231	ZENTRALE STARKSTROMANLAGEN					
	==========					
231.2.1	Lichtinstallation					

1	KRF 11		1051 40			
2	DRAHT T (4)	1,5	30120 60			
3	CU-DRAHT T	1,5	3012 70			
4	UP-DRUCKSCH. INKL.ANSCHLUESSE	3/1L 10/250	42241 30			
5	UP-STECKD. GR.1 T13 INKL.ANSCHLUESSE	10/250	46111 30	1	44.90	44.90
6	Anschluss an Spiegelkasten		4 N	1	75.00	75.00
7	Anschluss an Spiegelkasten		3 B	1	70.00	70.00N
8	Diverse Anpassungen in Regie		1 T			
9	MONTEUR (VSEI-REGIEANSATZ, PER STD.)		90161 09	2	52.00	104.00
10	Richtpreis Demontage des alten Spiegelkasten		2 R	1	200.00	200.00N

231.2.1	Lichtinstallation		10.00% auf	410.50		639.45
						==========

231	ZENTRALE STARKSTROMANLAGEN					639.45
						==========

23	E L E K T R O A N L A G E N					639.45
	**********					==========
	Gesamttotal					639.45
						**********

Autorisierter ELIBA-Hän

- Interrupt-Routinen, oder zumindest deren Einsprungpunkte,
- das Betriebssystem, falls vorhanden,
- das Hardware-Software-Initialisierungsprogramm.

Diese für die Software zwingend notwendige Massnahme hat zur Folge, dass der genutzte physikalische Speicherbereich (Fig. 3) unzusammenhängend wird. Es muss deshalb noch ein gewisser Hardware-Aufwand für die Adressdecodierung auf den Speicherkarten betrieben werden, um die in Figur 3 schraffiert gezeichneten Lücken auch physikalisch wegzulassen.

Der grösste Nachteil jedoch bleibt; eine Ausnützung des maximalen physikalischen Speichers ist nicht mehr möglich. Das Verhalten eines solchen Systems ist aus dem Blickwinkel des Software-Entwurfes tückisch: Je grösser ein System, desto mehr gemeinsamer Code und gemeinsame Daten werden benötigt. Auch die Anzahl der Interrupt-Routinen nimmt zu. Je grösser aber der residente Teil des Speichers ist, um so kleiner wird der gesamthaft zur Verfügung stehende Speicher. Bei Erreichen einer bestimmten kritischen Grösse klappt das System zusammen. Will man sich vor solchen Überraschungen hüten, so muss der gemeinsame Teil des Speichers von Anfang an genügend gross dimensioniert werden. Als Vorteil des Paging verbleibt, dass mit einfachen Hardwaremitteln die Speichergrenzen eines bestehenden Mikroprozessorchips erhöht werden können.

Memory-Management-Verfahren

Memory-Management (Fig. 4) wird, wie Paging, bei Systemen eingesetzt, deren physikalischer Speicher grösser als das logische Memory ist. Dabei wird zwischen den logischen und den physikalischen Adressbus eine *Memory Management Unit* (MMU, Speicherverwaltungseinheit) geschaltet – sie kann natürlich auch bereits im CPU-Chip integriert sein – welche jede logische Adresse in eine physikalische Adresse umrechnet. Dies geschieht nach folgendem Verfahren: Die logischen Adressen werden in eine Anzahl verschiedener Bereiche unterteilt (z.B. drei wie in Fig. 4). Für jede der MMU vorgelegte logische Adresse wird als erstes festgelegt, aus welchem Bereich sie stammt. Dann wird ihr eine für den ermittelten Bereich gültige Basisadresse zuaddiert.

Diese Basisadressen werden mit Hilfe von E/A-Operationen verändert (geschrieben) oder gelesen (Es gibt auch CPUs, welche diese Adressen wie Register behandeln.)

Die Einteilung des logischen Speichers kann bei gewissen MMU ebenfalls durch die Software definiert werden [2; 3], bei anderen [4] ist sie aber fest (z.B. 8 Bereiche zu je 8 KByte Speicher). Die MMU eines Mikroprozessors muss sehr *schnell* arbeiten, da jede Adresse der CPU innerhalb der Bus-Zykluszeit in die physikalische Speicheradresse umgerechnet werden muss.

Auch beim Memory-Management gilt, dass aus Softwaresicht ein Bereich vorteilhafterweise dauernd abgebildet ist. Mit mindestens 3 Bereichen des logischen Speichers lassen sich sehr elegante Multitasking-Systeme verwirklichen (Fig. 4): Im ersten Bereich liegt ein Betriebssystem mit allen E/A-Treibern, in den zweiten Bereich kommen zu verschiedenen Zeiten verschiedene Tasks zu liegen, während im dritten Bereich Daten und Tabellen abgebildet werden, welche für verschiedene Tasks von Bedeutung sind oder sehr viel Platz beanspruchen.

