5

5 Prozessortechnologie

5.1 Einleitung

Informationen werden durch Hardwarekomponenten verarbeitet. Die Beschränkung der Fähigkeit von Maschinen, Informationen zu verarbeiten, liegt in der Beschränkung der Prozessor Architekturen, die heute verwendet werden. Die Gesamtheit aller Prozessoren kann durch eine Reihe von Klassierungsschemata dargestellt werden. In diesem Kapitel werden das allgemeine Rechnermodell als Basis für Prozessor Architekturen, Flynn's Taxonomie sowie das "Erlangen Classification System" (ECS) dargestellt.

Das allgemeine Rechnermodell umfasst drei grundlegende Architekturtypen, die Allgemeine sequentielle Maschine (RAM - Random Access Machine), die Allgemeine parallele Maschine (PRAM - Parallel Random Access Machine) und die Vektorielle Maschine (VRAM - Vector Random Access Machine).

Die wichtigste Klassifikation stammt von Michael J. Flynn aus dem Jahre 1966, Er unterscheidet zwischen Monoprozessoren (SISD - Single Instruction Single Data), Multiprozessoren und verteilte Systeme (MIMD - Multiple Instruction Multiple Data), Vektorrechner und Arrayrechner (SIMD - Single Instruction Multiple Data) und Pipelinerechner (MISD - Multiple Instruction Single Data).

Das Erlangen Classification System (ECS) unterscheidet die Art der Parallelität (Pipelining und Nebenläufigkeit) sowie die Ebenen der Parallelität (Befehlsebene, Verarbeitungsebene, Verarbeitungsbreite).

5.2 Allgemeines Architektur-Modell

Prozessor Architekturen basieren auf zwei generalisierten allgemeinen Modellen, der Random Access Machine (RAM) die eine allgemeine sequentielle Maschine definiert und der Parallel Random Access Machine (P-RAM), welche eine allgemeine parallele Maschine

beschreibt. Eine spezialisierte Variante der RAM ist das V-RAM (Vector Random Access Machine), welche als Grundlage für viele Vektorprozessoren dient.

Das Entwicklungsteam des Alpha AXP Chip hat 1992 den Begriff Mikroprozessorachitektur geprägt. *"Thus, the architecture is a document that describes the behaviour of all possible implementations; an implementation is typically a singe computer chip. The architecture and software written to the architecture are intended to last several decades, while individual implementations will have much shorter lifetimes. The architecture must therefore carefully describe the behaviour that a machine-language programmer sees, but must not describe the means by which a particular implementation achieves that behaviour."*

(Demnach ist die Architektur ein Dokument, welches das Verhalten aller möglichen Implementationen beschreibt; Eine Implementation ist typischerweise ein einzelner Computerchip. Die Architektur, sowie die Software, die für die Architektur geschrieben wurde, sind dazu gedacht, mehrere Jahrzehnte zu überdauern, während einzelne Implementationen viel kürzere Lebenszeiten haben. Die Architektur muss desshalb das Verhalten, das ein Maschinensprachenprogrammierer sieht, sehr sorgfältig beschreiben, darf aber die Art und Weise, mit welcher eine spezifische Implementation dieses Verhaltens erzielt wird, nicht beschreiben.)

5.2.1 Random Access Machine

Die Grundarchitektur serieller Computer wird als Random Access Machine (RAM) dargestellt.

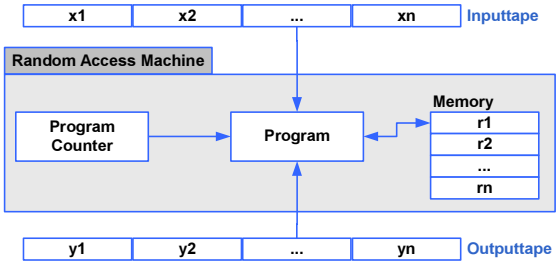

Abbildung 5.1 Random Access Machine [Aho et al. 1974]

Eine Random Access Machine bildet das Modell von Turing ab und besteht aus einem Programm Counter, dem Programm und einem Memory, wie in **Abbildung 5.1** dargestellt.

- *Inputtape:* Es besteht aus einer Sequenz von Integern. Jedes Mal, wenn das Programm einen Wert gelesen hat, bewegt sich das Band einem Schritt nach links.
- *Outputtape:* Es besteht aus einer Sequenz von Integern. Nach jedem Schreiben eines Wertes durch das Programm bewegt sich das Band einen Schritt nach links.
- *Program:* Das Programm besteht aus einer fixen Sequenz von Operationen, wie load, store, jump, test etc. Ein Programm liest die Informationen des Inputtapes, bearbeitet

diese Daten aufgrund seiner Instruktionssequenz, benutzt die Register des Memory zur Zwischenspeicherung und schreibt das Resultat auf das Outputtape.

- *Program Counter:* Der Programm Counter steuert den Ablauf des Programms.
- *Memory:* Die Register des Memory sind die Ablage für die Zwischenspeicherung von Werten.

Inputtape und Outputtape sind eigentlich zwei Teile eines Turingsbandes, welches die allgemeine Turingmaschine beschreibt, die die Basis des RAM-Modells darstellt.

5.2.2 Parallel Random Access Machine

Eine allgemeine parallele Machine wird mit dem PRAM (Parallel Random Access Machine) Modell beschrieben. Dies ist ein idealisiertes Modell, welches in seiner reinen Form nicht implementiert ist. Das Modell basiert auf dem RAM Modell und besteht aus einer Reihe von unabhängigen Random Access Maschinen, die miteinander über ein Shared Memory kommunizieren.

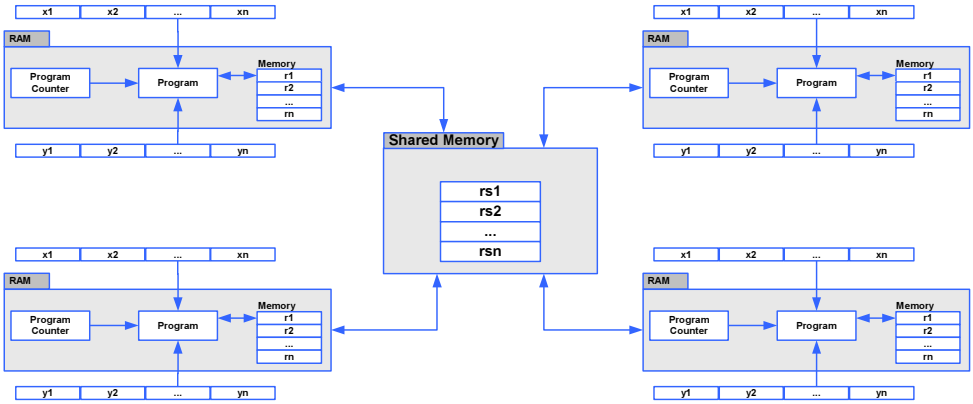

Abbildung 5.2 Parallel Random Access Machine.

Die einzelnen Prozesse (RAM) können in jedem Ausführungsschritt eine globale oder lokale Speicheradresse lesen, ein einzelner Befehl kann durch den Prozessor ausgeführt werden oder es kann auf eine globale oder lokale Speicheradresse geschrieben werden (**Abbildung 5.2**).

5.2.2.1 Vijaya Ramachandran's Klassifikation

Die Professorin Vijaya Ramachandran der University of Texas at Austin hat eine Klassifikation von PRAM anhand der Zugriffsmethode auf das Shared Memory vorgeschlagen

- *EREW-PRAM*: *Exclusive Read – Exclusive Write:* Lediglich ein RAM kann zu einem gegebenen Zeitpunkt lesen und schreiben.

- *CREW-PRAM*: *Concurrent Read – Exclusive Write:* Mehrere Prozessoren können lesen und schreiben. Lediglich ein RAM kann zu einem gegebenen Zeitpunkt schreiben.
- *CRCW-PRAM*: *Concurrent Read – Concurrent Write:* Mehrere RAM können gleichzeitig lesen und schreiben. Diese Variante wird am häufigsten realisiert. Kern einer CRCW-PRAM Umsetzung ist der so genannte Workspace, die im weitesten Sinn einen virtuellen Prozessor mit zugewiesenen Ressourcen (Memory, I/O, etc.) darstellt.

5.2.3 Vector Random Access Machine

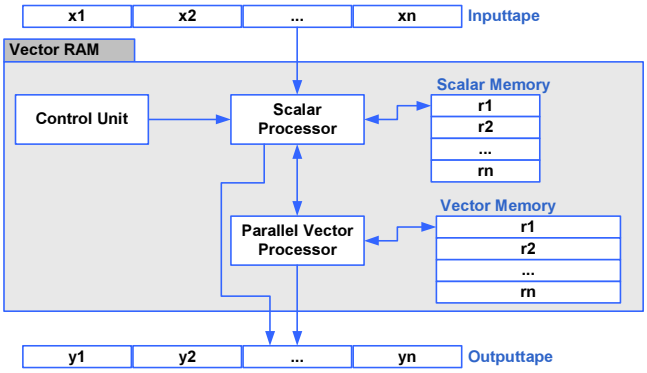

Abbildung 5.3 V-RAM Aufbau [Blelloch 1990]

Die Vector Random Access Machine (V-RAM) funktioniert analog der RAM, sie besitzt jedoch zusätzlich einen Vektorspeicher, einen Vektorprozessor und einen Vektor I/O Anschluss, wie in **Abbildung 5.3** skizziert. Der Vektorprozessor führt Operationen auf ganzen Vektoren aus.

