EIN BYTE BESTEHT AUCH HEUTE NOCH AUS 8 BITS

30 Jahre alter Atari 130 XE Computer veranschaulicht Informatik und Digitaltechnik

>> von Peer Johannsen > Die rasante Entwicklung der Digitaltechnik und der Computer in den letzten vierzig Jahren hat dazu geführt, dass Jugendliche heute ganz selbstverständlich mit diesen Technologien aufwachsen und ganz selbstverständlich mit den neuesten digitalen Medien umgehen. Die Generation heutiger Studienanfänger ist mit der Anwendung moderner Kommunikations- und Unterhaltungstechnik bestens vertraut. Fast jede Studentin und jeder Student besitzt ein eigenes Smartphone und einen eigenen Laptop, und sie alle sind Experten im Installieren von Apps und in der Nutzung von sozialen Netzwerken und virtuellen Clouds zum Datenaustausch. Gleichzeitig fehlt vielen von ihnen jedoch oftmals jedes Verständnis dafür, wie ein technisches Gerät, ein Computer oder ein Programm „im Inneren" funktionieren, und eine Welt und eine Zeit, in der es noch keine Mobiltelefone gab und in der wenige Computer nur wenigen Experten zugänglich waren, ist für sie kaum vorstellbar.

Ein bewusster Schritt zurück in genau diese Zeit und eine interaktive Vorführung historischer Computertechnik im Hörsaal kann hier einen motivierenden Beitrag leisten, um bei den Studierenden ein Interesse für die Grundprinzipien der Funktionsweise digitaler Hardware und für die Grundprinzipien der Berechnungsabläufe in Programmen zu wecken.

Moderne Computer haben heutzutage einen Prozessor mit 64-Bit Daten- und Adressbreite, sind mit einem Hauptspeicher von mindestens 4 Gigabyte RAM ausgestattet, haben Festplatten mit einer Kapazität im Terabyte-Bereich und werden mit einem 64-Bit Betriebssystem ausgeliefert. Die ersten Definitionen, die Studierende jedoch im Rahmen ihres Studiums der Informatik oder Informationstechnik kennenlernen, sind auch heute noch die Begriffe „Bit" und „Byte".

Ein Bit kann genau zwei verschiedene Zustände annehmen und speichern. Die geordnete Zusammenfassung von 8 Bits wird als ein Byte bezeichnet, und ein Byte ist die wichtigste Einheit zur Codierung und Speicherung digitaler Information. In Zeiten, in denen Ressourcenknappheit hinsichtlich der digitalen Speicherung von Information nahezu unbekannt ist, zählen diese Begriffe weiterhin zu den wesentlichen Grundlagen, die für das Verständnis der Codierung von Information in Digitalrechnern notwendig sind. Der Speicher eines Computers ist in Einheiten von Bytes organisiert, seine Größe und die Kapazität von Festplatten werden in Vielfachen von Bytes angegeben.

Doch warum sollte für die Speicherung des Ergebnisses eines Würfelwurfs ein Byte und keine Fließkommavariable verwendet werden, wenn doch „genug" Hauptspeicher vorhanden ist? Warum ist nicht die Taktgeschwindigkeit eines Prozessors für die Laufzeit eines Programmes entscheidend, sondern die Komplexität des zugrunde liegenden Algorithmus? Selbst bei den Studierenden, die Vorwissen oder Programmierkenntnisse mitbringen, fehlt oft das Bewusstsein für solche Fragen. Warum sollten Programme und Algorithmen speichereffizient und laufzeiteffizient entwickelt werden, wenn doch Speicher aufgerüstet und eine CPU einfach durch eine doppelt so schnell getaktete ersetzt werden kann?

Das Verständnis für genau solche Fragen und die dafür notwendigen Kenntnisse gilt es im Studium der Informatik zu vermitteln. Doch wie lässt sich dieses erreichen, wenn moderne Betriebssysteme den direkten Zugriff auf einzelne Bytes des Hauptspeichers verhindern, wenn Hochsprachen von den eigentlichen Vorgängen der Berechnungen im Prozessor abstrahieren, wenn Compiler den erzeugten Maschinencode bis zur Unkenntlichkeit optimieren, und wenn Prozessoren so hoch getaktet sind, dass die feinen Auswirkungen algorithmischer Details auf die Laufzeit eines Programmes bei überschaubaren Beispielen nicht mehr messbar sind?

