Systemprogrammierung

Betriebsarten: Stapelverarbeitung

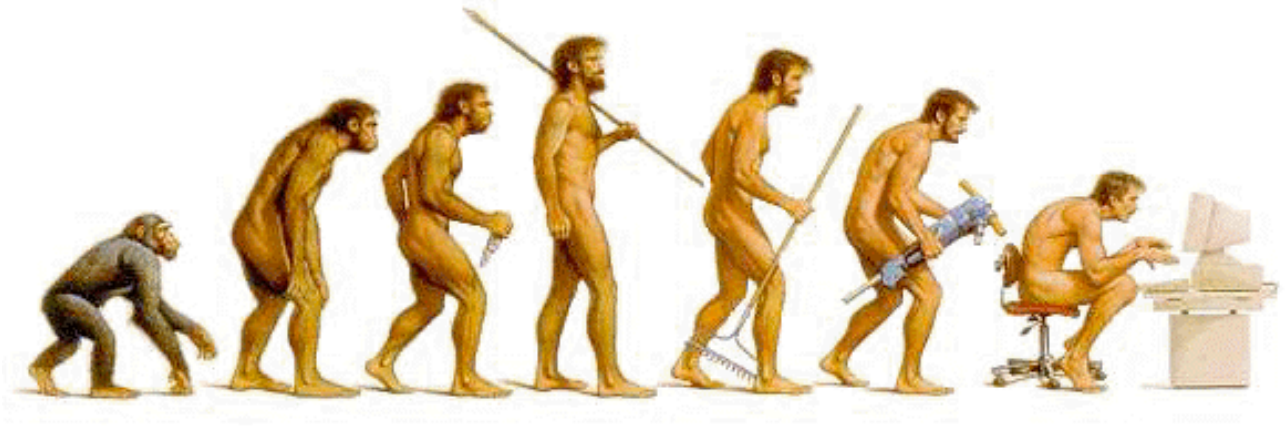
Wolfgang Schröder-Preikschat, Jürgen Kleinöder

Lehrstuhl Informatik 4

14. Januar 2012

Gliederung

Wenn wir nicht absichtlich unsere Augen verschließen, so können wir nach unseren jetzigen Kenntnissen annähernd unsere Abstammung erkennen, und dürfen uns derselben nicht schämen. (Charles Darwin)



Zusammenhänge erkennen — Gestern & Heute

Sensibilität für (die Notwendigkeit von) Betriebssystemfunktionen schärfen

Betriebssystem als Dienstleister verstehen, nicht als Selbstzweck

- eine **Maschine**, mit dessen Hilfe Rechnerhardware für (Anwendungs- und/oder System-) Programme nutzbar gemacht werden soll
- die Fähigkeiten dieser Maschine bestimmen sich in erster Linie durch den Anwendungszweck und die Hardware des Rechners
- in zweiter Linie stehen Fertigkeiten des Entwicklungspersonals, obwohl diese wesentlich zur Qualität des Endprodukts beitragen

Evolutionsgeschichte von Betriebssystemen im Zeitraffer

- mehr als nur ein „Abfallprodukt“ der nachfolgenden Betrachtungen
- zeigt das „Aufwachsen“ von Systemsoftware in funktionaler Hinsicht

Generationen von Hardware und Betriebssoftware

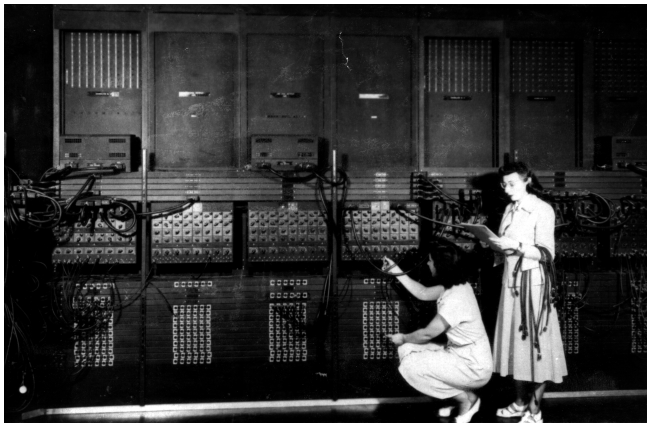
Zeitliche Einordnung von Technologiemerkmalen: „unscharf“, Übergänge fließend

Generation	Epoche	Hardware	Rechnerbetriebsart
1	1945	Röhre	Stapelbetrieb
2	1955	Halbleiter, Gatter	Echtzeitbetrieb
3	1965	MSI/LSI, WAN	Mehrprogrammbetrieb
4	1975	VLSI, LAN	Mehrzugangsbetrieb
5	1985	ULSI, RISC	Netzbetrieb
	1995	WLAN, RFID, SoC	Integrationsbetrieb

Legende: MSI, LSI, VLSI, ULSI — *medium, large, very large, ultra large scale integration*
LAN — *local area network* (Ethernet)
WAN — *wide area network* (Arpanet)
WLAN — *wireless* LAN
RISC — *reduced instruction set computer*
RFID — *radio frequency identification* (Funkerkennung)
SoC — *system on chip*
VM — *virtual memory*

Am Anfang war das Feuer...

ENIAC (*electronic numerical integrator and computer*), 1945



A small amount of time is required in preparing the ENIAC for a problem by such steps as setting program switches, putting numbers into the function table memory by setting its switches, and establishing connections between units of the ENIAC for the communication of programming and numerical information.
(Pressemitteilung, DoD, 1946)

ENIAC [20]

- elektronischer Allzweckrechner von 30 Tonnen Gewicht
- $15\,m \times 3\,m \times 5\,m$ große Zentraleinheit, $30\,m$ lange Frontplatte
- für seinen Betrieb waren 18 000 Röhren zuständig
- die elektrische Anschlussleistung belief sich auf etwa 174 000 Watt

Gliederung

Lochkartenverarbeitung

Hermann Holerith (1860–1929), Begründer der maschinellen Datenverarbeitung. IBM ließ sich 1928 das Format patentieren: 80 Spalten, 12 Zeilen und rechteckige Löcher an den Schnittpunkten.

