All About Memory

Lennart Armbrust

Sommerakademie in Leysin, August 2016

Abstract

Speicher spielen im Computer eine zentrale Rolle. Sie werden sowohl zur dauerhaften Sicherung von Daten als auch zum Vorhalten von Arbeitsdaten und Befehlen gebraucht. Im Folgenden gehe ich auf die Funktionsweise verschiedener Speichertypen ein und erläutere die Speicherhierarchie eines Computers.

Inhalt

1 Einleitung

Ohne Speicher könnte ein Computer nicht funktionieren. Nicht nur, dass wir keine Daten speichern könnten, um sie bei der nächsten Benutzung wieder verwenden zu können. Auch fehlte uns die Möglichkeit, dem Prozessor ständig neue Befehle und Daten zuzuführen, mit denen dieser arbeitet. Es deutet sich schon an: Diese verschiedenen Aufgaben mit verschiedenen Bedürfnissen erfordern verschiedene Wege, Daten zu speichern.

2 Verschiedene Speichertypen

Von den Anforderungen, die wir an Computer stellen, haben zwei Zentrale etwas mit dem Speicher zu tun:

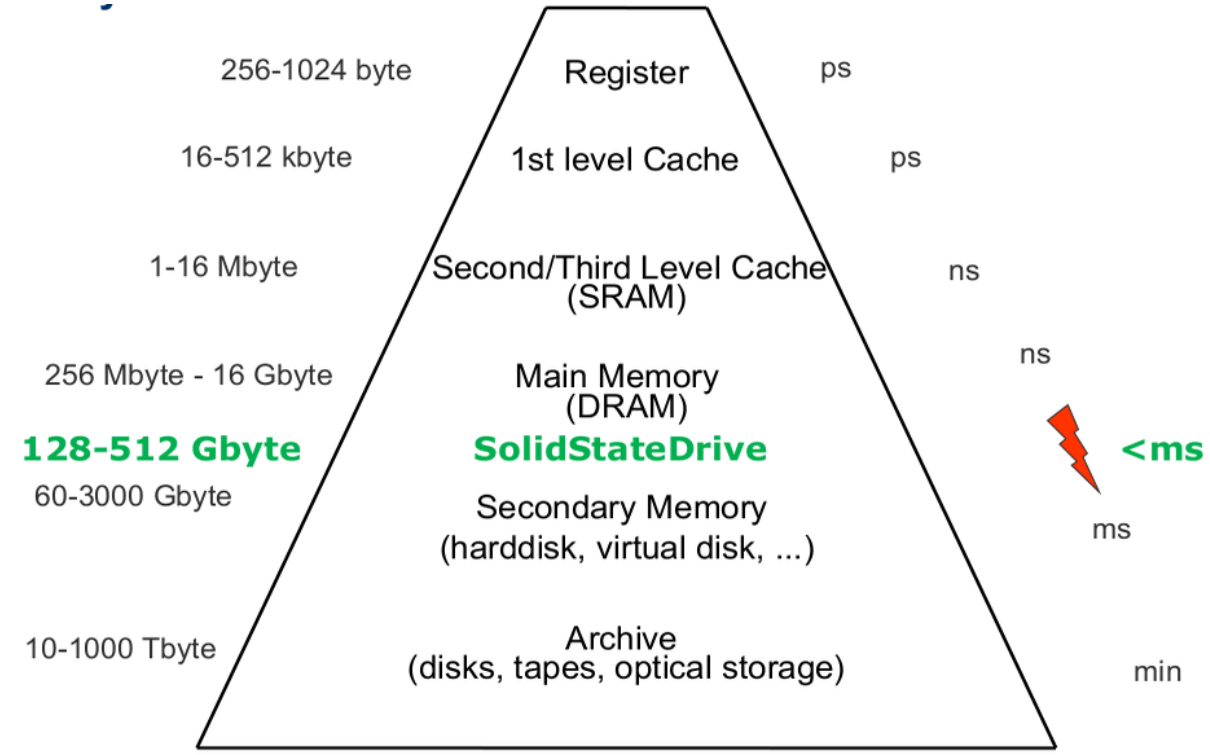
Zum einen die Geschwindigkeit. Wie wir wissen, hängt diese in erster Linie davon ab, wie schnell der Computer – bzw. genauer: sein Prozessor – rechnen kann. Damit der Prozessor seine Möglichkeiten voll ausschöpfen kann, ist es wichtig, dass ihm möglichst immer alle Daten, die er zum Rechnen braucht – die Daten, die er bearbeiten soll, sowie die Befehle, die er auf diese Daten anwenden soll – schnell zur Verfügung stehen, damit er nicht warten und so die möglichst zügige Abarbeitung verzögern muss.

