- **IT-Lösungen**
  - **Dokumentationen**
    - **Präsentationen**

PCT-Solutions

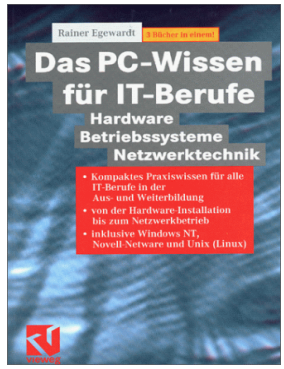
by
Rainer Egewardt

www.pct-solutions.de
info@pct-solutions.de

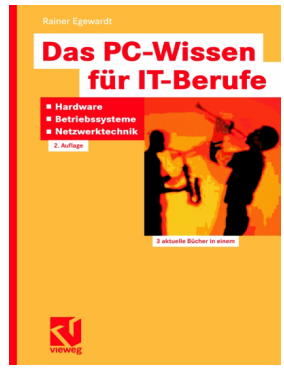
Unser "PC-Wissen für IT-Berufe" ist zu einem Bestseller im IT-Buchmarkt geworden

IT-Buchprojekte
von
PCT-Solutions

1. Auflage
600 Seiten



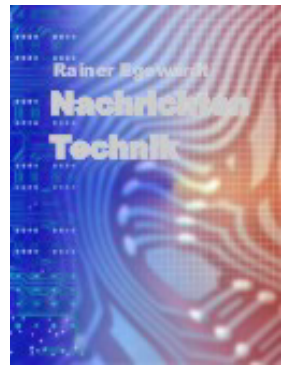
2. Auflage
1200 Seiten



Unser weiteren Buch-Projekte:

600 Seiten

600 Seiten



Nachfolgend
Das PC-Wissen für IT-Berufe

Micro-Prozessor-Technik

2. Auflage

1 Hardware-Technik

1.1 Micro-Prozessor-Technik

Jeder Prozessor-Typ hat seine Eigenarten. In den grundsätzlichen Funktionen sind allerdings alle Prozessoren gleich. Diese Funktionen sollen in diesem Abschnitt beschrieben werden.

1.1.1 Computerarten

Merkmale für die Leistungsfähigkeit von Computern sind:

- Architektur der CPU (CISC, RISC)
- Taktfrequenz und Wortbreite
- Anzahl der CPU’s
- Größe des Arbeitsspeichers
- Speicherkapazität insgesamt
- Schnittstellen zur Peripherie
- Eigenschaften des Betriebssystems

Weil von den über 100 internen Befehlen der CISC-CPU’s meistens weniger als die Hälfte benötigt werden, entstanden die RISC-CPU’s (ca. 60 Befehle). Diese können Befehle mit nur wenigen Takten ausführen, was mit nur wenigen Anweisungen zur selben Zeit erreicht wird (Pipeline-Verfahren).

Je größer die Taktfrequenz und die Wortbreite der CPU ist, desto schneller werden Programmanweisungen ausgeführt.

In diesem Zusammenhang werden Computer nach MIPS klassifiziert. (1 MIPS = 1 Million Anweisungen pro Sekunde)

PC's (80x86) 20–200 MIPS
Alpha Chip 400 MIPS
Pentium 400–2000 MIPS
AMD-K6 1500 MIPS

1.1.2 Funktionseinheiten

Die minimale Konfiguration eines Computers besteht aus:

- Prozessor
- Rechenwerk, Steuerwerk, Register
- Taktgeber
- Systemsteuerung
- Adress-, Daten-, Steuerbus
- Speicher (RAM, ROM)
- I/O Einheit

1.1.3 Funktion

Grundsätzliche Funktion:

Der Prozessor liest über seinen Datenbus Daten aus dem Speicher. Die entsprechende Stelle im Speicher wird dabei durch eine Adresse festgelegt, die der Prozessor mit Hilfe der Adressierungseinheit berechnet und über den Adressbus an das Speichersubsystem übergibt (siehe Abschnitt 1.2.10 „BUS-Systeme"). Je nach Mode ist die Adressierung unterschiedlich aufwändig.

Der Lese-Schreib-Vorgang läuft über die Busschnittstelle (BU), wozu die BU die Adresse und ggf. den Wert der zu schreibenden / speichernden Daten ausgibt. Der Prozessor spricht in gleicher Weise den Speicher an, um einen Befehl zu lesen. Die gelesenen Daten (Befehle) werden in der Prefetch-Queue gespeichert. In einem späteren Schritt liest die Befehlseinheit den

Befehl hier heraus, decodiert ihn und übergibt ihn an die Ausführungseinheit.