Ziffernlochkarte (engl. *numeric punch card*)
- Datenaufzeichnung durch Lochung (Loch $\mapsto$ 1, kein Loch $\mapsto$ 0)
  - Dezimalzahlen werden mit einer Lochung dargestellt
  - Buchstaben und Sonderzeichen mit zwei oder drei
  - negative Vorzeichen ggf. durch eine Lochung in Zeile 11
- manuell ca. 15 000 Zeichen/h, maschinell 4 000–10 000 Karten/h

Manuelle Bestückung des Rechners

Vollständige Kontrolle beim Programmierer/Operateur

1. Programme mit **Lochkartenstanzer** (engl. *card puncher*) ablochen
2. Übersetzerkarten in den **Lochkartenleser** (engl. *card reader*) geben
3. Lochkartenleser durch Knopfdruck starten
4. Programmkarten (zur Übersetzung) in den Kartenleser einlegen
5. Eingabekarten in den Kartenleser einlegen
6. Leere Lochkarten für die Ausgabe in den Kartenstanzer einlegen
7. Ausgabekarten in den Kartenleser des Druckers einlegen
8. Ergebnisse der Programmausführung vom Drucker abholen

Problem

- Urlader (z.B. Lochkartenleseprogramm) in den Rechner einspeisen

Lochkartenleseprogramm

Urlader (engl. *bootstrap loader*)

Rechnersystem durch Ureingabe (engl. *bootstrap*) laden:

1. Bitmuster einer Speicherwortadresse über Schalter einstellen
   - die Adresse der nächsten zu beschreibenden Stelle im Arbeitsspeicher
2. eingestellten Adresswert in den PC der CPU laden
3. Bitmuster des Speicherwortinhalts über Schalter einstellen
   - ein Befehl, ein Direktwert oder eine Operandenadresse des Programms
4. eingestellten Datenwert in den Arbeitsspeicher laden

- heute sind Urlader nicht-flüchtig im ROM/EPROM gespeichert!!!

Problem

- Bedienung, Mensch, Permanenz

Automatisierte Bestückung des Rechners

Verzicht auf Maßnahmen zur manuellen Einspeisung von Dienstprogrammen

Dienstprogramme sind zum Abruf bereit im Rechnersystem gespeichert:

- Systembibliothek, „Datenbank“ (Dateiverwaltung)

Anwendungsprogramme fordern Dienstprogramme explizit an:

- spezielle **Steuerkarten** sind dem Lochkartenstapel hinzugefügt
  - Systemkommandos, die auf eigenen Lochkarten kodiert sind
  - Anforderungen betreffen auch Betriebsmittel (z.B. Speicher, Drucker)
- der erweiterte Lochkartenstapel bildet einen **Auftrag** (engl. *job*)
  - an einen **Kommandointerpretierer** (engl. *command interpreter*)
  - formuliert als Auftragssteuersprache (engl. *job control language*, JCL)
- die Programmausführung erfolgt ohne weitere Interaktion

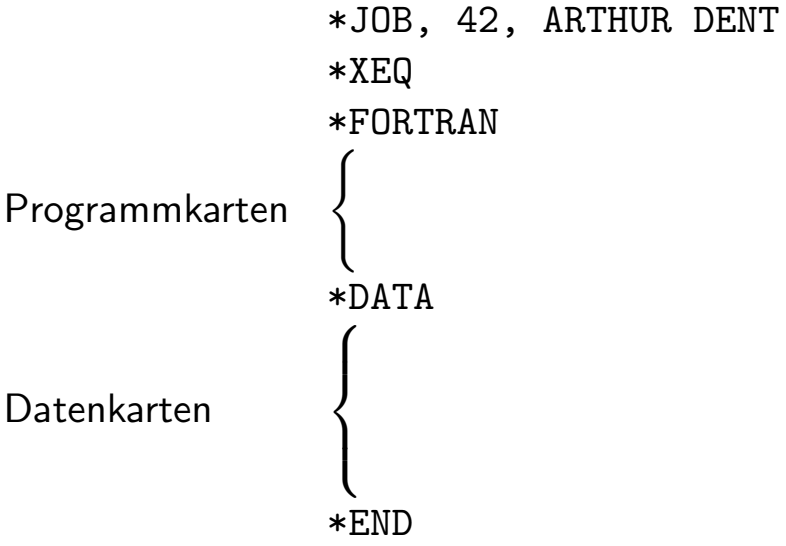
- eignet sich (nach wie vor) zur Bewältigung von „Routineaufgaben“

Problem

- vollständige Auftragsbeschreibung (inkl. Betriebsmittelbedarf)

Auftragssteuersprache

FORTRAN Monitoring System, FMS [11], um 1957



Residentes Steuerprogramm (engl. *resident monitor*)

Auftragssteuerung durch (Arbeitsspeicher-) residente Systemsoftware:

- Kommandointerpretierer, Lochkartenleseprogramm, E/A-Prozeduren
- Systemprogramme, die ein „embryonales Betriebssystem“ bilden

Solitär: einzelstehende und getrennt übersetzte Einheit, verschiedenartig vom Anwendungsprogramm entkoppelt

- Einsprungtabelle (engl. *jump table*)
- partielle Interpretation von Monitoraufrufen
- getrennte physikalische Adressräume (Schutzgatter)
- Arbeitsmodi (Benutzer-/Systemmodus, privilegiert/nicht-privilegiert)

Problem

- Arbeitsgeschwindigkeit der Peripherie, sequentielle Ein-/Ausgabe

Adressraumschutz durch Abteilung

Partitionierung des physikalischen Adressraums

Schutzgatter trennt den vom residenten Steuerprogramm und vom transienten Programm belegten Arbeitsspeicherbereich

- Anwendungsprogramme werden komplett, d.h. statisch gebunden
  - Schutzgatter ≡ unveränderliche, physikalische Speicheradresse
  - Relokationskonstante beim Binden, Ladeadresse für die Programme
- ggf. ist gewisse Flexibilität durch ein Schutzgatterregister gegeben
  - veränderliche, physikalische Speicheradresse; programmierbarer Wert
  - Relokationsvariable beim Binden, relative Programmadressen

