1 Einführung

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.

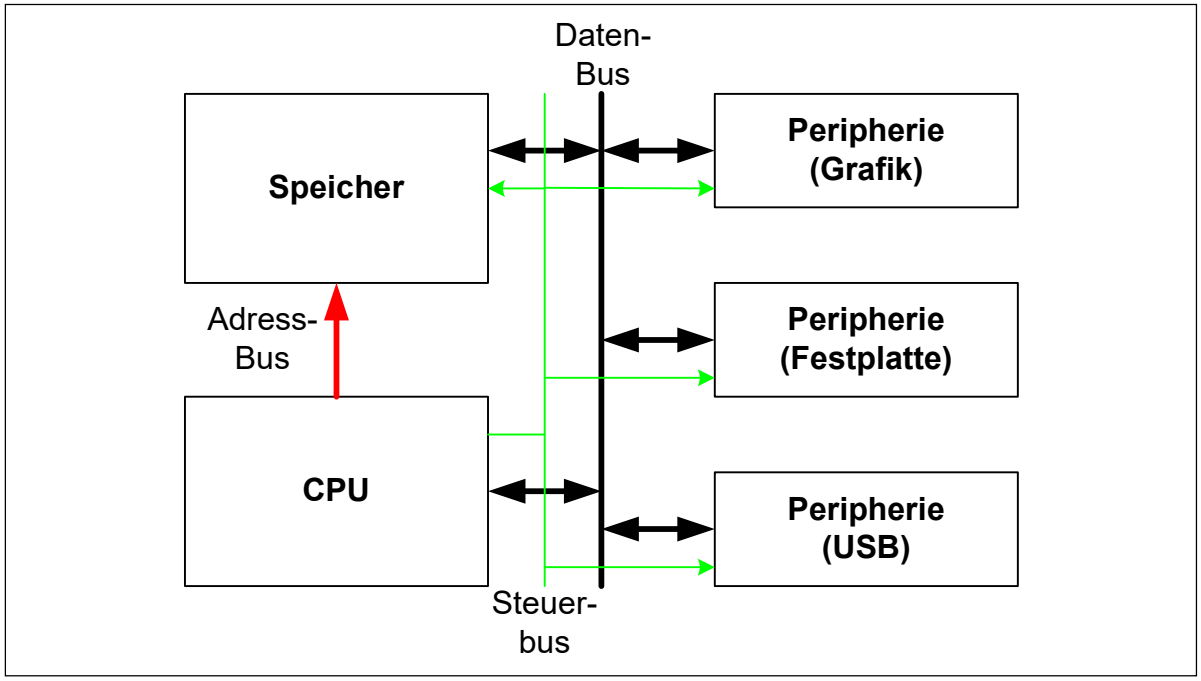

Abbildung 1: Prinzipielle Rechnerarchitektur

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. 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 Daten stehen auch im Speicher, 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.
Hier sind die Konstanten (wie die 0 oder das +1 bei i++) als Teil des Programmspeichers gekennzeichnet, weil sie später meist dort zu liegen kommen.

```
int vektor_betrag(int *vektor, int laenge)
{
  int i, summe=0;

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

  return summe;
}
```

Listing 1: Vektorbetrag

Es gibt 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.