Die Eigenschaften des V-RAM Modells sind:

- Es kann eine variable Anzahl von Operationen gleichzeitig auf einzelnen Elementen ausgeführt werden.
- Das Modell verwendet als zentrale Datenstruktur einen Vektor.
- Die Kontrolle über den Ablauf der Verarbeitung ist seriell.
- Die Parallelität des Systems wird über parallele Primitiven (Operationen auf Vektoren) ermöglicht.

5.3 Flynn's Taxonomie

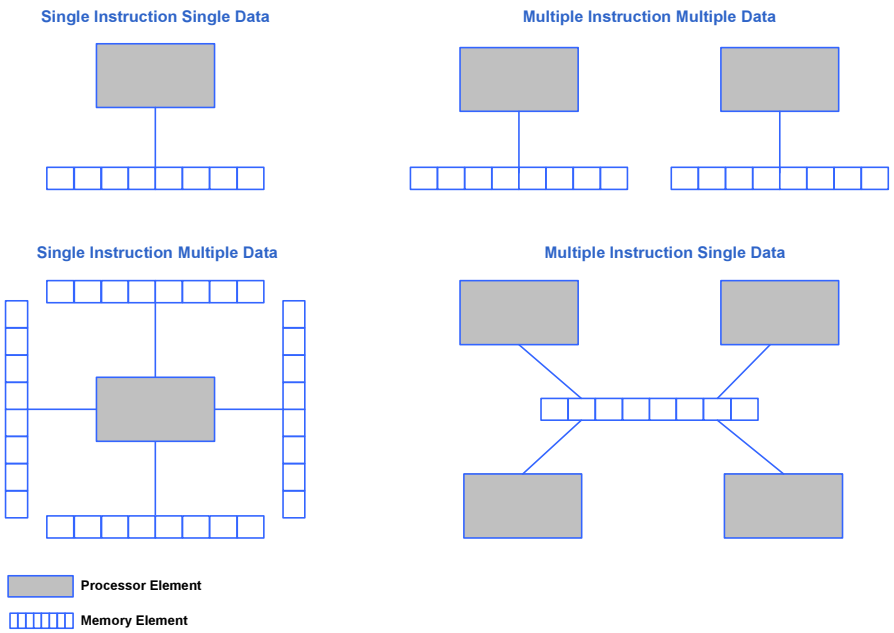

Abbildung 5.4 Flynn Taxonomie

Die Etablierteste Klassifikation von Prozessor Architekturen stammt von Michael J. Flynn, ehemaliger Professor für Computer Architecture an der Universität Stanford [Flynn 1966]. Seine Klassifikation entwarf M.J. Flynn bereits im Jahr 1966. Sie unterscheidet vier Rechnerklassen, wie in **Abbildung 5.4** dargestellt.

- *SISD:* single instruction stream, single data stream – Monoprozessoren.
- *MIMD:* multiple instruction stream, multiple data stream – Multiprozessoren und verteilte Systeme
- *SIMD:* single instruction stream, multiple data stream – Vektorrechner und Arrayrechner.
- *MISD:* multiple instruction stream, single data stream – Pipelinerechner.

5.3.1 Single Instruction Single Data

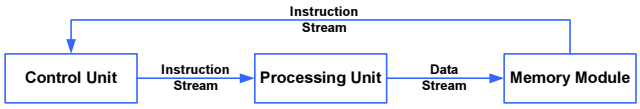

Abbildung 5.5 Single Instruction Single Data Aufbau

Die Klasse SISD (Single Instruction Single Data) ist eine direkte Umsetzung des RAM (Random Access Machine) Modells (**Abbildung 5.5**). Eine mögliche allgemeine Beschreibung dieser Klasse ist der "von Neumann Computer", der aus 4 Teilen besteht, der CPU, der ALU, dem Memory und den I/O Komponenten.

- *Central Processing Unit:* Die Kontrolleinheit zur Interpretierung der Instruktionen.
- *Arithemic and Logical Unit (ALU):* Eine arithmetische und logische Einheit zur Bearbeitung von Daten.
- *Memory:* Einem einzigen Memory zur Speicherung von Instruktionen und Daten.
- *I/O:* Eine Ein- und Ausgabeeinheit als Schnittstelle.

5.3.1.1 Von Neumann Bottleneck

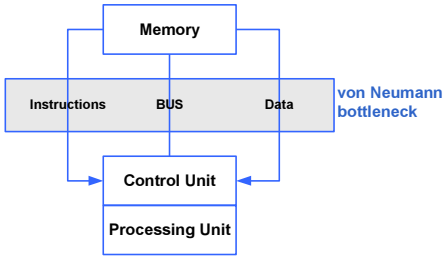

Abbildung 5.6 Das von Neumann Bottleneck

Die meisten konventionellen Prozessoren fallen unter die Klasse SISD, wenn auch nicht in reiner Form, da jeder Prozessor spezielle Instruktionen oder Operationen parallel ablaufen lassen kann. Die Beschränkung dieser Art von Prozessor Architekturen besteht in der Verbindung zwischen dem Memory und der Control Unit respektive der Processing Unit, dem Bus. Dieser Bus ist die einzige Verbindung und ist damit der entscheidende Einflussfaktor für die Geschwindigkeit des Gesamtsystems. Diese Beschränkung wird auch "von Neumann bottleneck" genannt (**Abbildung 5.6**).

5.3.2 Multiple Instruction Multiple Data

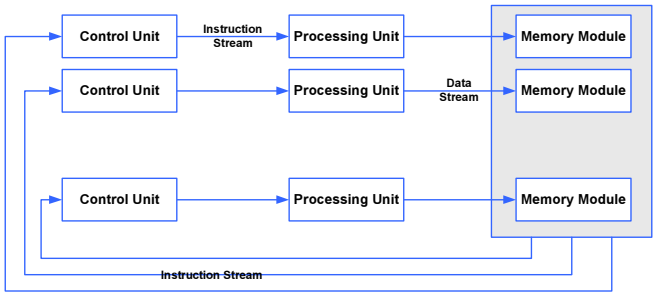

Abbildung 5.7 Multiple Instruction Multiple Data Aufbau

Die Klasse MIMD (Multiple Instruction Multiple Data) umfasst die meisten heute eingesetzten Parallelrechner (**Abbildung 5.7**). Unter diese Klasse fallen verteilte Systeme und Mehrprozessorrechner. Solche Systeme bestehen aus einer Menge von vollständig unabhängigen Prozessoren, die jeweils ihre private Sequenz von Instruktionen auf privaten oder globalen Daten ausführen.

MIMD geht davon aus, dass Ressourcen wie Speicher und Peripherie zwischen Prozessoren geteilt werden und die einzelnen Prozessoren interagieren. Die Intensität der Interaktion wird auch "Coupling" genannt.

- *Loosely Coupled MIMD:* Prozessoren und Memory Modules sind durch ein Netzwerk (Cross-Bar-Switch) voneinander getrennt.
- *Tightly Coupled MIMD:* Prozessoren und Memory Modules sind auf einem Node (Knoten). Die verschiedenen Knoten sind durch ein Netzwerk miteinander verbunden.

5.3.2.1 23 Teraflop

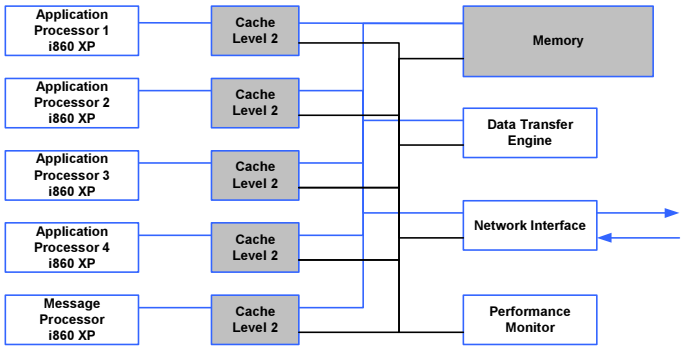

Abbildung 5.8 Beispiel eines MIMD Systems: die Intel Paragon XP/S Systemarchitektur

Einer der letzten Rechner auf der Liste der schnellsten Computer der Welt (top500) mit einer reinen MIMD Architektur, war im Jahr 2004 auf Platz 2 der Liste (**Abbildung 5.8**). Das Lawrence Livermore National Laboratory setzte einen Cluster basierend auf 4096 Intel Itanium2 Prozessoren ein. Dieses MIMD System basierte auf 1016 Knoten mit je 4 Prozessoren. Das System leisteet 32 Teraflops. Vorbild dieses Systems ist die Intel Paragon XP/S Architektur. Der Paragon XP/S bestand aus 1024 "Computing Nodes", die jeweils 4 Intel i860 Prozessoren sowie 128 Mbyte Speicher und einen 2x16 Kbyte grossen Cache umfassen. Ein spezieller Message Processor verwaltete die Kommunikation zwischen den einzelnen Knoten.