Befehle und Daten sind im selben physikalischen Speicher untergebracht. Diese sind in verschiedene Abschnitte (Segmente) unterteilt. Die Segmente enthalten Programmcodes oder Daten.

Die Businterfaceeinheit regelt den Datenverkehr zwischen den externen Einheiten und dem Prozessor. Zur schnellen Befehlsabarbeitung gibt es Befehlswarteschlangen.

In der Segmentierungseinheit wird zusammen mit der Pagingeinheit die Berechnung der Adressen zur virtuellen Speicherverwaltung durchgeführt.

1.1.4 Befehlsverarbeitung (s. Abb. 1)

Wesentliche Bestandteile des Prozessors sind:

- Befehlszähleinrichtung mit Adressregister
- Befehlsregister mit Befehlsdecoder
- ALU mit Akkumulator und Statusregister

1. Der Prozessor sendet auf den Adressleitungen die Adresse der Speicherzelle aus, in der der nächste zu verarbeitende Befehl steht.
2. Er liest mit einem Steuersignal diesen Befehl und speichert ihn im Befehlsregister.
3. Er stellt fest, um welchen ihm bekannten Befehl es sich handelt (Befehl decodieren (vom Hersteller vordefiniert)). Abhängig von dieser Entschlüsselung holt er weitere Daten aus dem Speicher oder führt den Befehl aus.

1.1.5 Befehlszähl-Einrichtung

Sie steuert die Reihenfolge, in der die Befehle abgearbeitet werden. Dafür ist intern ein Zähler eingebaut, der Program-Counter, der nach jedem abgearbeiteten Befehl um 1 erhöht wird, da die Befehle im Speicher hintereinander ge-

speichert sind. So wird die nächste Adresse im Speicher geliefert.

Das Adressregister wird zu Beginn einer jeden Befehlsholphase mit dem Inhalt des Program-Counters geladen.

1.1.6 Befehlsregister und Befehlsdecoder

Das über den Programmzähler und das Adressregister adressierte Befehlsbyte gelangt während der Befehlsholphase in das Befehlsregister.

Im Befehlsdecoder wird ermittelt, welchem Befehl diese Byte entspricht. Das Ergebnis wird der Ablaufsteuerung mitgeteilt.

1.1.7 Ablaufsteuerung

Sie erzeugt alle internen und externen Steuersignale für das Holen und Ausführen der Befehle.

Alle Steuersequenzen aller möglichen Prozessorbefehle werden in der Ablaufsteuerung in einem Microprogramm-Speicher aufbewahrt.