Zum anderen die Größe des Speichers, der uns zum Ablegen von Daten wie Dateien, Bildern, Musik, Filmen etc. zur Verfügung steht.

Wir unterteilen also in *Arbeits*- und *Massenspeicher*. Wie wir im Folgenden sehen werden, fällt die Unterteilung allerdings noch deutlich feingliedriger aus. Dazu müssen wir zunächst verstehen, wie der Arbeitsspeicher im Computer organisiert und mit dem Prozessor verbunden ist.

In der Vergangenheit ging man davon aus, dass Arbeitsspeicher teuer und somit sparsam einzusetzen sei. Man entwickelte eine Architektur, die den teuren Speicher möglichst gut ausnutzen sollte: die *Von-Neumann-Architektur*. Kern dieser Architektur ist die Idee, den Arbeitsspeicher sowohl für Daten als auch für Befehle zu benutzen. Dies war kein Problem, da der limitierende Faktor bei der Geschwindigkeit des Computers die Rechengeschwindigkeit war. Verzögerungen durch Speicherzugriffe sind allerdings immer stärker zum Problem geworden, dem man begegnen musste, da die Geschwindigkeit der Prozessoren deutlich schneller wuchs als die Geschwindigkeit der Speicheranbindungen, der sogenannten *Speicherbusse*. Wie dieses Problem abgemildert werden konnte, werde ich später erklären.

Zunächst einmal wollen wir einen Blick auf die verschiedenen Speichertypen und deren Zugriffszeiten werfen:



Anhand dieser Abbildung wird deutlich, wie sich Größenverhältnisse und Zugriffszeiten unterscheiden. Eine weitere Idee wird durch die angedeutete Pyramide dargestellt: die *Speicherhierarchie*. Wie wir schon wissen, gibt es verschiedene Anforderungen an den Speicher: Er soll einerseits möglichst groß und andererseits möglichst schnell sein. Der Nutzer möchte im Idealfall davon nichts mitbekommen und einen sehr großen, sehr schnellen Speicher zur Verfügung haben. Da dies sehr teuer und aufwändig wäre, bedient man sich eines Tricks: Man hierarchisiert den Speicher und setzt verschiedene Speicherformen mit unterschiedlichen Eigenschaften ein. Es gibt kleinen und sehr schnellen Speicher sowie sehr großen und weniger schnellen Speicher. Durch geschickte Verwaltung können damit mit sehr viel geringerem Aufwand die Eigenschaften Geschwindigkeit, Größe und (scheinbare) Homogenität gleichermaßen bedient werden.

3 Die Hardware

Dieser Speicherhierarchie liegen, wie in der Abbildung oben deutlich wird, verschiedene Arten von Speicher zu Grunde. Auf verschiedene Bauformen und Konzepte, die sich sehr grundlegend unterscheiden können, gehe ich im Folgenden ein.

3.1 Register

Register sind so etwas wie die Arbeitsfläche des Computers. Für jede ausgeführte Instruktion werden die benötigten Daten in Register geladen. Die General-Purpose-Register stehen dabei zur freien Verfügung, während in speziellen Registern Werte gespeichert werden, die nicht überschrieben werden dürfen. Ein Beispiel hierfür ist der Instruction Pointer, der stets die Adresse des aktuell ausgeführten Befehls enthält.