Heute werden hybride Architekturen, die aus einer Kombination von SIMD und MIMD Architekturen bestehen, eingesetzt. Die Nummer 1 der Liste der schnellsten Supercomputer des Jahres 2013 beispielsweise verwendet 3 Typen von Prozessoren in Kombination. Der Milkyway-2 Rechner verfügt über insgesamt 3‘120’496 Prozessorkernen, davon 48.000 GPUs der Art Intel Xeon Phi 31S1P3 mit je 57 Kernen, 32.000 CPUs der Art Intel Xeon E5-26924 mit 12 Kernen und 4.096 CPUs der Art Galaxy FT-15005 Frontend-Prozessoren mit 16 Kernen. Es stehen 1.375 TB RAM und 54 Pflop Peak Performance zur Verfügung. Zudem werden 64 Speicherserver mit insgesamt 12.4 PB Speicher eingesetzt.

Die Nummer 2 der schnellsten Supercomputer ist ebenfalls ein hybrides Modell, welches ebenfalls auf einer Kombination von drei Prozessortypen basiert. In diesem Fall sind es Cray XK7, Opteron 6274 mit 16 Cores und Nvidia Kepler mit 14 Cores. Insgesamt werden 560‘640 Cores verwendet, um ca. die halbe Leistung des Milkyway-2 zu erreichen.

5.3.3 Single Instruction Multiple Data

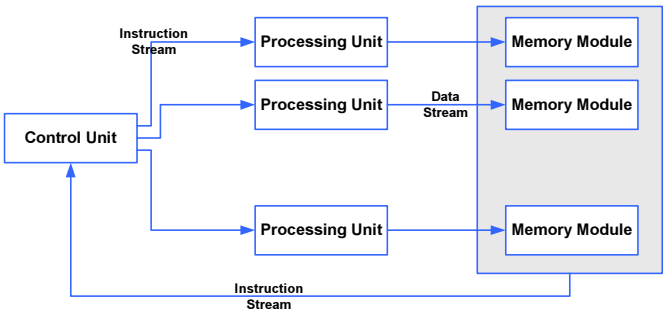

Abbildung 5.9 Single Instruction Multiple Data Aufbau [Nagel 2005]

Die Klasse SIMD (Single Instruction Multiple Data) umfasst Prozessorarrays, Vektorrechner und systolische Arrays (**Abbildung 5.9**). Jede beteiligte Einheit folgt derselben Instruktionssequenz, führt diese jedoch auf privaten Daten durch. Alle Processing Units Einheiten werden zentral gesteuert. Während Prozessorarrays und Vektorrechner aus vielen relativ einfachen von Neumann Computern bestehen, ist das charakteristische an systolischen Arrays, dass sie lediglich add oder multiply Operationen durchführen und auf einem matrixartigen Netzwerk basieren. Der klassische Vertreter eines systolischen Arrays ist der so genannte Xputer [Meinks, Pietsch 1998].

5.3.3.1 Array Computer

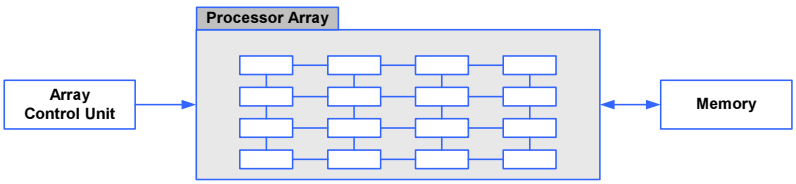

Abbildung 5.10 Array Computer

Ein Array Computer besteht aus einem einzigen Control Processor und einem Array von vielen einfachen Processing Elements (**Abbildung 5.10**). Der erste Vektorcomputer wurde 1976 von Seymour Cray auf den Markt gebracht. Der Cray-1 war der Startschuss zur Entwicklung von Hochleistungsrechnern, die vor allem im Bereich der Klimaforschung und der Berechnung von Simulationen eingesetzt worden sind.

5.3.3.2 Connection Machine von Thinking Machines Corporation

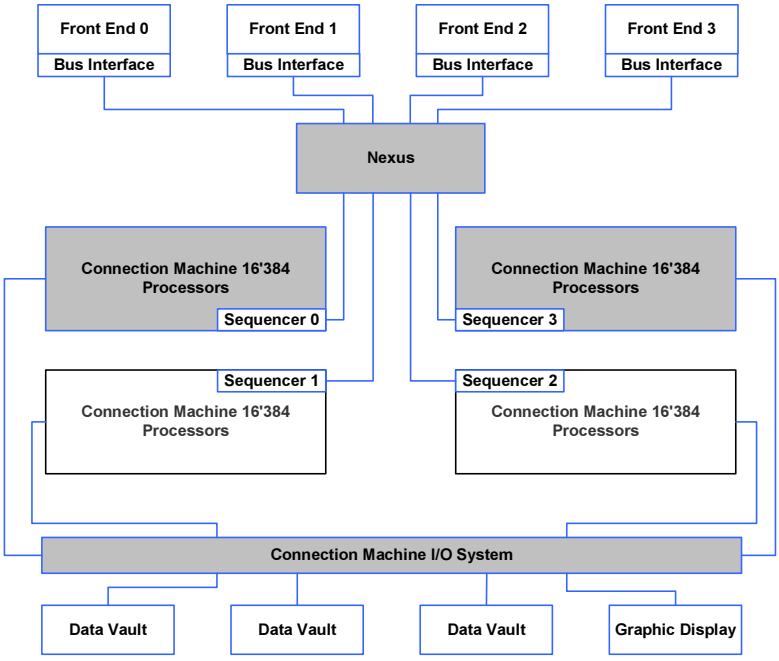

Abbildung 5.11 Die Systemorganisation der Conncetion Machine

Das bekannteste Beispiel eines Array Computers ist die Connection Machine, deren Architektur in **Abbildung 5.11** dargestellt ist. Die Connection Machine basiert auf den Ideen von Danny Hilis, einem Studenten von Marvin Minski am MIT Artificial Inteligence Lab, und wurde von 1983 – 1984 in der ersten Version gebaut von der Firma Thinking Machines [Tucker, Robertson 1988].

- *Front End 0 – 3:* Diese Rechner (DEC VAX) stellen die Schnittstelle der Connection Machine zur Aussenwelt dar. Auf diesen Rechnern wird die Software gespeichert und zur Ausführung vorbereitet.
- *Bus Interface:* Die Schnittstelle zum Nexus.
- *Nexus:* Der Nexus ist ein 4x4 Cross-Bar-Switch, der eine schnelle Verbindung zwischen den Front End Systemen und den einzelnen Connection Machine Prozessorfarmen erlaubt.
- *Sequencer 0 – 3:* Die Schnittstelle wischen der Connection Machine Prozessorfarm und dem Nexus. Die Sequencer sind haben die Funktionen einer Array Control Unit.
- *Connection Machine 16'384 Processors:* Das Herzstück der Connectionmachine, ein Array bestehend aus 16'384 einfachen Prozessoren.
- *Data Vault:* Massenspeicher der Connection Machine.
- *Graphic Display:* Anzeigesystem der Connection Machine

5.3.4 Multiple Instruction Single Data

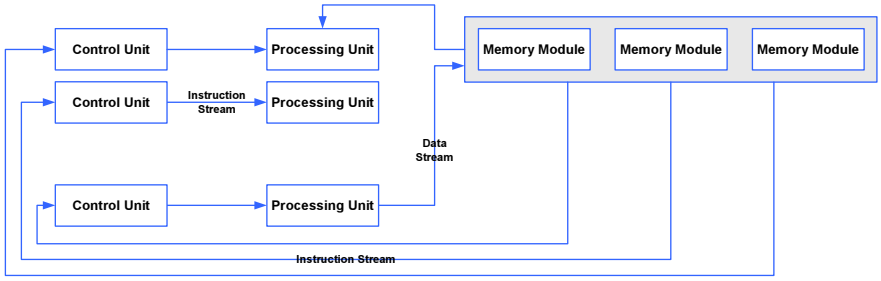

Abbildung 5.12 Multiple Instruction Single Data Aufbau

Zur Klasse MISD zählt Flynn die Pipeline Rechner (**Abbildung 5.12**). Die Technik des Pipelinging wird benutzt, wenn identische Prozesse mehrfach in einem System vorkommen, wenn sukzessive Operanden durch verschiedene Processing Units verarbeitet werden können und wenn Operationen von verschiedenen Processing Units ausgeführt auf dieselben Systemressourcen zugreifen. Graphikprozessoren verwenden oft diese Technik.