Im Unterschied zum Paging gilt für das Memory-Management-Verfahren, dass auf jeden Fall der ganze physikalische Adressbereich ausgenützt werden kann. Auch ist die Hardware-Realisierung des Speichers etwas einfacher. Er kann nämlich zusammenhängend decodiert werden, da die Einteilung über die Software vorgenommen werden kann.

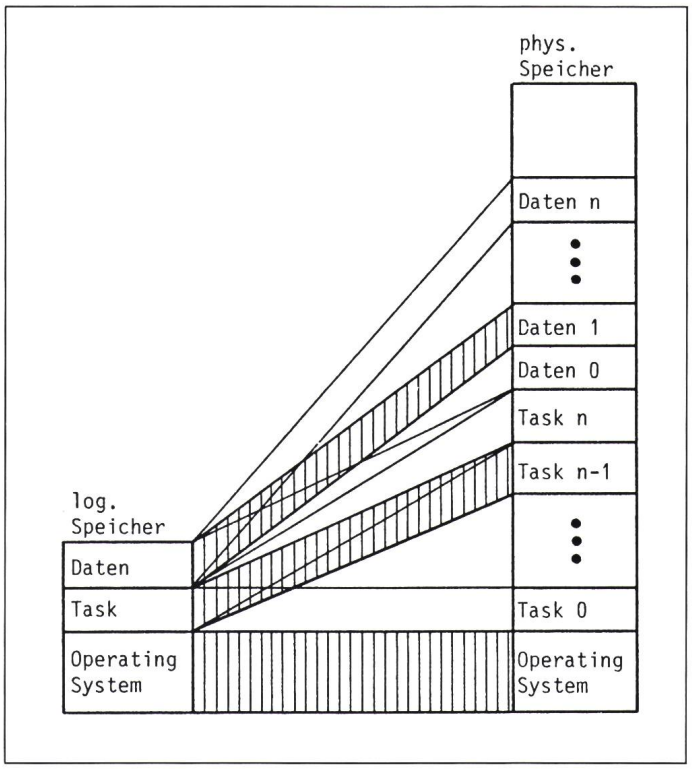
Der Entwurf von Software für ein MMU-System bringt erheblich weniger Schwierigkeiten mit sich als derjenige für ein Paging-System, auch wenn die Benützung einer MMU auf den ersten Blick eher komplizierter erscheint.

Memory-Segmentation-Verfahren

Das Memory-Segmentation-Verfahren (Speicheraufteilung) hat viele Ähnlichkeiten mit dem Memory-Management-Verfahren. Der wesentliche Unterschied besteht bei der Bereichsaufteilung der logischen Adressen. Bei der Memory Segmentation wird die Tatsache ausgenutzt, dass der Programmcode und die Daten eigentlich nicht im gleichen logischen Adressraum stehen müssen [5]. Der CPU ist ja bei jedem Speicherzugriff bekannt, ob es sich um das Laden eines Befehls oder um einen Datenzugriff handelt. Somit kann sie *selbständig* verschiedene Basisadressen für die Adressberechnung verwenden.

Neben der Unterteilung in Daten und Code bieten sich auch noch die Zugriffe auf den Stack als eigener Bereich an. Dieses Verfahren allein

Figur 4
Speicherverwaltung mittels Memory-Management Unit (MMU)



bringt allerdings erst die Möglichkeit, für jedes Segment den ganzen logischen Adressbereich ausnutzen zu können. Soll ein System aber mehr Programmcode aufweisen, als ein Segment fasst, so müssen weitere Massnahmen getroffen werden. Es sind Sprungbefehle nötig, die über die Segmentgrenzen hinausgehen können. Für die Realisierung eines Multitasking-Systems müssen auch die Interrupt-Einsprünge über die Segmentgrenzen hinweg möglich sein.

Beim Software-Entwurf erweist sich das Verfahren der Memory Segmentation als sehr heikel. Es gibt keinen Speicherbereich, der jederzeit resident abgebildet ist. Das Verfahren scheint sich als Notlösung etabliert zu haben, um den Übergang von 16- zu 32-Bit-CPUs voll kompatibel gestalten zu können und trotzdem die störende Begrenzung des Speichers auf 64 KByte umgehen zu können.

Neuerdings gibt es allerdings eine CPU (ZILOG Z-280, [6]), die dieses Verfahren zusätzlich zum Memory-Management anwendet. Hier kommt dann die an sich gute Idee doch noch voll zum Tragen.

