Das Betriebssystem z/OS und die zSeries

Die Darstellung eines modernen Großrechnersystems

Von
Michael Teuffel und Robert Vaupel

Oldenbourg Verlag München Wien

Bibliografische Information Der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

Rosenheimer Straße 145, D-81671 München
Telefon: (089) 45051-0
www.oldenbourg-verlag.de

Lektorat: Margit Roth
Herstellung: Rainer Hartl
Umschlagkonzeption: Kraxenberger Kommunikationshaus, München
Gedruckt auf säure- und chlorfreiem Papier
Druck: R. Oldenbourg Graphische Betriebe Druckerei GmbH

ISBN 3-486-27528-3

Vorwort

Die zSeries und das Betriebssystem z/OS stehen synonym für den Begriff Mainframe. Um einen Überblick über z/OS zu geben, ist es notwendig, die geschichtliche Entwicklung des Großrechners zu beleuchten, den momentanen technologischen Entwicklungsstand zu erläutern und einen Ausblick auf die zukünftige Entwicklung zu geben. Wichtig dabei ist es jedoch, vorab zu verstehen, wo und wie diese Rechnersysteme eingesetzt werden. Damit ergeben sich folgende Fragestellungen, die neben der reinen Technologie behandelt werden sollen:

- Wo und wie werden zSeries-Rechner eingesetzt?
- Wie haben sich diese Rechner entwickelt?
- Worauf legen Benutzer Wert, die diese Rechnersysteme einsetzen?
- Worin unterscheiden sich zSeries-Rechner von anderen Rechnerarchitekturen?
- Welche Besonderheiten zeichnet die zSeries insbesondere mit dem z/OS Betriebssystem aus?
- Welche Entwicklung zeichnet sich für die zSeries und für z/OS ab?

Das Ziel diese Buches ist nicht nur, einen geschichtlichen Überblick zu geben, sondern die Technologien zu beleuchten, die die zSeries und das Betriebssystem z/OS ausmachen, und einige wesentliche technologische Eigenschaften detailliert zu beschreiben. Daraus ergibt sich folgende Gliederung:

- *Ein Blick zurück - Wie es angefangen hat*

  In diesem Kapitel wird die Geschichte der Rechner dargestellt - die mechanischen Rechner von Schickard und Leibniz, der Beginn moderner Datenverarbeitung mit Hollerith, Zuse und Aiken, sowie die Entwicklung der IBM Rechner von der S/360-Architektur bis zur Gegenwart.

- *Die z/Architektur und die zSeries*

  Es ist interessant, den Rechner sowohl mit einem Blick in seine inneren Strukturen wie auch aus der Sicht des Betriebssystems und der Anwendungen kennenzulernen.

- *Die I/O-Architektur der zSeries*

  Die Daten für die Datenverarbeitung - ein komplexes, aber höchst effektives System von Platten, Kanälen und Steuereinheiten wird hier vorgestellt.

- *z/OS - Das Betriebssystem*

  Mit einer Tradition von fast 40 Jahren ist z/OS ein top-aktuelles Betriebssystem, das alle Anforderungen eines modernen und leistungsfähigen Großrechnersystems erfüllt.

- *Die Parallel Sysplex-Architektur*

  Der Parallel Sysplex verbindet zusammen mit der Coupling Facility viele Rechner zu einer Einheit, die höchsten Ansprüchen komplexer Installationen genügt.

- *z/OS Workload Management*

  Die Zielsetzung der Benutzer ist die Basis für den Workload Manager, um die vorhandenen Ressourcen eines Systems optimal zum Einsatz zu bringen.

- *Intelligent Resource Director*

  Um ein großes System für verschiedene Anwendungsbereiche flexibel aufteilen zu können, bietet der Intelligent Resource Director die notwendigen Funktionen an.

- *Performance - Ein Problem der Datenverarbeitung*

  Trotz aller Optimierung der Hardware- und Software-Kapazitäten kommt ein Leistungsverhalten des Systems, so wie es seine Benutzer erwarten, nicht von selbst. Wie Performance-Charakteristiken definiert und erreicht werden können, wird in diesem Zusammenhang beschrieben.

- *Anwendungsintegration unter z/OS*

  Ein Optimum an Leistung kann nur erreicht werden, wenn die Anwendungen effektiv mit dem Betriebssystem zusammenpassen. SAP R/3 und WebSphere sind dafür typische Beispiele in der heutigen Großrechnerwelt.