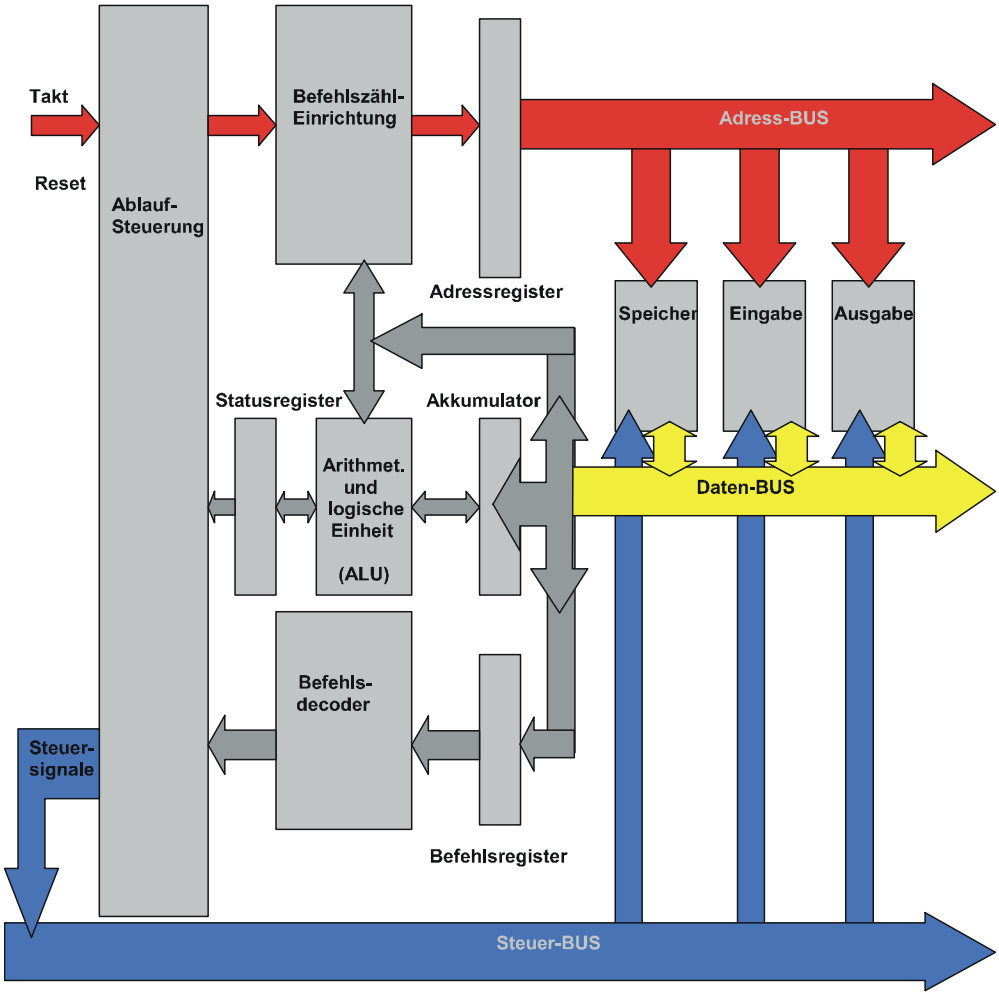


Abb. 1 Abarbeitung von Microprozessor-Befehlen

1.1.8 Takt und Reset

Zur Erzeugung des Systemtaktes besitzen einige Prozessoren interne Oszillatoren. Bei anderen Typen muss der Takt extern erzeugt werden.

Damit der Prozessor beim Anlegen einer Betriebsspannung nicht willkürlich mit der Pro-

grammabarbeitung beginnt, muss er in einen Grundzustand gebracht werden. Dafür hat jeder Prozessor einen Reset-Eingang. Nach einem Impuls an diesem Eingang wird der Program-Counter gelöscht und die Befehlsholphase eingeleitet. Der Prozessor beginnt dann bei der Adresse 0000H mit der Befehlsabarbeitung.

1.1.9 ALU und Akkumulator

Die Verarbeitung der Daten erfolgt in der ALU. Hier werden arithmetische-, logische- und Schiebeoperationen durchgeführt. Die Daten, die verarbeitet werden, werden Operanten genannt.

Die ALU kann einen oder zwei Operanten verarbeiten. Wie und in welcher Form Operanten verarbeitet werden, hängt vom auszuführenden Befehl ab. Für die Zuführung beider Operanten besitzt die ALU zwei Kanäle. Der eine Operant gelangt über den Datenbus in die ALU, der andere über den Akku.

Das Ergebnis einer logischen / arithmetischen Operation wird im Akku abgelegt.

Alle Prozessoren haben Befehle, die das Laden des Akkus mit Daten aus dem Speicher oder das Ablegen das Akkuinhalts im Speicher bewirken. Außerdem läuft der Datenverkehr zu/von den Ein/Ausgabeeinheiten über den Akku ab.

Der Akku hat eine zentrale Funktion.

1.1.10 Statusregister

Im Statusregister werden Zustandbits (Flags) gesetzt, die unmittelbar an die Ablaufsteuerung übergeben werden, da es Befehle für den Prozessor gibt, die in Abhängigkeit vom Zustand der Flags ausgeführt werden.

Carry Bit (CY) =1 Überlauf bei einer Operation,
=0 kein Überlauf

Zero Bit (Z) =1 Ergebnis ist 0,
=0 Ergebnis ist nicht 0

Negativ Bit (N) =1 Ergebnis ist negativ,
=0 Ergebnis ist positiv

Parity Bit (P) =1 Anzahl Bits ist gerade,
=0 Anzahl ist ungerade

1.1.11 Register

Es existieren in einem Prozessor eine Vielzahl von Registern.

