Vorlesungsmodul Mikroprozessortechnik 1
- VorlMod MpTk1 -

Matthias Ansorg

29. September 2003 bis 23. März 2004

Zusammenfassung

Studentische Mitschrift zur Vorlesung Mikroprozessortechnik 1 bei Prof. Dr. Klaus Wüst (Wintersemester 2003/2004) im Studiengang Informatik an der Fachhochschule Gießen-Friedberg. Diese Mitschrift ist identisch zum offiziellen Skript der Vorlesung [2] gegliedert, zumindest bis inkl. der zweiten Ebene.

Die Veranstaltung MpTk1 ist eine reine Vorlesung ohne Programmier- und Rechenübungen. MpTk2 ist dagegen ein reines Praktikum, in dem einige der in der Vorlesung besprochenen Geräte programmiert werden. DIeses Praktikum wird jedes Semester angeboten und kann auch parallel zu MpTk1 absolviert werden, weil die Versuchsanleitungen sehr ausführlich sind.

Für eine Klausurvorbereitung sind notwendig: alternativ das Buch [1]oder eine Kombination aus dem Skript [2], dieser Mitschrift und darin angegebenen Ergänzungen zum Skript. Außerdem die Klausuren [3] und die Lösungen dazu [5].

- **Bezugsquelle:** Die vorliegende studentische Mitschrift steht im Internet zum Download bereit. Quelle: Persönliche Homepage Matthias Ansorg :: InformatikDiplom http://matthias.ansorgs.de/InformatikDiplom/.
- **Lizenz:** Diese studentische Mitschrift ist public domain, darf also ohne Einschränkungen oder Quellenangabe für jeden beliebigen Zweck benutzt werden, kommerziell und nichtkommerziell; jedoch enthält sie keinerlei Garantien für Richtigkeit oder Eignung oder sonst irgendetwas, weder explizit noch implizit. Das Risiko der Nutzung dieser studentischen Mitschrift liegt allein beim Nutzer selbst. Einschränkend sind außerdem die Urheberrechte der angegebenen Quellen zu beachten.
- **Korrekturen und Feedback:** Fehler zur Verbesserung in zukünftigen Versionen, sonstige Verbesserungsvorschläge und Wünsche bitte dem Autor per e-mail mitteilen: Matthias Ansorg <mailto:matthias@ansorgs.de>.
- **Format:** Die vorliegende studentische Mitschrift wurde mit dem Programm LyX(graphisches Frontend zu LaTeX) unter Linux geschrieben und mit pdfLaTeXals pdf-Datei erstellt. Grafiken wurden mit dem Programm xfig unter Linux erstellt und als pdf-Dateien exportiert.
- **Dozent:** Prof. Dr. Klaus Wüst.
- **Verwendete Quellen:**
- **Klausur:** Die Klausur wird stets nach den Semesterferien geschrieben. Außer dem Taschenrechner sind keine Hilfsmittel erlaubt. Vorteil: es gibt einfache Wissensfragen und einfache Verständnisfragen. Es gibt keine Hausübungen und damit keine formalen Teilnahmevoraussetzungen zur Klausur.

Inhaltsverzeichnis

1 Einführung

MpTk1 setzt seit WS 2003/2004 einiges aus ReArch2 voraus, das nicht mehr wiederholt wird. Stoffüberblick:

- Systembus und Adressverwaltung
- Einfache Mikroprozessoren
- Besondere Betriebsarten
  - Interruptbetrieb
  - DMA
- Speicherverwaltung
  - Segmentierung
  - Paging
  - Caching
- RISC
- Superskalare Prozessoren, SIMD
- Mikrocontroller
- Digitale Signalprozessoren

1.1 Geschichtliches

1.1.1 Mechanische Rechner

- Mechanischer Rechner von Blaise Pascal, 1642. Der älteste Computer der Welt.
- Leibniz 1672
- Babbage 1833

Mechanische Rechner waren zu unpräzise und störanfällig, um sich breit durchzusetzen.

