4.5 PC-Hardware/Mikroprozessor (CPU)

4.5.1 Wie funktioniert eine CPU?

4.5.1.1 Elemente Die CPU (*central processing unit*) ist ein wichtiger Bestandteil eines Computers; sie enthält nämlich mindestens die Funktionsblöcke:

a) ein Rechenwerk (oder *arithmetic logic unit*, ALU)

b) ein Steuerwerk (oder Leitwerk)

c) ein internes Bussystem

d) ein oder mehrere Register (Speicher für Datenwörter)

4.5.1.2 Aufbau I Abbildung 1 zeigt ein Beispiel für den Aufbau einer CPU mit Harvard-Architektur. Im oberen Bereich findet man die Blöcke des Steuerwerks: Befehlsablaufsteuerung,

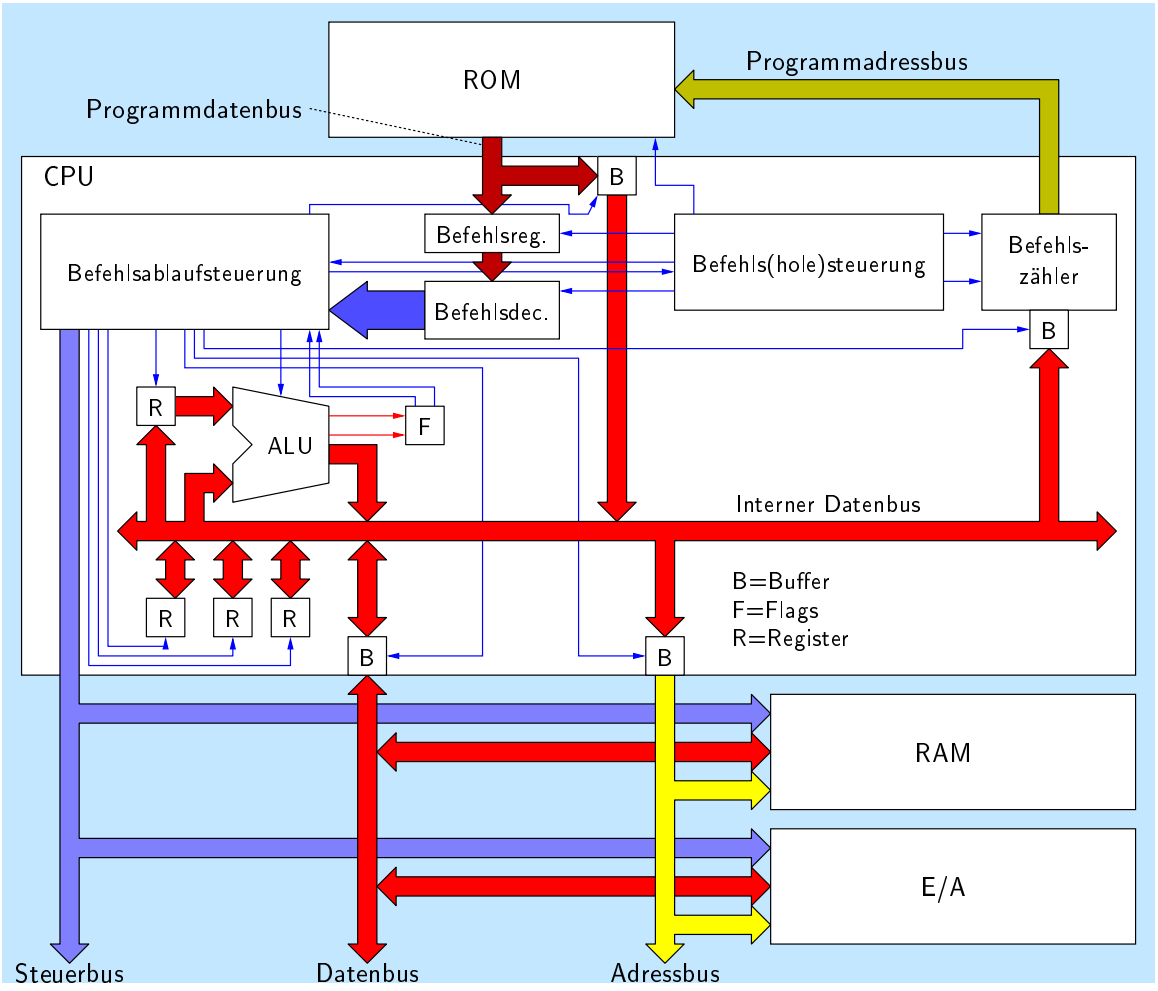

Abbildung 1: CPU mit Harvard-Architektur

Befehlssteuerung, Befehlszähler, Befehlsregister und Befehlsdecoder. Die ALU ist ein eigener kleiner Block. Die Register sind mit R gekennzeichnet. Das interne Bussystem ist durch rote (Datenbus) und blaue (Steuerleitungen) Pfeile gekennzeichnet.

4.5.1.3 Ablauf eines Befehls Für den Befehlsholezyklus (*fetch cycle*) ist die Befehlssteuerung zuständig. Nach dem Einschalten des Rechners wird der Befehlszähler auf den Wert 0 gesetzt (beim PC auf $FFFF0_{16}$). Damit liegt auf dem Programmadressbus die Befehlsadresse 0. Das ROM liefert daraufhin das erste Befehlswort über den Programmdatenbus an die CPU. Es wird im Befehlsregister abgespeichert. Im Befehlsdecoder wird es decodiert (daher der Name).

Nun liegt der Befehl vor und die Befehlssteuerung übergibt per Steuerleitung die Kontrolle an die Befehlsablaufsteuerung. Der Befehlsausführungszyklus (*execution cycle*) kann beginnen. Sie arbeitet wie ein einfaches Schrittschaltwerk. In jedem Schritt nimmt sie festgelegte Leitungen aus dem Befehlsdecoder und schaltet sie durch auf die Steuerleitungen, die von ihr (der Befehlsablaufsteuerung) ausgehen.

Das heißt, dass der Befehlsdecoder für jeden Befehl, den die CPU ausführen kann, eine bestimmte Abfolge von Nullen und Einsen für jede Steuerleitung ausgibt.[1]

