TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik | Institut Systemarchitektur | Professur Rechnernetze

WS 2012
LV Informatik-I für Verkehrsingenieure

2. Rechnerarchitektur
2.1 einfache Computer

Dr. rer.nat. D. Gütter

Mail: Dietbert.Guetter@tu-dresden.de
WWW: wwwpub.zih.tu-dresden.de/~guetter/

Analogrechner

physikalische Abbildung eines mathematischen Modells

1825 Gonella - mechanisches Integriergerät
Weiterentwicklungen bis ca. 1960

1965

endim 2000
Rechenelektronik Glashütte

Analogrechner eignen sich für
schnelle Berechnungen,
sind aber ungenau

Einsatz z.B. bei Artillerie

Mechanische Rechenmaschinen/-automaten

Zifferndarstellung durch Staffelwalzen, Zahnräder, …

1623 Schickard mechanische Rechenmaschine
Weiterentwicklungen bis ca. 1960

1962

Supermetall 215 (SAR IIc K)
Rheinmetall Sömmerda

Hochentwickelte Maschinen mit
- Steuerautomatik,
- allen Grundrechenarten,
- Zwischenspeicher,
- Druckwerk,
- …

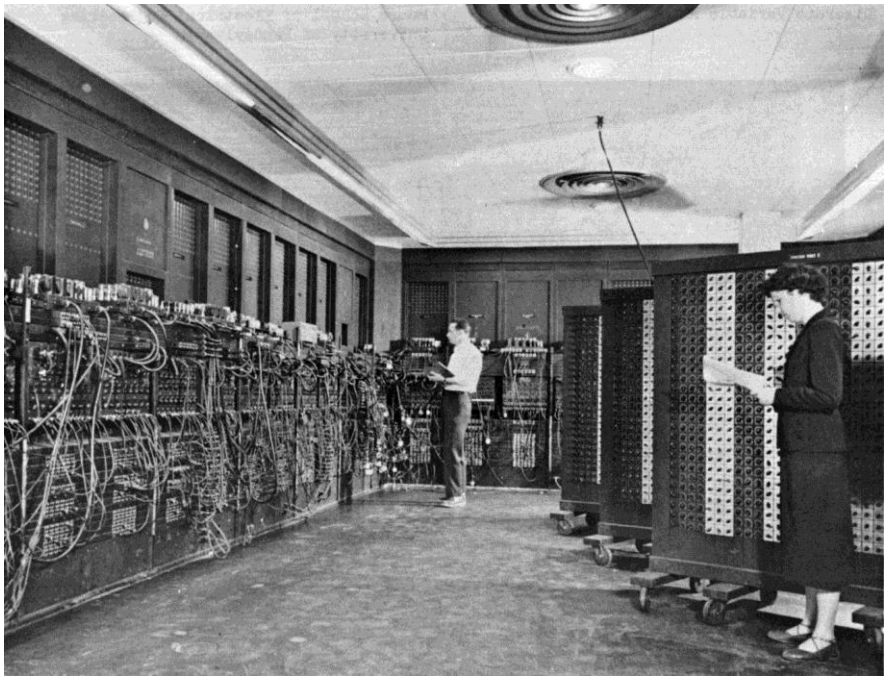
Programmierbare Rechenautomaten

1807 Jaquard - Webstuhlsteuerung
1832 Babbage – erster Computer (funktionsunfähig)
1996 Hollerith – Lochkartentechnik
1941 Zuse – elektromechanischer Computer Z1