5.4 Erlangen Classification System

Der Deutsche Computerpionier Wolfgang Händler (11.12.1920 – 19.2.1998) hat 1974 das Erlangen Classification System (ECS) erwickelt [Hwang, Briggs 1984]. Diese Klassierung unterscheidet die Art der Parallelität (Pipelining und Nebenläufigkeit) sowie die Ebenen der Parallelität (Befehlsebene, Verarbeitungsebene, Verarbeitungsbreite).

Formale Definition ECS

Ein Rechnertyp T ist definiert durch das Tripel (K[*K'], D[*D'], W[*W'])

K: Anzahl der nebenläufigen Steuerwerke (Control Units)

K´: Anzahl der Control Units für Programm- bzw. Prozessor-Pipelining

D: Anzahl der nebenläufigen Rechenwerke (Processor Units) je Control Unit

D´: Anzahl der Pipelining- Processor Units je Control Unit

W: Anzahl der parallelen Bitstellen im Processor Unit (Arithmetisch Logische Einheit)

W´: Anzahl der elementaren Teilwerke

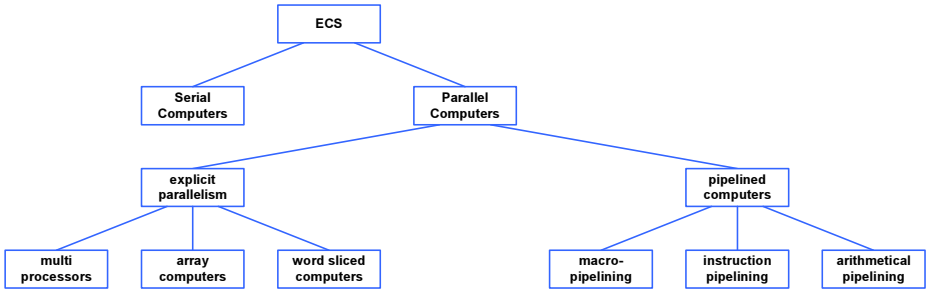

Abbildung 5.13 Erlangen Classification System [Weper et al. 1998]

Das Erlangen Classification System unterscheidet serielle von parallelen Rechnern (**Abbildung 5.13**). Die Klassifikation selbst bezieht sich lediglich auf die Parallelrechner.

- *Multi Processors:* Multiple Instructions, Multiple Data, Different Programs
- *Array Computers:* Single Instruction, Multiple Data, Singe Program
- *Word Sliced Computers:* Single Instruction, several bits of Single Data Stream
- *Computers Using Macropipelining:* Multiple Instructions of different Programs, Singe Data Stream
- *Computers using Instruction Pipelining:* Multiple Instructions, Single Data, Single Program
- *Computers using Arithmetical Pipelining:* Multiple Instructions, Parts of one Program, Single Data

5.4.1.1 Einfaches Rechenbeispiel ECS

Beispiel 1: T = (4,1, 32 * 5)

Ein Multiprozessorsystem mit 4 Prozessoren, ein Rechenwerk (z.B.Integer Unit) pro Prozessor, 32 Bit Intstruction Length und 5 Stage Pipeline.

5.4.1.2 ECS Erweiterungen

Die Erweiterungen des ECS sind:

Verbindungsoperator "+" für inhomogene Zusatzeinheiten.

Verbindungsoperator "v" für verschiedene Betriebsarten.

Verbindungsoperator "x" für Markopipelining inhomogener Systeme.

5.4.1.3 Erweiterte ECS Beispiele

Beispiel 1: Der i860 Prozessor

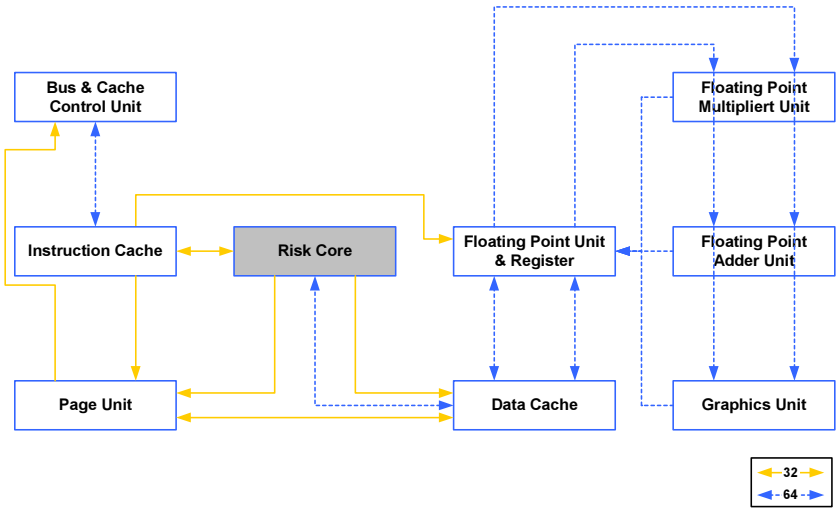

Abbildung 5.14 Blockschema des i860 Prozessor

Der i860 ist ein Prozessor mit einem CPU (Risk Core), zwei Processing Units (Floating Point Multiplier Unit und Floating Point Adder Unit), wie in **Abbildung 5.14** skizziert. Der Prozessor verwendet maximal drei 32 Bit Verbindungen (vom "Risk Core" zur "Page Unit" und zum "Data Cache" sowie zwischen "Risk Core" und "Instruction Cache") und maximal drei 64 Bit Verbindungen (Zwischen "Floating Point Unit & Register" und "Floating Point Multiplier Unit").

ECS-Wert: Ti860 = (1*1, +*2,[32 * 3] + [64 * 3])

Beispiel 2 Distributed Array Prozessor

Der Prozessor kennt drei Betriebsarten:

Betriebsart 1: 1 Control Prozessor mit 64 Rechenwerken und einer 64 stufigen Pipeline.

Betriebsart 2: 1 Control Prozessor mit 128 Rechenwerken und einer 32 stufigen Pipeline.

Betriebsart 3: 1 Control Prozessor mit 4096 Rechenwerken und einer 1 stufigen Pipeline.

ECS-Wert: TDAP= (1, 64, 64) v (1, 128, 32) v (1, 4096, 1)

5.5 Prozessorfamilien – Konzepte und Techniken

5.5.1 Der logische Aufbau eines Rechners

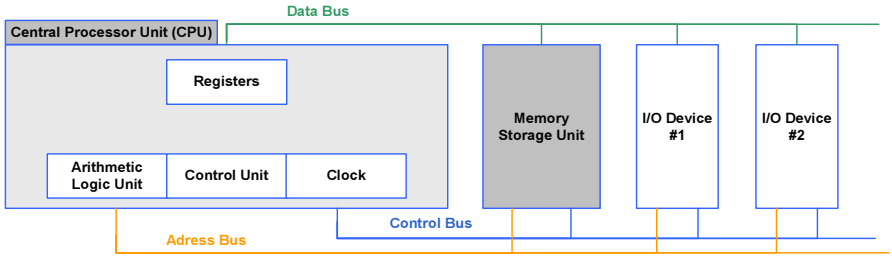

Abbildung 5.15 Logischer Aufbau eines Rechners

Der logische Aufbau eines Rechners trennt zwischen der Central Prozessor Unit (CPU), einem System von mehreren logischen Bussystemen (Adress Bus, Control Bus), den externen Memory und einer Reihe von I/O Komponenten (**Abbildung 5.15**).

- *CPU:* Die CPU besteht aus Register, einer Arithmetic Logic Unit (ALU), einer Control Unit sowie einer Clock.
- *Registers:* Die Register umfassen eine beschränkte Anzahl Speicherplätze zur internen Zwischenablage von Werten.
- *Arithmetic Logic Unit:* Die ALU führt arithmetische Operationen sowie logische Operationen aus.
- *Control Unit:* Die Control Unit koordiniert die Ausführung der Maschinenbefehle im Prozessor.
- *Clock:* Die Clock synchronisiert die internen Operationen des Prozessors mit den anderen Systemkomponenten.
- *BUS:* Der Systembus eines Rechners umfasst normalerweise drei verschiedene Bus-Systeme, den Data Bus (Daten), den Control Bus (Synchronisation) und den Address Bus (Instruktionen).

5.5.2 Virtuelle Maschinen

Um Kompatibilität zwischen einzelnen Prozessorgenerationen zu gewährleisten, verwenden alle Prozessorhersteller das Konzept der virtuellen Maschinen. Das Konzept stammt von Andrew Tannenbaum, der an der "Faculty of Sciences" der Universiteit Amsterdam unterrichtet [Tannenbaum 2000]. Die klassische Definition stammt jedoch vom Informatikwissenschaftler David Gelernter, heute Professor in Yale.