Was nun passiert, hängt vom Befehl ab. Es gibt Befehle, die mit den Registern und der ALU zusammenhängen, z. B. „nimm den Wert von Register A, addiere 1 hinzu und speichere das Ergebnis wieder in Register A ab“. Andere hängen mit dem RAM zusammen, z. B. „nimm den Wert von Register A und speichere ihn an der Adresse, die in Register D steht“. Je nach Befehl werden diese Sachen dann ausgeführt. Wichtig dabei sind die blau eingezeichneten Steuerleitungen, die an jede Baugruppe gehen und Signale freischalten oder sperren.

4.5.1.4 Aufbau II Abbildung 2 zeigt ein Beispiel für den Aufbau einer CPU mit Von-Neumann-Architektur. Man sieht, dass der innere Aufbau sehr ähnlich zu dem der Harvard-Struktur ist. Lediglich der Ausgang zum Adressbus muss durch einen digitalen Umschalter, einen sogenannten Multiplexer, zwischen Befehlszähler und internem Datenbus umgeschaltet werden. Im *fetch cycle* wird der Inhalt des Befehlszählers nach außen geschaltet, im *execution cycle* gegebenenfalls der Inhalt des internen Datenbusses.

4.5.2 Schnelle CPU – schneller Rechner?

Viele Nutzer sind nach dem Kauf eines neuen Rechners („mit doppelt so schneller CPU“) enttäuscht: Man kann nicht so viel schneller arbeiten, wie es die Kennzahlen der CPU (f_{CPU}, *DBB* usw.) erwarten ließen. Woran liegt das?

4.5.2.1 Computer vs. Mensch Oft ist es so, dass ein Prozess (=ein laufendes Programm) im größten Teil seiner Laufzeit wartet. Manchmal wartet er auf langsame Hardware oder ein langsames Netz. Meistens wartet er auf seinen Benutzer. Das liegt aber häufig nicht (nur) am Nutzer, sondern am Programm selbst, welches nicht ergonomisch konstruiert ist. Gerade bei Programmen mit graphischer Benutzeroberfläche muss oft viele Male geklickt werden, bis eine bestimmte Aktion ausgeführt werden kann.