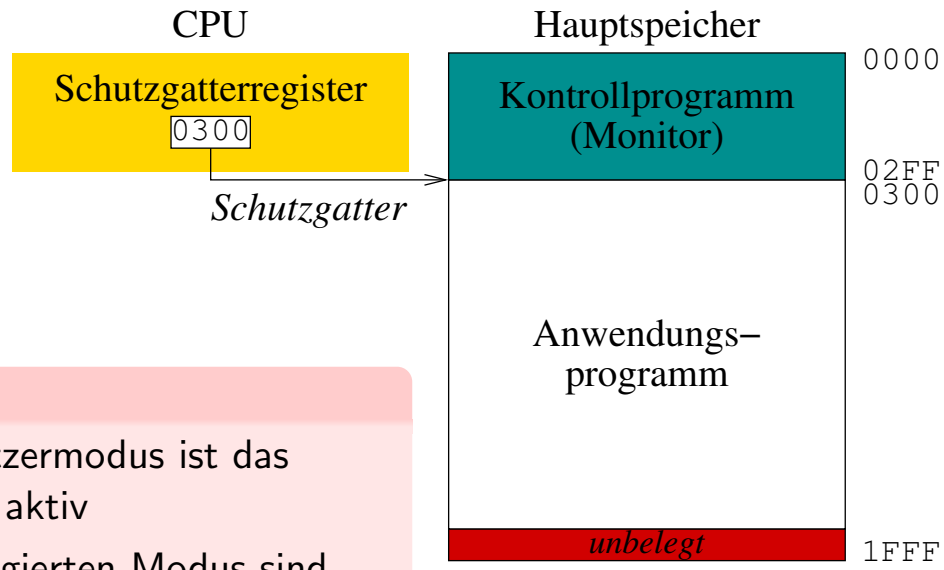
Speicherverwaltung ist statisch, geschieht vor Programmausführung

- ein verschiebender Lader legt die Lage im Arbeitsspeicher fest
  - obligatorisch bei Verwendung des Schutzgatterregisters, optional sonst
- zur Programmlaufzeit ist zusätzlicher Speicher nur bedingt zuteilbar
  - spätestens zum Ladezeitpunkt muss weiterer Bedarf bekannt sein

Problem

- Überbelegung des Arbeitsspeichers (zu großes Programm)

Schutzgatterregister (engl. *fence register*)

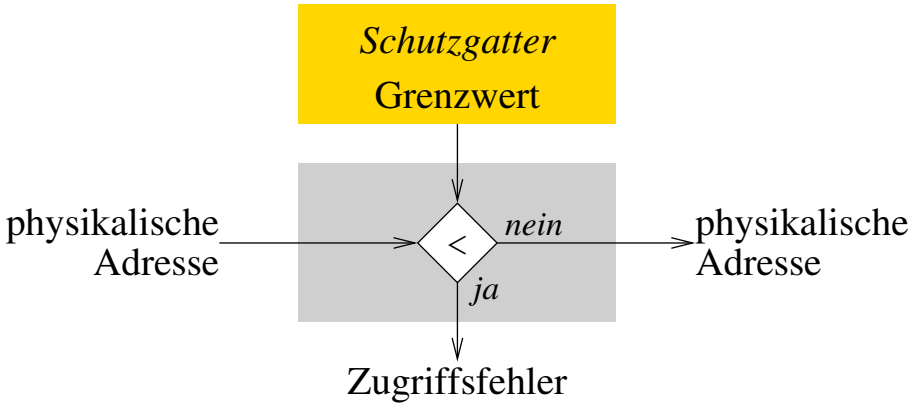


Arbeitsmodi

- nur im Benutzermodus ist das Schutzgatter aktiv
- nur im privilegierten Modus sind Änderungen am Inhalt des Schutzgatterregisters erlaubt

Schutzgatterregister (Forts.)

Zugriffsfehler führt zum Abbruch (synchrone Programmunterbrechung, Trap)



Problem

- „interne Fragmentierung“ (Zugriff auf unbelegten Bereich)

Abgesetzter Betrieb

Berechnung erfolgt getrennt von Ein-/Ausgabe (engl. *off-line*)

Satellitenrechner zur Ein-/Ausgabe mit „langsamer Peripherie“

- Kartenleser, Kartenstanzer, Drucker
- Ein-/Ausgabedaten werden über **Magnetbänder** transferiert

Hauptrechner zur Berechnung mit „schneller Peripherie“

- Be-/Entsorgung des Hauptspeichers auf Basis von **Bandmaschinen**
- dadurch erheblich verkürzte Wartezeiten bei der Ein-/Ausgabe

- heute finden zusätzlich **Wechselplatten** Verwendung

Problem

- sequentieller Bandzugriff, feste Auftragsreihenfolge

Dedizierte Rechner/Geräte für verschiedene Arbeitsphasen

Phase	Medium			Rechner
Eingabe	Text/Daten	⇒	Lochkarten	Satellitenrechner
	Lochkarten	⇒	Magnetband	
↓				↓
Verarbeitung	Magnetband	⇒	Arbeitsspeicher	Hauptrechner
	Arbeitsspeicher	⇒	Magnetband	
↓				↓
Ausgabe	Magnetband	⇒	Lochkarten	Satellitenrechner
	Daten	⇒	Drucker	

Überlappte Ein-/Ausgabe

Nebenläufige Ausführung der Gerätetreiber

Speicherdirektzugriff (engl. *direct memory access*, DMA, [12]):

- unabhängig von der CPU arbeitende E/A-Kanäle
  - zwischen einem E/A-Gerät und dem Arbeitsspeicher (RAM)
- Datentransfer durch ein **Kanalprogramm** (*cycle stealing*, [8, 6])

Fall für asynchrone Programmunterbrechungen [10, S. 225–227]:

- die E/A-Geräte fordern der CPU weitere E/A-Aufträge ab
- E/A und Berechnung desselben Programms überlappen sich
- gleichzeitige Aktivitäten sind ggf. explizit zu koordinieren
  - **Synchronisation** [5, S. 28–33]

Problem

- Leerlauf beim Auftragswechsel

Überlappte Auftragsverarbeitung

(engl. *single-stream batch monitor*)

Verarbeitungsstrom sequentiell auszuführender Programme/Aufträge

- bei Ausführung eines laufenden Auftrags den nachfolgenden Auftrag bereits in den Arbeitsspeicher einlesen (d.h. zwischenpuffern)
- Programme im **Vorgriffsverfahren** (engl. *prefetching*) abarbeiten

Auftragseinplanung (engl. *job scheduling*) liefert Abarbeitungsreihenfolge

- statische/dynamische Einplanung nach unterschiedlichsten Kriterien
  - Ankunftszeit, erwartete Laufzeit, erwarteter Betriebsmittelbedarf
- die Aufträge werden dazu im Hintergrundspeicher zusammengestellt
  - *wahlfreier Zugriff* (z.B. Plattenspeicher [7]) vermeidet Engpässe