"*A running program is often referred to as a virtual machine - a machine that doesn't exist as a matter of actual physical reality. The virtual machine idea is itself one of the most elegant in the history of technology and is a crucial step in the evolution of ideas about software. To come up with it, scientists and technologists had to recognize that a computer running a program isn't merely a washer doing laundry. A washer is a washer whatever clothes you put inside, but when you put a new program in a computer, it becomes a new machine.... The virtual machine: A way of understanding software that frees us to think of software design as machine design.*" [Gelernter 1997]

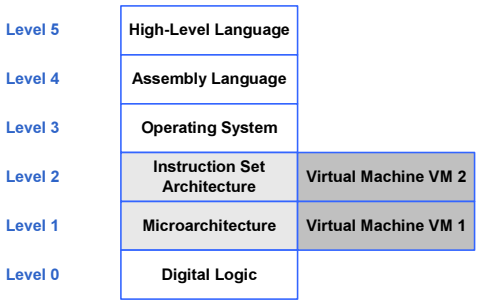

Abbildung 5.16 Der Prozessor als Virtuelle Maschine

Der entscheidende Punkt ist die Tatsache, dass eine Virtuelle Maschine wiederum auf einer anderen Virtuellen Maschine aufsetzen kann, wie in **Abbildung 5.16** dargestellt. Eine Virtuelle Maschine bildet immer das logische Modell eines Rechners ab.

- *Microarchitecture:* Die Hersteller von Prozessoren veröffentlichen normalerweise die spezifischen Befehlssätze der Microarchitektur (Microinstructions) nicht. Auf dieser Ebene werden primitive Operationen im Prozessor, wie beispielsweise ein "Fetch" eines Operanden aus dem Memory umgesetzt.
- *Instruction Set Architecture:* Das Instruction Set ist die Zugriffsebene auf den Prozessor. Der Hersteller veröffentlicht diesen Instruction Set, damit die Hersteller von Assemblern und Betriebssysteme darauf zugreifen können.

5.5.3 Der Instruction Execution Cycle

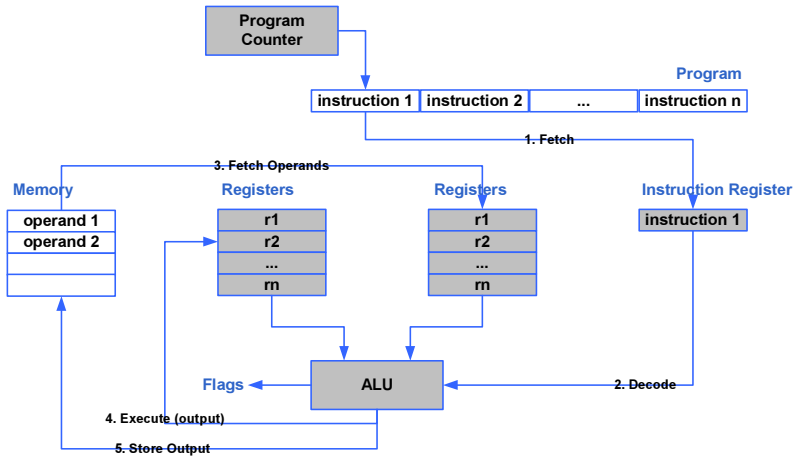

Abbildung 5.17 Der Logische Aufbau eines Rechners und der Ablauf einer Instruction

Die Ausführung einer einzelnen Maschineninstruktion wird in einzelne Schritte aufgeteilt. Diese Schritte werden "Instruction Execution Cycle" genannt (**Abbildung 5.17**). Ein einzelner Schritt eines Instruction Execution Cylces stellt eine „Micro Instruction“ dar. Alle möglichen Instruktionen eines Prozessors ergeben den Befehlssatz des Prozessors.

- *1.Fetch:* Die Control Unit ruft eine Instruction ab, indem er diese aus dem Memory in das Instruction Register der CPU kopiert.
- *2.Decode:* Die Control Unit ermittelt den Type der Instruction, die ausgeführt werden soll. Null oder mehrere Operanden werden an die ALU übertragen. Anschliessend wird der ALU das Signal zur Ausführung der Instruction gesendet.
- *3.Fetch Operands:* Wird ein Operand verwendet, der im Memory gespeichert ist, so initiiert die Control Unit einen Read, um den Operanden in eines der Register zu übertragen.
- *4.Execute:* Die Arithmetic Logic Unit führt die Instruction aus. Daten (Zwischenresultate der Instruction) werden an den Output Operanden (Register) gesendet.
- *5.Store Output:* Das Resultat (Output Operand) wird im Memory abgelegt.

Die einzelnen Schritte des Instruction Execution Cycle werden auch "Stage" (Etappen) genannt. Je nach Prozessor variiert die Anzahl der Stages.

5.5.4 Multi-Stage Pipelines

Jeder Schritt des Instruction Execution Cycle braucht mindestens einen Clock Cycle (Clock Tick). Dies bedeutet jedoch nicht, dass ein Prozessor wartet, bis alle Schritte abgearbeitet sind, bevor mit der Ausführung der nächsten Instruction begonnen werden kann.

Ein Prozessor kann jeden Schritt des Instruction Execution Cycle parallel abarbeiten. Diese Technik wird Piplining genannt. Am Piplining sind meist mehrere Komponenten eines Prozessors beteiligt.

5.5.4.1 Intel IA-32 Six-Stage Pipeline

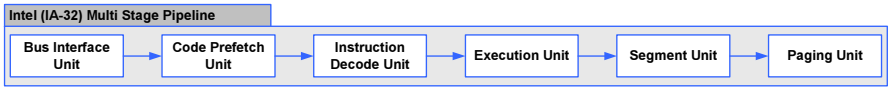

Abbildung 5.18 Intels Six Stage Pipeline

Die Intel IA-32 Architektur (ab 386er Generation) arbeitet mit einem "six-stage execution cycle", wie in **Abbildung 5.18** dargestellt.

- *Bus Interface Unit:* Die BIU greift auf das Memory zu.
- *Code Prefetch Unit:* Die Code Prefetch Unit erhält Instruction von der BUI und legt diese in einer Instruction Queue ab.
- *Instruction Decode Unit:* Die Instruction Decode Unit dekodiert die Maschineninstruktionen aus der Instruction Queue und übersetzt sie in Microcode.
- *Execution Unit:* Die Execution Unit führt die Microcode Instructions aus.
- *Segment Unit:* Die Segment Unit übersetzt logische in lineare Adressen und führt Zugriffstests (Protection Checks) durch.
- *Paging Unit:* Die Paging Unit übersetzt lineare Adressen in physische Adressen, führt Page Checks und hält eine Liste derjenigen Pages, auf die zuletzt zugegriffen worden ist.

5.5.4.2 RISC Five-Stage Pipeline

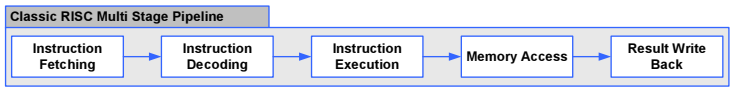

Abbildung 5.19 RISC Five-Stage Pipeline

Prozessoren wie der Alpha oder der PowerPC verwenden die klassische "five-stage execution pipeline" der RISC Rechner Architektur (**Abbildung 5.19**).

- *Instruction Fetching:* Der Programm Counter ruft die nächste auszuführende Instruction ab. Die Instructions sind normalerweise in einem Instruction Cache gespeichert.
- *Instruction Decoding and Operand Fetching:* Der Opcode (Kode des entsprechenden Befehls im Instruction Set des Prozessors) und die Operanden werden geprüft und die entsprechend Kontrollsignale werden gesendet. Operanden werden allenfalls in Register geladen.

- *Instruction Execution:* Die durch den Opcode definierte Operation wird ausgeführt. Zugriffe auf das Memory sind als einzelne Operation definiert.
- *Memory Access:* Daten werden von oder ins Memory geladen. Typischerweise wird ein Datencache verwendet.
- *Result Write-Back:* Das Resultat der Operation wird in ein so genanntes Register File zurück geschrieben.

5.5.4.3 Vergleichendes Beispiel

Ausgangspunkt der Überlegungen ist eine "six-stage pipeline", also eine Pipeline mit sechs Schritten, die eine vollständige Abarbeitung eines Befehls durchlaufen muss. Es wird die Anzahl der benötigten Prozessorzyclen zur Abarbeitung eines Befehls verglichen.

Es werden vier Fälle verglichen:

- Fall 1: Non-Pipelined Instruction Execution
- Fall 2: Pipelined Instruction Execution
- Fall 3: Superscalar Execution Pipelines
- Fall 4: Parallel Superscalar Execution Pipelines

Ein Superscalar Prozessor hat zwei oder mehr Execution Pipelines. Hautmotivation für diese Architekturen ist die Tatsache, dass viele Einzelschritte (Stages) eines Befehls mehr als einen Clock Cycle brauchen, um vollständig abgearbeitet werden. In einer parallelen Superscalaren Pipeline Prozessor Architektur werden Teile der Pipeline parallel geführt. So wird der Stage 4 in zwei parallelen Versionen Stage 4A und Stage 4B realisiert. Die Befehle werden abwechslungsweise durch den einen oder den anderen Stage geleitet.

