1 Grundlagen 1
1.1 Rechnerarchitektur 1
1.2 Takt 2
1.3 Speicherarchitektur 2
2 Mikroprozessor 3
2.1 Begriffsbestimmung 4
2.2 Geschichte 4
2.3 Caches 5

1 Grundlagen

1.1 Rechnerarchitektur

Die meisten Rechner sind im Inneren ähnlich aufgebaut: Es gibt eine zentrale Recheneinheit (CPU, Central Processing Unit), einen Speicher (in dem Daten und Maschinenprogramm gespeichert werden) und eine Vielzahl von Peripherieeinheiten, die für die Kommunikation mit der Umwelt sorgen. Diese Komponenten sind untereinander (Abbildung 1) über sog. *Busse* verbunden.

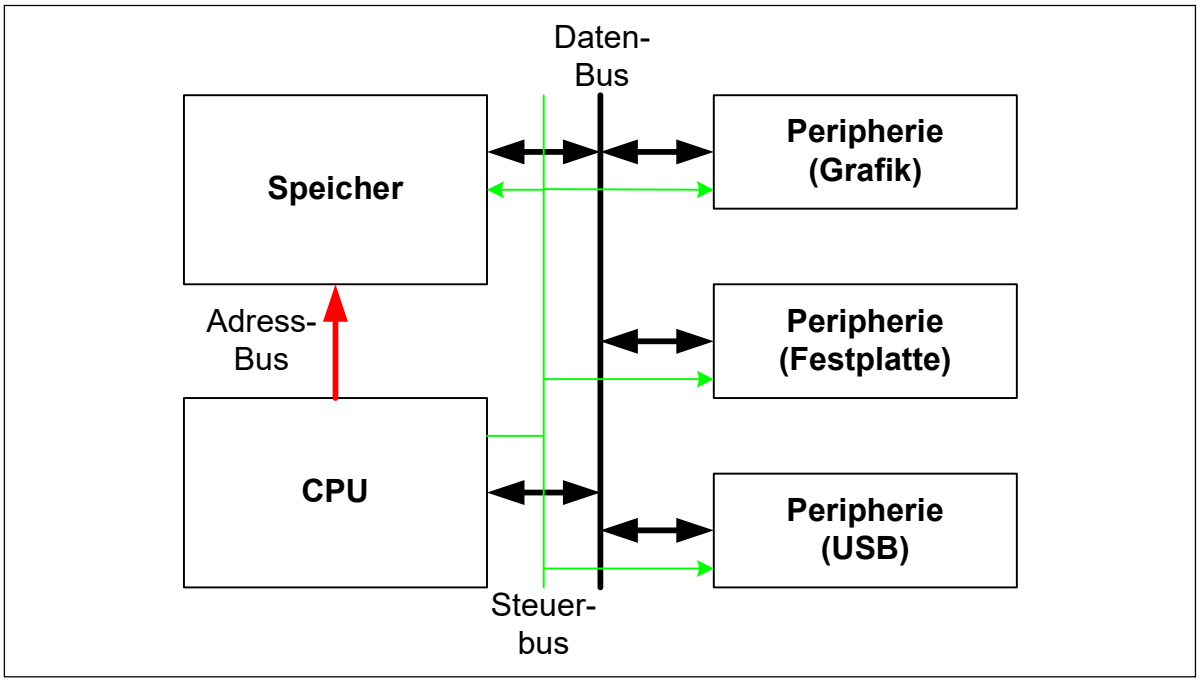

Abbildung 1: Prinzipielle Rechnerarchitektur

Ein *Bus* ist der Sammelbegriff für ein Leitungsbündel, für das wenigstens eine der drei folgenden Eigenschaften zutrifft:

1. Die Leitungen haben alle dieselbe Funktion, zur Geschwindigkeitserhöhung werden eben nur parallel mehrere Signale übertragen. Typisch ist dafür der Datenbus, der z.B. 32 Datenleitungen umfasst, damit ein 32 Bit Codewort (z.B. ein Integerwert mit 32 Bit Länge) auf einmal übertragen werden kann.
2. Die Leitungen haben zwar unterschiedliche Funktion, dienen aber in ihrer Gesamtheit demselben Zweck. Typisch ist der Steuerbus, mit dessen unterschiedlichen Signalen die CPU die anderen Komponenten koordiniert.
3. An die Leitungen sind mehr als zwei Teilnehmer angeschlossen. Typisch ist wieder der Datenbus oder auch der USB-Bus. Bei letzterem erfolgt die Datenübertragung zwar

seriell, aber an die beiden im Bus vorhandenen Datenleitungen können (nahezu) beliebig viele Teilnehmer angeschlossen werden.

In einem PC sind beispielsweise im PCI-Bus alle drei Teilbusse (Adressbus, Datenbus, Steuerbus) zusammengefasst, so dass ein auf einem Steckplatz eingestecktes Gerät mit allen anderen wesentlichen Komponenten des PC verbunden ist. Teilweise werden auch spezielle Busse verwendet, um eine nochmals schnellere Verbindung zwischen zwei Komponenten herzustellen. Ein Beispiel wäre im PC die separate Anbindung einer Grafikkarte über einen eigenen Bus (AGP o.ä.).

1.2 Takt

Eine wesentliche Eigenschaft nahezu aller heutigen Rechner ist, dass alle Komponenten *synchron* arbeiten. Damit ist gemeint, dass Operationen mit einem fest definierten Zeitraster, dem Takt, synchronisiert ablaufen. Früher gab es in einem Rechner einen einzigen Takt, mit dem alle Komponenten gleichermaßen versorgt wurden. Heute haben viele Komponenten ihre eigenen, lokalen Takte. So kann jede Komponente mit der Geschwindigkeit arbeiten, die für die jeweilige Aufgabe angemessen ist, ohne dass sie dabei auf andere (langsamere oder schnellere) Rücksicht nehmen müsste. Allerdings erfordert dies, dass an den Grenzen zweier solcher Taktsysteme jeweils eine Synchronisation zum Zweck der geordneten Datenübergabe erforderlich ist. Ein Beispiel bei heutigen PCs wären die drei Bereiche Speicher, CPU und PCI-Bus, die alle mit unterschiedlichen Takten (und damit Geschwindigkeiten) betrieben werden.

1.3 Speicherarchitektur

Ein Programm selbst besteht aus den Befehlen in der Maschinensprache für eine bestimmte CPU. Dieses Programm muß zur Laufzeit im Speicher stehen, damit es ausgeführt werden kann. Das Programm selbst ändert sich dabei nicht. Zudem verarbeitet ein Programm Daten, die zur Laufzeit ebenfalls im Speicher stehen werden. Diese Daten werden sich in der Regel häufig ändern. Ausgenommen davon sind Daten, die als Konstanten im Programm vereinbart sind – diese müssen ebenfalls im Speicher stehen, werden sich aber während der Laufzeit nicht ändern.

Sieht man sich das C-Programm aus Listing 1 an, dann werden die blau markierten Anteile im Speicher als Daten behandelt, während der schwarze Rest in Maschinenbefehle umgesetzt wird und im Speicher als Programm behandelt wird.

```
int vektor_betrag(int *vektor, int laenge)
{
  int i, summe;

  summe=0;
  for (i=0; i<laenge; i++)
  {
     summe += vektor[i] * vektor[i];
  }

  return summe;
}
```

Listing 1: Vektorbetrag

Es gibt nun zwei weit verbreitete Speicherarchitekturen, *von-Neumann* und *Harvard*. In der von-Neumann-Architektur (Abbildung 2 links) werden sowohl Daten als auch Programm in demselben physikalischen Speicher abgelegt. In der Harvard-Architektur (Abbildung 2 rechts)

dagegen gibt es zwei physikalisch getrennte Speicher, von denen der eine nur die Daten enthalten kann und der andere nur das Programm. Beide Varianten haben ihre Vor- und Nachteile und werden dementsprechend je nach Aufgabenstellung eingesetzt.