Problem

- Arbeitsspeicher, Monopolisierung der CPU, Leerlauf bei Ein-/Ausgabe

Abgesetzte Ein-/Ausgabe

Spooling (engl. *simultaneous peripheral operations online*)

Pufferbereiche zur Entkopplung von Berechnung und Ein-/Ausgabe [16]

- Programmausführung ist eine periodische Abfolge von zwei Phasen:
  CPU-Stoß (engl. *CPU burst*) vergleichsweise schnell
  - Aktivitäten eines Programms, die Berechnungen betreffen

  E/A-Stoß (engl. *I/O burst*) vergleichsweise langsam
  - Aktivitäten eines Programms, die Ein-/Ausgabe betreffen
- Pufferung im Hintergrundspeicher (engl. *secondary/backing store*)
  - Berechnungen im Vordergrundspeicher (engl. *primary/main store*)
- per DMA werden die Daten („nebenbei“) hin- und hertransferiert

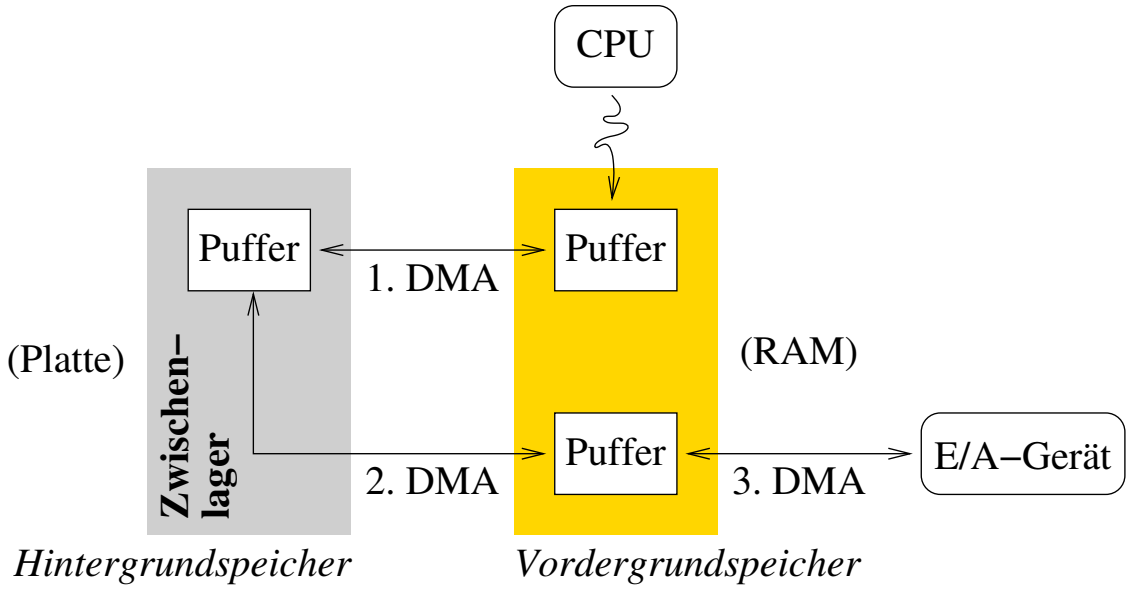
Systemprogramme starten/überwachen die E/A-Vorgänge

- heute z.B. `lpr(1)` und `lpd(8)` unter Unix

Problem

- Leerlauf im Wartezustand (Einprogramm(-/-prozess)betrieb)

Abgesetzte Ein-/Ausgabe (Forts.)



Gliederung

Maßnahmen zur Leistungssteigerung

Multiprogrammbetrieb (engl. *multiprogramming*) und Simultanbetrieb (engl. *multiprocessing*), auch Maßnahmen zur Strukturierung:

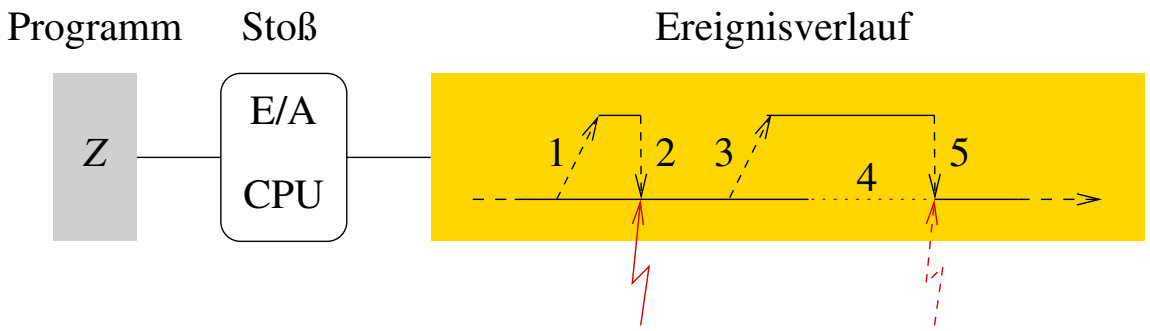
- Multiplexen der CPU zwischen mehreren Programmen[1]
  - **Nebenläufigkeit** (engl. *concurrency*)
- Abschottung der sich in Ausführung befindlichen Programme
  - **Adressraumschutz** (engl. *address space protection*)
- Überlagerung unabhängiger Programmteile

> *If we believe in data structures, we must believe in independent (hence simultaneous) processing. For why else would we collect items within a structure? Why do we tolerate languages that give us the one without the other? (Alan Perlis [18])*

[1] Zunächst als ein eher „befremdliches Ergebnis“ von *Interrupts* angesehen [14, S. 43].