Abhilfe kann man durch Software schaffen, bei der der erfahrene Benutzer durch Voreinstellungen, durch Funktionstasten oder durch Befehlszeilen den Großteil der Mausklicks einsparen kann.

4.5.2.2 Hardware vs. Software Wenn ein Prozess für eine Aufgabe lange (oder zu lange) braucht, sagt man leicht, der Computer sei zu langsam. Es kann aber sein, dass die Software die Aufgabe nicht effizient löst. Besonders bei großen Datenmengen scheitern etliche Programme, weil sie nicht die richtigen Algorithmen (=Vorgehensweisen) benutzen.

Bei Standard-Software tritt das in der Regel seltener auf (es sei denn, es handelt sich um Makro-Programmierung). Bei Branchen-Software oder Maßanfertigungen kann es durchaus der Fall sein, dass der Programmierer davon ausging, dass man nur kleine Datenmengen verarbeiten

[1] Man kann den Ausgang des Befehlsdecoders vergleichen mit einem Liedblatt für ein Tasteninstrument, bei dem zu jeder Taktzeit festgelegt ist, welche Taste gedrückt ist und welche nicht. Und der Befehlsdecoder kennt eben nicht nur ein Lied (=Befehl), sondern mehrere davon. Die Summe der Befehle, die eine CPU kennt, nennt man den *Befehlssatz* der CPU. Übrigens ist der Befehlsdecoder in vielen CPUs als ein kleines ROM ausgeführt, ein sogenanntes *microcode rom*.