Ich selber habe den technologischen Fortschritt der Computertechnik in den letzten Jahrzehnten hautnah miterlebt. Im Alter von 11 Jahren kaufte ich mir 1982 gegen den Willen meines Vaters, der reines Spielinteresse befürchtete, einen der ersten verfügbaren 8-Bit Heimcomputer, einen Atari 600 XL, und brachte mir anhand des eingebauten Basic-Interpreters das Programmieren bei. Es folgten eine Datasette, ein 800 XL, eine 1050 Diskettenstation und kurz darauf ein 130 XE sowie die frühzeitige Erkenntnis, später im Leben einmal „etwas mit Computern" studieren zu wollen. Die mangelnde Geschwindigkeit der Basic-Programme führte zu den ersten Programmierschritten in Maschinensprache. Später dann, mit Aufkommen der ersten 16-Bit Systeme, schaffte ich mir zusätzlich einen Atari 1040 ST an, den ich ebenfalls begeistert in Basic und Assembler programmierte, bis ich eines Tages von einer Programmiersprache namens C erfuhr und feststellte, wie viel einfacher sich hiermit doch vieles hätte realisieren lassen.

Die Bedeutung dieser Entwicklung, von den ersten 8-Bit Heimcomputern und Basic hin zu heutigen modernen Mikrocontrollern und PCs mit 32-Bit oder 64-Bit Betriebssystemen und objektorientierten Programmiersprachen wie C++, lässt sich der heutigen Generation von Studierenden im Unterricht sehr schön anschaulich erklären, wenn das Wissen über die verschiedenen Rechnergenerationen nicht nur theoretisch vermittelt wird, sondern wenn ihnen solche Rechnersysteme auch einmal live vorgeführt werden.

Die Generation der ersten 8-Bit Heimcomputer aus den frühen 80er Jahren eignet sich hervorragend zur Veranschaulichung vieler Grundlagen heutiger Computertechnologien und heutiger Programmiersprachen. In einem spontanen Experiment habe ich vor einem Jahr einmal meinen 30 Jahre alten Atari 130 XE Heimcomputer mit in eine Vorlesung ge-

Professor Peer Johannsen setzt seinen 1987 geschriebenen Zeilenassembler in der Vorlesung ein.

nommen und den Studierenden im Betrieb gezeigt. Ich habe festgestellt, dass sich gerade dieser alte 8-Bit Rechner sehr schön dazu verwenden lässt, um Prinzipien moderner digitaler Rechner und elementare Inhalte der Informatik und der Software-Entwicklung zu demonstrieren.

Alleine schon die Tatsache, dass bei diesem Computer die Tastatur noch fest im Gehäuse integriert ist, dass er keinen Lüfter braucht, dass nach dem Anschalten das Betriebssystem aus dem ROM geladen wird und gleich vorhanden ist, weckt Erstaunen und Interesse bei den Studierenden. Ebenso das Vorführen des Ursprungs der Bedeutung des Wortes „Floppy" oder die Erkenntnis, dass Diskettenlaufwerke einmal fast so groß waren wie das Computergehäuse selber, und dass der Speicherplatz im RAM oder auf Disketten einmal so beschränkt war, so dass es sich auszahlte, sich beim Programmieren darüber Gedanken zu machen, wie viele Bytes eine Variable im Speicher verbraucht, oder ob man ein Programm speichereffizient geschrieben hat.

Grundprinzipien der Software-Entwicklung und die Wirkungsweise der elementaren Bausteine von Algorithmen (Folgen von elementaren Anweisungen, Fallunterscheidungen, Schleifen) lassen sich sehr leicht anhand des eingebauten Basic-Interpreters des Atari erklären und vorführen. Einfache Programme lassen sich ohne große Vorkenntnisse und ohne Wissen über die Funktionsweise eines Compilers schnell entwickeln und direkt ausprobieren. Eingabe und Ausgabe von Daten und das Zeichnen von Graphiken kann schnell und einfach realisiert werden, ohne sich vorher mit einzubindenden Bibliotheken oder der Ansteuerung einer modernen Graphikkarte auseinandersetzen zu müssen. Das unterliegende Konzept eines Bildschirmspeichers, in dem Bitmuster den auf dem Bildschirm dargestellten Pixeln entsprechen, lässt sich am Atari unmittelbar mit ein paar PEEK und POKE Befehlen zeigen (das sind Befehle zum direkten Zugriff auf einzelne Zellen des Speichers, die es heute so in modernen Programmiersprachen nicht mehr gibt).

Es ist ungleich schwerer, Anfängern diese Inhalte und dieses Grundverständnis der Arbeitsweise von Programmen und den Zusammenhang zwischen Bits im Speicher und Komponenten der Hardware auf einem PC unter Windows mit Hilfe eines C-Compilers zu vermitteln. Dabei sind gerade bei Anfängern, die ihre ersten Schritte in Richtung Software-Entwicklung unternehmen, das schnelle Erfolgserlebnis und die damit einhergehende Motivation äußerst wichtig.