1.1.2 Relais-Rechner

- Z3 von Konrad Zuse. Konrad Zuse hat einmal in der Ingenieurshochschule in Gießen gearbeitet, aus der dann später die FH Gießen-Friedberg entstand. Die Z3 hat gut gearbeitet und ist heute noch im Museum zu sehen.
- Mark I von Aiken 1944.

Relais-Rechner sind langsam und brauchen enorm viel Strom.

1.1.3 Elektronenröhren-Rechner

Sie waren schon wesentlich schneller als die Relais-Rechner. Über ein negativ geladenes Steuergitter kann bei Bedarf der Stromfluss im Vakuum einer Elektronenröhre unterbrochen werden. Um Rechner zu realisieren, braucht man immer Bauelemente, die gleichartige Bauelemente steuern können. Das können Relais, Elektronenröhren oder (wie heute) Transistoren sein.

Bekanntestes Beispiel für Elektronenröhren-Rechner ist Kolossus, mit dem der ENIGMA-Kode der deutschen U-Boot-Flotte geknackt wurde. Bekanntester Vertreter der Arbeitsgruppe um Kolossus war Alan Turing.

Größter Vertreter der Elektronenröhren-Rechner war ENIAC, ein Forschungsrechner in den USA. Von Neumann erfand dafür die Ablage der Programme im Datenspeicher statt auf Lochstreifen. IBM 701 war der erste Rechner von International Business Machines.

1.1.4 Transistorrechner

Bardeen, Brattain und Shockley erfanden 1948 in den Bell Labs den Transistor. Man erkannte schnell, dass auch damit Rechner gebaut werden können. Nachteil dieser Rechner waren, dass diskrete Transistoren nicht miniaturisiert werden können.

- TX-0: der erste Transistorrechner, ein Experiment des MIT
- DEC PDP-8, DEC PDP-11

1.1.5 Mikroprozessoren mit integrierten Schaltkreisen

Erfunden 1958 durch Robert Noyce. Er gündete dann mit Gordon Moore[1] die »Integrared Electronics Corp.« (Intel). Der erste Mikroprozessor war der i4004 von 1971. Er hatte 2250 Transistoren, 16 Anschlüsse, 4bit Register, 4 Datenleitungen, 12 Adressleitungen. 1972 kam der Nachfolger i8008 und der i8080, 1978 der i8086.

1978 baute IBM den ersten PC mit dem Intel 8086 - alle heutigen PCs sind abwärtskompatibel zu diesen. IBM erfand auch den PC: einen Computer für einen Menschen allein. Allerdings nahm IBM den PC selbst nicht ernst und beschräkte sich auf Midframes und Mainframes. Das lag an einer völligen Fehleinschätzung des Marktes: IBM konnte sich nicht vorstellen, dass jemand einen PC zu Hause braucht. Kurz darauf gab es PC-Clones zum halben Preis und IBM machte keinen Gewinn damit. Diese Geschichte wird oft als Lehrbeispiel für marktwirtschaftliche Fehleinschätzungen benutzt.

[1] Er ist der Verfasser des »Mooreschen Gesetzes«!

1.2 Dynamik des Bereiches Mikroprozessortechnik

Die Komplexität der Prozessoren kann sehr gut an der Anzahl der Transistoren gemessen werden, die in einer CPU integriert sind. Entwicklung: 2300 Transistoren beim i4004, 40 Millionen Transistoren heute.

Integrationsgrad: von 100 zu 100000 Gatter pro Chip.

Arbeitstakt: von weniger als 1 MHz bis zu 4GHz heute.

Verarbeitungsbreite: von 4Bit beim i4004, heute 64Bit beim Itanium und Athlon64.

Bereits 1965 erkannte Gordon Moore anhand sehr weniger Daten: die Komplexität der Bausteine verdoppelt sich regelmäßig (Mooresches Gesetz). Der Zeitraum der Verdopplung ist etwa 18 Monate bei Speicherbausteinen und 24 Monate bei Prozessoren. Dieses Gesetz gilt schon seit 30 Jahren!