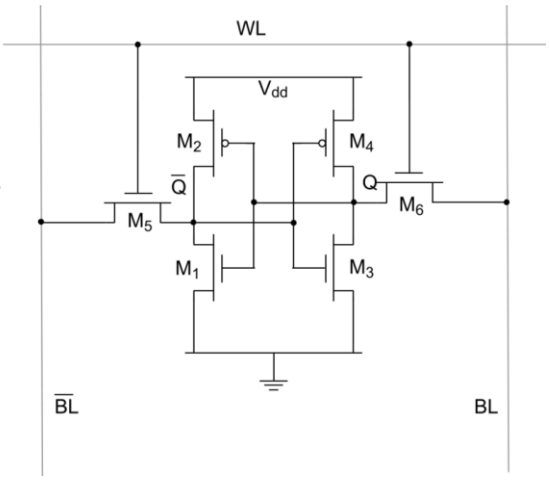
3.2 Caches

Caches sind kleine Speicher, die oft direkt im Prozessor eingebaut sind. Sie sind jene Speicher, die man zur Dämpfung des oben angesprochenen *Von-Neumann-Flaschenhalses* verwendet. Sie halten bestimmte Abschnitte aus dem Arbeitsspeicher vor, auf die oft zugegriffen wird. Fragt der Prozessor eine Speicherzelle an, die im Cache vorgehalten wird, so bekommt er den Inhalt direkt und sehr viel schneller, als wenn er dafür auf den Arbeitsspeicher zugreifen müsste. Moderne Prozessoren verfügen über mehrere Ebenen an Caches. Diese unterscheiden sich durch Größe, Realisierungsform und Geschwindigkeit. Möchte der Prozessor ein Datum aus dem Speicher haben, dann kann es sein, dass der erste Cache dieses Datum nicht vorhält. In diesem Fall müsste der zweite Cache gefragt werden usw. Ist ein Cache voll und sollen weitere Daten eingelagert werden, helfen Ersetzungsstrategien dabei, zu entscheiden, welche Daten aus dem Cache verdrängt werden. Wichtige Ersetzungsstrategien sind: *First In First Out* (die Daten, die sich am längsten im Prozessor befinden, werden verdrängt), *Least Recently Used* (die Daten, auf die am längsten nicht zugegriffen wurde, werden verdrängt), *Least Frequently Used* (die Daten, auf die am seltensten zugegriffen wurde, werden verdrängt) etc. Solange nur Daten aus diesen Caches gelesen werden, reicht uns dieses Konzept aus. Wenn aber Daten aus dem Speicher, die im Cache vorgehalten werden, verändert werden sollen, müssen wir uns Gedanken um die Konsistenz dieser Daten machen. Hierzu gibt es verschiedene Ansätze. Die wichtigsten zwei sind *write-through* und *write-back*. Beim write-through wird einfach gleichzeitig das Datum im Cache (in allen Caches) sowie im Arbeitsspeicher verändert. So ist die ständige Kohärenz des Speichers gewährleistet, allerdings braucht ein schreibender Zugriff auf den Cache auch länger, da eben auch auf dem Arbeitsspeicher zugegriffen werden muss, auch wenn das Datum eigentlich im Cache liegt. Eine weitere Möglichkeit ist das write-back, das die Daten zunächst nur im Cache verändert und mit einem sogenannten *Dirty Bit* versieht. Dieses zeigt dann an, dass die Daten verändert wurden und vor dem Verdrängen im Arbeitsspeicher verändert werden müssen. Dies ist schneller als das write-back, erfordert aber mindestens ein Bit Speicherplatz pro Speicherzelle im Cache.

3.3 SRAM

SRAM steht für *Static Random Acces Memory* und ist eine Bauform von Arbeitsspeicher. SRAM wird in modernen Computern für den Cache benutzt. Der Speicher ist flüchtig, das heißt, dass die

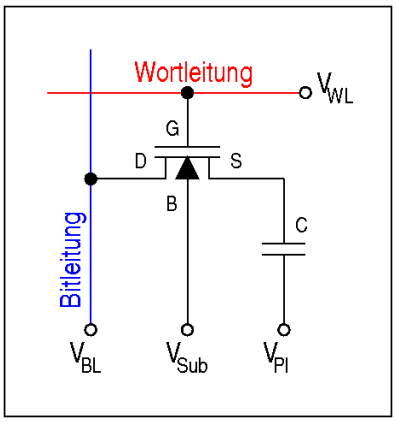
abgelegten Daten verloren gehen, sobald die Betriebsspannung wegfällt. SRAM ist schnell, aber die benötigte Hardware ist vergleichsweise aufwendig: Pro Bit werden sechs Transistoren benötigt. Ein einmal gespeicherter Wert bleibt im SRAM enthalten, solange die Betriebsspannung anliegt. Rechts ist eine SRAM-Zelle zu sehen.



