Programm- und Rechnerarchitektur

Logische Komponenten

Ein Programm definiert einen Datenverarbeitungsprozess auf einem Rechner. Jede Art von Datenverarbeitung setzt sich aus zwei logischen Komponenten zusammen:

- Daten und Vereinbarungen zu deren Organisation und
- Anweisungen zur Verarbeitung der Daten (*Weinmann: Programmieren, 12*).

1. Daten sind die konkreten Werte, mit denen ein Programm arbeitet. Sie können vom Programm fest vorgegeben sein (Konstanten) oder ihr Anfangswert kann definiert sein, jedoch im Programmablauf geändert werden (Variablen). Daten können vom Anwender eingegeben oder von einem Speichermedien bzw. einer Datenbank eingelesen werden.

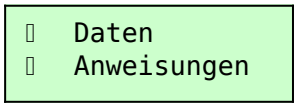
2. Es gibt Anweisungen, die den Programmablauf steuern (Kontrollbefehle). Sie legen beispielsweise die Bedingungen und die Reihenfolge fest, in der einzelne Anweisungen ausgeführt werden (Alternativen) oder ob ein Programmteil (Anweisungsblock) mehrmals wiederholt werden soll (Wiederholungen, Schleifen). Die Ablauflogik eines Programms besteht aus der Gesamtheit und dem Zusammenspiel seiner Kontrollstrukturen.

Daneben gibt es Anweisungen, die die spezielle Art der Datenverarbeitung regeln (Operationen und Funktionen). Sie legen fest, was mit den Daten gemacht werden soll und wie dies zu geschehen hat. Es gibt eine Anzahl von elementaren Operationen, die in jeder Programmiersprache vorhanden sind. Dazu gehören beispielsweise arithmetische und logische Operationen, sowie Eingabe- und Ausgabeanweisungen.

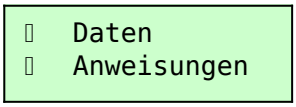
Diese Grundbestandteile jeder Programmiersprache unterliegen gewissen Regeln. Die *Syntax* regelt die Orthographie (richtige Schreibweise) und die Grammatik (erlaubte Kombination) von Anweisungen. Die *Semantik* beschreibt die inhaltliche Bedeutung von Anweisungsfolgen. Regeln bestimmen unter anderemhttps://wiki.debian.org/dmesg auch die Anordnung und Reihenfolge der logischen Komponenten in einem Hochsprachenprogramm. So müssen Daten in (fast) allen Programmiersprachen in bestimmten Abschnitten des Quellcodes erklärt (deklariert) werden.

Trennung von Daten und Anweisungen

Das Prinzip der Trennung von Daten und Anweisungen ist ein fundamentales Arbeitsprinzip der Datenverarbeitung. Es bestimmt auch die Architektur eines Computers. Die physikalische Struktur und die Speicherorganisation eines Rechners entspricht dem Prinzip der Trennung von Daten und Anweisungen in einem Hochsprachenprogramm.



Symbolisches Programm



Speicherorganisation

Rechnerarchitektur (von-Neumann-Architektur)

Daten und Anweisungen sind im Computerspeicher immer in binärer Form ohne zusätzliche Kennzeichen abgelegt. Der Prozessor kann an ihrer Struktur nicht erkennen, um welche Art logischer Komponente es sich handelt. Damit er Daten und Anweisungen unterscheiden kann, ist eine bestimmte Art der Repräsentation im Arbeitsspeicher erforderlich. John von Neumann (1903-1957) wies nach, dass diese Tatsache ein prinzipielles Problem darstellt. Wie kann verhindert werden, dass der Computer Anweisungen als Daten und Daten als Anweisungen interpretiert?