Die Wichtigsten sind:	
Akkumulator:	Arith., log. und Schiebeoperationen
Baseregister:	Zeiger auf Basisadresse
Zählregister:	speichert Wiederholungen
Datenregister:	speichert Daten
Base Pointer:	Zeiger auf Basisadressen im Stacksegment
Source Index:	Indexzeiger
Befehlszähler:	Offset der Befehle
Stackpointer:	Stackzeiger
Codesegment:	Befehle und Daten, die unmittelbar adressiert werden
Datensegment:	Daten, die dem gerade laufenden Programm zugeordnet sind
Stacksegment:	Daten, auf die mittels Stack-Befehlen zugegriffen wird
Flagregister:	Zustandsbits
Steuerregister:	Page Directory Basis Register
Debugregist	Ablegen der linearen Adressen

er:	
Speicherverwal-tungsregister:	dienen der Speicherverwaltung

1.1.12 Stackpointer

Der Stackpointer ist ein Register, das Adressen der im Stapelspeicher (Stack) aufbewahrten Registerinhalte enthält. Die Zwischenspeicherung der Registerinhalte im Stack wird benötigt, wenn bei einer Unterbrechung der momentanen Arbeit des Prozessors (Interrupt) andere momentan benötigte Daten in die Register geladen werden müssen.

Nach Abarbeitung des Interrupts holt sich der Prozessor die zwischengespeicherten Daten wieder aus dem Stack und fährt mit der Arbeit an der Stelle fort, an der er vor dem Interrupt aufgehört hat.

Der Stack beginnt immer an der obersten Adresse und wird nach der niedrigeren hin belegt (LIFO-Speicher, Last In – First Out).

1.1.13 Interrupt

Allgemein:

Durch einen Interrupt wird der Prozessor in seiner momentanen Arbeit unterbrochen. Dies könnte z.B. beim Darstellen von Zeichen auf dem Bildschirm sein, damit er ein Zeichen von der Tastatur einliest. Damit der Prozessor erkennt, mit welcher Arbeit er vor Auftreten des Interrupts beschäftigt war, werden zuvor der Zustand des Prozessors und die Inhalte der Register auf dem Stack zwischengespeichert und nach Bearbeitung des Interrupts wieder eingelesen. Jeder Interrupt hat einen Vektor auf eine Interruptroutine, die den Interrupt behandelt.

Hardwareinterrupt:

Hier wird der Interrupt durch einen Hardware-Baustein oder ein Peripheriegerät (zB. Festplatte) ausgelöst. Es wird zwischen maskierbaren (sperrbaren) und nicht maskierbaren Interrupts unterschieden. Löst ein Gerät einen NMI (nicht maskierbarer Interrupt) aus, so wird dem NMI-Anschluss des Prozessors ein aktives Signal zugeführt. Der Prozessor arbeitet den gerade ausgeübten Befehl ab und führt unmittelbar anschließend einen Interrupt 2 durch. Beim PC wird ein NMI nur ausgelöst, wenn ein schwer wiegender Hardware-Fehler vorliegt.

Ein NMI hat absolute Priorität und wird immer als erstes Bearbeitet.

Dem gegenüber können die Interrupt-Anforderungen IRQ maskiert werden. Diese Interrupts liegen am INTR Anschluss des Prozessors.

Für die Bedienung von Hardware-Interrupts spielt der Interrupt-Controller eine wichtige Rolle. Er verwaltet mehrere Interrupt-Anforderungen und gibt sie geordnet nach Priorität an den Prozessor weiter (siehe Abschnitt 1.2.14 „Chipsatz“).

Softwareinterrupt:

Ein Softwareinterrupt wird gezielt durch einen INT-Befehl beim Programmieren ausgelöst.

Exeption:

Exeptions sind Interrupts, die vom Prozessor selbst erzeugt werden. Die Auswirkungen entsprechen einem Softwareinterrupt.

Ursache für eine Exeption ist im Allgemeinen eine prozessorinterne Fehlerbedingung, die den Eingriff von Systemsoftware erfordert und vom Prozessor nicht mehr selber behandelt werden kann.

1.1.14 Betriebsarten

Real-Mode:

Betriebsmodus der 80x86 Prozessoren, bei dem der Segmentwert einfach mit 16 multipliziert, und der Offset dazu addiert wird, um eine Speicheradresse zu erzeugen (16 × Segment + Offset). Es findet keine Zugriffsprüfung auf Code-, Datensegmente und den I/O Bereich statt.

Alle 80x86 Prozessoren, einschließlich Pentium, können im Real Mode betrieben werden.