Zusammenfassend kann gesagt werden, daß z/OS zusammen mit der z/Architektur beste Voraussetzungen für die Datenverarbeitung sowohl heutzutage wie für die Zukunft bietet.

Begriffe und verwendete Namen

Das Thema dieses Buches sind die Rechner der zSeries als Implementierung der z/Architektur und das Betriebssystem z/OS. Sie sind die Ergebnisse der neuesten Entwicklungen, die auf dem System S/390 und dem Betriebssystem OS/390 basieren. Bei allen wichtigen neuen Aspekten und Fähigkeiten gibt es auch noch viele Gemeinsamkeiten, speziell im I/O-Bereich, und deshalb tauchen die Begriffe S/390 und OS/390 in den Ausführungen dort noch auf, wo es keine signifikanten Unterschiede gibt.

Die Rechner der zSeries, auf denen z/OS läuft, haben die Bezeichnungen z900 und z990, sie sind die Nachfolger des Parallel Enterprise Servers S/390 G5 und G6. MVS, OS/390 und z/OS sind jeweils aufeinander folgende Generationen desselben Betriebssystems, ähnlich wie beispielsweise Windows NT und Windows 2000.

Eine Bemerkung zur Terminologie: mit den Buchstaben K, M, G und T wird ein Vielfaches von 2 bezeichnet:

K (Kilo)	$1024 = 2^{10}$	G (Giga)	$1073741824 = 2^{30}$
M (Mega)	$1048576 = 2^{20}$	T (Tera)	$1099511627776 = 2^{40}$

Obwohl diese Bezeichnungen aus dem Dezimalsystem übernommen sind und dort die Bedeutung Kilo (10^3), Mega (10^6), Giga (10^9) und Tera (10^{12}) haben, stehen sie hier für Vielfache von 2, die dem entsprechenden Dezimalwert am nächsten kommen. Damit hat dann beispielsweise MB die Bedeutung Megabytes.

Wirtschaftliche Bedeutung der zSeries

Eine im Jahr 2000 von der International Technology Group durchgeführten Studie, die sich mit dem weltweiten Transaktionsaufkommen von Großunternehmen im Finanzsektor beschäftigt, kommt zu dem Ergebnis:

- 95% der weltweit 2000 größten Unternehmen setzen OS/390 für ihren zentralen Server ein. Insgesamt 20 000 Unternehmen verfügen über einen S/390 Rechner.
- Zwischen 65% und 70% aller geschäftsrelevanten Daten werden im EBCDIC-Format auf S/390 Rechnern gespeichert.
- 60% aller geschäftsrelevanten Daten, auf die mittels des World Wide Web zugegriffen werden kann, sind in Mainframe-Datenbanken gespeichert, hauptsächlich DB2, IMS und VSAM.

Daraus kann man schließen, daß nahezu alle deutschen Großunternehmen Rechner der S/390- und inzwischen der z/Architektur mit dem OS/390- oder z/OS-Betriebssystem einsetzen. Dies ist in der Tat der Fall. Während vor etwa 15 Jahren in Großunternehmen neben Rechnern zur Steuerung von Prozessen fast ausschließlich Rechner der S/370-Architektur eingesetzt wurden, werden seit der Einführung des Internets eine Vielzahl von unterschiedlichen Rechnern eingesetzt. Während unternehmenskritische Daten nach wie vor ausschließlich auf S/390-Rechnern liegen, werden andere Rechnerarchitekturen beispielsweise für die Nutzung des Internets eingesetzt. Eine wesentliche Aufgabe, die sich daraus ergibt, besteht in der Anbindung von Internet-Technologien in die /390-Landschaft: WebBrowser, WebServer, Java Applets, CORBA, Enterprise JavaBeans, Business Objekte und Java Frameworks. Praktisch alle größeren Unternehmen beschäftigen sich mit dieser Thematik.

Anforderungen der Benutzer

Wir werden im geschichtlichen Überblick der z/Architektur sehen, daß die Benutzeranforderungen ursprünglich hauptsächlich auf kompatible Systeme ausgerichtet waren. Damit verbunden war der Druck auf größere Rechnersysteme, um die steigenden Anforderungen einer zentralen IT[1] zu befriedigen.

Seit Mitte der 80er Jahre und verstärkt in den 90er Jahren liegen die Anforderungen der IT in:

Verfügbarkeit: Nachdem Rechner das zentrale Nervensystem der IT bilden, besteht die Notwendigkeit, sie für 24 Stunden an 7 Tagen der Woche in Betrieb zu halten.