Clock Cycle	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6
1	Instruction 1					
2		Instruction 1				
3			Instruction 1			
4				Instruction 1		
5					Instruction 1	
6						Instruction 1
7	Instruction 2					
8		Instruction 2				
9			Instruction 2			
10				Instruction 2		
11					Instruction 2	
12						Instruction 2

Abbildung 5.20 Fall 1: Non-Pipelined Instruction Execution

Die Instruction Nr. 2 beginnt erst, wenn die Instruction Nr. 1 vollständig abgearbeitet ist. Anzahl notwendiger Zyklen: 12

Clock Cycle	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6
1	Instruction 1					
2	Instruction 2	Instruction 1				
3		Instruction 2	Instruction 1			
4			Instruction 2	Instruction 1		
5				Instruction 2	Instruction 1	
6					Instruction 2	Instruction 1
7						Instruction 2

Abbildung 5.21 Fall 2: Pipelined Instruction Execution

Die Instruction Nr. 2 beginnt, wenn der erste Stage der Instruction Nr. 1 abgearbeitet ist. Anzahl notwendiger Zyklen: 7

Generell werden für k Ausführungschritte (Execution Stages) und n Befehle (Instructions) k + (n - 1) Prozessorzyklen benötigt,

Clock Cycle	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6
1	Instruction 1					
2	Instruction 2	Instruction 1				
3	Instruction 3	Instruction 2	Instruction 1			
4		Instruction 3	Instruction 2	Instruction 1		
5			Instruction 3	Instruction 1		
6				Instruction 2	Instruction 1	
7				Instruction 3		Instruction 1
8				Instruction 3	Instruction 2	
9				Instruction 3		Instruction 2
10					Instruction 3	
11						Instruction 3

Abbildung 5.22 Fall 3: Superscalar Execution Pipelines

Ausgangsituation des Falls 3 ist ein Stage 4, der zwei Clock Cycles braucht. In diesem Fall gehen Zyklen verloren und für k Ausführungschritte (Execution Stages) und n Befehle (Instructions) werden (k + 2n - 1) Prozessorzyklen benötigt.

Clock Cycle	Stage 1	Stage 2	Stage 3	Stage 4A	Stage 4B	Stage 5	Stage 6
1	Instruction 1						
2	Instruction 2	Instruction 1					
3	Instruction 3	Instruction 2	Instruction 1				
4		Instruction 3	Instruction 2	Instruction 1			
5			Instruction 3	Instruction 1	Instruction 2		
6				Instruction 3	Instruction 2	Instruction 1	
7				Instruction 3		Instruction 2	Instruction 1
8						Instruction 3	Instruction 2
9							Instruction 3

Abbildung 5.23 Fall 4: Parallel Superscalar Execution Pipelines

Der Fall 4 entspricht dem Fall 3 mit einem Stage, der zwei Clock Cycles braucht. In diesem Fall gehen keine Zyklen verloren und für k Ausführungschritte (Execution Stages) und n Befehle (Instructions) werden (k + n) Prozessorzyklen benötigt.

5.5.5 Processor Based Branch Prediction

Branch Prediction ist eine wichtige Eigenschaft moderner Prozessorarchitekturen. Die Voraussage, welche Programmverzweigungen als nächstes zu erwarten sind und der schnelle Zugriff auf die entsprechenden Befehle beeinflussen die Rechenleistung eines Prozessors in entscheidendem Masse. Branch Prediction lässt dich auf zwei Grundproblematiken zurückführen:

- *Branch Direction Prediction:* Die Voraussage, welche Verzweigung die wahrscheinlichste ist.
- *Branch Target Buffering:* Das Zwischenspeichern einer Befehlsfolge innerhalb einer einmal gefundenen Verzweigung.

Es wird zwischen statischer Branch Prediction und dynamischer Branch Prediction unterschieden. Static Branch Prediction eignet sich vor allem für Programme, die sehr viele Schleifen verwenden. Dynamic Branch Prediction eignet sich für Programme, die wenige Schleifen, jedoch viele "Conditional Branches" (if, case) verwenden.

5.5.5.1 Static Branch Prediction

Static Branch Prediction wird vor allem im Compilerbau (siehe Kapitel 4) verwendet und eignen sich für Schleifen sehr gut.

Folgende Methoden sind gebräuchlich [Smith 1998]:

- *All Branches Taken:* Die Voraussage geht davon aus, dass alle Branches durchlaufen werden. Da sämtliche Branches einer Schlaufe immer durchlaufen werden, ist eine Erfolgsrate von 50% zu erwarten.
- *Same Branches Execution Taken:* Diese Voraussage geht davon aus, dass bei konditionalen Verzweigungen immer dieselbe Variante durchlaufen wird, die bereits beim letzten Durchlauf gewählt wurde (LRU - Last Recently Used).
- *Backward Branch Only:* Es wird davon ausgegangen, dass Schlaufen meistens mit Rücksprungadressen beendet werden.
- *Most Not Taken:* Es wird eine Tabelle mit denjenigen Branchadressen geführt, die am meisten nicht durchlaufen worden sind.

Die komplexeren Algorithmen für Static Branch Prediction arbeiten mit einer Klassierung der verschiedenen Branches. Diese Klassierung wird dann bewertet und je nach Resultat wird zunächst die am besten Bewertete Variante durchlaufen. Diese Bewertungen arbeiten mit dem Ausschlussverfahren und schliessen Branches mit folgenden Eigenschaften aus:

- Pointervergleiche
- Exception Handlers
- Methoden- und Funktionsaufrufe

5.5.5.2 Dynamic Branch Prediction

Heute werden im Prozessorbau lediglich Dynamic Branch Prediction Algorithmen verwendet, da die statischen Voraussagen bereits durch Compiler optimiert worden sind. Folgende Methoden werden verwendet [Yoon 2005)]

- *Bimodal Branch Prediction:* Die Voraussage nutzt die bimodale Verteilung des Verhaltens von Verzweigungen. Die Voraussage wird Aufgrund der letzten paar Verzweigungen gemacht und entsprechend in einer Tabelle gehalten auf die die Control Unit eines Prozessors zugreift.
- *Local Branch Prediction:* Die Vergangenheit jeder Verzweigung wird unabhängig bewertet. Die Richtung, die jede einzelne Verzweigung in der Vergangenheit durchlaufen hat, wird als die wahrscheinlichste angesehen.
- *Global Branch Prediction:* Diese Voraussage kombiniert die History der einzelnen Verzweigungen mit der History aller Verzweigungen und wählt dann die wahrscheinlichste Variante als Voraussage aus.
- *Combined Branch Prediction:* In diesem Fall werden verschiedene Methoden kombiniert und schrittweise angepasst. Typischerweise werden in einem ersten Schritt die letzten k durchlaufenen Verzweigungen verwendet. In einem zweiten Schritt werden dann die letzten n Instanzen einer bestimmten Verzweigungsart betrachtet.

5.6 CISC Prozessoren

CISC (Complex Instruction Set Computer) Rechner Architekturen gehen von knappen Systemressourcen wie Speicher und andere Hardware aus.

Typische Merkmale von CISC:

- *Long Instructions:* Befehle unterschiedlicher Länge von 1 - 6 Byte werden verwendet. Da der Zugriff auf den Speicher byteweise erfolgt und Speicher knapp ist, wird in einem Schritt auf das Memory zugegriffen. Je nach Opcode werden weitere Bytes nachgeladen werden, um den Befehl zu komplettieren.
- *Complex Instruction Set:* Um die Arbeit des Assemblerprogrammierers zu erleichtern, werden viele verschiedene Maschinen-Befehle eingeführt. Sie erlaubten das Schreiben von kompaktem Code. Die Folge sind komplizierte Prozessor Architekturen, um die Befehle abzuarbeiten, die viele Taktzyklen brauchten.
- *Direct Memory Operations:* Da die Zahl der Register begrenzt ist (8 - 16 Register), werden viele Operationen in den Speicher ausgelagert. Maschinenbefehle unterstützten diese Auslagerung: Speicher: Speicherplatz 1 "Operation (load, move, delete) Speicherplatz 2.

- *Complex Addressing:* Die kurzen Wortlängen der CPU (z. B. 16-Bit Adressregister) bedeutet, dass die Adressierung von mehr als 64 k Byte in mehreren Schritten erfolgen muss.

Der heute verbreitete CISC Prozessor sind der Pentium III von Intel.

5.6.1 Der 8086 Prozessor

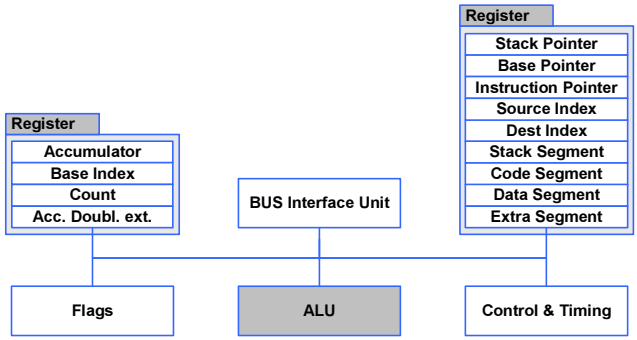

Abbildung 5.24 Intel 8086 Architektur

Die CISC Architektur des 8086 ist zum heute noch verwendeten 8080 auf Ebene "Instruction Set Architecture" kompatibel und wird in vielen so genannten „Embedded Systems“ eingesetzt. Der Aufbau des Prozessors ist in **Abbildung 5.24** dargestellt. Der 8086 hat 40 arithmetische und logische Befehle, 43 Steuerbefehle, 19 Datenübertragungsbefehle und 25 Zeichenkettenbefehle. Das Instruction Set des Prozessors umfasst 127 Instructions. Der 8086 besitzt 7 verschiedene Befehlsformate.