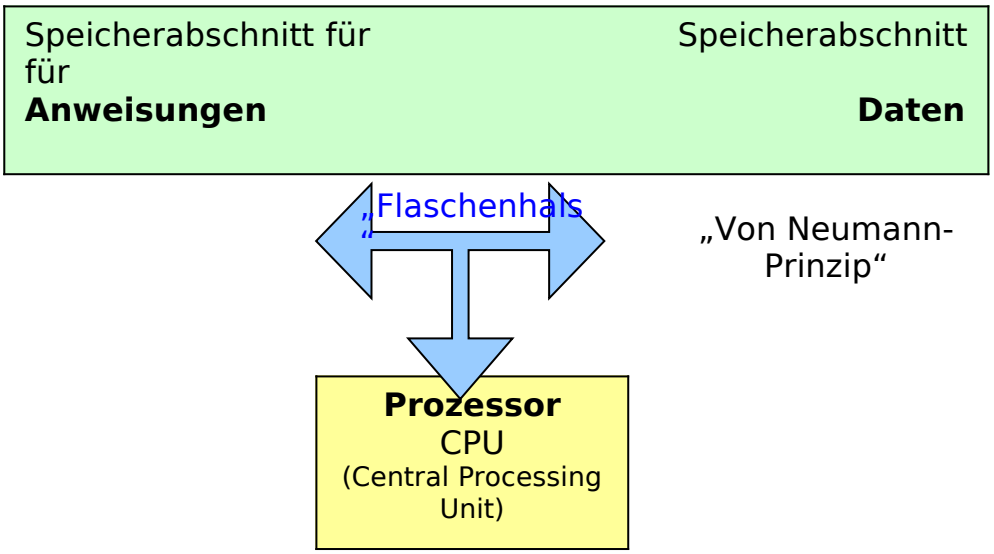
http://tal.cs.tu-berlin.de/ifp/fiff/Stach.html

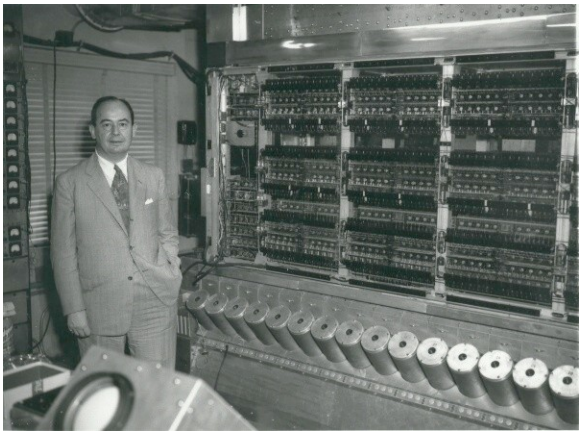
”First Draft of a Report on the EDVAC”, 1945

Die einfachste und effektivste Lösung ist es, Anweisungen und Daten an festgelegten Orten innerhalb des Computerspeichers abzulegen. Dann kann der Computer an der ihm bekannten Speicheradresse erkennen, ob es sich um eine Anweisung oder um ein Datum handelt. Alle gängigen Rechnersysteme basieren heute auf dem Prinzip der Trennung von Anweisungen und Daten im Computerspeicher.

Es wäre möglich, Anweisungen mit einem bestimmten Kennzeichner zu versehen, wie es mit der alternativen Harvard-Architektur vorgeschlagen wurde. Diese Alternative hat sich jedoch nicht auf breiter Front durchgesetzt.

Da der Speicherzugriff und die Datenübermittlung zwischen Speicher und Prozessor (aus Gründen der durchgängigen Adressierung) über einen gemeinsamen Weg (den Datenbus) erfolgt, entsteht hier eine Engstelle, der sogenannte „Flaschenhals“. Moderne Prozessoren überwinden den dadurch entstehenden Datenstau, aber sie müssen immer noch die Aufteilung des Speichers in einen Anweisungsteil und einen Datenteil berücksichtigen. Diese Tatsache hat grosse Bedeutung für die Programmierung.