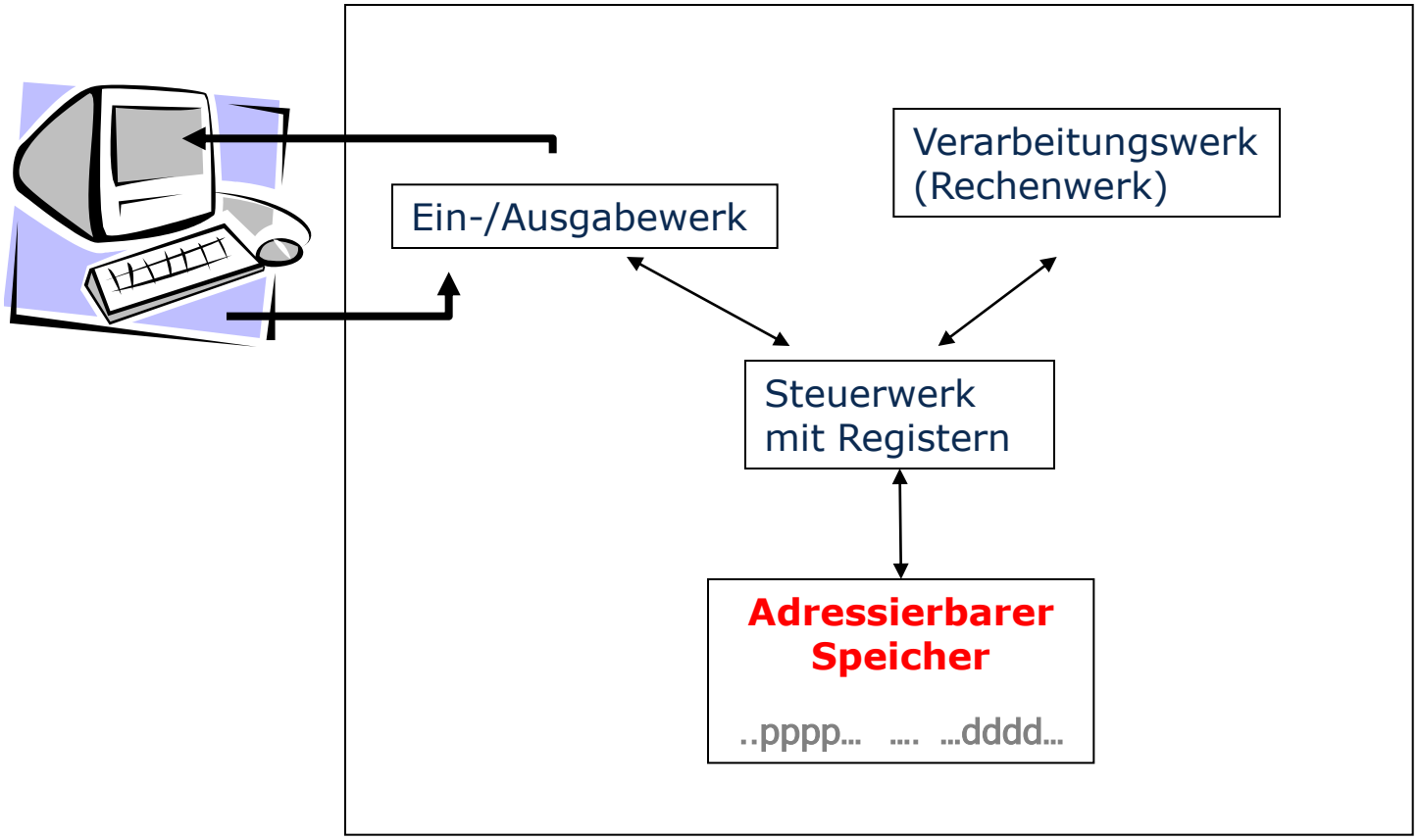
1944
v.Neumann
Computer-Architekturmodell

1946
Eckert/Mauchly

elektronischer Computer
ENIAC
(18000 Röhren, …)



J. v. Neumann - Automat



Voraussetzungen für Programmabarbeitung

Programmentwickler

Erarbeitung eines Speicherbelegungsplanes für den Programmkode

Hauptspeicheradresse	Hauptspeicherinhalt
…	…
…	…
…	…

Programmkode-Inhalt:

- Maschinenbefehle (Daten für Steuerwerk)
- Daten, belegt mit Initialwerten
- Speicherplatzreservierungen für Arbeitsdaten

Maschinenbediener

Füllen des Speichers gemäß Belegungsplan,
Einstellen der Adresse des 1. auszuführenden Befehles (Startadresse)

Programmabarbeitung

Start des Steuerwerkes, danach

Steuerschleife (zyklische Arbeit des Steuerwerkes;
Beendigung durch speziellen HALT-Befehl)

- Lesen des aktuellen Maschinenbefehls aus Speicher
- Befehlsinterpretation
- ggf. Lesen Operanden aus Speicher
- Starten/Steuern der Bearbeitung des Befehles durch Verarbeitungs- oder Ein-/Ausgabewerk
- Bestimmung der Adresse des Nachfolgebefehls

Maschinenbefehle

- Eingabedaten für Steuerwerk
- Befehlssatz - Gesamtheit der Maschinenbefehle
- Befehlskode regelt binäre Darstellung

  Inhalt: Operationskode
  Operandenwerte bzw. –adressen
  Ergebnisdatenadressen
  Adreßangabe für Nachfolgebefehl

- Beispiel: Multiplikation

  Faktor 1 sei die Zahl 16
  Faktor 2 befindet sich auf Speicheradresse 1000
  Produkt soll auf Adresse 2000 gespeichert werden
  Nachfolgebefehlsadresse 500 falls Produkt >0, sonst 600

- Befehlssätze sind in der Praxis meist einfacher als im Beispiel

Adressierungsarten

direkte Adressierung

Operandenadresse steht im Befehlswort

indirekte Adressierung

Befehlswort enthält die Adresse eines Speicherbereiches,
der die Operandenadresse enthält

registerbezogene Adressierung

Befehlswort enthält nur Relativadressen
Operandenadresse = Relativadresse + Inhalt Adreßregister