1.3 Motivation

1.4 Grundbestandteile eines Mikrorechnersystems

1.5 Grunsätzliche Funktionsweise eines Mikrorechnersystems

1.6 Testfragen

2 Technische Grundlagen

2.1 Informationseinheiten und -darstellung

2.2 Halbleiterbauelemente

2.3 Schaltkreisfamilien

2.4 Die Herstellung integrierter Schaltkreise

2.5 Testfragen

3 Bustreiber und einfache Bussysteme

3.1 Anforderungen

3.2 Ausgänge mit offenem Kollektor (Open Collector)

3.3 Tristate-Ausgänge

3.4 Bustreiber

3.5 Testfragen

4 Speicherbausteine

4.1 Allgemeine Eigenschaften

4.2 ROM (Nur-Lese-Speicher)

4.3 Random Access Memory (RAM)

4.4 Testfragen

5 Ein-/Ausgabebausteine

5.1 Allgemeine Eigenschaften

Ein-/Ausgabebausteine verbinden Bausteine, die nicht direkt am Systembus betrieben werden, mit diesem. Ein Tastaturcontroller oder ein Parallelport haben z.B. selbst kein Businterface und werden über Ein-/Ausgabebausteine dort angebunden. Eine Netzwerkkarte oder Soundkarte hat dagegen ein eigenes Businterface und wird nicht über zusätzliche Ein-/Ausgabebausteine angebunden[2]! Das bedeutet nicht, dass Netzwerkkarten und Soundkarten keine I/O-Geräte wären! Sie haben ja ebenso wie Tastaturcontroller und Parallelport eine I/O-Adresse.

5.2 Einfache Ein-/Ausgabebausteine

Zähler/Zeitgeber-Baustein mit Reload- und Compare-Register Auch genannt »Zählereinheit mit Capture-, Compare- und Reload-Modus«. Dies ist ein Bestandteil eines Mikrocontrollers (also auf dessen Chip integriert), der entweder Systemtakte mitzählt (und damit als Zeitgeber funktioniert) oder externe Signale zählt (und damit als Zähler funktioniert). Die Reaktion des Bausteins auf bestimmte Ereignisse kann programmiert werden. Mögliche Ereignisse und Reaktionen in einem einfachen Baustein:

- Compare-Ereignis (Zählerregister erreicht den Wert im Compare-Register). Mögliche Reaktionen:
  - Pegel von Portleitungen[3] einstellen (high oder low).
  - Zählrichtung des Zählers neu festlegen (aufwärts oder abwärts). Manche Zähler haben nur eine vorgegebene Zählrichtung.
- Reload-Ereignis (Overflow oder Underflow im Zählerregister)[4]. Mögliche Reaktionen:

[2] Es mag sein, dass solche Erweiterungskarten E/A-Bausteine on board haben.
[3] Portleitungen sind Ausgangsleitungen des Bausteins. Sie sind nummeriert.
[4] Bei manchen realen Bausteinen lässt sich einstellen, dass das Reload-Ereignis auch bei einer fallenden Taktflanke einer bestimmten Ausgangsleitung auftritt.

- Reload. Der Wert des Reload-Registers wird neuer Anfangswert des Zählerregisters. Diese Reaktion tritt immer ein.
- Pegel von Portleitungen einstellen (high oder low).

Diese Reaktionen sind zusammen mit den Werten in Compare- und Reload-Register die einzigen Einstellmöglichkeiten dieser Bausteine. Damit lässt sich z.B. ein periodisches pulsweitenmoduliertes Signal auf einer Ausgangsleitung erzeugen. Pulsweitenmodulierte Signale sind Rechteckschwingungen; sie unterscheiden sich in der Breite des HIGH-Anteils (»Puls«) einer Periode. Diesen HIGH-Anteil gibt man durch das Tastverhältnis $p_h = \frac{T_{high}}{T_{Periode}}$ an. Ein Tastverhältnis $p_h = 0,5$ steht für gleich breite HIGH- und LOW-Anteile. Verfahren zur Formung solcher pulsweitenmodulierten Signale:

1. Lass den Zähler stets abwärts zählen.
2. Trage die Gesamtdauer einer Periode in Zählerschritten n_G in das Reload-Register ein. Jede Periode des Eingangssignals (z.B. mit $f_{in} = 1MHz$) entspricht einem Zählerschritt.
3. Lass die Reaktion auf das Reload-Ereignis den Ausgangszustand einer Periode herstellen, also »Ausgangsleitung HIGH setzen« oder »Ausgangsleitung LOW setzen«.
4. Trage als Reaktion auf das Compare-Ereignis die zum Reload-Ereignis komplementäre Reaktion ein. Dies erzeugt einen Pegelwechsel.
5. Trage die verbleibende Gesamtdauer nach dem Pegelwechsel in Zählerschritten n_R in das Compare-Register ein.

Quelle: [11], [3].

6 Busanschluss und Adressverwaltung

6.1 Allgemeines

6.2 Die Trennung von Speicher und Ein-/Ausgabebausteinen

6.3 Aufgaben

7 Einfache Mikroprozessoren

7.1 Grundbegriffe

7.2 Interner Aufbau eines Mikroprozessors

7.3 Busleitungen

7.4 Befehlsaufbau

7.5 Zeitraster bei der Befehlsausführung

7.6 Fallbeispiel: Motorola 6800

7.7 Fallbeispiel: 6502

7.8 Fallbeispiel Z80

8 Aufbau einfacher Systeme

8.1 Hilfsschaltungen

8.2 Testfragen

9 Besondere Betriebsarten

9.1 Direct Memory Access (DMA)

9.2 Interrupts

Interrupt-Vektorisierung Interrupt-Vektorisierung ist ein Verfahren, bei einem IRQ nicht eine Standard-ISR aufzurufen sondern in eine von mehreren ISRs zu verzweigen. Die Bezeichnung stammt wohl von den mathematischen Vektorpfeilen: man stellt sich vor, dass zu jedem IRQ solch ein Pfeil gehört, der auf die aufzurufende ISR zeigt.

9.3 Testfragen

10 Der Intel 8086 - Urvater der PC-Prozessoren

10.1 Eckdaten

10.2 Einige charakteristische Merkmale des 8086

Das Adress-Latch In einem 8086-System sind Adress- und Datenbus gemultiplext. Die Adresse wird zuerst übertragen und durch das Adress Latch weiter an den Speicher bzw. die I/O-Bausteine angelegt, während die Daten über den Bus übertragen werden. Das Adress Latch dient also als Zwischenspeicher für 20 Bit breite Adressen. Quelle: http://www.aleksis.de/ausbild/school/ct/ct_mainboard.html.

10.3 Der Intel 8088

10.4 Aufgaben

11 Der Aufbau von Maschinencode

11.1 Allgemeines

Was ist Adress-Skalierung? Adress-Skalierung ist eine Erweiterung der indizierten Speicheradressierung. Bei gewöhnlicher indizierter Speicheradressierung wird eine Adresse aus der Summe des Inhalts von Basisregister, Indexregister und Displacement gebildet. Anschaulich ist dies die Adressierung von Feldern (Arrays) über Basisadresse (im Basisregister), Offset für den Datensatzbeginn (im Indexregister) und Zeiger auf einen Teil des Datensatzes (im Displacement). Bei Adress-Skalierung wird der Inhalt des Indexregisters vor der Additon mit der Datensatzgröße (z.B. 4 Byte) multipliziert. Anschaulich enthält es jetzt also nicht mehr den Offset des Datensatzbeginns, sondern die Datensatznummer.

Quelle: http://stilzchen.kfunigraz.ac.at/~pgk/lehre/comput.pdf.