Protected-Mode:

Betriebsmodus ab 286er, bei dem der Zugriff eines Tasks auf Code-, Datensegmente und I/O Bereich, selbstständig geprüft wird.

Die Adressbildung ist völlig inkompatibel zum Real-Mode. Real-Mode Anwendungen wie DOS können im Protected-Mode nicht laufen.

Virtual-Mode:

Betriebmodus ab 386, der Zugriffe wie im Protected-Mode prüft, aber die Adressbildung wie im Real Mode handhabt. Dadurch können Real-Mode-Anwendungen in einer geschützten Umgebung ablaufen.

Zusammen mit dem Paging sind mehrere Tasks parallel möglich.

1.1.15 Pipelining

Pipeline-Architektur:

Bei der Befehlsabarbeitung ist eine parallele Verarbeitung möglich, was sich besonders bei Langwortbefehlen auswirkt, weil dadurch die Rechenzeit verkürzt wird. Es können so viele Befehle auf einmal parallel abgearbeitet werden, wie die Architektur der Pipeline dies erlaubt.

Bei einer 3-teiligen Pipeline z.B. wird während der Prefetch-Phase (Befehlsvorhol-Phase) gleichzeitig ein weiterer Befehl bearbeitet. Der vorgeholte Befehl wird in die Einheit A geholt und schrittweise durch die Verarbeitungs-

einheit B nach C gebracht. In C liegt der Befehl vollständig decodiert vor und ist für die Steuereinheit zur Ausführung gültig. Befehle mit unmittelbaren Daten finden ihre Daten in B vor und können ausgeführt werden. Es wird also eine Funktionsausführung eingeleitet, noch bevor die gegenwärtige Funktion abgeschlossen ist.

1.1.16 Busschnittstelle

Neben anderen Bestandteilen enthält sie die Prefetch-Queue.

1.1.17 Ausführungseinheit

Die Ausführungseinheit ist für die Datenverarbeitung zuständig und enthält:

- die Steuereinheit
- die ALU
- verschiedene Register

1.1.18 Adressierungseinheit

Die Adressierungseinheit hat häufig noch eine Speicherverwaltungseinheit integriert. Sie besteht aus der Segmentierungseinheit und der Pagingeinheit.

1.1.19 Prefetch-Queue

Die Prefetch-Queue ist ein kleiner Zwischenspeicher in der CPU, in der der Prefetcher bereits die nächsten Befehle ablegt, bevor der Prozessor den gegenwärtigen Befehl abgearbeitet hat (Pipelining). Sie dient zur Entlastung des Bussystems und zur Vorcodierung der Befehle bei CISC-Prozessoren.

1.1.20 Operationen

Register-Register-Operationen:

Die Vielzweckregister dienen als interne Datenspeicher der CPU. Die Ausführungseinheit liest

Werte aus diesen Registern und führt sie der ALU zu, um Berechnungen anzustellen. Das Ergebnis wird wieder in den Registern abgelegt.

Speicher-Speicher-Operationen:

Daten werden direkt aus dem Speicher gelesen und nach Verarbeitung wieder im Speicher abgelegt.