SRAM-Zelle, Abelsson, CC BY-SA 3.0

3.4 DRAM

DRAM steht für *Dynamic Random Access Memory* und ist ebenfalls eine Bauform für Arbeitsspeicher. DRAM wird in modernen Computern für den eigentlichen Arbeitsspeicher verwendet und ist ebenfalls flüchtig. Der Vorteil von DRAM ist, dass er anstatt der sechs Transistoren bei SRAM nur einen Transistor und einen Kondensator pro Bit benötigt, was ihn sehr viel kompakter und günstiger macht. Allerdings ist ein einmal gespeicherter Wert nur so lange sicher, bis der Kondensator sich entladen hat. Deswegen muss der Speicher periodisch aufgefrischt werden. Rechts ist eine DRAM-Zelle zu sehen.



DRAM-Zelle, JürgenZ., CC BY-SA 3.0

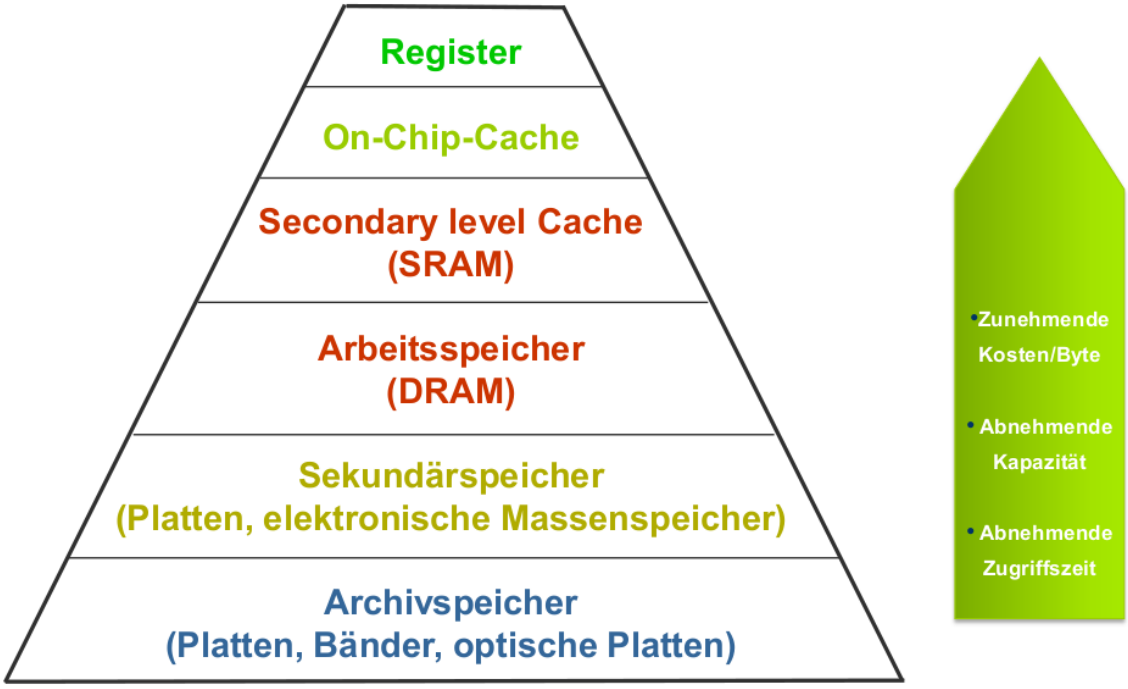
3.5 SSD

SSD steht für *Solid State Drive* und ist eine Bauform von Massenspeicher. SSDs basieren wie alle bisher vorgestellten Speicherformen auf Halbleitern. Sie sind allerdings nicht flüchtig, weswegen sie zum dauerhaften Speichern von Daten geeignet sind. Für einen Massenspeicher verfügen SSDs über eine hohe Geschwindigkeit und sind energieeffizient. Außerdem sind sie unempfindlich gegen Stöße, weswegen sie prädestiniert sind für den Einsatz in mobilen Geräten.