11.2 Fallstudie: Intel 8086-Maschinencode

11.3 Aufgaben

12 Die Unterstützung von Multitasking-Betriebssystemen durch Mikroprozessoren

12.1 Anforderungen bei Multitasking

12.2 Fallbeispiel: Intel 80386

12.2.1 Eckdaten

12.2.2 Betriebsarten

12.2.3 Privilegierungsstufen

12.2.4 Geschützte Speichersegmente, Selektoren und Deskriptoren

Am Ende dieses Unterkapitels in [2] ist ein Algorithmus angegeben, wie beim Intel 80386 im Protected Mode auf Daten im Hauptspeicher zugegriffen wird. Wir geben ihn hier in einer ausführlich erläuterten Version an. Vorab: dieser Algorithmus wird vom Prozessor selbständig ausgeführt, als Teil des Fetch-Zyklus in jedem Befehl, der Speicheroperanden verwendet. Für das ausgeführte Programm (etwa `MOV EAX, [BP+10]`) ist dieser Algorithmus unsichtbar, man sagt auch »transparent«. Es sei denn, der Algorithmus endet mit der Fehlermeldung »General Protection Fault«. Dieser Algorithmus hat auch nichts mit der Umsetzung von linearen[5] in physikalische Adressen

[5] Eine lineare Adresse ist eine 32bit-Ganzzahl, die ein Byte im flachen Speichermodell adressiert. Der Speicher wird dabei gesehen als $4GB$ lange Folge von Bytes. Eine lineare Adresse entspricht also dem, was im Betriebssystemsektor als »virtuelle Adresse« bezeichnet wird. Hier ist dieses Wort jedoch schon reserviert für 48bit-Adressen mit dem Aufbau 16bit-Selektor:32bit-Offset. Diese werden durch das hier beschriebene Verfahren der Segmentierung in lineare Adressen umgesetzt.

zu tun, er prüft lediglich ob der aktuelle Befehl auf eine bestimmte virtuelle Adresse zugreifen darf. Die virtuelle Adresse wird dann der MMU (memory management unit) übergeben und erst dann in eine physikalische Adresse transformiert.

Die Datenstrukturen Eine kurze Beschreibung aller Datenstrukturen, die am Speicherzugriff im Protected Mode beteiligt sind, in der Reihenfolge ihrer Verwendung:

Offset-Speicheradresse Jede 32-Bit-Adresse[6] ist ein 32-Bit-Offset in ein bestimmtes Segment. Jedes Segment kann also einen vollständigen virtuellen Adressraum von $2^{32}Byte = 4GB$ zur Verfügung stellen und wirkt für das Programm wie »ganz leerer, ganz privater Arbeitsspeicher«. Programme haben mindestens zwei Segmente (Code- und Datensegment), können also theoretisch $8GB$ eigene Daten haben. Dies ist möglich obwohl der physikalische RAM nur $4GB$ groß sein kann: mindestens die Hälfte der Daten wird dann auf Festplatte ausgelagert.

Segmentregister Die 16bit-Register CS, DS, SS, ES, FS, GS. Sie enthielten im 8086 Segmentadressen. Im 80386 enthalten sie die Selektoren aller für das aktuelle Programm zugreifbaren Segmente.

Selektor Eine 16bit-Datenstruktur, die einen Deskriptor aus einer Deskriptorentabelle auswählen kann. Ein Selektor ist also genau einem Segment zugeordnet. Er hat folgende Felder:

RPL Requested Priviledge Level des zugeordneten Segments.

TI Entscheidet, ob das zugeordnete Segment über die GDT oder LDT zu suchen ist.

Index Position des zugeordneten Deskriptors in der durch TI angegebenen Deskriptorentabelle.

Deskriptorentabelle Ein Feld von Deskriptoren. Das Feld hat eine Startadresse. Die Globale Deskriptorentabelle (GDT) enthält die Deskriptoren aller Segmente des Systems außer den Segmenten der ISRs. Die lokalen Segmente von Prozessen sind nur indirekt enthalten: sie werden in einer Loaklen Deskriptorentabelle (LDT) zusammengefasst, die ein eigenes Segment bildet, das ihrerseits in der GDT eingetragen ist.