Überlappte Ein-/Ausgabe im Einpro{gramm,zess}betrieb

Leerlauf der CPU im Wartezustand



1. Programm$_z$ löst einen E/A-Stoß nebenläufig zum CPU-Stoß aus
2. ein Interrupt zeigt die Beendigung des E/A-Stoßes an
3. die Unterbrechungsbehandlung startet einen weiteren E/A-Stoß
4. Programm$_z$ muss auf E/A warten, sein CPU-Stoß endet
5. die CPU läuft leer, bis der E/A-Stoß zum Abschluss kommt

Problem

- Durchsatz, Auslastung

Multiplexen des Prozessors

Die CPU „gleichzeitig“ von mehreren Programmen benutzen

Wartezeit von Programm$_x$ wird als Laufzeit von Programm$_y$ genutzt

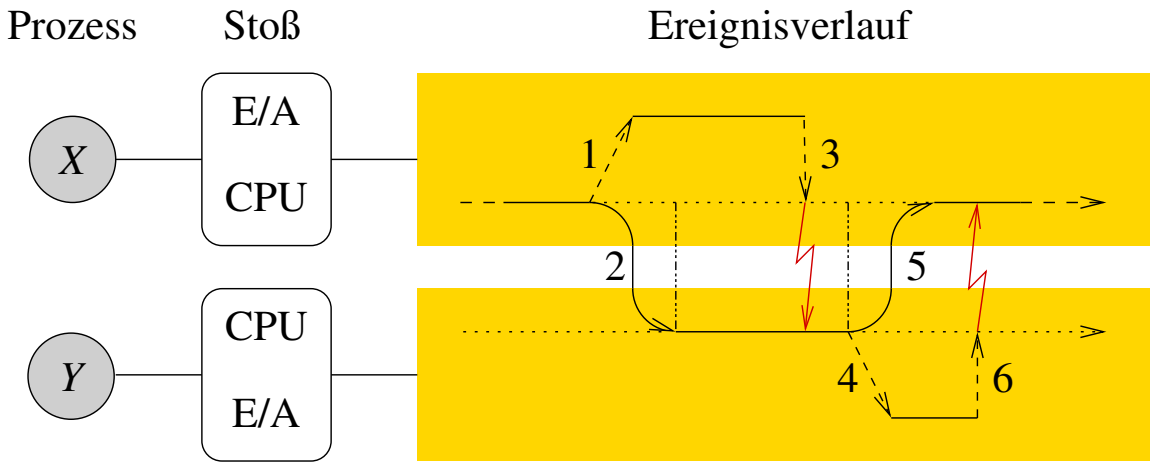
- aktives Warten (engl. *busy waiting*) auf Ereigniseintritte vermeiden
- stattdessen passives Warten betreiben und die CPU einem anderen lauffähigen Programm zuteilen

Abstraktion von wartenden/laufenden Programmen, liefert ein generisches Konzept zur Programmverarbeitung -und verwaltung

- der Begriff „**Prozess**“: ein beliebiges „Programm in Abarbeitung“
- für jedes zu ladende Programm wird ein Prozess erzeugt

Einplanung (engl. *scheduling*) und Einlastung (engl. *dispatching*) von Prozessen regeln die Verarbeitung lauffähiger Programme [3, 2, 9]

Programmverarbeitung im Mehrprozessbetrieb



BS ≡ mehrfädiges Programm (engl. *multi-threaded program*)

- ein Programm, in dem mehrere Prozesse „gleichzeitig“ aktiv sind

Programmverarbeitung im Mehrprozessbetrieb (Forts.)

Kooperation von Prozess$_x$ und Prozess$_y$, nach erfolgter Einplanung:

1. Prozess$_x$ löst einen E/A-Stoß und beendet seinen CPU-Stoß
2. Prozess$_y$ wird eingelastet und beginnt seinen CPU-Stoß
3. Beendigung des E/A-Stoßes von Prozess$_x$ unterbricht Prozess$_y$
   - die Unterbrechungsbehandlung überlappt sich mit Prozess$_y$
4. Prozess$_y$ löst einen E/A-Stoß und beendet seinen CPU-Stoß
5. Prozess$_x$ wird eingelastet und beginnt seinen CPU-Stoß
6. Beendigung des E/A-Stoßes von Prozess$_y$ unterbricht Prozess$_x$
   - die Unterbrechungsbehandlung überlappt sich mit Prozess$_x$

Problem

- Adressraumschutz, Koordination

Adressraumschutz

Abschottung von Programmen durch Eingrenzung/Segmentierung

Bindungszeitpunkt von Programm- an Arbeitsspeicheradressen beeinflusst das Modell zur Abschottung der Programme — und umgekehrt:

vor Laufzeit Schutz durch **Eingrenzung**

- Programme laufen (weiterhin) im physikalischen Adressraum
- ein **verschiebender Lader** besorgt die Bindung zum *Ladezeitpunkt*
- die CPU überprüft die generierten Adressen zur Ausführungszeit

zur Laufzeit Schutz durch **Segmentierung** [4]

- jedes Programm läuft in seinem eigenen **logischen Adressraum**
- der Lader bestimmt die Bindungsparameter (phys. Basisadresse)
- die CPU (bzw. MMU) besorgt die Bindung zur Ausführungszeit
  - dynamische Programmumlagerung (engl. *program relocation*, [15])

In beiden Fällen richtet der Binder Programme relativ zu einer logischen Anfangsadresse aus, meistens zur Basis 0.

Adressraumschutz durch Eingrenzung

Begrenzungsregister (engl. *bounds register*) legen die Unter-/Obergrenze eines Programms im physikalischen Adressraum fest

- für jedes Programm wird ein solches Registerpaar verwaltet
  - die sog. *Softwareprototpypen* der Adressüberprüfungshardware
- bei Prozesseinlastung erfolgt die Abbildung auf die Hardwareregister

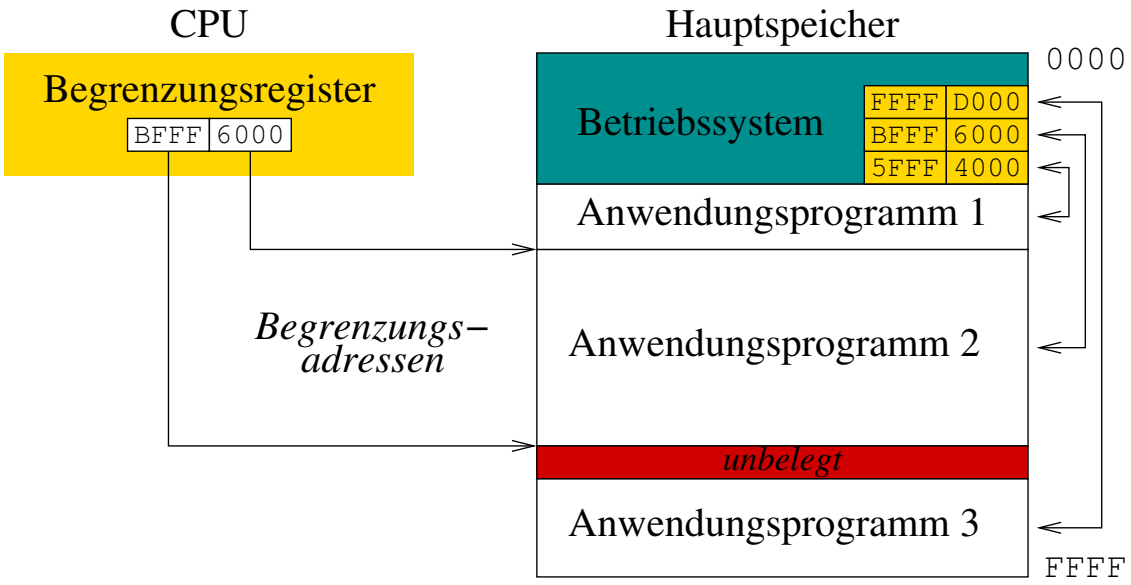
Speicherverwaltung ist statisch, geschieht vor Programmausführung