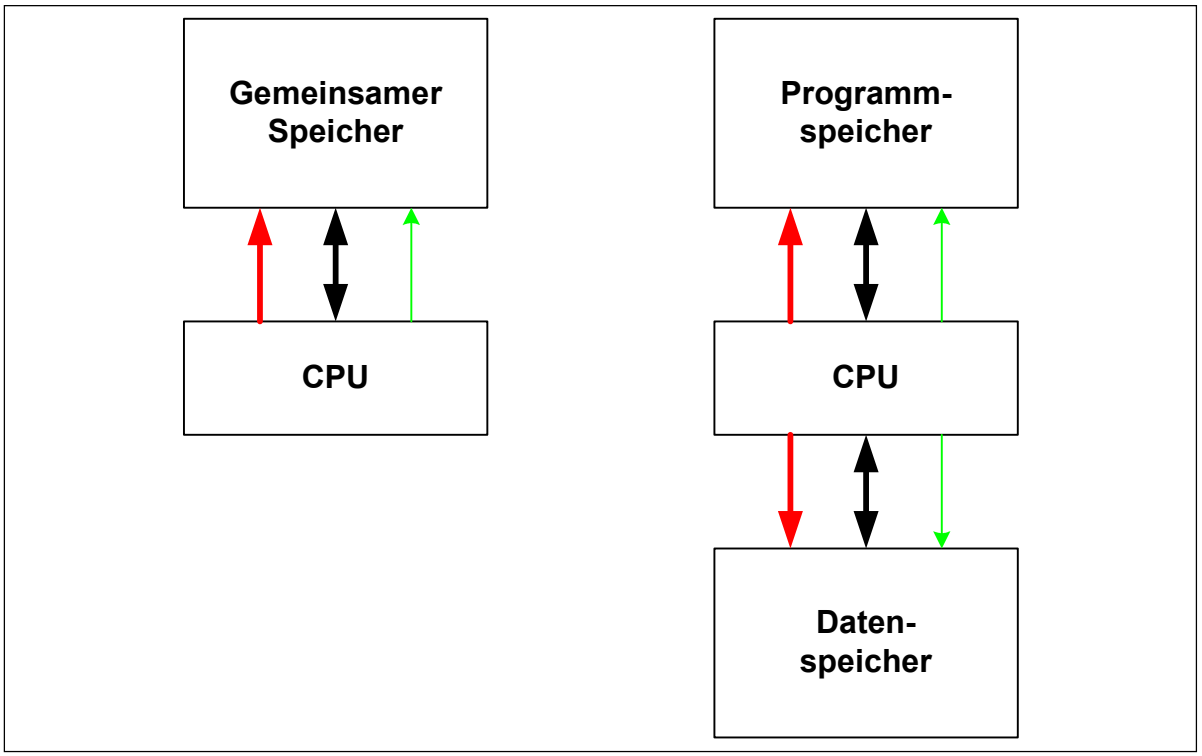

Abbildung 2: von Neumann vs. Harvard

In der von-Neumann-Architektur kann der zur Verfügung stehende Speicher vollständig mit einer beliebigen Mischung aus Daten- und Programmanteile gefüllt werden. Ein Bildbearbeitungsprogramm ist selbst sehr klein, verarbeitet aber möglicherweise eine sehr große Datenmenge. Der Speicher wäre zu 5% mit dem Programm belegt, 95% bleiben frei für die Daten. In einer Büroumgebung sind sehr viele Programme gleichzeitig aktiv, sie benötigen aber dafür nur wenig Platz für Daten. Hier könnten 70% des Speichers mit Programmcode belegt sein, die restlichen 30% reichen dann immer noch für die Daten.
In einer Harvard-Architektur ist dagegen eine „Übertrag“ nicht benutzten Programmspeichers in Datenspeicher (oder umgekehrt) nicht möglich. Wenn also die Nutzung des Rechners unvorhersehbar ist, dann ist die von-Neumann-Architektur die bessere Wahl, da sie in der Speicherverteilung flexibler ist.
In der Harvard-Architektur kann die CPU gleichzeitig einen Befehl aus dem Programmspeicher holen und einen Datenwert aus dem Datenspeicher holen. Dafür stehen ja zwei völlig getrennte Speicher mit allen nötigen Anschlüssen zur Verfügung. Die Harvard-Architektur erlaubt also eine deutlich schnellere Programmausführung als die von-Neumann-Architektur, da dort der Zugriff auf Daten und Programm nur abwechselnd möglich ist. Wenn also der Programmanteil einmalig festgelegt wird und sich dann nicht mehr ändert, dann ist die Harvard-Architektur im Vorteil.