Virtual-Memory-Verfahren

Im vorigen Kapitel ging es hauptsächlich darum, durch Speicherabbildungsverfahren den durch den Programmcode oder die Zahl der Prozessor-Adressbit beschränkten Speicher zu erweitern. Bei Mikroprozessoren mit grossen Adressbusbreiten (z.B. 24 oder 32 Bit) stellt sich diese Problematik nicht. Vielmehr ist es bis heute gar nicht möglich, z.B. einen Speicher von 4 GByte Grösse physikalisch auch zu realisieren.

Beim Entwurf eines Softwaresystems für solche Mikroprozessoren müsste somit Rücksicht genommen werden auf den physikalisch in jedem speziellen System tatsächlich vorhandenen Speicher. Diese Forderung ist natürlich völlig unhaltbar. Die Lösung dieses Problems bringt das Konzept des *virtuellen Speichers*.

Bei den früher beschriebenen Abbildungsverfahren wurde davon ausgegangen, dass jede logische Adresse in eine physikalische Adresse umgerechnet werden kann, welche auch tatsächlich einem physikalischen Speicherplatz entspricht. Diese Annahme wird beim virtuellen Abbilden fallengelassen. Das Verfahren ist im übrigen dem Memory-Management-Verfahren sehr ähnlich. Das Ergebnis der Umrechnung einer logischen in eine physikalische Adresse kann, wie beim Memory-Management, eine gültige physikalische oder aber eine virtuelle Adresse ergeben. Ist das Ergebnis eine virtuelle Adresse, so heisst dies, dass der von einem Task angesprochene Speicherbereich derzeit nicht im physikalischen Speicher enthalten ist, sondern z.B. auf einem Swap-File auf der Systemdisk gespeichert (oder überhaupt nicht vorhanden) ist. Das Betriebssystem muss nun durch die MMU aktiviert werden, um diesen Speicherblock - zuungunsten irgend eines anderen - in den physikalischen Speicher zu laden. Die dabei überschriebenen Speicherblöcke müssen natürlich vorgängig gerettet werden, falls sie gültige Daten enthalten. Zudem muss der verursachende Task solange angehalten werden, bis sein Speicherbefehl erledigt werden kann. Beim Wiederstart muss dann erst noch diese zuletzt ausgeführte Instruktion wiederholt werden. Gerade diese letzte Bedingung verlangt nach einer in der CPU integrierten Lösung [7], da sie ausserhalb keineswegs einfach gelöst werden kann.

Die grosse Gefahr des virtuellen Abbildungsverfahrens besteht darin, dass der Softwareentwurf von einem unendlich grossen Speicher ausgehen kann. Ist man sich der Tatsache des virtuellen Abbildens zu wenig bewusst, werden Tasks geschrieben, welche sich dadurch auszeichnen, dass sie einen unheimlichen Verkehr zwischen Swap-File (auf Disk) und Speicher verursachen. In einem solchen Falle beginnt der Gepard wie eine Schnecke zu kriechen. Wird z.B. ein Sortiertask als reiner Memory-Sort aufgebaut, so kann der Swap-I/O ein Vielfaches des Disk-I/O eines auf sequentiellem Mischen beruhenden Verfahrens betragen. Umgekehrt ist der Vorteil offensichtlich: Die Grösse des zur Verfügung stehenden physikalischen Speichers hat beim Softwareentwurf überhaupt keinen Einfluss mehr.

Schlussbemerkung

Während das Paged-Memory-Verfahren heute kaum mehr angewendet wird, spielen das Memory-Management- und das Memory-Segmentation-Verfahren immer noch eine grosse Rolle. Mittelfristig wird sich allerdings mit dem zunehmenden Einsatz von 32-Bit-Mikrocomputern (Motorola 68010, Intel 80386, NS 32532 usw.) das Virtual-Memory-Verfahren - eventuell kombiniert mit andern Verfahren - durchsetzen.

Literatur

[1] Eurolog - system: Grundlagen. Publ.-Nr. P 2000. Dietikon, F.J. Furrer - W.M. Gloor AG, 1985.

[2] User's manuel. Hitachi microcomputer system CMOS 8-bit microprocessor HD 64180. - Preliminary - Tokyo, Hitachi Ltd., 1986.

[3] Introduction to the NS32000 architecture. Third edition. Santa Clara/California, National Semiconductor Corporation, 1984.

[4] PdP-11 processor handbook. Maynard/Massachusetts, Digital Equipment Corporation, 1985.

[5] iAPX 86/88, 186/188 user's manuel. Programmer's reference. Santa Clara/California, Intel Semiconductor Corporation, 1985.

[6] Z 280™ MPU. Microprocessor unit. Advance information. Campbell/California, Zilog Incorporation, 1986.

[7] NS32000: The benefits of demand paged virtual memory. Santa Clara/California, National Semiconductor Corporation, 1984.