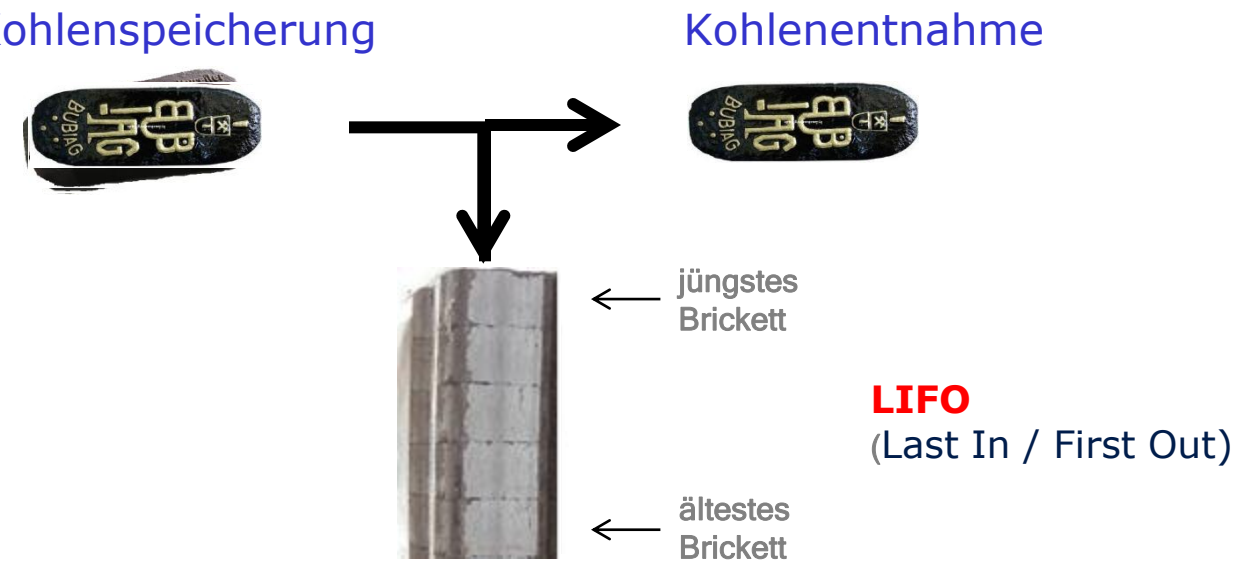
Die indirekte und die registerbezogene Adressierung
erleichtern dem Programmierer die Arbeit
mit Zeigern und Feldern.

Klassifikation der Maschinenbefehle

- **arithmetische und logische Befehle**
  Addition, bitweise logische ODER-Verknüpfung, …
- **Bitmanipulationsbefehle**
  Löschen, Setzen, Ändern einzelner Operandenbits
- **Testbefehle** (evtl. implizit in anderen Befehlsarten)
  Testen Gerätebereitschaft, Operandenvorzeichen, …
- **Sprungbefehle**
  unbedingte Sprünge für nichtsequentielle Befehlsabarbeitung
  bedingte Sprünge für Verzweigungen und Zyklen
  Unterprogramm-Aufruf und -Rücksprung
- **Transportbefehle**
  Operandentransfer Speicher-Speicher
  Operandentransfer Steuerwerk-Speicher usw.
- **Steuerbefehle**
  HALT, Schrittbetrieb usw.

Stack – (Stapelspeicher, Kellerspeicher)

Begriff abgeleitet von Kohlelagerung im Keller



- Nur **2 Zugriffsfunktionen** (keine Speicheradressierung)
  - Legen auf Stapel — sequentielles Stapeln
  - Holen vom Stapel — des zuletzt gestapelten Objektes
- Höhe des Stapels abhängig von der Menge der gestapelten Objekte

Befehlsabarbeitungsbeispiel (1)