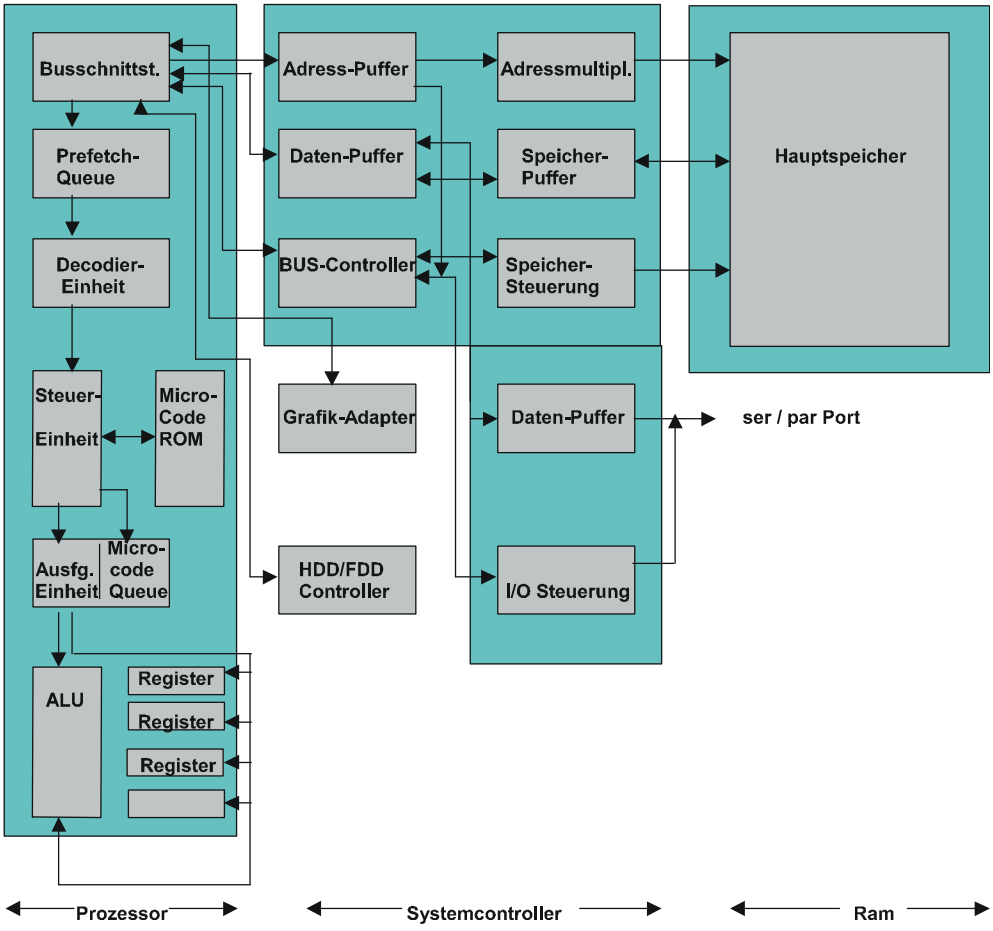


Abb. 2 Kommunikation zwischen Prozessor, Systemcontroller und RAM

1.1.21 Logische Speicheradressierung und Speicherzugriff

Codesegmentregister und Befehlszähler bilden die Adresse, aus der ein Befehl geladen werden soll (Befehlsfetching). Da die Adresse berechnet wird, nennt man diesen Vorgang auch eine indirekte Adressierung im Gegensatz zur direkten, bei der die physikalische Adresse ausgegeben wird. Bei der indirekten Adressierung können unterschiedliche Segmentadressen + Offsets die gleiche physikalische Adresse meinen. Damit kann der Prozessor diesen so adressierten Befehl einlesen und ausführen.

Am Code des Befehls erkennt der Prozessor, wie viele Bytes er einlesen muss. Nach der Ausführung des Befehls wird der Befehlszähler um die Zahl der Bytes inkrementiert, die der gerade ausgeführte Befehl aufwies. Das Paar Codesegment und Befehlszähler weisen dann auf den nächsten Befehl.

Beim Befehlsfetching spielen Prefetch-Queue und Busschnittstelle eine wichtigen Rolle. Befehle werden zuerst in die Prefetch-Queue eingelesen, die diese dann an die Befehlseinheit weitergibt. Sind in der Prefetch-Queue durch die Weitergabe der Befehlsbytes an die Steuereinheit so viele Bytes frei geworden, wie der Datenbus breit ist, liest die Busschnittstelle entsprechend viele Bytes aus dem Speicher neu in die Prefetch-Queue ein.

Führt der Prozessor gerade einen Befehl aus, der keinen unmittelbaren Speicherzugriff erfordert, kann die Busschnittstelle die Prefetch-Queue nachladen, ohne dass die Abarbeitung des gerade aktiven Befehls behindert wird. Damit steht nach Beendigung der momentanen Arbeit des Prozessors sofort der nächste Befehl zur Verfügung und muss nicht extra aus dem Speicher nachgeladen werden. Allerdings hat der Speicherzugriff eines aktiven Befehls Vorrang.

Parallel zur Ausführung des aktiven Befehls und nach dem Nachladen der Prefetch-Queue decodiert die Befehlseinheit den nächsten Befehl und bereitet ihn für die EU (Execution Unit) zur Ausführung vor. Die Ausführungseinheit EU führt den Befehl nun in bestimmten Taktzyklen aus. Einfache Befehle benötigen zwei, umfangreichere Befehle bis zu 300 Taktzyklen.