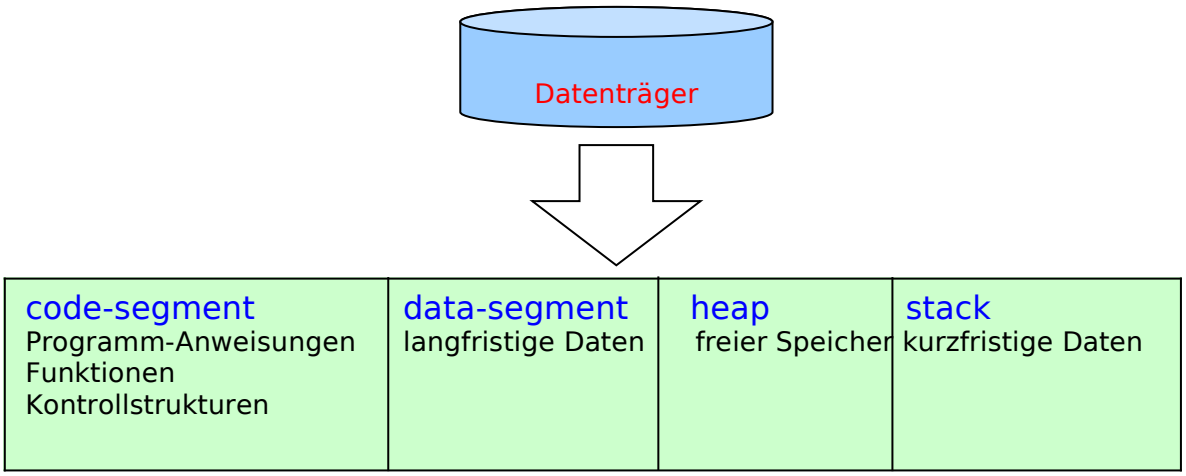


John von Neumann (1903 – 1957)

http://www.eingang.org/Lecture/neumann.html
http://www.csupomona.edu/~hnriley/www/VonN.html

Speicherorganisation

Ein ausführbares Programm, das in binärer Form auf einem Datenträger (lokales Laufwerk oder Netzwerklaufwerk) gespeichert ist, muss vor der Ausführung in den Arbeitsspeicher kopiert (geladen) werden. Das Betriebssystem sorgt dafür, dass bereits beim Laden der für das Programm zur Verfügung gestellte Speicher in einen Anweisungsteil (code-segment) und einen Datenteil (data-segment) aufgeteilt wird. Die Programmanweisungen kommen in das code-segment und die Daten in das data-segment. Hierdurch sind beim Start des Computerprogramms die hardwaremässigen Voraussetzungen für eine logische Trennung gewährleistet.



Beim Start des Computerprogramms holt der Prozessor die erste Anweisung aus dem Anfang des Codesegments und führt sie aus. Danach werden alle folgenden Anweisungen des Programmes sequentiell abgearbeitet. Der Prozessor führt genau Buch über die jeweils abgearbeitete Speicherzelle (Befehlsregister) und weiss deshalb, welche Anweisung als nächste auszuführen ist. Bei Programmsprüngen wird die Information (Speicheradresse) des zuletzt ausgeführten Befehls zwischengespeichert.