Addition aller Zahlen von 1 bis 100, Ergebnis soll in Register 1 stehen

Sequentielle Abarbeitung

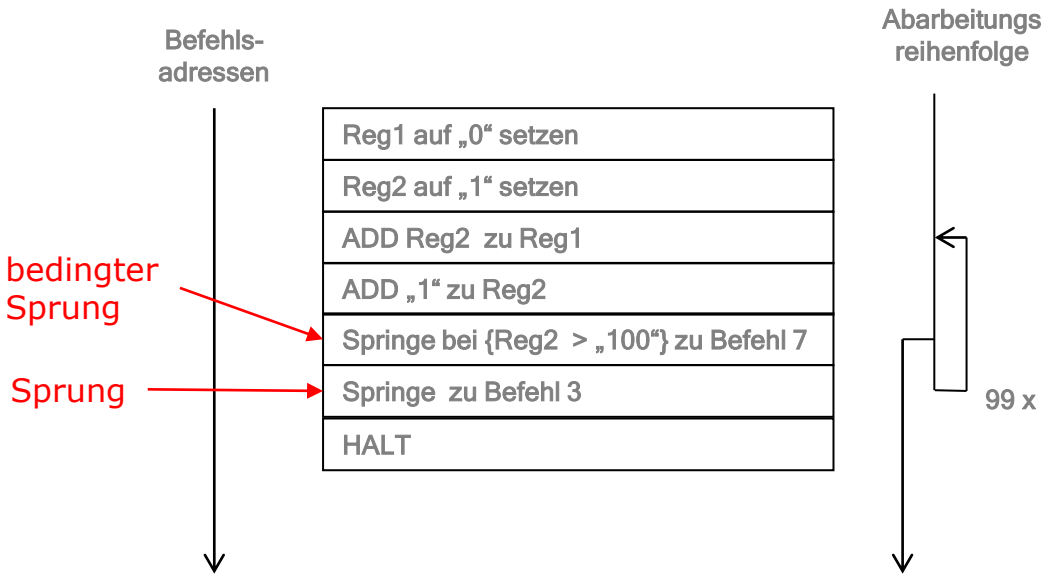
Fazit

- speicheraufwendig
  202 Befehle
- unflexibel

Befehlsabarbeitungsbeispiel (2)

nichtsequentielle Abarbeitung

mit
Sprüngen, bedingten Sprüngen, Schleifen



Fazit

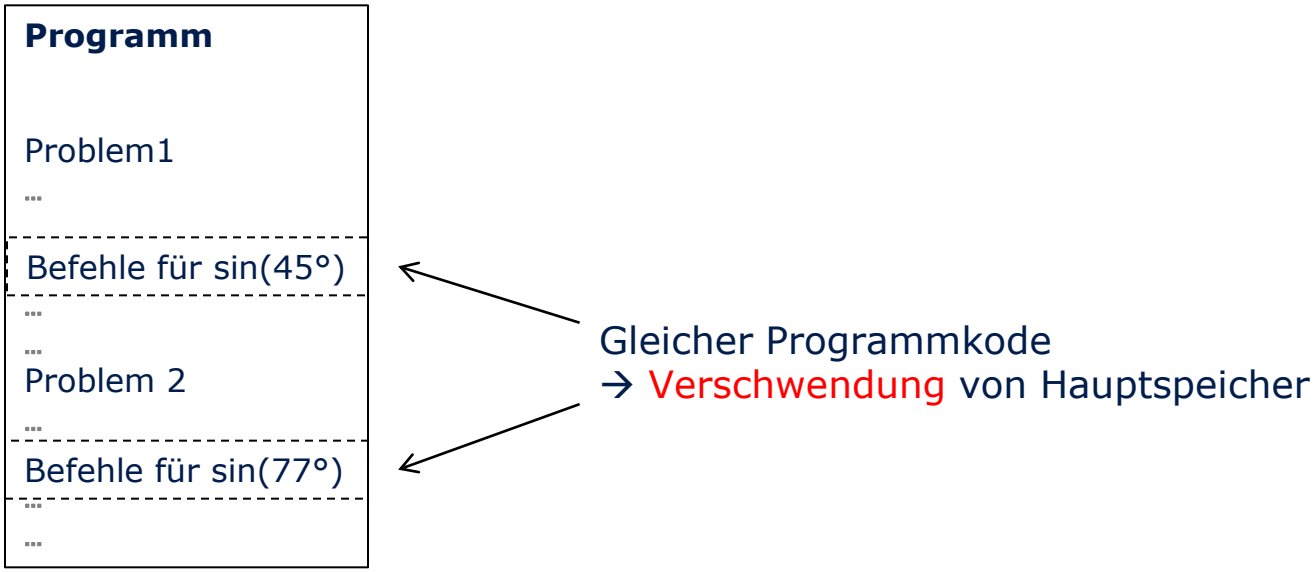
- Speichereffizient
  7 Befehle
- flexibler