Deskriptor Eine 64bit-Datenstruktur, der genau ein Segment zugeordnet ist. Sie enthält Informationen über dieses Segment. Für Zugriffsschutzmechanismen sind davon relevant:

Basis Virtuelle 32bit-Anfangsadresse des Segments.

Limit Größe des Segments. Maximaler Wert $4GByte$.

DPL Desciptor Priviledge Level. Zum Zugriff ist ein Priviledge Level nötig, der numerisch gleich oder kleiner DPL ist.

Der Algorithmus Im Skript steht: »Ein Zugriff auf Daten im Hauptspeicher bei unbekannter LDT und unbekanntem Selektor erfolgt in folgenden Schritten« [2, Kap.12.2.4, S.115]. Eigentlich sollte es heißen: »bei unbekannter LDT und unbekanntem Segment«! Denn die Selektoren für Codesegment, Datensegment und ggf. weitere Segmente müssen bereits gültig und in den entsprechenden Segmentregistern des 80386 geladen sein. Der vollständige Algorithmus:

1. Sei eine Speicheradresse m, auf die zugegriffen werden soll.
2. Bestimme, auf welche Art von Segment zugegriffen werden soll: Codesegment, Datensegment, Stacksegment oder ein Extrasegment. Die Selektoren in Segmentregistern führen zum Ort der Segmente dieser Arten, die dem aktuellen Prozess zugeordnet sind.
3. Bestimme das Segmentregister zur gewünschten Segmentart und lies den Selektor s darin aus.
4. Lies das TI-Bit von s und sein Indexfeld i_1.
   (a) Wenn $TI = 0$, so ist der gesuchte Deskriptor in der GDT enthalten.
      i. Entnehme die Anfangsadresse der GDT aus dem Register GDTR.
      ii. Lese den Deskriptor d an der Position $i_1 \cdot 8$ in der GDT. Faktor 8, weil jeder Deskriptor $8Byte = 64bit$ hat.

[6] Sog. »virtuelle Adresse«. Es sind die Adressen, mit denen ein Programm arbeitet. So, wie sie vom Compiler generiert werden.

(b) Wenn $TI = 1$, so ist der gesuchte Deskriptor in der LDT des Prozesses enthalten.
   i. Entnehme die Anfangsadresse der GDT aus dem Register GDTR.
   ii. Lese den Selektor, der zur LDT des Prozesses führt, aus dem Register LDTR.
   iii. Lese den Index i_2 dieses Selektors.
   iv. Lese den Deskriptor an der Position $i_2 \cdot 8$ in der GDT.
   v. Lese die Basisadresse dieses Deskriptors. Es ist die Basisadresse der LDT des Prozesses.
   vi. Lese den Deskriptor d an der Position $i_1 \cdot 8$ in dieser LDT. Es ist der Deskriptor des gesuchten Segments.

5. Erste Zugriffsprüfung: Aus dem RPL des Selektors s und aktuellem CPL[7] wird der schlechtere ausgewählt, der Effective Privilege Level, EPL. Der EPL muss gleichgut oder besser sein als der DPL des Deskriptors d. Ist das nicht der Fall, wird die Exception »allgemeine Schutzverletzung« (General Protection Fault) ausgelöst, Taskabbruch!
6. Bilde die Summe bm aus der Basisadresse des Deskriptors d und der verlangten Speicheradresse m als Offset.
7. Bilde die Summe bl aus der Basisadresse des Deskriptors d und seinem Limit l als Offset.
8. Zweite Zugriffsprüfung: wenn $bm > bl$, so liegt die Adresse ausserhalb der Segmentgrenzen. Erzeige eine »allgemeine Schutzverletzung«.

12.2.5 Kontrolle von I/O-Zugriffen

12.2.6 Taskwechsel

12.2.7 Paging

[2] ist hier unvollständig. Notwendige Ergänzungen bieten z.B.:

- [9, Kap. 6.7.1]
- [10, Kap. 3.1]

13 Caching

13.1 Allgemeines und Speicherhierchie

13.2 Cache-Strukturen

[2] ist hier unvollständig. Es beschreibt z.B. nicht den Adressaufbau in einem 4-Weg-Cache. Vergleiche deshalb [14, S. 8].