2 Mikroprozessor

In fast allen Rechnersystemen übernimmt den Teil der CPU ein *Mikroprozessor*. Dieser ist damit das Herzstück des Rechners, denn er koordiniert das Geschehen im gesamten Rechner.

2.1 Begriffsbestimmung

Unter einem *Mikroprozessor* versteht man allgemein ein elektronisches Bauelement, das in der Lage ist, ein Programm auszuführen. Das Programm muss dazu in der für diesen Mikroprozessor definierten Maschinensprache vorliegen. Die übliche Abkürzung ist µP bzw. uP. Der Vorsatz „Mikro“ soll dabei anzeigen, dass es sich um ein einziges (und damit vergleichsweise kleines) Bauelement handelt, nicht um eine Zusammenschaltung vieler einzelner Bauelemente. Letzteres ist auch heute noch in Großrechnern üblich, während heutige PCs oder Workstations ausnahmslos nur noch Mikroprozessoren enthalten. In einem Rechner können aber durchaus mehrere Mikroprozessoren enthalten sein, diese arbeiten dann parallel und können verschiedene Aufgaben gleichzeitig übernehmen.
Mikroprozessoren werden gerne nach der größten Länge einer Dualzahl eingeteilt, die sie noch effizient bearbeiten können.. Ein µP, der mit einem Befehl Zahlen der Länge 8, 16 oder 32 Bit addieren kann, wird daher in der Regel als 32bit-Prozessor klassifiziert.

2.2 Geschichte

Der erste µP namens „4004“ wurde ab 1969 von der Firma Intel entwickelt und war 1971 fertig. Es war ein 4bit µP, der für einen Tischrechner (analog zu den heutigen Taschenrechnern) gedacht war. Der Durchbruch auf den Massenmarkt erfolgte jedoch erst mit dem 1974 ebenfalls von Intel entwickelten „8080“, einem 8bit-Prozessor.

Jahr	Prozessor	Bit	Firma	Bemerkung
1971	4004	4	Intel	Erster µP, Einsatz in Tischrechnern, 2300 Transistoren, 500 kHz Takt
1974	8080	8	Intel	6000 Transistoren, 2 MHz Takt, erstes PC-Betriebssystem CP/M verfügbar
1978	8086	16	Intel	Großvater der heutigen x86-Familie, 29000 Transistoren, 5 MHz Takt, erstes Auftreten von MS-DOS als Betriebssystem
1979	68000	16/ 32	Moto-rola	Einziger zeitweise erfolgreicher Gegenspieler im PC-Markt, 16 Bit nach außen, 32 Bit intern, ca. 69000 Transistoren, 8 MHz Takt. Einsatz in Mac, Atari, Amiga sowie Workstations
1985	R2000	32	MIPS	Erster allgemein verfügbarer RISC-Prozessor, 110000 Transistoren, 16 MHz Takt
1985	80386	32	Intel	Erweiterung des 8086 auf 32 Bit, dabei erhebliche Erweiterung des Befehlssatzes, heute der Standardbefehlsatz aller x86-Prozessoren, 275000 Transistoren, 16 MHz Takt
1986	ARM2	32	Acorn	Vater der heutigen ARM-Prozessoren in Hochleistungs-Embedded Systems (Mobiltelefone, Router), 8 MHz
1995	Pentium Pro	32	Intel	Erster x86 mit interner RISC-Verarbeitung, wegen hohen Preises aber geringer Erfolg, 21 Mio. Transistoren, 150 MHz Takt, „RISC-Schocker“
2003	Athlon 64	64	AMD	Abermalige Erweiterung des x86-Grundbefehlssatzes auf 64 Bit, der in allen aktuellen (2008) x86-Prozessoren verwendet wird, 106 Mio. Transistoren, 1.8 GHz Takt