5.6.2 Der 80386 Prozessor

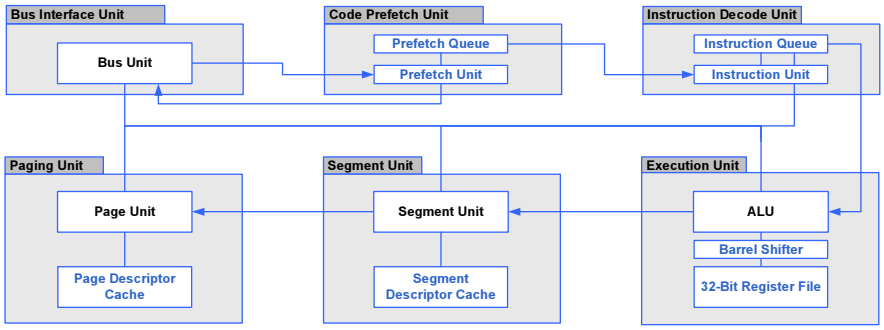

Abbildung 5.25 Intel 80386 Blockschema

Der Intel 80386 Prozessor war eine wichtige Weiterentwicklung des 8086. Das Blockschema des Prozessors ist in **Abbildung 5.25** skizziert.

Zentrales Merkmal des 80386 Rechners ist die differenziertere Adressierung des Speichers über zwei Cache-Speicher, die die Deskriptoren für Segmente und Seiten (Paging) vorhalten.

5.6.3 Die Intel Itanium 2 Architektur

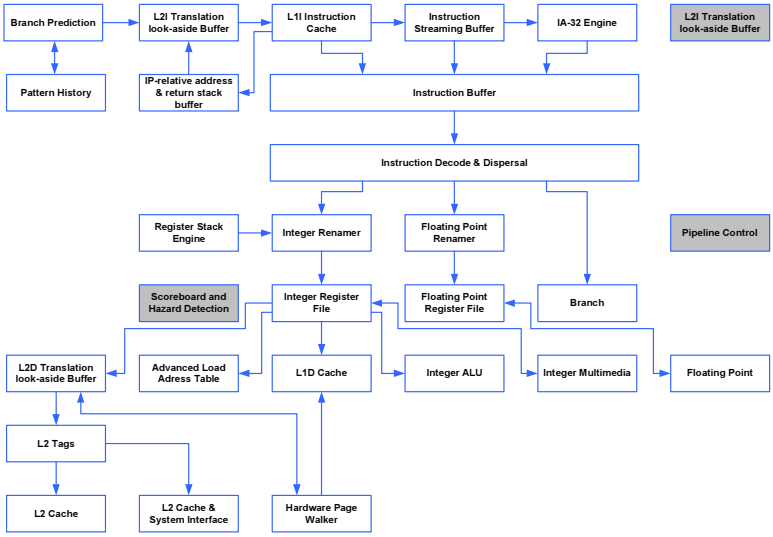

Abbildung 5.26 Intel Itanium 2 Systemarchitektur

Im Jahr 2002 hat Intel den ersten Prozessor mit der neuen Itanium 2 Architektur produziert (**Abbildung 5.26**). Diese Architektur wurde aufgrund einer Zusammenarbeit der Entwicklungslabors von Intel und Hewlett-Packard definiert. Die Prozessoren dieser Familie werden sowohl für Hochleistungs- PCs als auch für HP Workstations und Supercomputer verwendet [McNairy, Soltis 2003].

Zentrale Eigenschaften dieser Prozessoren sind:

- *Nine-stage Execution Pipeline:* Statt 6 (IA-32) werden neu Neun Schritte für die Abarbeitung eines Befehls benötigt. Darstellung in **Abbildung 5.27**.
- *Superscalar Processor:* Gleichzeitig können Sechs Befehle abgearbeitet werden.

- *Memory Management:* Der Zugriff auf das Memory erfolgt über eine drei-stufige Cache Architektur.
- *Branch Prediction:* Es wird eine Combined Branch Prediction mit zwei Levels eingesetzt.

5.6.4 RISC Elemente

In einem gewissen Sinne kann die Intel Itanium Prozessorfamillie auch zu den RISC (Reduced Instruction Set Computer) Rechnern gezählt werden. Typische RISC-Merkmale sind in der Architektur zu finden.

Abbildung 5.27 Intel Itanium 2 Nine Stage Pipeline

- *IPG:* Instruction Pointer Generation and Fetch
- *ROT:* Instruction Rotation
- *EXP:* Instruction Template Decode, Expand, and Disperse
- *REN:* Rename (for register stack and rotating registers) and decode
- *REG:* Register File Read
- *FP1 EXE:* Floating-Point Pipe Stage 1 - ALU execution
- *FP2 DET:* Floating-Point Pipe Stage 2 - Exception Detection
- *FP3 WRB:* Floating-Point Pipe Stage 3 - Write Back
- *FP4:* Floating-Point Pipe Stage 4

5.7 RISC Prozessoren

RISC (Reduced Instruction Set Computer) vereinfachen den Befehlssatz eines Prozessors durch Vereinheitlichung. Diese Vereinheitlichung ist aufgrund der höheren Integrationsdichte durch VLSI Chip Technologie möglich. Diese Technologie erlaubt die Verlagerung der Komplexität weg von der Programmiersprache (Assembler) hin zum Chip selbst.

Beschleunigung wird vor allem durch folgende Techniken erreicht:

- *Data Transport:* Reduktion des Datentransports durch Reduktion der Zugriffe auf den Hauptspeicher. Der Ausbau der Register und die Realisierung einer Load-Store Architektur minimieren die Memory Access Rate.
- *One Step Memory Access:* Zugriff auf den Hauptspeicher mit einem Befehl anstatt ein byteweiser Zugriff.

- *Specialized Arithmetic Components:* Spezialisierte Komponenten für die Abarbeitung arithmetischer Befehle
- *Current Window Pointer:* Der CWP unterstütz Methoden- und Prozeduraufrufe durch ein Registerband statt durch eine fixe Anzahl von Register mit einer fixen Grösse.

Kennzeichen der RISC Prozessor Architektur

- Befehle gleicher Länge (meist 32 Bit)
- Abarbeiten mit gleicher Taktzahl: erlaubt Befehlspipelines
- Eingeschränkter Befehlssatz (32 - 128 Befehle)
- Explizite Lade/Speicher-Befehle
- Registerband mit Registerfenstern

5.7.1 Befehlsverteilung in CISC – Argumente für RISC

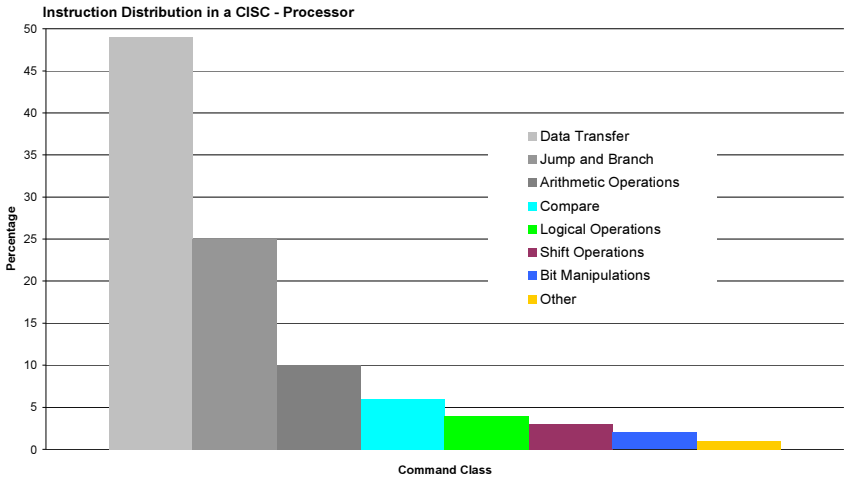

Abbildung 5.28 Verteilung der Befehlsarten in einem CISC Prozessor

Die Verteilung der Befehlsklassen in einem CISC Prozessor sind ein wichtiges Argument für die Realisierung von RISC Architekturen als Alternative. Die Statistik der Befehlsklassen lassen darauf schliessen, dass lediglich eine sehr kleiner Teil des oft sehr umfangreichen Befehlssatzes eines CISC Prozessors überhaupt benötigt werden (**Abbildung 5.28**).