Unterprogrammorganisation

Häufig benötigt man in einem Programm

- mehrmals die gleichen Befehlsfolgen
- aber in unterschiedlichen Zusammenhängen

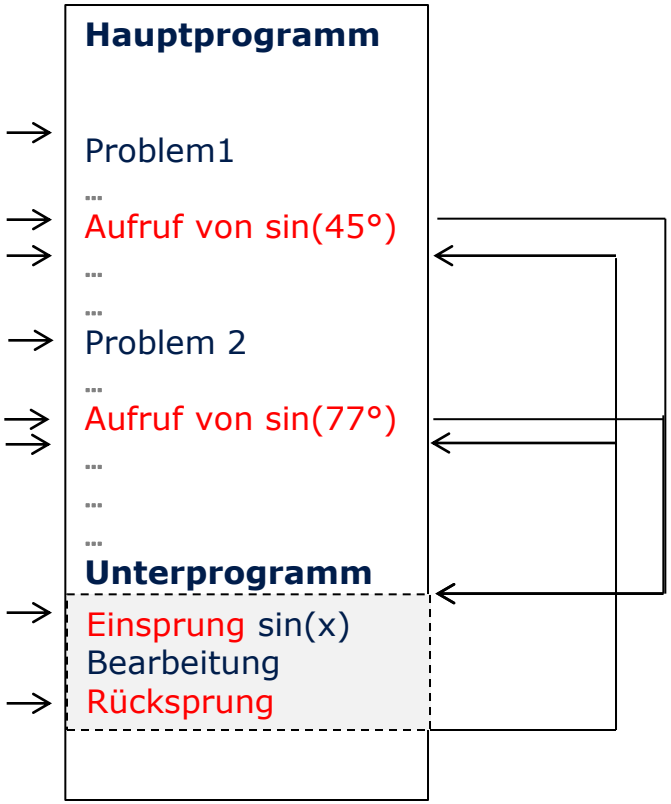
z.B. für numerische Berechnungen, wie sin(x), log(x), …



Unterprogrammorganisation

Hauptprogramm ruft an mehreren Stellen das gleiche Unterprogramm auf.

Vorteil: Programmkode nur 1x im Hauptspeicher



Problem

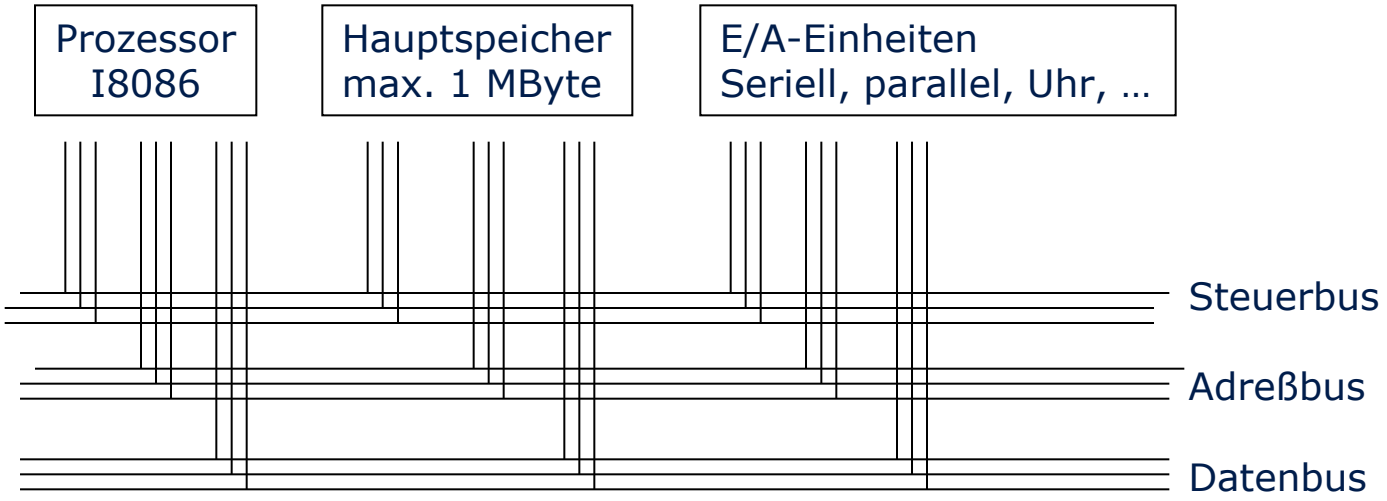
- Einsprungadresse eindeutig
- Rücksprungadresse nicht eindeutig