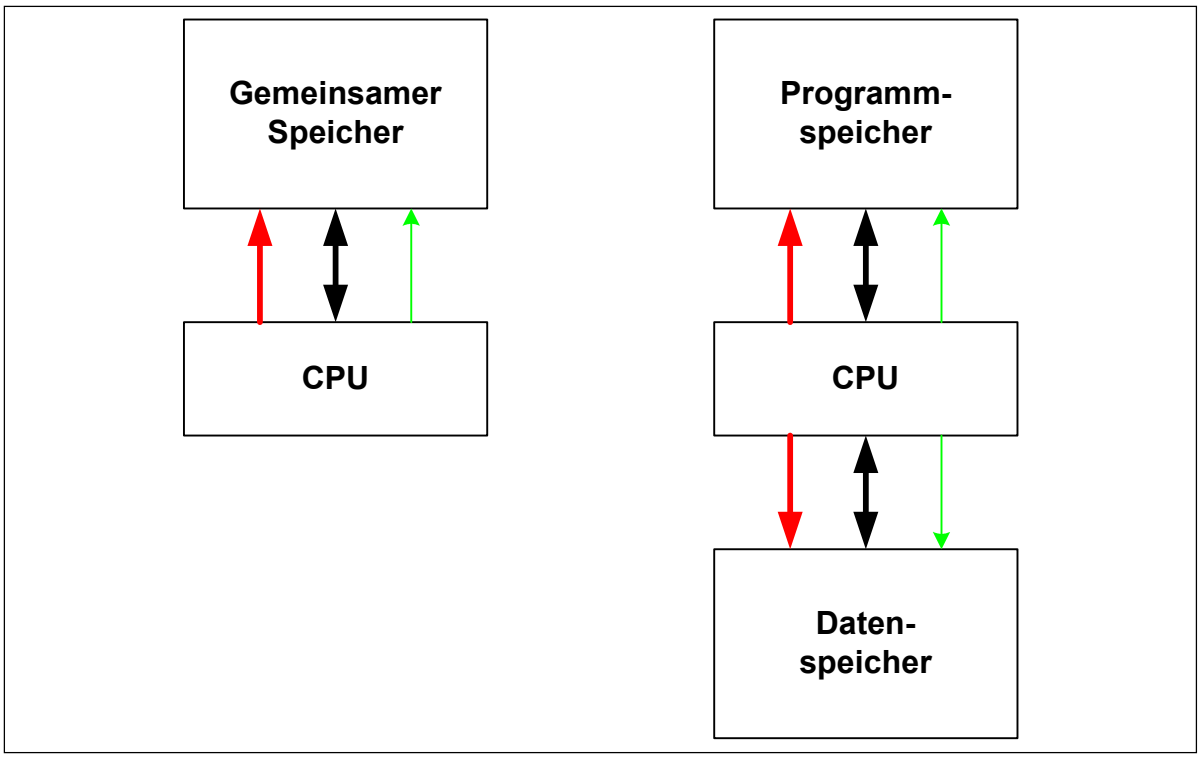

Abbildung 2: von Neumann vs. Harvard

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 (reinen) 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. Die Harvard-Architektur erlaubt also eine 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.

Heute (2024) verwenden nahezu alle Mikroprozessoren *nach außen* die von-Neumann-Architektur. Eine Ausnahme stellt die AVR8-Architektur dar, die in den üblichen 8 Bit Arduinos verwendet wird. Intern

1.4 Mikroprozessor (µP, uP, CPU)

Unter einem *Mikroprozessor* (uP, µP, CPU) versteht man allgemein ein elektronisches Bauelement, das in der Lage ist, ein Programm auszuführen. Das Programm muss dazu in der für diesen uP definierten Maschinensprache vorliegen. Der Vorsatz „Mikro“ (das µ in µP) soll dabei anzeigen, dass es sich um ein einziges und damit kleines Bauelement handelt.
Mikroprozessoren werden gerne nach der größten Länge einer Dualzahl eingeteilt, die sie noch in einem Maschinenbefehl 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.

Mikroprozessoren als eigenständige Bauelemente gibt es heute (2024) nicht mehr bzw. allenfalls als Ersatzteile für alte Rechner.
Die Funktion, die sie erfüllen, ist natürlich zentral und auch nicht verschwunden, aber sie ist jetzt Teil eines Bauteils, das noch andere Funktionen erfüllt. Damit hat sich auch der Name geändert: Ein Mikroprozessor als Teilfunktion heißt jetzt *core* oder auf Deutsch *Rechenkern*.

1.5 Mikrocontroller (µP, SoC)

Ein Mikrocontroller oder SoC (System-On-a-Chip) ist ein Bauteil, das zusätzlich zu mindestens einem Rechenkern noch weitere wesentliche Funktionen enthält, die für ein System wie in Abbildung 1 dargestellt typisch bzw. nötig sind. Die Abgrenzung zwischen µC und SoC ist unscharf. Im Allgemeinen beinhaltet ein SoC leistungsfähigere Rechenkerne (die Mehrzahl ist typisch), dafür aber nur wenig oder gar keinen integrierten Speicher.
Ein typischer Anwendungsfall für ein SoC ist ein Smartphone.
Ein µC ist dafür häufig mit so viel Zusatzfunktionen, inklusive Speicher, ausgestattet, dass er das einzige digitale Bauteil im gesamten System ist. Die damit ausgestatteten Produkte reichen vom Funkautoschlüssel über Maschinensteuerungen (z.B. weiße Ware, E-Bike-Antriebe) bis zu einfachen grafikfähigen Systemen.