- der verschiebende Lader legt die Lage im Arbeitsspeicher fest
- zur Programmlaufzeit ist zusätzlicher Speicher nur bedingt zuteilbar
  - spätestens zum Ladezeitpunkt muss weiterer Bedarf bekannt sein

Problem

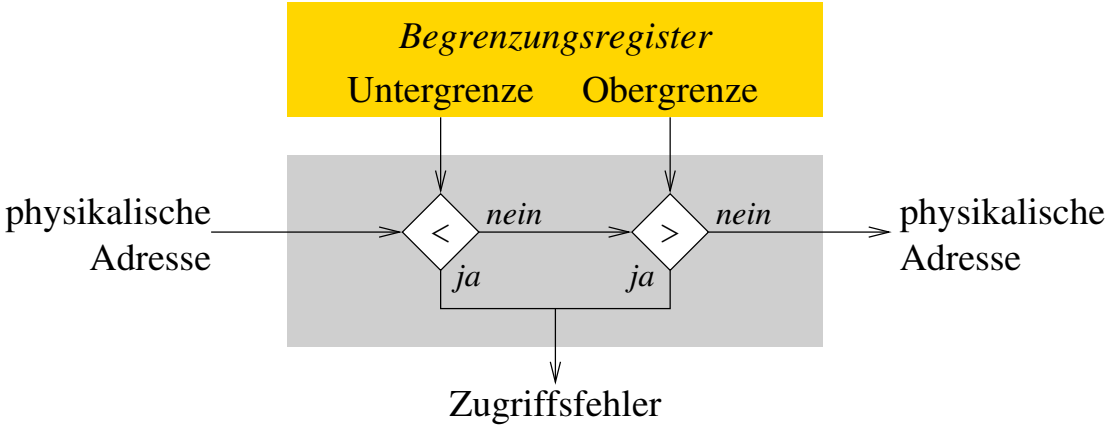
- Fragmentierung, Verdichtung

Begrenzungsregister (engl. *bounds register*)



Begrenzungsregister (Forts.)

Zugriffsfehler führt zum Abbruch (synchrone Programmunterbrechung, Trap)



Adressraumschutz durch Segmentierung

Basis-/Längenregister (engl. *base/limit register*) geben Programme vom physikalischen Adressraum abstrahierende logische Adressräume

- für jedes Programm wird ein solches Registerpaar verwaltet
  - die sog. *Softwareprototpypen* der Adressumsetzungshardware
- bei Prozesseinlastung erfolgt die Abbildung auf die Hardwareregister

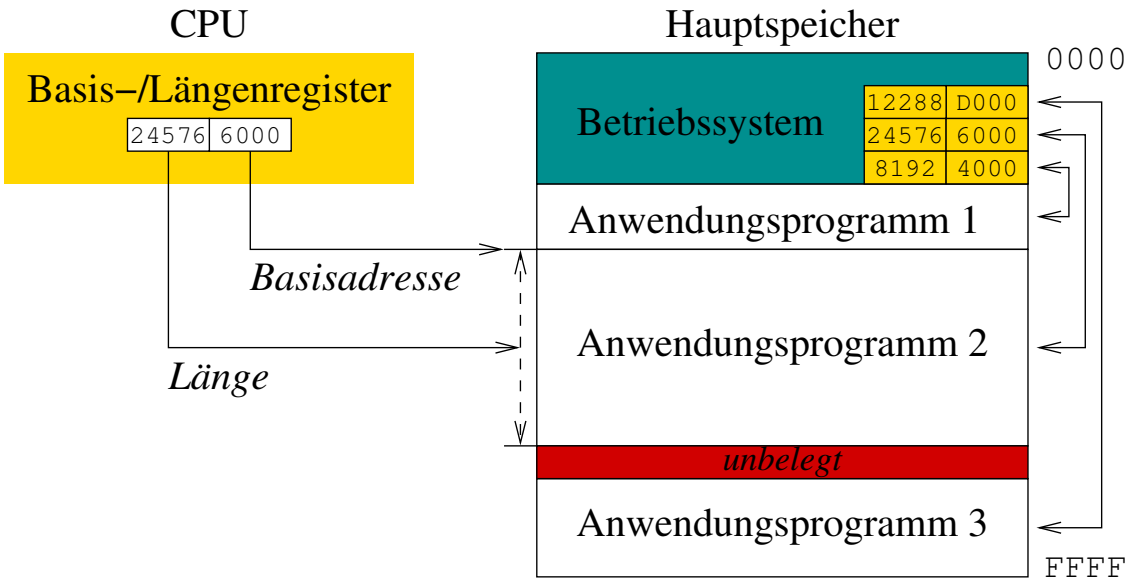
Speicherverwaltung ist dynamisch, geschieht zur Programmausführung

- der Lader legt die initiale Lage im Arbeitsspeicher fest
- zur Programmlaufzeit ist zusätzlicher Speicher „beliebig“ zuteilbar
  - limitierend ist der jeweils physikalisch noch freie Arbeitsspeicher