  → muß beim Aufruf dem UP übergeben werden

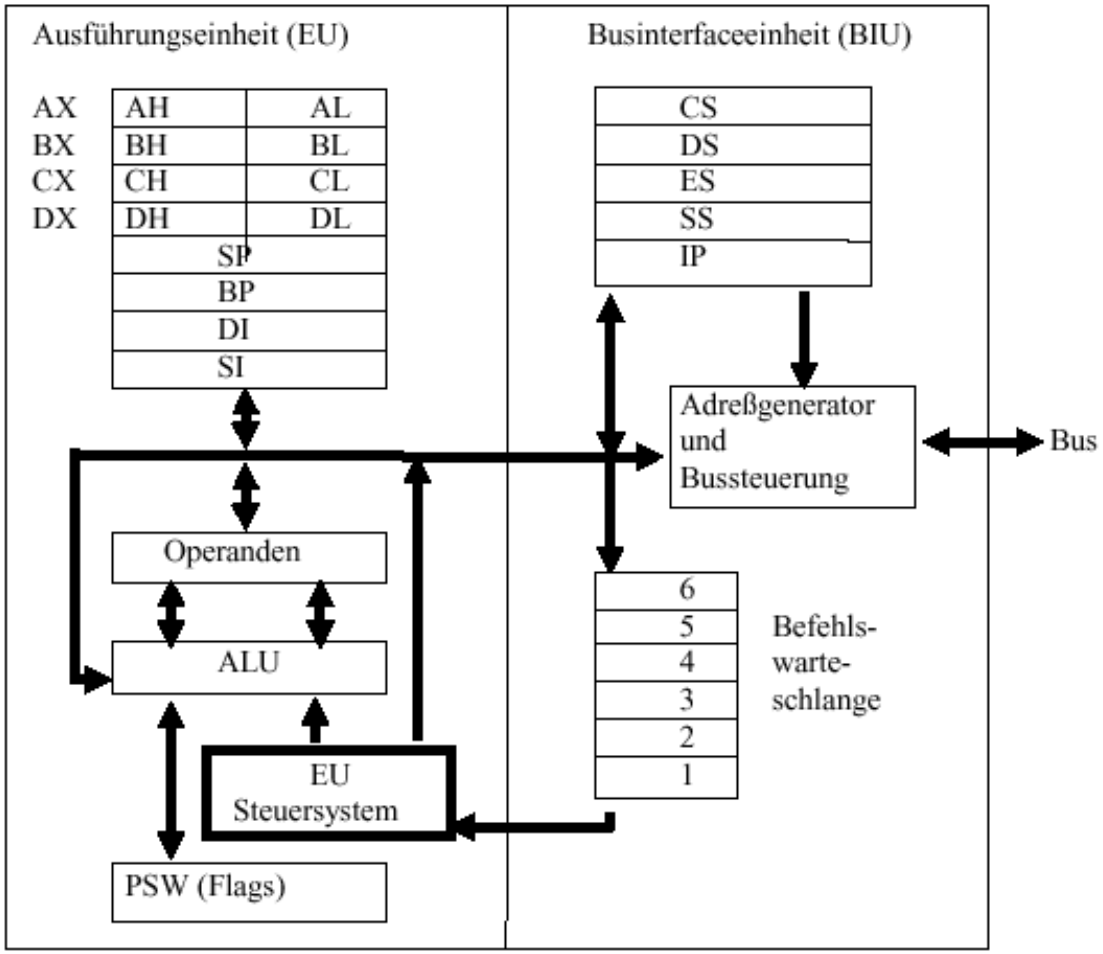
IBM- Personalcomputer PC XT

nach 1980 erster PC im Masseneinsatz

- Prozessor — Intel 8088
  auch I8086 mit Verarbeitungsbreite 16 Bit
- Hauptspeicher — max. 1 Mbyte Größe, byteweise adressiert
  Speicherzugriff mit Wortbreite 16 Bit (2 Byte)
- Architektur



Architektur Prozessor I8086



Adressierung des Hauptspeichers

Segmentierung

- Einteilung des Hauptspeichers in Segmente
- Adressierung des Hauptspeichers über zwei Adreßangaben

  - Anfangsadresse in einem Segmentregister
  - Offsetadresse (Relativadresse bezogen auf Segmentanfang)

- Segmentadressierung gewährleistet Programmverschieblichkeit

Speicherzugriff (Adreßberechnung durch BIU)