Ausfallsicherheit: Heutzutage können ungeplante Ausfälle das *Business* von Firmen in Gefahr bringen. Es wird geschätzt, daß ein eintägiger Ausfall der zentralen IT einer Großbank einen Verlust in Milliardenhöhe beschert. Ein Ausfall von drei Tagen oder mehr kann zum Zusammenbruch von Firmen führen.

Skalierbarkeit: Für Unternehmen ist die Möglichkeit, die Leistung ihrer Rechnerinfrastruktur inkrementell zu erweitern, ein wesentlicher Faktor zur Sicherung von Investitionen.

Sicherheit: Letztendlich ist die Sicherheit des zentralen Nervensystems von Unternehmen ein Faktor, der nicht weiter erläutert werden muß.

[1] IT = Informationstechnologie, im Zusammenhang mit Firmen-IT meint man damit häufig das *Glasshouse*.

Managebarkeit: Rückt seit Mitte der 90er Jahre ins Zentrum der IT. Dies trägt der Tatsache Rechnung, daß heutige ITs eine Vielzahl von unterschiedlichen Rechnersystemen betreuen müssen und dies im Prinzip mit immer weniger Leuten.

In den folgenden Kapiteln werden wir sehen, wie die oben genannten Aspekte in der Realisierung der Hard- und Software gewährleistet werden. Im Mittelpunkt stehen dabei die Aspekte: Verfügbarkeit, Ausfallsicherheit, Skalierbarkeit und Managebarkeit.

Kundeninvestitionen

Das folgende Beispiel weist auf die enormen Investitionen in Anwendungen hin, die auf der S/390-Architektur laufen, die damit eine Motivation für die technologische Darstellung in den folgenden Kapiteln bieten:

- 16 000 Unternehmen weltweit (darunter 490 der Fortune 500 Companies) setzen CICS als Transaktionssystem ein.
- Es sind 30 Millionen CICS-Terminals installiert, auf denen täglich 20 Milliarden Transaktionen ausgeführt werden. Das ist mehr, als das World Wide Web im gleichen Zeitraum an Hits erzeugt.
- Mit CICS-Transaktionen werden täglich 64 Billionen (10^{12}) $ transferiert oder abgerechnet.
- Eine Überschlagsrechnung mit den folgenden Annahmen ergibt:
  1. 20 000 S/390-Server haben durchschnittlich 1 Million Zeilen aktiven Anwendungscode (LOC[2]) (zwischen 200 000 und 50 Millionen pro Server).
  2. Das sind kumulativ 20 Milliarden LOC.
  3. Mit einer Produktivität von 2 000 LOC/Mannjahr ergibt das eine Investition von 10 Millionen Mannjahren.
  4. Rechnet man mit 100 000 $ /Mannjahr, so summiert sich das zu einer Investition von 1 Billion $ in S/390-Anwendungssoftware.
- Zum Vergleich: das Bruttosozialprodukt der USA betrug 1999 9 Billionen $.

Das Betriebssystem z/OS und die zSeries

Die einleitenden Bemerkungen sollten einen Einstieg in die vor dem Leser liegende Thematik geben. Wenn es vor einigen Jahren im Überschwang der PC-Euphorie noch hieß: *Die Dinosaurier sind tot* - so hat es sich zwischenzeitlich deutlich gezeigt, daß genau das Gegenteil richtig ist. Die Großen und die Kleinen (wobei diese längst leistungsfähiger sind, als es Großrechner noch vor einigen Jahren waren) gehören zusammen und bieten ihren Benutzern unter Ausnutzung der neuesten Technologien das Umfeld, das zu einer optimalen Datenverarbeitung notwendig ist.

Allen Lesern, seien es PC-Freaks, die sich in Neuland wagen, oder Systemprogrammierer im z/OS-Umfeld, wünschen wir einen guten Einstieg in diese komplexe Materie.

Robert Vaupel Michael Teuffel

[2] Zeilen Anwendungscode = Lines of Code = LOC

INHALTSVERZEICHNIS

1.0 Ein Blick zurück - Wie es angefangen hat

- **Mit dem Abakus hat es angefangen**
- **Hollerith und Zuse als Pioniere der Datenverarbeitung**
- **System /360 ist die Basis**
- **Der rasante Fortschritt bis zur z/Architektur**

Ein Blick zurück - man muß schon sehr weit zurückblicken, wenn man sich die Geschichte der Rechenmaschinen ansehen will[3]. Rechenmaschinen, Rechenautomaten, so wurden Computer ja sehr lange bezeichnet, obwohl diese Wunderwerke der Technik heutzutage natürlich sehr viel mehr als nur rechnen können. Aber mit dem Rechnen fing alles an.