Tabelle 5.1 Die Verteilung der Befehlsklassen in Zahlen

Instruction Distribution	Percentage
Data Transfer	49

Jump and Branch	25
Arithmetic Operations	10
Compare	6
Logical Operations	4
Shift Operations	3
Bit Manipulations	2
Other	1

5.7.2 Die Sparc Prozessoren

Die Sparc Prozessoren gelten als klassische RISK Realisierung. Sun Microsystems hat 1988 den ersten RISC Prozessor auf dem Markt eingeführt. Der erste Sparc Prozessor verwendet lediglich eine "four-stage execution pipeline", wie in **Abbildung 5.29** dargestellt.

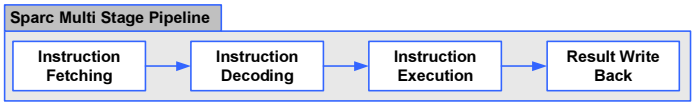

Abbildung 5.29 Sparc Four Stage Pipeline

- Instruction Fetching: Befehl holen
- Instruction Decoding and Operand Fetching: Befehl dekodieren.
- Instruction Execution: Befehl Ausführen
- Result Write-Back: Resultat zurück schreiben

In Prozeduren können 8 Aufrufparameter, 8 lokale und 8 globale Variablen verwendet werden. Der Sun Microsystems SPARC 10 Computer war ein großer Erfolg. Durch die einfache interne Struktur konnte die Maschine erheblich schneller arbeiten als entsprechende CISC-Rechner. Allerdings wurde der vom Compiler zu erzeugende Code länger, da mehr einfache Befehle zu verarbeiten waren.

5.7.2.1 Der UltraSPARC-I Prozessor

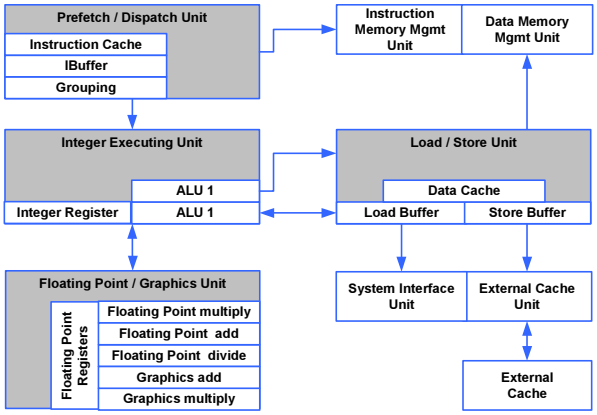

Abbildung 5.30 UltraSPARC-I Blockdiagramm

Der UltraSPARC-I wurde von SUN Microsystems 1995 als erster 64-bit Superscalar RISC Prozessor angekündigt (**Abbildung 5.30**). Er besteht aus 3 zentralen Komponenten (Prefetch / Dispatch Unit, Integer Executing Unit, Floating Point / Graphics Unit) und einer Reihe von Hilfsmechanismen [Greenley 1995].

- *Prefetch / Dispatch Unit:* Diese Komponente ist zuständig für das Laden und Dekodieren von Befehlen. Die Komponente arbeitet mit einem Instruction Cache mit 4 Plätzen, einem Buffer für die Decodierten Befehle (Ibuffer) und einem Groupingmechanismus, der internen Instruktionen nach Instruktionsart zusammenfasst.
- *Integer Executing Unit:* Sämtliche Integeroperationen werden in dieser Komponente verarbeitet.
- *Floating Point / Graphics Unit:* Die Graphic und Floating Point Operationen, sowie die die entsprechende Register sind die wesentlichen Bestandteile der FGU.

5.7.2.2 Der UltraSPARC-III Prozessor

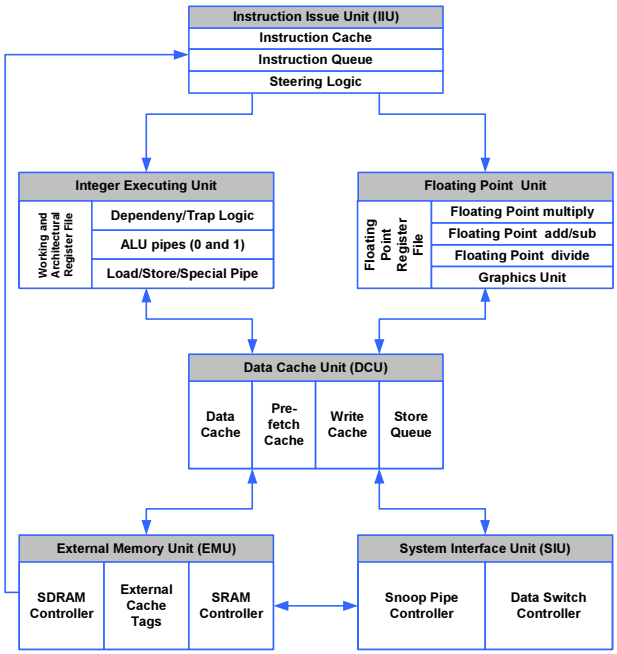

Abbildung 5.31 UltraSPARC-III Blockdiagramm

Die ursprüngliche SPARC Architektur ist in den nachfolgenden Prozessormodellen lediglich erweiter worden (**Abbildung 5.31**). Die Instruction Execution Pipeline des UltraSPARC-III Prozessor [Horel, Lauterbach 1999]:

- *Address Generation:* Generate instruction fetch addresses, generate predecoded instruction bits on cache fill
- *Instruction Prefetch:* Fetch first cycle of instructions from cache; access first cycle of branch prediction
- *Instruction Fetch:* Fetch second cycle of instructions from cache; access second cycle of branch prediction - translate virtual-to-physical address
- *Branch Target Calculation:* Calculate branch target addresses; decode first cycle of instructions
- *Instruction Decode:* Decode second cycle of instructions; enqueue instructions into the queue
- *Instruction Steer:* Steer instructions to execution units
- *Register File Read:* Read integer register file operands; check operand dependencies

- *Integer Execute:* Execute integers for arithmetic, logical, and shift instructions; read, and check dependency of, first cycle of data cache access floating-point register file
- *Data Cache Access:* Access second cycle of data cache, and forward load data for word and double-word loads - execute first cycle of floating-point instructions
- *Memory Bypass:* Load data alignment for half-word and byte loads; execute second cycle of floating-point in-structions
- *Working Register File Write:* Write speculative integer register file; execute third cycle of floating-point instructions
- *Pipe Extent:* Extend integer pipeline for precise floating-point traps; execute fourth cycle of floating-point instruc-tions
- *Trap:* Report traps
- *Done:* Write architectural register file

5.8 Multicore Prozessoren

Multicore Prozessoren sind Prozessoren, die mehr als einen vollständigen Prozessor auf einem einzigen Chip enthalten. Neben der Erhöhung der Taktfrequenz sind sie eine wirtschaftliche Möglichkeit, Performancesteigerungen zu erreichen, ohne ein neues Chipdesign zu entwerfen.

Die Idee, ganze oder aber wichtige Bestandteile eines Prozessors auf einem Chip zu replizieren, kann auch mit anderen Mitteln erreicht werden. So hat beispielsweise Intel das so genannte "Hyper-Threading Technology" entwickelt, was nichts anderes als eine Replikation der Prozessorbestandteile Register, Stackpointer, Programcounter und Instructionpointer ist. Diese Replikationen werden "Siblings" genannt. Diese Technik wird auch "Simultaneous Multithreading" genannt. Sie sieht minimal eine mehrfache Ausführung von Pipelines und Registern vor.

Die klassische Multicore Realisierung ist der Power4 Prozessor von IBM, der im Jahre 2001 auf dem Markt gekommen ist. Die meisten grossen Chiphersteller wie IBM, Intel, SUN und AMD produzieren heute Multicore Prozessoren. Der heute wohl prominenteste Prozessor ist der von IBM, Sony, and Toshiba entwickelte Cell Prozessor der Sony Playstation 3. Er leistet in der Playstation 216 GFlops Peak Performance (Die Performance eines konventionellen PC ist unter 10 GFlops).

5.9 Fragen zum Kapitel

Nr	Frage
1	Sie sieht der Instruction Execution Lifecycle aus?
2	Was ist der Zweck des Erlangen Classification Systems?
3	Auf welcher Prozessortechnologie basiert der schnellste Rechner der Welt?
4	Erklären Sie den Unterschied zwischen CISC und RISC.
5	Was ist ein Superscalar Processor?
6	Was ist der Sinn von Multistage Pipelines?
7	Was ist eine virtuelle Maschine?
8	Was ist der Nexus?
9	Welche Klassen umfasst Flynn's Taxonomie?
10	Was ist der Unterschied zwischen den RAM, dem PRAM und dem VRAM Modell?

5.10 Übung zum Kapitel

Nachfolgend finden Sie den Artikel „Introduction to the Cell Multiprocessor“ von verschiedenen Autoren aus den IBM Systems Journal July / September 2005.

Im Artikel werden die Designziele der Prozessorarchitektur formuliert. Fassen sie auf einer 1/2 Seite zusammen, wie diese Ziele erreicht worden sind.