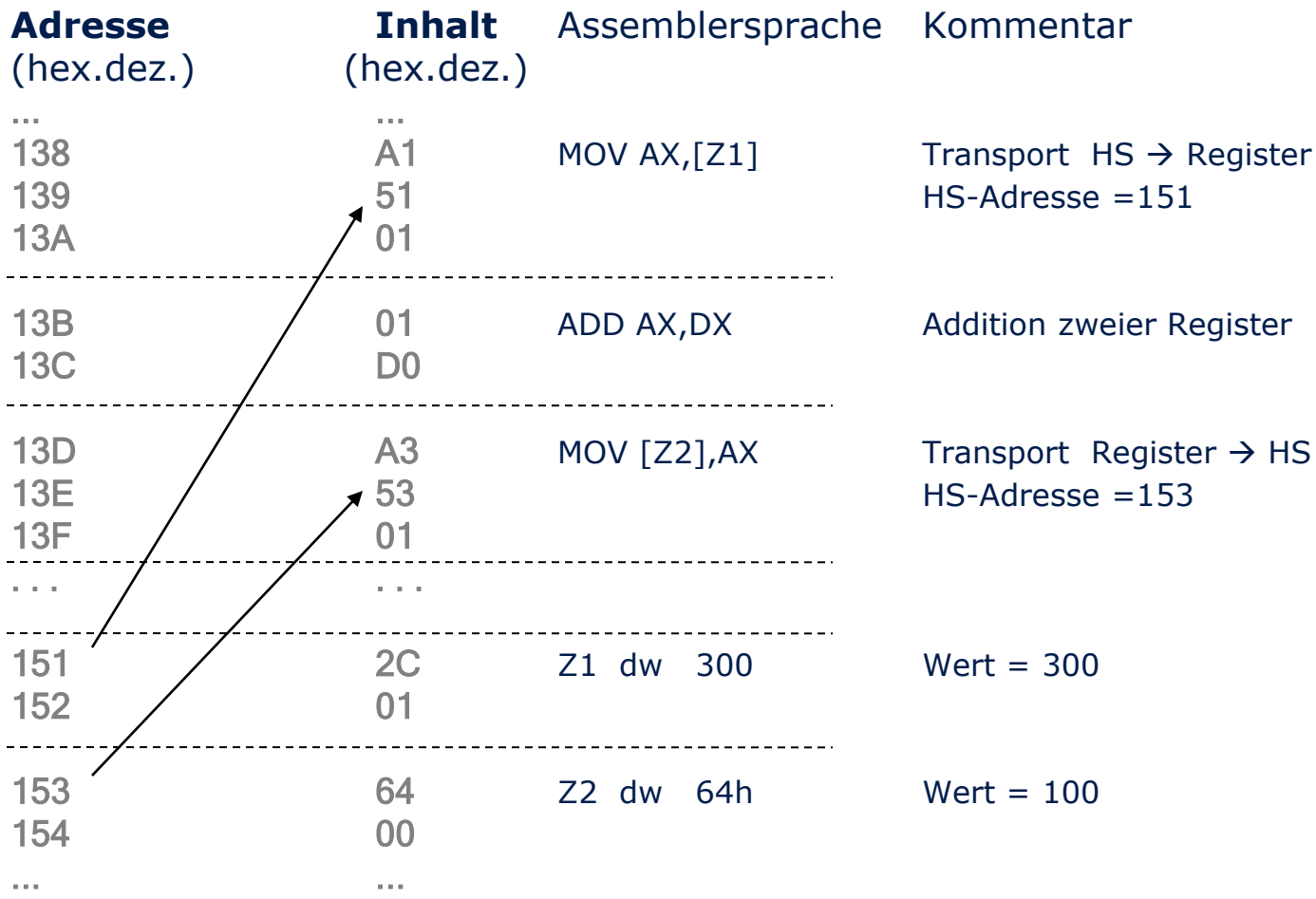
$$
\begin{array}{ll}
 & S_{16}\ S_{15}\ S_{14}\ S_{13}\ S_{12}\ S_{11}\ S_{10}\ S_9\ S_8\ S_7\ S_6\ S_5\ S_4\ S_3\ S_2\ S_1\ 0\ 0\ 0\ 0 \\
+ & \qquad\qquad\qquad\qquad O_{16}\ O_{15}\ O_{14}\ O_{13}\ O_{12}\ O_{11}\ O_{10}\ O_9\ O_8\ O_7\ O_6\ O_5\ O_4\ O_3\ O_2\ O_1 \\
\hline
= & A_{20}\ A_{19}\ A_{18}\ A_{17}\ A_{16}\ A_{15}\ A_{14}\ A_{13}\ A_{12}\ A_{11}\ A_{10}\ A_9\ A_8\ A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1
\end{array}
$$

Segmente

- Datensegment (Segmentregister DS)
- Stacksegment (Segmentregister SS : Stack Pointer SP)
- Extrasegment (Segmentregister ES)
- Kodesegment (Segmentregister CS : Instruktion Pointer IP)