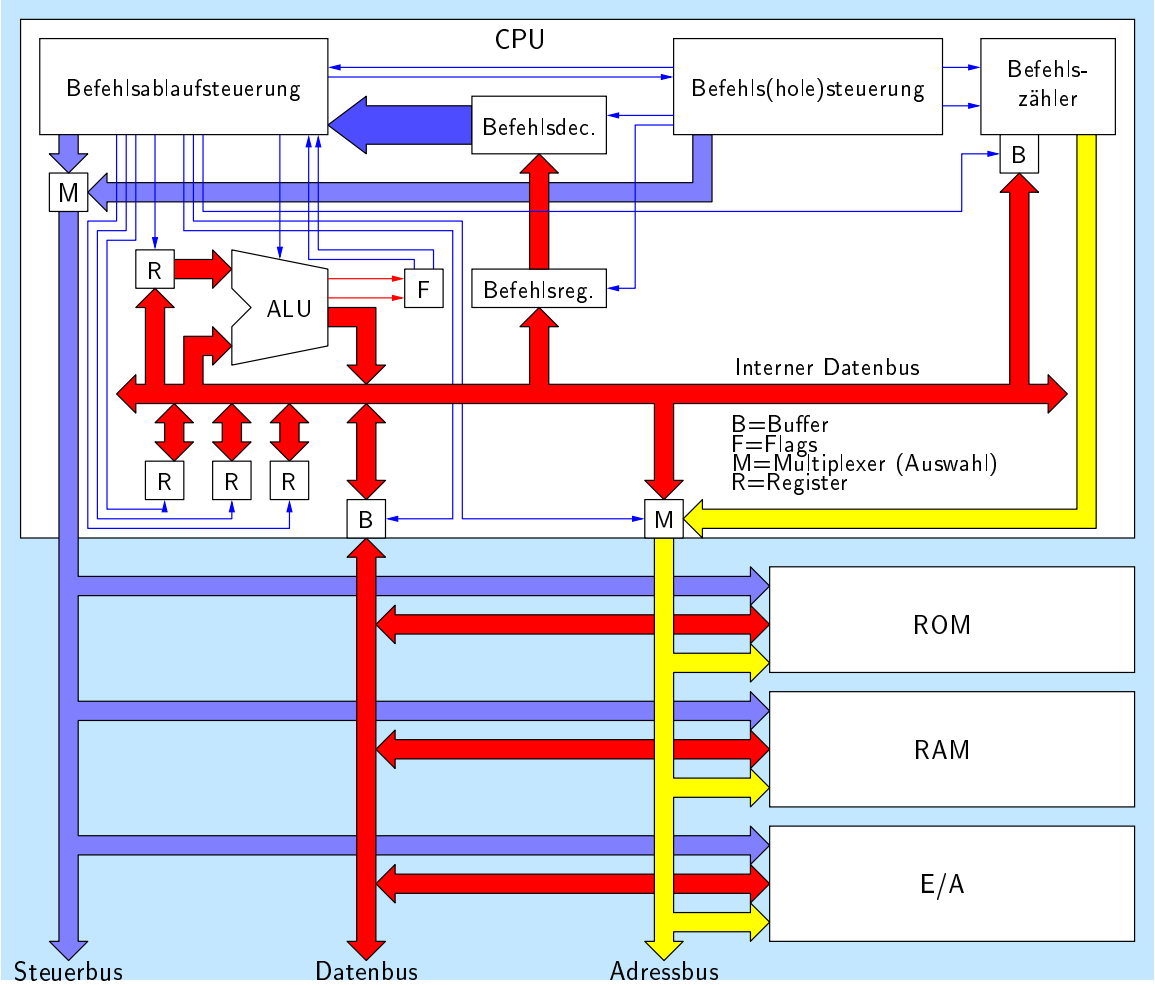

Abbildung 2: CPU mit Von-Neumann-Architektur

würde; dann hat er eventuell für Sortierung oder Datenzugriff einen einfachen, aber langsamen Algorithmus verwendet. Das lässt sich nachträglich ändern.

4.5.2.3 Zentraleinheit vs. Massenspeicher Massenspeicher wie Festplatte, CD, DVD usw. sind prinzipbedingt in ihrer Zugriffszeit viel langsamer als der Hauptspeicher. Das Verhältnis liegt bei 1:1000000. Eine Sortierung, die im Hauptspeicher eine Sekunde dauert, benötigt auf der Festplatte bis zu elf Tage. Sobald ein Programm Daten in kleinen Happen von der Festplatte lädt oder darauf schreibt, spielen CPU und Hauptspeicher keine Rolle mehr.

Mit dem Aufkommen von SSDs (*solid state discs*, Massenspeicher, deren Speichermedium Flash-EPROMs sind) ist dieser Aspekt eher in den Hintergrund getreten, wirkt sich aber trotzdem noch aus (weil die Datenraten zum Massenspeicher geringer sind als die Datenraten zum Hauptspeicher und weil auf den Massenspeicher über Dateien statt über Adressen zugegriffen wird, was einen weiteren Geschwindigkeitsverlust bedeutet). Dennoch ist die Benutzung von SSDs eine echte Abhilfe.

4.5.2.4 CPU vs. Bussystem und RAM Selbst wenn man einen Prozess vor sich hat, der nicht auf einen Nutzer wartet und auch nicht auf die Festplatte zugreift, bedeutet eine doppelt so schnelle CPU trotzdem noch nicht, dass der Prozess doppelt so schnell läuft:

- Jeder Befehl wird beim Von-Neumann-System einzeln über das Bussystem aus dem ROM oder dem RAM geholt
- Viele Befehle arbeiten so, dass Datenworte im RAM gespeichert oder von dort geladen werden

In beiden Vorgängen ist das Bussystem beteiligt: Die Busdatenrate begrenzt hier sowohl das Holen aller Befehle (*fetch cycle*) als auch das Ausführen vieler Befehle (*execution cycle*). Wenn das Bussystem zu langsam ist, nützt die „schnellste“ CPU nichts.

Abhilfe schafft ein Bussystem mit hoher Busdatenrate (f_b und DBB!) und ein möglichst schnelles RAM.

4.5.2.5 Befehlsrate (Verarbeitungsgeschwindigkeit) Wie misst man nun, ob eine CPU „schnell“ ist? Dafür gibt es die Befehlsrate (z. T. auch Rechenleistung genannt). Sie gibt die Verarbeitungsgeschwindigkeit einer CPU an.

- $Befehlsrate = \frac{Anzahl der verarbeiteten Befehle}{Zeit}$
- $BR = \frac{N}{t}$

Eine gebräuchliche Einheit ist MIPS, d. h. *million instructions per second* und bedeutet Millionen Befehle pro Sekunde.