Natürlich ist das Ziel der Programmierausbildung im Studium das Heranführen der Studierenden an moderne Entwicklungsumgebungen und moderne Programmiersprachen, wie zum Beispiel C, C++ oder Java. Der Weg hierhin kann jedoch deutlich leichter fallen, wenn erst einmal ein Grundverständnis für Befehlsabläufe in Programmen, für Variablen und für Bits und Bytes vorhanden ist. Der Einstieg über ein paar einfache Basic-Programme kann hier auch Motivator für diesen Schritt sein, zum Beispiel, indem auch einige der Nachteile einer Interpreter-Sprache wie Basic demonstriert werden, wie die Inflexibilität durch die notwendige Nummerierung von Zeilen oder der Geschwindigkeitsnachteil. Auch viele andere Konzepte einer modernen hardwarenahen Sprache wie C (Zeiger, Gültigkeitsbereiche von Variablen, Parameterübergabe an Funktionen) lassen sich besser verstehen, wenn der Zusammenhang zwischen der Hochsprache und der Assemblersprache eines Prozessors (Register, Bitoperationen, Arbeitsweise eines Stacks) einmal direkt veranschaulicht wird.

Ein schneller Einstieg in die Assemblerprogrammierung ist auf dem 8-Bit Atari mit seinem 6502 Prozessor deutlich leichter als auf einem PC mit Intel i7 Prozessor oder einem 64-Bit ARM Mikrocontroller. Ich habe hierfür in den Vorlesungen einen von mir im Jahre 1987 geschriebenen Zeilenassembler eingesetzt (ASS-Zeilenassembler, Happy-Computer, 2. Atari XL/XE Sonderheft), der die zeilenweise Assemblierung einzelner Prozessorbefehle und ein direktes Ablaufen und Ausprobieren von Assemblerroutinen erlaubt.

Zunächst habe ich mit den Studierenden gemeinsam ein Basic-Programm geschrieben, das mit Hilfe von PEEK die Tastatur abfragt und dann durch POKE die Farbe des Bildschirmhintergrundes je nach Tastendruck ändert. Anschließend haben wir dieses Programm in Assemblerbefehle umgesetzt. Ziel war es, die Vorgänge, die in einem Hochsprachen-Compiler ablaufen, sowie den Zusammenhang zwischen Bytes im Speicher und angeschlossener Hardware „hautnah" zu verdeutlichen. Die Prinzipien, die hierbei angewendet werden, sind >

Aus der Frühzeit der Computertechnologie: ein 30 Jahre alter Atari 130 XE Heimcomputer.

dieselben, die auch heute noch für moderne eingebettete Mikrocontroller gelten.

Im weiteren Verlauf der Vorlesungen wurden die Studierenden dann schrittweise an die heutzutage aktuellen und in der Industrie eingesetzten Programmiersprachen und Prozessortechnologien herangeführt. Dieses ist natürlich der Sinn und das Ziel der Ingenieursausbildung im Studium. Das Verständnis für die Funktionsweise eines modernen 64-Bit ARM Mikrocontrollers und für die Besonderheiten seiner Programmierung fällt aber eben deutlich leichter, wenn man vorher einmal die Grundzüge früherer Prozessorgenerationen anschaulich erlebt und verstanden hat.

Meine Grundüberzeugung in der Lehre ist, dass sich Programmieren nicht an der Tafel erlernen lässt. Im Hörsaal sollten Rechner direkt eingesetzt werden und es sollte gemeinsam programmiert werden, um insbesondere Studierenden, die keine Vorkenntnisse mitbringen oder keine Vorstellungen vom Innenleben eines Computers haben, einen anschaulichen und interessanten Einstieg in die Welt der Informatik zu geben. Gerade hier hat sich mein alter 8-Bit Atari 130 XE bewährt, weil eben kein großes Betriebssystem im Wege steht und ein Zugriff auf die wesentlichen Hardwarekomponenten noch direkt möglich ist, und weil die reale Vorführung eines echten Gerätes eben doch viel beeindruckender und anschaulicher ist als die Verwendung eines Simulators oder Emulators für solche Lehrinhalte. Ein herzlicher Dank geht an dieser Stelle an den Atari Bit Byter User Club e.V. (ABBUC) für seine Unterstützung bei Reparaturen dieses antiken Rechners und bei der Instandhaltung der zugehörigen Laborausstattung. Ich bin gespannt darauf, welche weiteren Einsatzmöglichkeiten im Rahmen der Hochschulausbildung mir für meinen Atari noch einfallen werden

Dr. Peer Johannsen
ist Professor für Informatik und Software-Engineering in der Fakultät für Technik.