13.3 Aktualisierungsstrategien

Die im Skript [2, Kap. 13.3] genannten Bits V und D, die zu jeder Cache-Zeile abgespeichert werden können, bedeuten:

- V (valid, gültig) wird gesetzt, wenn die Cache-Zeile nicht leer ist, also tatsächlich gecachten Inhalt aus dem Hauptspeicher repräsentieren soll.
- D (dirty, verändert) wird gesetzt, wenn V gesetzt ist und die Cache-Zeile inkohärent mit dem nachgeordneten Speicher ist. Das D-Bit wird nur beim Copy-Back-Verfahren verwendet. Beim Write-Through-Verfahren können keine Inkohärenzen auftreten.

Das Skript [2] behandelt nicht das MESI-Kohärenzprotokoll in der benötigten Tiefe. Ergänzend konsultiere man deshalb etwa [12, Kap. 3.1].

[7] Current Priviledge Level

14 RISC- und CISC-Prozessoren

14.1 CISC-Prozessoren

14.2 RISC-Prozessoren

14.2.1 Registersatz

14.2.2 Pipelining

Diskussion der Vor- und Nachteile von superpipelined Prozessoren, d.i. von Prozessoren mit mehr als 10 Pipeline-Stufen [8]: Die Performance eines superpipelined Prozessors hängt hauptsächlich von seiner Sprungbehandlung ab. Je länger die Pipeline, desto schwieriger ist eine verzögerungsfreie Sprungbehandlung. Superpipelined Prozessoren stellen also hohe Anforderungen an optimierende Compiler, passende Instruktionen für die »delayed slots« zu finden, also für die Plätze nach Sprungbefehlen. Natürlich kann man ebenso das Verfahren der dynamischen Sprungvorhersage anwenden - Fehlvorhersagen führen bei längerer Pipeline jedoch zu größeren Zeitverlusten. Quelle: http://www.isi.edu/acal/tech-reports/Abstracts/1993/tr-93-14.html.

15 PC-Prozessoren der 90er Jahre

15.1 Intel Pentium

15.2 Intel Pentium Pro

15.3 Intel Pentium MMX, Pentium II und Pentium III

Literatur

[1] Prof. Dr. Klaus Wüst: »Mikroprozessortechnik«; Vieweg-Verlag. Es spiegelt sehr gut den Vorlesungsinhalt in Art und Weise der Erklärung und der Stoffauswahl wieder. Es ist eine gute Klausurvorbereitung. Preis: 21,90 EUR.

[2] Prof. Dr. Klaus Wüst: »Mikroprozessortechnik«; Skript zur Vorlesung und Vorläufer von [1]. Quelle: http://homepages.fh-giessen.de/~hg6458/mpt_a4.pdf, referenziert auf http://www.fh-giessen.de/fh/script/. Weitere Quelle: http://www.thesunscreenman.com/download/mct0002.pdf, referenziert auf http://www.thesunscreenman.com/html/elek0300.htm.

[3] Prof. Dr. Klaus Wüst: Klausuren zur Mikroprozessortechnik. Beinhaltet alle Klausuren seit Sommersemester 2000. Quelle: http://homepages.fh-giessen.de/~hg6458/.

[4] A. Schmidtberger: »Mikroprozessortechnik«. Studentische Mitschrift zur Veranstaltung »Mikroprozessortechnik« bei Prof. Dr. Klaus Wüst aus dem Sommersemester 1995. Quelle: http://www.fh-giessen.de/fachschaft/mni/data/skripte/Microproz_ganz.zip (113867 Byte), referenziert auf Skriptsammlung der Fachschaft MNI, FH Gießen-Friedberg http://www.fh-giessen.de/fachschaft/mni/cgi-bin/navi01.cgi?skripte. Die entpackte Datei ist `Mikroprozessortechnik.doc` (422400 Byte). Dieses Dokument ist durch das offizielle Skript [2] völlig obsolet geworden.