Eine andere Einheit ist FLOPS, d. h. *floating point operations per second* und bedeutet Anzahl der Gleitkomma-Rechenoperationen pro Sekunde.

Offenbar handelt es sich um verschiedene Einheiten für verschiedene Größen: Mit FLOPS (MFLOPS, GFLOPS) wird die Befehlsrate für Gleitkomma-Rechenoperationen angegeben. Sie ist nur für technisch-wissenschaftliche Software relevant, z. B. das Programm SPICE für Berechnung elektronischer Schaltungen.

Mit MIPS wird die Befehlsrate für übliche Befehle angegeben. Das ist sinnvoll für Standard-Software wie Office-Pakete oder viele Spiele, weil Standard-Software aus Geschwindigkeitsgründen keine oder nur wenige Gleitkomma-Rechenoperationen enthält.

Gemessen wird die Befehlsrate von sogenannten Benchmark-Programmen. Leider hat die Beliebtheit von Benchmark-Programmen dazu geführt, dass Hersteller ihre CPUs so optimiert haben, dass die in den Benchmarks enthaltenen Befehle und Befehlsfolgen besonders schnell ausgeführt wurden.

4.5.3 CPU-Tuning

Wie kann man nun die Befehlsrate erhöhen? Dazu gibt es mehrere gebräuchliche Ansätze.

4.5.3.1 Befehlspipeline Bei der Befehlspipeline wird ausgenutzt, dass jeder Befehl nacheinander mehrere Stufen durchläuft. Zuerst wird jeder Befehl aus dem Befehlsspeicher geholt, dann im Befehlsregister zwischengespeichert, dann decodiert und schließlich ausgeführt. Da liegt es nahe, dass man – wie bei einem Fließband – einen Befehl ausführt, in dieser Zeit aber schon den nächsten decodiert und den übernächsten zwischenspeichert.

Diese Technik kann man noch etwas verfeinern und die Pipeline damit auf acht oder 16 Stufen verlängern, danach ist die Verbesserung nur noch minimal.

Nur leider kann man diese Technik nur auf Befehle anwenden, weil man die Reihenfolge der Befehlsdatenwörter schon kennt[2]. Bei Daten (also Variablen, die irgendwo im RAM stehen) ist sie nicht verwendbar.

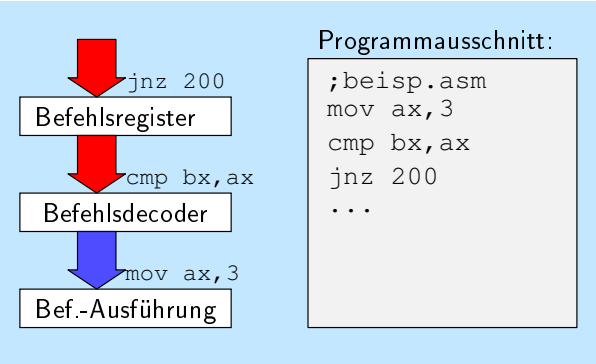

Abbildung 3: Befehlspipeline

4.5.3.2 Daten-Cache Für Daten eignet sich eine andere Technik, die man als Verallgemeinerung der Pipeline ansehen könnte: Variablen, die für einen bestimmten Prozess häufig gebraucht werden, liegen meist relativ nah zueinander im RAM. Vielleicht liegen sie nicht alle direkt nebeneinander, aber der Abstand ist nicht groß.

Und die Idee liegt darin, kleine Speicherbereiche (Blöcke) von z. B. 4 kiB, die bei einem Prozess andauernd gebraucht werden, als ganzes in der CPU zwischenzuspeichern. Dazu wird in oder bei der CPU ein kleines, schnelles, statisches RAM eingebaut, welches mehrere dieser Blöcke aufnehmen kann (Abbildung 4).