3.6 HDD

HDD steht für *Hard Disk Drive* und ist ebenfalls eine Bauform für Massenspeicher. Es handelt sich bei HDDs um die klassischen *Festplatten*. HDDs bestehen aus rotierenden Magnetscheiben, auf denen sich Spuren befinden, die mit einem Arm gelesen und beschrieben werden können. Die Vorteile von Festplatten sind die hohe Kapazität, der günstige Preis sowie die vergleichsweise hohe Beständigkeit. Durch ihre beweglichen Teile (rotierende Scheiben sowie Schreib-Lese-Kopf) sind sie anfällig für Stöße. Außerdem haben Sie eine vergleichsweise lange Zugriffszeit, insbesondere, wenn Daten nicht hintereinander auf der Scheibe liegen, da sich dann der Arm ständig zum neuen Ort bewegen muss.

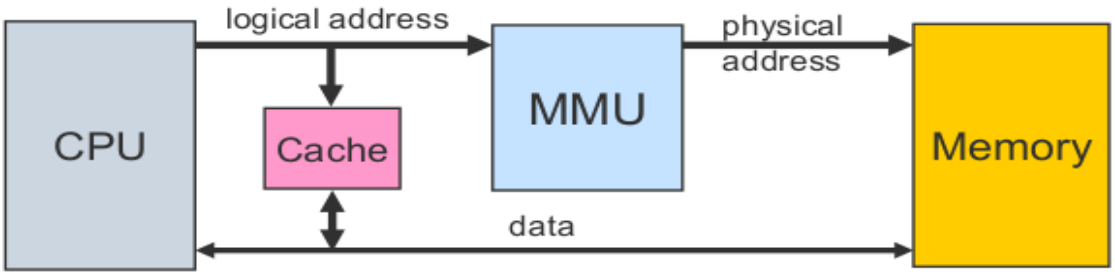
3.7 Vergleich



4 Speicherhierarchie

Wie oben bereits angesprochen ist es das Ziel der Hierarchisierung von Speicher, den Zielkonflikt zwischen Größe und Geschwindigkeit von Speicher aufzulösen. Wenn bestimmte Voraussetzungen eingehalten werden, wirkt hierarchisierter Speicher wie ein großer, zusammenhängender Speicher. Es muss die *Lokalität der Programmverarbeitung* gegeben sein. Das bedeutet, dass im Arbeitsspeicher Programme zusammenhängende Blöcke bekommen sollen und im Idealfall die folgende Instruktion stets in der folgenden Speicherzelle steht. Dies ist wichtig, damit recht erfolgreich vorhergesehen werden kann, auf welchen Speicher in naher Zukunft zugegriffen wird. Dies wiederum ist wichtige Voraussetzung dafür, dass es zur *rechtzeitigen Umlagerung von Informationen* kommen kann. Dies ist wichtig, damit möglichst viele Daten, auf die zugegriffen wird, bereits bei ihrem ersten Zugriff im Cache liegen. Des Weiteren darf die *Inhomogenität des Speichersystems für Benutzer nicht sichtbar* sein. Das bedeutet, dass der Benutzer den Eindruck haben soll, er habe eben jenen sehr großen und sehr schnellen, homogenen Speicher. Dies erreicht man durch *Virtualisierung.*

5 Virtualisierung



Virtualisierung hat, wie bereits beschrieben, das Ziel, den Speicher homogen erscheinen zu lassen. Hierfür stattet man den Computer mit einer *Memory Management Unit* (MMU) aus, die die physikalischen Adressen des Speichers in logische Adressen übersetzt. Der Computer verfügt so über einen sehr großen Bereich virtueller Adressen. So kann jeder Anwendung ein eigener Bereich an Speicher zur Verfügung gestellt werden, obwohl die Bereiche der Programme zusammengenommen deutlich größer als der real existierende Speicher sein können. Dies bietet unter anderem auch den Vorteil, dass die Memory Management Unit Versuche von Programmen entdecken kann, auf Speicherbereiche anderer Programme zuzugreifen, und so zur Sicherheit beiträgt.

Literatur

Sehr geholfen hat mir bei der Erstellung des Vortrags die Vorlesung *Rechnerarchitektur* von Prof. Jochen Schiller an der Freien Universität Berlin. Aus dieser Vorlesung stammen drei Grafiken, die ich hier verwendet habe. Des Weiteren möchte ich auf folgende Artikel in der Wikipedia verweisen:

- https://de.wikipedia.org/wiki/Cache
- https://de.wikipedia.org/wiki/Static_random-access_memory
- https://de.wikipedia.org/wiki/Dynamic_Random_Access_Memory