Heute versucht man immer mehr unabhängige Schritte parallel auszuführen, um die Leistungsfähigkeit des Prozessors zu steigern. (Ein normaler Pentium (PI) besitzt 3 unabhängige Pipelines, die parallel arbeiten.)

1.1.22 Stack

Jedes Programm besitzt ein eigenes Stacksegment, auf dem der Wert eines Registers oder Speicherwortes abgelegt wird. Es können die Flags und alle Vielzweckregister auf dem Stack gespeichert und wieder entnommen werden.

Der Stack wächst bekanntlich von oben nach unten. Werden Daten auf dem Stack abgelegt, vermindert sich der Wert des Stackpointer.

Der Stackpointer ist ein Register, das Adressen der im Stack aufbewahrten Registerinhalte enthält. Der Stack kann zur Zwischenspeicherung von Daten benutzt werden, aber am häufigsten wird er zur Übergabe von Parametern an Prozeduren und Subroutinen benutzt. Auch Daten, die der Prozessor bei einem Interrupt zwischenspeichern muss, werden auf den Stack geschoben.

1.1.23 Datensegmentregister

Das Datensegmentregister ist immer dann wichtig, wenn ein Befehl Daten aus dem Speicher liest oder in ihm abspeichert, d.h., wenn Speicheroperanten betroffen sind. Der Offset des Speicheroperanten wird in einem Vielzweckregister bereitgehalten, und das Paar DS:Offset verweist auf den anzusprechenden Wert (indirekte Adressierung).

DS wird standardmäßig als das einem Offset zugeordneten Segmentregister verwendet. Wenn ein Wert in einem anderen Segment geschrieben oder gelesen werden soll, muss das Segmentregister DS durch ein Extrasegmentregister ersetzt werden. Die Daten des Codesegments sollten dabei nur ausführbar oder lesbar sein, nicht aber überschreibbar. Ein Überschreiben des Code führt zum Absturz der Programme. Nur ausführbare Daten sind zwar in die Prefetch Queue einlesbar, nicht aber in ein Vielzweckregister oder Segmentregister. Ein Programm kann also nicht im Sinne von Daten verwendet werden, die bearbeitet werden.

Die Verwendung verschiedener Segmente für Daten, Stack und Code gestattet eine Trennung der verschiedenen Abschnitte eines Programms.

(Wichtig im Protected Mode).

1.1.24 Real Mode, High Memory Area und Himem.sys

Da der 8086 in der Adressierungseinheit wegen seines 20 Bit breiten Datenbusses nur ein 20 Bit Addierer zur Verfügung steht, ist die ausgegebene Adresse auf 1 MB beschränkt. Segment- als auch Offsetregister haben eine Breite von 16 Bit. Damit ist ihr max. Wert jeweils gleich FFFFH (64 kB).

Wird die dem Real Mode zu Grunde liegende Operation für Adressbildung angewandt, so ergibt sich: Segment FFFFH $\times$ 16D (10H) = FFFF0H + Offset FFFFH = 10FFEFH = 1114095D / 1024 = 1.087 MB. Somit ergibt sich ein Adressbereich von 1 MB.

Bei einem 20 Bit Adressaddierer fällt die führende (1)0FFEFH als Übertrag weg und wird ignoriert. Es bleibt 0FFEFH als physikalische Adresse übrig. Diese fällt in das erste 64 kB Segment des Adressraumes (Wrap Around).

Da ab dem 386er aber ein 32 Bit breiter Datenbus, und damit auch ein 32 Bit Addierer in der

Adresseinheit zur Verfügung steht, und damit lineare Adressen mit 32 Bit erzeugt werden können, wird die führende 1 hier nicht ignoriert, sondern es werden 21 Adressleitungen aktiviert. Aus diesem Grunde können die Adressen 100000H bis 10FFEFH angesprochen werden. Die 1 MB Grenze ist damit um 64 kB überschritten. Dieser Speicherbereich wird High-Memory-Area (HMA) genannt. Er ist für DOS und Programme, die im Real Mode laufen, über dem 640 kB Basisspeicher, eine 10% ige Speichererweiterung. Hier werden Teile von DOS und Treiber abgelegt, damit so viel wie möglich Chip-Speicher zur Verfügung bleibt.

Die Ausgabe der Adresse 10FFFFFH entspricht allerdings nicht einem echten 21 Bit breiten Adressbus, der ja bis 1FFFFFH adressieren kann.