Tabelle 1: Ausgewählte Mikroprozessoren

Tabelle 1 zeigt eine Reihe von Mikroprozessoren, die einen wesentlichen Einfluss hatten oder eine wesentliche Neuerung mitbrachten. Nach einer sehr großen Vielfalt an Prozessoren in den Jahren 1974-1985, in denen viele verschiedene Konzepte ausprobiert wurden, blieb im PC-Markt nur noch die x86-Familie von Bedeutung. Diese Familie hat unter anderem auch dadurch

überlebt, dass für gut befundene Konzepte aus anderen Prozessoren mit übernommen wurden, so dass der interne Aufbau eines heutigen (2016) x86-Prozessors nichts mehr mit dem ursprünglichen Stammvater 8086 zu tun hat. Demgegenüber blieb aber der Befehlssatz *abwärtskompatibel*. Darunter ist zu verstehen, dass Programme, die für frühere Mitglieder der Prozessorfamilie geschrieben wurden, auch noch auf den aktuellsten Vertretern (die noch weit mehr könnten) ablauffähig blieben. Dies bedeutete einen Investitionsschutz für Firmen, die noch alte Programme haben, aber nicht die Möglichkeit oder das Interesse, diese für die neuen Prozessoren neu zu kompilieren oder anzuschaffen.
Die Tabelle bricht 2003 ab, obwohl selbstverständlich die Entwicklung weitergegangen ist. Mikroprozessoren dieser Leistungsklasse spielen in „Embedded Systems“, das ist die weit überwiegende Anzahl an Geräten mit Mikroprozessoren, keine Rolle. Dort kommt es vielmehr auf eine jeweils passende Kombination aus einem Rechenkern mit mittlerer Leistungsfähigkeit und zur Aufgabenstellung passenden, im Chip bereits vorhandenen, Peripheriemodulen an.
Hier hat sich (Stand 2016) am Markt eine von der Firma ARM entwickelte Architektur für den Rechenkern als Quasistandard durchgesetzt.

2.3 Caches

Damit auch diejenigen Systeme, die mit einer von-Neumann-Architektur arbeiten, in den Geschwindigkeitsvorteil der Harvard-Architekturen kommen, haben alle neueren µP