1.1 Von null auf 1900

Der Abakus - Vor Jahrtausenden erfunden

Die vermutlich erste Rechenmaschine war der *Abakus*. Unter dem Abakus verstehen wir heute den in östlichen und fernöstlichen Ländern noch weit verbreiteten Holzrahmen mit den darin senkrecht eingebauten Stäben, an denen durchbohrte Kugeln auf- und abgeschoben werden können. Damit unterscheidet er sich allerdings wesentlich von dem, was man später als Rechenmaschine bezeichnete, da der Benutzer selbst aktiv den Vorgang durchführen muß, der zu einem Ergebnis führt. Deshalb sollte man eher von einer Rechenhilfe sprechen.

Da der Abakus ein sehr altes Gerät ist, das wohl schon seit Jahrtausenden in unterschiedlichen Formen in Gebrauch ist, fällt es schwer, seinen genauen Ursprung festzulegen. Er ist aber, wenn auch in unterschiedlichen Ausführungen, immer noch in Benutzung. In Japan beispielsweise lernt man den Umgang mit ihm in der Grundschule als Pflichtfach, und er ist aus dem modernen japanischen Leben nicht wegzudenken.

Schickard entwickelt das Zählrad

Die erste urkundlich nachweisbare Rechenmaschine der Welt erfand Wilhelm Schickard, Schwabe und Professor für biblische Sprachen in Tübingen, 1623 mit seiner *Rechenuhr*. Dazu verwertete er das Prinzip der beweglichen Rechenstäbe Napiers, mit dem er über die Logarithmen korrespondiert hatte. Die Rechenuhr ging in den Kriegswirren des 30jährigen Kriegs verloren, aber sie konnte nach Schickards überlieferten Notizen und Skizzen für seinen Mechaniker rekonstruiert werden.

Sie war für die vier Grundrechenarten geeignet. Ihr dekadisches Zählrad, Standardteil aller folgenden mechanischen Rechenmaschinen, erlaubte zehn Stellungen und drehte

[3] Ein Teil der Informationen ist der Reihe "*Kleine Kulturgeschichte des Rechnens*" aus den IBM Nachrichten (1987-1988) entnommen.

mit jeder Umdrehung über den Übertragszahn und das Zwischenrad das nächste Zählrad einen Schritt weiter, insgesamt über sechs Stellen. Jedem Zählrad war zur Ziffernanzeige eine Ablesetrommel beigefügt. Bewegt wurden die Zählräder mit einem Stichel, nach rechts zum Addieren, nach links zum Subtrahieren.

Schickards grundlegend neue Idee vom Zählrad und der selbsttätigen Zehnerübertragung wurde vom Krieg ausgelöscht. Er selbst starb 43-jährig an der Pest.

Leibniz und die Nachfolger

Anders als Schickard hatte Gottfried Wilhelm Freiherr von Leibniz das Glück, von Krieg, Plünderung und Seuchen unbehelligt, ganz seine Studien leben zu können. Als Philosoph schuf er ein neues, physikalisch begründetes theologisches Weltbild, aber er befasste sich auch mit Rechenmaschinen. Dabei erweiterte er die Idee des Zählrads um die Multiplikation - durch fortgesetzte, gezählte Addition - und bereitete so den Weg für die noch bis zur Mitte des 20. Jahrhunderts aktuellen mechanischen Rechner.

Er stellte 1693 in London das erste Modell seiner Maschine vor und beschreibt sie selbst als aus einem unbeweglichen zwölfstelligen Resultatwerk und einem beweglichen achtstelligen Einstellwerk bestehend. Letzteres war rechts mit einem Rad ausgestattet, das beim Multiplizieren die Umdrehungen zählte, beim Dividieren den Quotienten anzeigte. Links ließ sich mit einer Kurbel das Einstellwerk gegenüber dem Rechenwerk verschieben. Insgesamt wurde die Maschine mit einem in der Mitte liegenden Handrad getrieben.

Die Mechanik der Maschine bereitete ihm allerdings große Schwierigkeiten. Die Ursache lag in den Fertigungstoleranzen, oder genauer gesagt darin, daß man noch nicht verstand, in der Fertigung das Prinzip der Toleranzen anzuwenden. Die Idee war aber geboren.