[8] Fehlt in [2].

[5] Andreas Ditze: »Microprotz (Lösungen) 1.0 beta«. Lösungen zu Aufgaben, die Prof. Dr. Klaus Wüst in der Veranstaltung »Mikroprozessortechnik« bis zum Sommersemester 1999 als Klausuraufgaben gestellt hat. Quelle: http://www.andreas-ditze.de/studium/Wuest-Mikroprotz-WS99.zip (19623 Byte), entpackt Wuest-Microprotz.doc (102400 Byte).

[6] Sammlung von Unterlagen zur Elektrotechnik für einen Abschluss am Berufskolleg Technik. Darunter sind etliche nützliche Dokumente zur Mikroprozessortechnik. http://www.thesunscreenman.com/html/elek0300.htm

[7] onlinelesen.de :: Sammlung von Fachbüchern zur Prozessortechnik zum freien Download. http://www.computer-literatur.de/buecher/Hardware_nachTitel.html#Prozessoren

[8] Björn Köster: »I/O-Grundlagen«; Ausarbeitung im Ausbildungsgang zum Mathematisch-Technischen Assistenten. Eine gute und leicht verständliche Einführung in PIO, Interrupts und DMA. Quelle: http://www.bjoern-koester.de/iogrundlagen/index.html

[9] Prof. Dr. Michael Jäger: »Betriebssysteme I - Eine Einführung«. Quelle: http://homepages.fh-giessen.de/~hg52/lv/bs1/skripten/bs1skript/pdf/bs1skript.pdf.

[10] Jens Hohmuth, Prof. Dr. Lenk: »Protected Mode Tutorial (Version 1.3.1)«. Quelle: http://www.fh-zwickau.de/doc/prmo/start.htm.

[11] »Der Mikrocontroller MC80515«. Datenblätter und allgemeine Einführung in die Mikrocontrollertechnik http://www.goblack.de/desy/mc8051chip/.

[12] Dr.-Ing. Thomas Flik: »Bus- und Speicherorganisation«. Quelle: kapitelweise auf http://rosw.cs.tu-berlin.de/lehre/bsorg/.

[13] »Intel 80386 Programmer's Reference Manual 1986«. Quelle: http://www7.informatik.uni-erlangen.de/~msdoerfe/embedded/386index.htm.

[14] Swetlana Fries: »Rechnerarchitektur Skript Speicher«. Studentische Mtschrift zur Veranstaltung Rechnerarchitektur an der Universität Karlsruhe. Quelle: http://www.stud.uni-karlsruhe.de/~uke3/Klausuren/Rechnerarchitektur_Skript_Speicher.pdf.

[15] Bähring: »Mikroprozessortechnik«. Es gibt einige ältere Exemplare in der Lehrbuchsammlung der Bibliothek.

[16] Beierlein, Hagenbruch: »Taschenbuch der Mikroprozessortechnik«. Alles, was in der Klausur gefragt wird, steht hier drin. Es ist ein günstiges Buch, von kompetenten Fachleuten geschrieben. Es orientiert sich an gängigen Produkten.

[17] Flik, Liebig: »Mikroprozessortechnik«. In einer älteren Auflage in etlichen Exemplaren in der Lehrbuchsammlung der Bibliothek vorhanden.

[18] Hennessy, Patterson: »Rechnerarchitektur«. Die Autoren sind Pioniere der RISC-Architektur. Hier werden auch quantitative Aspekte behandelt. Es gibt einige Exemplare in der Lehrbuchsammlung der Bibliothek.

[19] Hans-Peter Messmer: »PC-Hardwarebuch«.

[20] Müller, Balz: »Mikroprozessortechnik«; Vogel Fachverlag. Sehr leicht verständlich, dafür nicht alle Details. In älterer Auflage in etlichen Exemplaren in der Lehrbuchsammlung der Bibliothek vorhanden.

[21] Tanenbaum, Goodman: »Computerarchitektur«. Peerson-Verlag. Sehr locker zu lesen.