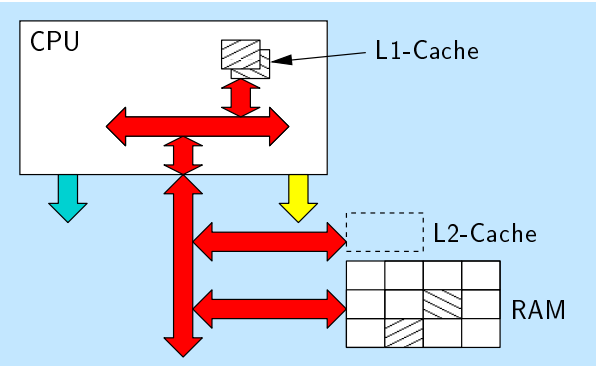

Abbildung 4: Cache

Dieses RAM nennt man Cache[3].

[2] außer bei bedingten Sprungbefehlen

[3] Ab und zu muss man nach Schreibzugriffen den Cache wieder mit dem „normalen“ RAM synchronisieren. Das wird selbsttätig von der CPU gemacht.

Aufgrund der Nähe zum Rest der CPU ist die Datenrate zum Cache sehr groß. Diese Tatsache, dass so die meisten Datenzugriffe nicht mehr auf das Bussystem gehen, sondern intern abgehandelt werden können, erhöht die Verarbeitungsgeschwindigkeit sehr. Nur, wenn große RAM-Bereiche beschrieben werden, treffen viele Zugriffe nicht mehr ins Cache. Dann sinkt die Geschwindigkeit wieder.

4.5.3.3 Microcontroller Wenn man nicht nur die oben genannten Baugruppen, sondern auch RAM und ROM und E-/A-Baugruppen in die CPU aufnimmt, erhält man eine MCU (*micro controller unit*), auch Microcontroller genannt. Hier liegt der ganze Hauptspeicher auf dem Chip und ist damit wie ein Cache mit hoher Datenrate angebunden (Abbildung 5).

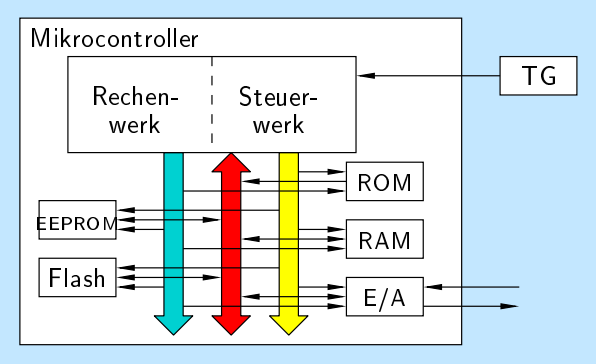

Abbildung 5: Microcontroller-System

Diese Struktur erlaubt hohe Verarbeitungsgeschwindigkeiten. In der Praxis wird sie oft *embedded systems* eingesetzt, also in Computern, die in Geräten, Fahrzeugen und Gebäuden selbständig Steuerungsaufgaben erledigen (z. B. Motor-Steuergeräte in Autos). Wenn viele E-/A-Funktionen integriert sind, spricht man heute auch von SoC (*system on a chip*).

4.5.3.4 RISC Im Laufe der Jahre ist der Befehlssatz bei vielen CPU-Familien immer weiter gewachsen. Die meisten Programmierer und die meisten Übersetzer (Compiler genannt) benutzen aber nur wenige (quasi immer die gleichen) CPU-Befehle. Aus diesem Grund kam die Idee auf, neue CPUs zu entwickeln, die nur wenige wichtige Befehle ausführen können. Diese Befehle allerdings werden so optimiert, dass sie möglichst schnell ablaufen.

CPUs, die nach diesem Grundsatz entwickelt wurden, heißen RISC (*reduced instruction set computers*). Es gelang damit, CPUs zu entwickeln, die pro CPU-Takt einen ganzen Befehl ausführen können ($BR = f_{CPU}$).

Alle anderen CPUs nennt man dann CISC (*complex instruction set computers*). Die üblichen CPUs für PCs sind CISC. Allerdings bemühen sich die CISC-Hersteller, darauf hinzuweisen, dass auch sie RISC-Merkmale in ihre CISC-CPUs eingebaut hätten.

Typische Vertreter der RISC-Architektur sind ARM-Prozessoren. Sie haben im Mobilbereich einen hohen Marktanteil.