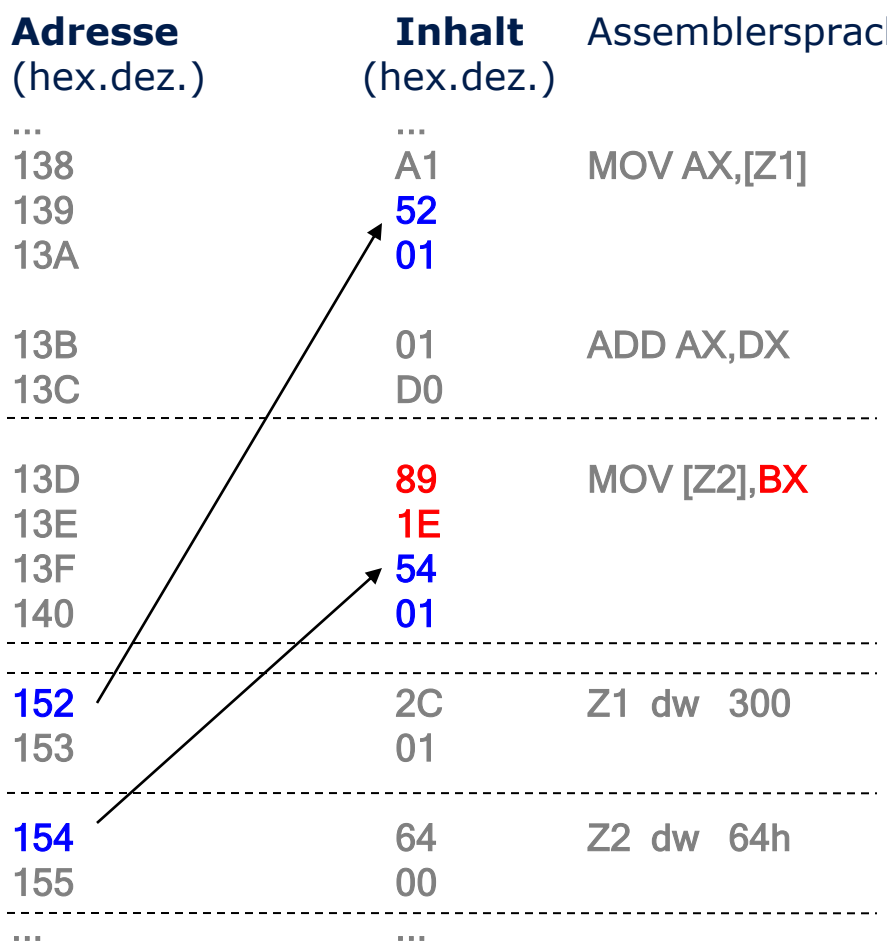
Speicherbelegungsplan

Maschinenkode Notation als Tabelle (Speicherplatzadresse/-Inhalt)



Adresse (hex.dez.)	**Inhalt** (hex.dez.)	Assemblersprache	Kommentar
...	...		
138	A1	MOV AX,[Z1]	Transport HS → Register
139	51		HS-Adresse =151
13A	01		
13B	01	ADD AX,DX	Addition zweier Register
13C	D0		
13D	A3	MOV [Z2],AX	Transport Register → HS
13E	53		HS-Adresse =153
13F	01		
. . .	. . .		
151	2C	Z1 dw 300	Wert = 300
152	01		
153	64	Z2 dw 64h	Wert = 100
154	00		
...	...		

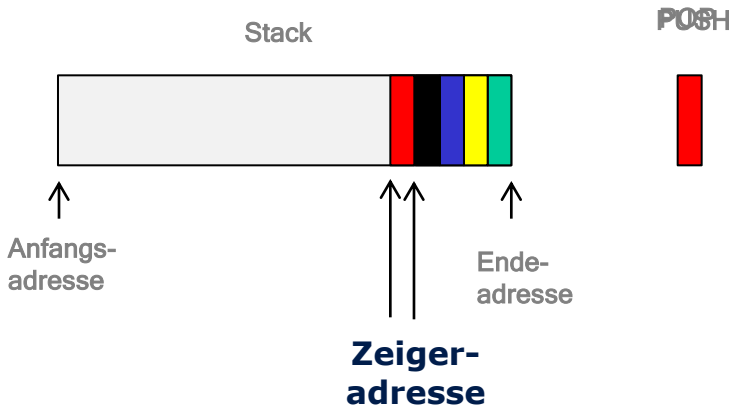
Speicherbelegungsplan nach Programmänderung



Adresse (hex.dez.)	**Inhalt** (hex.dez.)	Assemblersprache	Kommentar
...	...		
138	A1	MOV AX,[Z1]	Transport HS → Register
139	52		HS-Adresse =152
13A	01		
13B	01	ADD AX,DX	Addition zweier Register
13C	D0		
13D	89	MOV [Z2],BX	Befehl ist 1 Byte länger (!)
13E	1E		Transport Register → HS
13F	54		HS-Adresse =154
140	01		
152	2C	Z1 dw 300	Wert = 300
153	01		
154	64	Z2 dw 64h	Wert = 100
155	00		
...	...		

Stack

Teil des Hauptspeichers



Zugriffsbefehle

POP Register

1. Lesevorgang: Stackinhalt an Position der Zeigeradresse → Register
2. Zeigeradresse wird erhöht um Speicherwortlänge

PUSH Register

1. Zeigeradresse wird verringert um Speicherwortlänge
2. Schreibvorgang: Registerinhalt → Stack an Position der Zeigeradresse

Wenn Stack zu klein dimensioniert, **evtl. Stacküberlauf !!!**

Unterprogrammorganisation (vereinfacht)

Vorbereitung des Aufrufes
Alle Eingabeparameter für das Unterprogramm werden geschrieben

- in Prozessorregister
- oder in den Stack

CALL UP-Adresse

1. Schreibvorgang: Rücksprungadresse → Stack
   (= Inhalt IP-Register + Länge des Befehls „CALL")

2. Schreibvorgang: UP-Einsprungadresse → IP-Register

RETURN

1. Lesevorgang: Rückkehradresse vom Stack holen

2. Schreibvorgang: Rückkehradresse → IP-Register