Problem

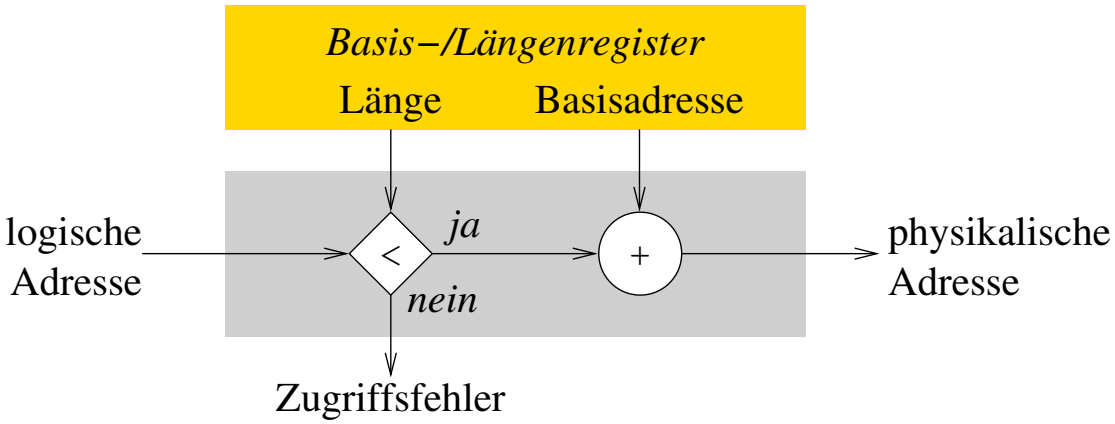
- Überbelegung des Arbeitsspeichers (zu große/viele Programme)

Basis-/Längenregister (engl. *base/limit register*)



Basis-/Längenregister (Forts.)

Zugriffsfehler führt zum Abbruch (synchrone Programmunterbrechung, Trap)



- Grundlage für eine MMU des heutigen Stands der Technik

Überlagerung von Programmteilen im Arbeitsspeicher

Technik der Überlagerung (engl. *overlay*, [17])

Ein Programm, das einschließlich Daten die Kapazität des Hauptspeichers übersteigt, wird in hinreichend kleine Teile zergliedert, die nicht ständig im Hauptspeicher vorhanden sein müssen sondern stattdessen im Hintergrundspeicher (Trommel, Platte) vorgehalten werden.

Überlagerungen werden nur bei Bedarf (engl. *on demand*) nachgeladen

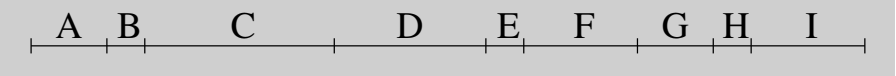
- das Nachladen ist programmiert, d.h. in den Programmen festgelegt
- die Entscheidung zum Nachladen fällt zur Programmlaufzeit

Problem

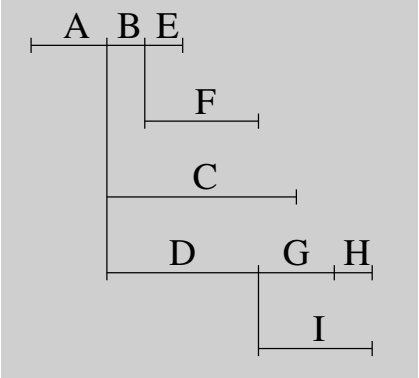
- finden der (zur Laufzeit) „optimalen“ Überlagerungsstruktur

Einsparung von Arbeitsspeicher zur Programmlaufzeit

statisch



dynamisch



eingelagert

ABE

ABF

AC

ADGH

ADI

- nicht alle Bestandteile (A – I) eines Programms müssen gleichzeitig im Arbeitsspeicher vorliegen, damit das Programm auch ausführbar ist
- wann welches Programmteil zur Ausführung gelangt, legt der dynamische Ablauf des Programms selbst fest

Überlagerungsstruktur eines Programms

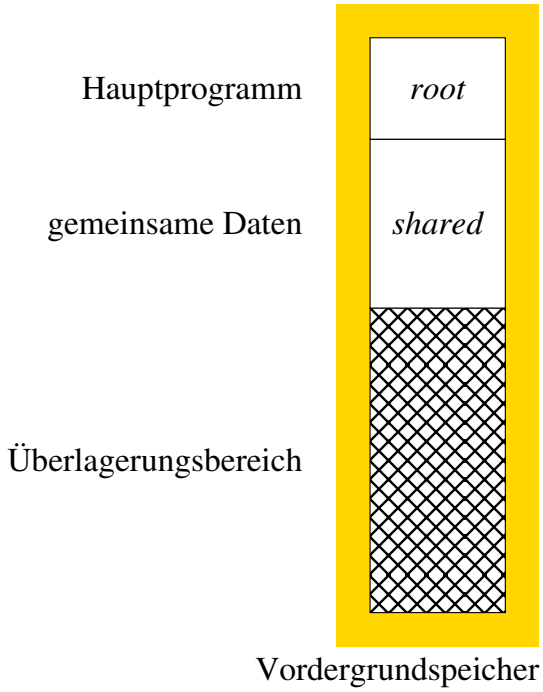
Programme, die Überlagerungen enthalten, sind (grob) dreigeteilt:

1. ein arbeitsspeicherresidenter Programmteil (engl. *root program*)
2. mehrere in Überlagerungen zusammengefasste Unterprogramme
3. ein gemeinsamer Datenbereich (engl. *common section*)

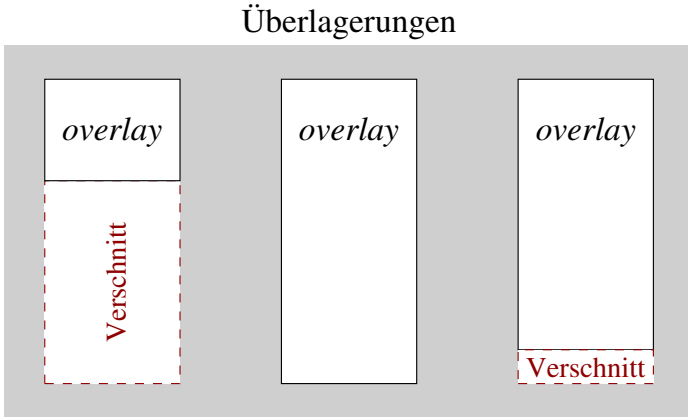
Überlagerungen sind das Ergebnis einer Programmzerlegung zur Programmier-, Übersetzungs- und/oder Bindezeit

- manuelle bzw. automatisierte Abhängigkeitsanalyse des Programms
- Anzahl und Größe von Überlagerungen werden statisch festgelegt
- lediglich aktivieren (d.h. laden) von Überlagerungen ist dynamisch

Organisation des Hauptspeichers



Überlagerungen sind Teile eines Programms und keine Elemente einer mehreren Programmen gemeinsamen Bibliothek (engl. *shared library*) oder DLL (engl. *dynamic link library*)!