Der Gerätetreiber HigMem.Sys wurde als Speichermanager für das HMA entwickelt, da auch Gerätetreiber wie Smartdrv.Sys und Ramdrive.Sys diesen Speicherbereich nutzen und es somit zu keiner Kollosion der Daten kommen kann.

Weiter gibt es einen Upper-Memory-Block (UMB). Im Bereich zwischen 640 kB und 1 MB, der für ROMs vorgesehen ist, aber nur selten auch von diesen belegt ist, gibt es, von modernen Speichercontrollern eingerichtet, ein 64 kB großes Fenster (EMS).

Ausführlichere Beschreibung, siehe Abschnitt 3.1.5 „Speicher-Bereiche im RAM“.

1.1.25 Paging

Allgemein:

Die Unterteilung eines großen virtuellen Adressraumes in kleinere physikalische Einheiten, den Pages (Segmenten).

Demand Paging:

Die Auslagerung der Pages eines Hauptspeichers auf einen externen Massenspeicher, wenn die Daten gerade nicht benötigt werden (swappen).

Möchte die CPU auf die ausgelagerten Daten zugreifen, so wird die gesamte Page wieder in den Hauptspeicher eingelesen und dafür eine gerade nicht benötigte Page ausgelagert. Dadurch kann ein wesentlich größerer virtueller Adressraum erzeugt werden, als physikalisch wirklich vorhanden ist. Nach Kombination von Segment und Offset zu einer linearen Adresse wird die so erhaltene lineare Adresse beim Paging in eine 10-Bit-Page-Directory-Adresse, eine 10-Bit-Page-Adresse und einen 12-Bit-Offset aufgespaltet.

Jede Page ist 4 kB groß, sodass der gesamte Adressraum beim Paging in 4 kB-Blöcke unterteilt wird.

1.1.26 Physikalischer Zugriff auf den Speicher

Um Daten aus dem Speicher lesen oder in diesen schreiben zu können, muss der Prozessor über seine Anschlüsse physikalisch adressieren (über den Adressbus) und die Daten übertragen (über den Datenbus).

Ein solcher Ablauf folgt einer streng definierten Abfolge von Adress-, Steuer- und Datensignalen. Der Prozessor stellt dazu die physikalische Adresse des Speicherobjekts nach Auflösung von Segmentierung und Paging über die Adresseinheit und die Busschnittstelle bereit. Die Signale des Prozessors werden nicht direkt zur Ansteuerung des Speichers oder des Systembusses verwendet. Vielmehr wird ein Bus-Controller dazwischen geschaltet, der alle Signale für das Speichersubsystem zur Verfügung stellt. Dieser Bus-Controller ist zusammen mit den Adress- und Datenpuffern sowie zusätzlichen Steuerschaltungen Bestandteil eines Systemcontrollers.

1.1.27 Waitstates

Kann der Speicher oder das Peripheriegerät innerhalb bestimmter Taktzyklen einen Schreib/Lesevorgang nicht abschließen, dann erhält die Speichersteuerung ein Ready* Signal (nicht fertig). Damit wird dem Prozessor signalisiert, dass er einen weiteren Zyklus einlegen soll, damit dem Speicher oder Peripheriegerät mehr Zeit gegeben werden wird, die Anforderung auszuführen.

Waitstates für Lesevorgänge können sich von solchen für Schreibvorgänge unterscheiden, weil DRAMs im Allgemeinen häufig schneller schreiben als lesen können.

Die Anzahl der Waitstates ist vom Ort abhängig.

Der Hauptspeicher läuft mit erheblich weniger Waitstates als der Video-RAM, der Bestandteil des Adressraumes ist.

Waitstates können bei älteren Boards mit Jumpern eingestellt werden. Bei neueren Boards werden für die jeweilige Situation flexible Waitstates benötigt, weswegen diese im BIOS eingestellt werden können.

1.1.28 I/O-Adressraum

Neben dem Speicherbereich gibt es noch den I/O-Adressraum.

Der Speicher- und der I/O-Bereich werden beide über den Adress- und Datenbus adressiert, wobei sich ein Zugriff nur durch ein anderes Signal unterscheidet. Der I/O-Bereich ist logisch nur mittels des Akkus, der Speicher dagegen mit allen Registern adressierbar. Die Segmentregister sind für den I/O-Bereich ohne Bedeutung.

Die Ports und der I/O-Bereich werden dazu verwendet, Register in Peripheriegeräten anzusprechen.