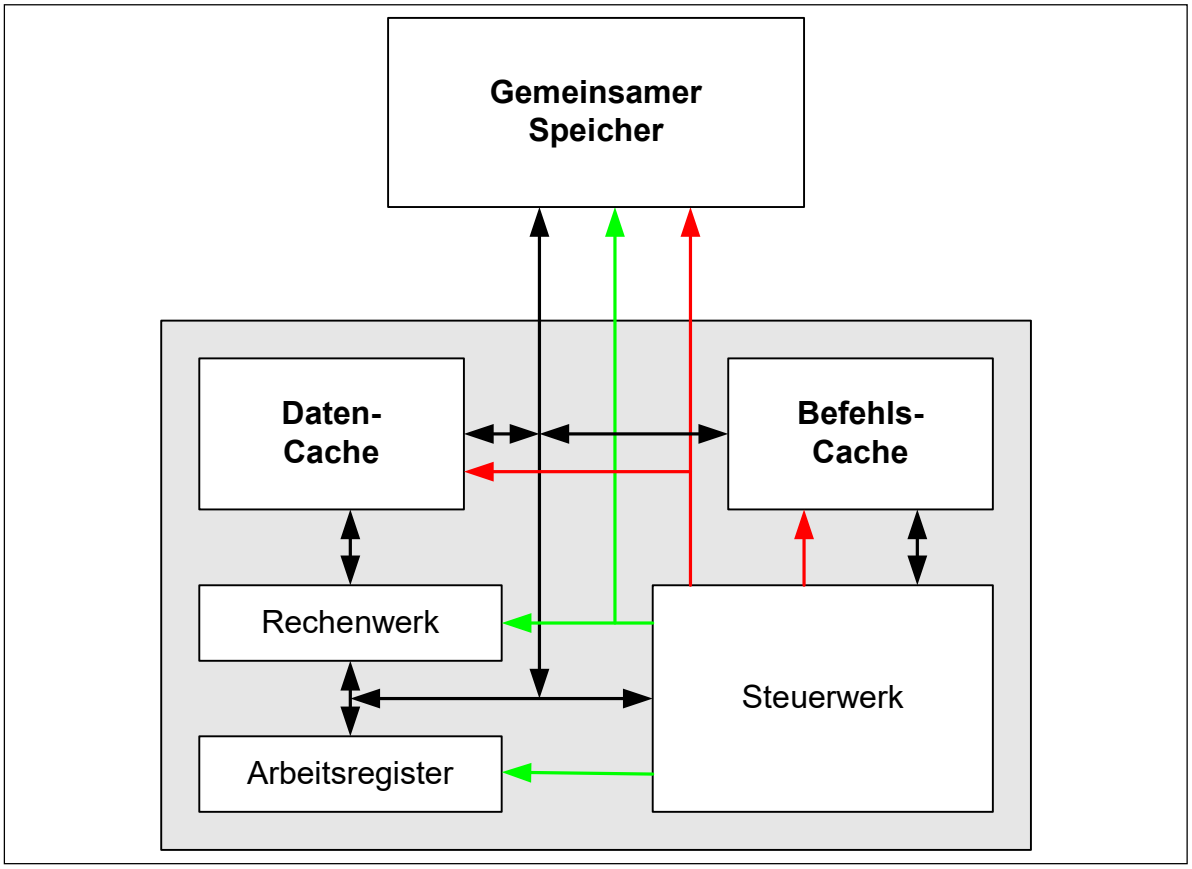

Abbildung 3: Prozessor mit Caches

sogenannte *Caches*. Dies sind kleine Speicher, die in den µP integriert sind (siehe Abbildung 3) und die den jeweils zuletzt benützten Teil des externen Speichers als Kopie enthalten. Im Unterschied zu den Arbeitsregistern sind die Caches Teil des Speichers, nur eben in den µP integriert. Im Beispielprogramm würde der Code für die for-schleife *einmalig* (beim ersten Durchlauf) aus dem externen Speicher in den Befehlscache kopiert. Für alle weiteren

Durchläufe braucht das Steuerwerk dann nicht mehr auf den externen Speicher zugreifen sondern kann sich die benötigten Befehle aus dem Befehlscache holen. Das geht erstens schneller und vermindert die Belastung der externen Busse. Ebenso werden im Datencache entsprechend die zuletzt benutzten Variablen gehalten. Zunächst sind das alle die Variablen, die in den Arbeitsregistern keinen Platz mehr gefunden haben. Im Beispiel wären das weiterhin die einzelnen Elemente des Vektors. Hier bringt das zwar nichts, aber bei einer Matrixmultiplikation C=A x B würden beispielsweise die Matrixelemente A[i,j] und B[i,j] immer wieder benötigt. Dann wären nach kurzer Zeit alle Elemente im Datencache und die folgenden Durchläufe könnten schneller ablaufen, da nicht mehr auf den externen Speicher zugegriffen werden muss. Mit Hilfe der Caches können also einerseits die langen Zugriffszeiten externer Speicher umgangen werden und zudem wird intern statt der von-Neumann-Architektur die schnellere Harvard-Architektur verwendet. Insgesamt bleibt aber die Flexibilität der von-Neumann-Architektur weiter erhalten.

Caches werden zunehmend auch bei Mikrocontrollern der höheren Leistungsklasse (typisch ab einer Arbeitstaktfrequenz von ca. 250 MHz) eingesetzt. Da die Zugriffszeiten des in solchen µC auf dem Chip vorhandenen Programmspeichers nicht in demselben Maß gesenkt werden kann wie die Arbeitsfrequenz steigt, würde ohne Cache kaum noch eine Steigerung der Rechenleistung (eines Kerns) eintreten.