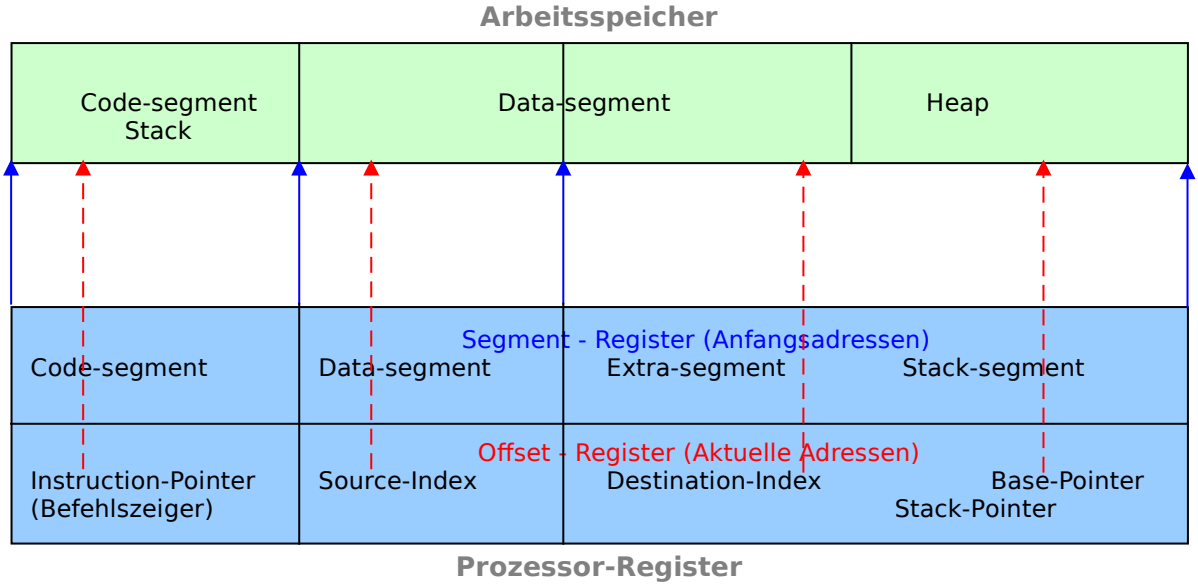
Wenn das Programm ausgeführt wird (zur Laufzeit eines Programms) sind immer auch aktuelle Daten zu verarbeiten, die beispielsweise vom Nutzer eingegeben oder die für die Berechnung von Zwischergebnissen benötigt werden. Der Computer kann beim Laden des Programms noch nicht entscheiden, wieviel Speicherplatz dafür benötigt wird.

Deshalb gibt es weitere Speicherbereiche, die zur Laufzeit nach Bedarf belegt, gelöscht und wieder neu belegt werden können. Hierbei wird unterschieden zwischen kurzfristig benötigtem Speicher und langfristig benötigtem Speicher. Der langfristige Speicher (heap, Halde) kann unter einem modernen Betriebssystem wie Windows nahezu beliebig vergrössert werden, indem Teile des Festplattenspeichers mitverwendet werden, falls der Hauptspeicher nicht ausreicht. Der stack (Stapel, Keller) ist grössenmässig begrenzt und sein Inhalt wird automatisch gelöscht, wenn er nicht mehr benötigt wird. Das Speicherverwaltungssystem teilt dem Prozessor mit, wo diese Speicherbereiche liegen und dieser merkt sich die Speicheradressen in speziellen Registern.

Prozessorarchitektur und Speicheradressierung

Der interne Register-Aufbau eines Intel-Prozessors der Familie x86 entspricht der aufgezeigten Struktur eines Maschinenprogramms und seiner Speicher-Repräsentation. Jedem der vier Speicherbereiche ist ein Segment-Register und ein Offsetregister zugeordnet. Jedes der Segmentregister weist auf den Beginn des zugeordneten Adressbereichs.

Das entsprechende Offsetregister repräsentiert den Abstand von der Anfangsadresse (Offset), also den momentanen Stand der Befehls- bzw. Datenfolge. Die Speicherverwaltung berechnet aus den relativen Adressen (Segment + Offset) die absolute Speicheradresse und kann damit auf den Speicherinhalt zugreifen.



Bei der Programmierung ist die Kenntnis dieser grundlegenden Speicherorganisation von grossem Nutzen. Viele Programmierprobleme sind auf eine unzureichende Kenntnis der computerinternen Vorgänge zurückzuführen. Auch wenn moderne Programmier-sprachen von diesen Vorgängen weitgehend abstrahieren, ist es dennoch manchmal für die erfolgreiche Fehlersuche (debugging) unerlässlich, davon eingehende Kenntnis zu haben.

Alle modernen Programmiersprachen bauen letztlich auf dieser Speicherorganisation auf. Das Verständnis für objektorientierte Programmierung befreit nicht von der Kenntnis diesen hardwaremässigen Voraussetzungen (Objekte werden in Java grundsätzlich auf dem Heap angelegt). Letztendlich müssen auch alle verteilten Anwendungen auf einem konkreten Computer mitsamt seiner Speicherorganisation ausgeführt werden.