1709 zeigte Giovanni Polenus in Padua eine Weiterentwicklung, aber auch er hatte große Toleranzprobleme, die ihn sein Werk am Ende selber zerstören ließen. 1727 brachte Antonius Braun in Wien das alte Leibniz-Konzept doch noch zum glatten Funktionieren. 1782 nahm der schwäbische Pfarrer Matthäus Hahn den Serienbau von Rechenmaschinen auf, und 1850 wurde in England das erste Patent für eine tastaturgesteuerte Addiermaschine erteilt.

Was bei alledem auch nach 200 Jahren noch Gültigkeit hatte, waren die Konzeptionen aus dem 17. Jahrhundert: Das nach einer Umdrehung wieder auf null gehende Zählrad, die durchlaufende Zehnerübertragung, die Verzweigung der Eingaben über Wellen und Zahnräder und die Speicherung der Eingaben durch Winkelstellungen und Abstände.

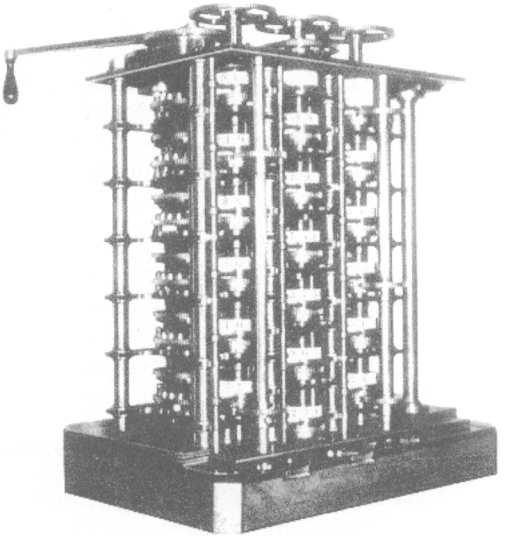
Babbages Absicht schlägt die Brücke ins 20. Jahrhundert

Einen bemerkenswerten Versuch, auf den Grundlagen der vorausgegangenen Konstrukteure einen Rechenautomaten allerdings wesentlich erweiterter Kapazität zu schaffen, unternahm Charles Babbage aus Cambridge in England. Kaum 20 Jahre alt, hatte er einmal Logarithmentafeln zu prüfen, eine Aufgabe, die den jungen Mathematiker ganz selbstverständlich nach Wegen suchen ließ, derartige Arbeiten zu vereinfachen. Die Lösung sah er in der Konzeption einer *difference engine*, die er 1822 präsentierte. Das Aufsehen, das er mit diesem für zwei Differenzen und acht Dezimalstellen ausgelegten Modell erregte, stimulierte ihn zu einem noch anspruchsvolleren Plan - eine Maschine für sieben Differenzen und 20 Stellen. Daß sie nie fertig wurde, kann im Rückblick nicht verwundern: Getriebe von der Komplexität, wie dieses Konzept sie

beanspruchte, würden selbst bei heutigen Fertigungstechniken Schwierigkeiten bereiten.

Aber selbst in seiner Erfolglosigkeit war Babbage nicht ohne Verdienst: In der Erkenntnis, daß die durchlaufende Übertragung nach Schickard, Pascal und Leibniz das Rechnen mit großen Zahlen zu langsam machen würde, konzipierte er den parallelen mechanischen Übertrag, das *anticipatory carry*, das als Prinzip selbst in den Elektronenrechnern des 20. Jahrhunderts eine Rolle spielte.

Elf Jahre nach seiner *difference engine* kündigte Babbage 1833 seine *analytical engine* an, die - wäre sie realisiert worden - der erste digitale Rechenautomat der Geschichte geworden wäre. Es sollte eine problemlösende Maschine werden, die den rechnenden Menschen nachahmt, ja selbst die meisten Funktionen der heutigen Maschinen erfüllt, als Komponenten also arithmetische Einheiten hat, Zahlenspeicher, Steuereinheit für Programm, Rechenoperationen und Datentransport sowie Ein- und Ausgabe. Babbage dachte an einen Speicher für tausend Zahlen zu 50 Stellen, in dem die Ziffern von Hand eingestellt werden. Das Programm sollte mit Jacquard-Karten gesteuert werden. Selbst Programmverzweigungen nach logischen Entscheidungen sah er vor. Je nach dem Vorzeichen der Zwischenergebnisse sollte das Kartenband vor- oder rückwärts laufen, sollte Programmschritte überspringen oder -schleifen wiederholen. Als Zeitbedarf für die Arbeit seiner Maschine hatte Babbage eine Sekunde für Addition und Subtraktion gerechnet, eine Minute für die Multiplikation zweier 50stelliger Zahlen, und ebenfalls eine Minute für die Division einer 100stelligen durch eine 50stellige Zahl.