Abbildung 3 zeigt wesentliche Daten eines der ersten µC überhaupt. Man kann sagen, weniger integrierte Funktionen sind nicht mehr sinnvoll. Entsprechend haben heute alle µC mindestens diese Funktionen integriert: µP (CPU), Speicher (PROGRAM MEMORY, DATA MEMORY), Ein-/Ausgabeleitungen (I/O LINES), Takterzeugung (CLOCK), Zähler (TIMER/EVENT COUNTER). Die Bezeichnungen in Klammern beziehen sich auf das Blockschaltbild links unten.

Device	Internal Memory		RAM Standby
8050AH	4K x 8 ROM	256 x 8 RAM	yes
8049H	2K x 8 ROM	128 x 8 RAM	yes
8048H	1K x 8 ROM	64 x 8 RAM	yes
8040AHL	none	256 x 8 RAM	yes
8039HL	none	128 x 8 RAM	yes
8035HL	none	64 x 8 RAM	yes
8749H	2K x 8 EPROM	128 x 8 RAM	yes
8748H	1K x 8 EPROM	64 x 8 RAM	yes

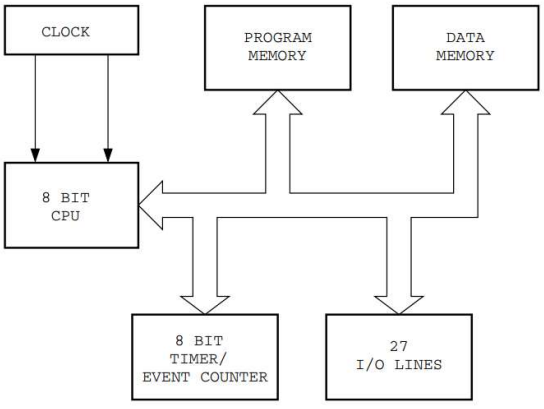

Figure 1. Block Diagram

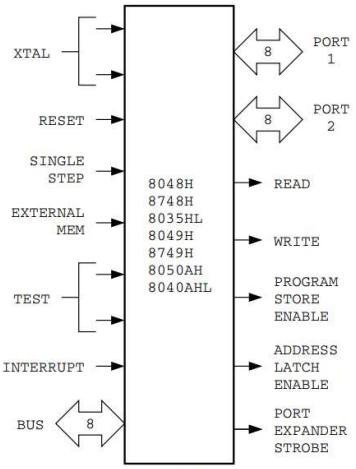

Figure 2. Logic Symbol

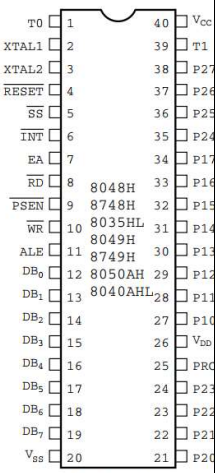

Figure 3. Pin Configuration

Abbildung 3: Eigenschaften eines sehr einfachen µC (8048)

Dieser µC verwendet die Harvard-Architektur. Der integrierte Speicher ist nach heutigen Begriffen winzig. Der Datenspeicher (RAM) des 8048 reicht gerade einmal für 16 Integer mit je 32 Bit (in 2024 eine typische Integerlänge, wenn man ein Programm an einem PC entwickelt). Dennoch war es z.B. mit dem Typ 8049 möglich, einen funktionsfähigen Schachcomputer ohne weitere digitale Bauteile zu entwickeln.
Der im Praktikum verwendete µC (STM32L432) ist nur ein Mittelklasse-µC (2024), enthält aber alleine schon einen 1000fach größeren Datenspeicher als der 8048.
Die Speicherarchitektur ist von-Neumann, aber intern kann auf unterschiedliche Speicherbereiche manchmal gleichzeitig zugegriffen werden. Dafür sind dann auch mehrere parallele Busse für Daten und Adressen vorhanden.
Mit diesem Zusatzaufwand erreicht man in den meisten Fällen die Geschwindigkeit einer echten Harvard-Architektur, behält aber die Flexibilität der von-Neumann-Architektur.
Das ist typisch für µC dieser Leistungsklasse.