Verarbeitung mehrerer Auftragsströme

(engl. *multiprocessing*, *multi-stream batch monitor*)

Verarbeitungsströme sequentiell auszuführender Programme/Aufträge

- pseudo-parallele Abarbeitung mehrerer Stapel im Multiplexverfahren
  - sequentielle Ausführung der Programme desselben Auftragstapels
  - überlappende Ausführung der Programme verschiedener Auftragstapel
- „wer [innerhalb des Stapels] zuerst kommt, mahlt zuerst“...

Stapelwechsel verlaufen **kooperativ** (engl. *cooperative*) oder kommen **verdrängend** (engl. *preemptive*) zustande

- kooperativ bei (programmierter) Ein-/Ausgabe und Programmende
  - d.h., bei Beendigung des CPU-Stoßes eines Prozesses
- verdrängend bei Ablauf einer Frist (engl. *time limit*)

Problem

- interaktionsloser Betrieb, Mensch/Maschine-Schnittstelle

Gliederung

Resümee

- **Stapelbetrieb** meint die sequentielle Abwicklung von Aufgaben
  - anfangs manuelle Bestückung des Rechners durch die Programmierer
  - später automatisierte Bestückung mittels **Kommandointerpretierer**
    - Programmierer organisierten ihren Auftragsstapel mittels Steuerkarten
    - Operateure übern{a,e}hmen die manuelle Bestückung des Rechners
  - **Aufgabenverteilung** dehnt(e) sich auch systeminterne Abläufe aus
    - abgesetzter Betrieb, überlappte Ein-/Ausgabe
    - überlappte Auftragsverarbeitung, abgesetzte Ein-/Ausgabe
- **Mehrprogrammbetrieb** hält mehrere Programme im Hauptspeicher
  - der Prozessor wird im Multiplexverfahren betrieben: **Prozesse**
  - Schutzvorkehrungen schotten Programmadressräume voneinander ab
  - Hauptspeicherknappheit wird durch dynamisches Laden begegnet
  - das Rechensystem damit sicher in **Simultanbetrieb** fahren können

Beachte

- Operateure bek{a,om}men interaktiven Zugang zum Rechensystem
- Anwender arbeit(et)en mit dem Rechensystem interaktionslos

Literaturverzeichnis

[1] Bell, C. G. ; Newell, A. :
Computer Structures: Readings and Examples.
New York, NY, USA : McGraw-Hill Inc., 1971. –
668 S.

[2] Coffman, E. G. ; Denning, P. J.:
Operating System Theory.
Prentice Hall, Inc., 1973

[3] Conway, R. W. ; Maxwell, L. W. ; Millner, L. W.:
Theory of Scheduling.
Addison-Wesley, 1967

[4] Dennis, J. B.:
Segmentation and the Design of Multiprogrammed Computer Systems.
In: *Journal of the ACM* 12 (1965), Okt., Nr. 4, S. 589–602

[5] Dijkstra, E. W.:
Communication with an Automatic Computer, Universiteit van Amsterdam, Diss., Okt. 1959

Literaturverzeichnis (Forts.)

[6] IBM:
Data Processing Machine Including Program Interrupt Feature.
U.S. Pat. No. 3,319,230 (1967), 1956

[7] IBM:
IBM 503 RAMAC.
http://www-03.ibm.com/ibm/history/exhibits/storage/storage_PH0305.html, 1956

[8] IBM:
709 Data Processing System.
http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_PP709.html, Jan. 1957

[9] Kleinrock, L. :
Queuing Systems. Bd. I: Theory.
John Wiley & Sons, 1975

[10] Knuth, D. E.:
The Art of Computer Programming. Bd. 1: *Fundamental Algorithms*.
Reading, MA, USA : Addison-Wesley, 1973

Literaturverzeichnis (Forts.)

[11] Larner, R. A.:
FMS: The FORTRAN Monitor System.
In: Butler, M. K. (Hrsg.) ; Brown, J. M. (Hrsg.) ; American Federation of Information Processing Societies, Inc. (Veranst.): *1987 Proceedings of the National Computer Conference, June 15–18, 1987, Chicago, Illinois, USA* Bd. 56 American Federation of Information Processing Societies, Inc., AFIPS Press, 1987, S. 815–820

[12] Leiner, A. L.:
System Specifications for the DYSEAC.
In: *Journal of the ACM* 1 (1954), Apr., Nr. 2, S. 57–81

[13] Leiner, A. L. ; Alexander, S. N.:
System Organization of DYSEAC.
In: *IRE Transactions on Electronic Computers* EC-3 (1954), März, Nr. 1, S. 1–10

[14] Loopstra, B. J.:
The X-1 Computer.
In: *The Computer Journal* 2 (1959), Nr. 1, S. 39–43

[15] McGee, W. C.:
On Dynamic Program Relocation.
In: *IBM Systems Journal* 4 (1965), Sept., Nr. 3, S. 184–199

Literaturverzeichnis (Forts.)

[16] Mock, O. ; Swift, C. J.:
The Share 709 System: Programmed Input-Output Buffering.
In: *Journal of the ACM* 6 (1959), Apr., Nr. 2, S. 145–151

[17] Pankhurst, R. J.:
Operating Systems: Program Overlay Techniques.
In: *Communications of the ACM* 11 (1968), Febr., Nr. 2, S. 119–125

[18] Perlis, A. J.:
Epigrams on Programming.
In: *SIGPLAN Notices* 17 (1982), Nr. 9, S. 7–13

[19] Smotherman, M. :
Interrupts.
http://www.cs.clemson.edu/~mark/interrupts.html, Jul. 2008

[20] US War Department, Bureau of Public Relations:
For Radio Broadcast: Physical Aspect, Operation of ENIAC are Described.
http://americanhistory.si.edu/collections/comphist/pr4.pdf, Febr. 1946

Rechnerbetrieb am RRZE um 1973

http://www.rrze.uni-erlangen.de/wir-ueber-uns/publikationen/das-rrze-der-film.shtml