Babbage - Analytical Engine

Wie bei den früheren Erfindern waren es Toleranzprobleme, die die Maschine am Funktionieren hinderten. Babbage verstarb verbittert 1871. Seine Gedanken ruhten bis weit in das 20. Jahrhundert und wurden erst wieder 1934 in Berlin von dem jungen Bauingenieur Konrad Zuse aufgegriffen.

1.2 Der Beginn der modernen Datenverarbeitung

Hollerith und die Lochkartentechnik

Es war Herman Hollerith, der erstmals die Lochkarte als Informationsspeicher benutzte. Als er 1880 an der amerikanischen Volkszählung mitarbeitete, begann er, um sich dem stumpfsinnigen Auswerten statistischer Angaben zu entziehen, die Angaben zur Person codiert in Karten zu lochen und in einer Zähl- und Registriermaschine auszuwerten. Den Lochpositionen auf den Karten ordnete er beliebige Bedeutung zu, etwa Geburtstag, Bekenntnis oder Geschlecht. Der Kontaktapparat schloß über

Abfühlstifte für jede Lochung einen Kontakt, der die Zählwerke jeweils um einen Schritt weiterbewegte oder die Klappen eines Sortierapparates öffnete, in den die Karten geordnet abgelegt werden konnten. In der darauffolgenden US-Volkszählung 1890 bewährte sich Holleriths Erfindung in unerwartet hohem Maß.

In Deutschland dauerte es bis 1910, bis Hollerith-Maschinen, allerdings inzwischen wesentlich verbesserte Versionen, für die Volkszählung in Berlin eingesetzt wurden. Mit der Zeit verbesserten Zusätze die Anlagen weiter. Zu Locher, Prüfer, Sortierer und Tabellierer traten ein Druckwerk für Listen, Mischer, Summenstanzer, Rechenlocher und anderes. Aber subtrahiert wurde immer noch, wie bei Pascal, durch komplementäre Addition, und multipliziert, wie bei Leibniz, durch fortgesetzte Addition und Stellenverschiebung. Die Verläßlichkeit war allerdings einsame Spitze: Unter zehn Millionen Signalen war höchstens ein gestörtes.

Hollerith - Tabulator und Sortierer

Gesteuert wurde die Lochkartenmaschine von der zunächst festen, später auswechselbaren Schalttafel. Ihre Buchsen führten zu den Steuerelementen der Maschine und wurden mit Schaltschnüren der gewünschten Maschinenfunktion entsprechend untereinander verbunden. Zu den Daten kamen noch weitere Löcher in die Karte, die in der Maschine dann Steuersignale erzeugten. Das Programmieren bestand also aus dem Stecken der Schaltschnüre und dem Stanzen dieser Steuerlöcher.

Zuse entwickelt den programmgesteuerten Rechner

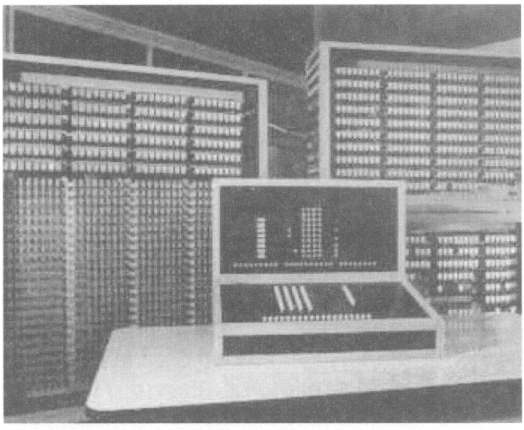
Konrad Zuse griff 1934 die Ideen von Charles Babbage wieder auf, konnte dabei aber der Idee des programmgesteuerten Rechners freilich entscheidend Neues hinzufügen. Er entschied sich erstens für die Darstellung von Zahlen und Befehlen für den Rechner in dualer Form - durch die jeweilige Stellung bistabiler Schalter -, zweitens für die Verwendung halblogarithmischer Zahlen, heute Gleitkomma oder *floating point* genannt, drittens für den Gebrauch logischer Operationen Und, Oder und Negation.

Nun waren duale Zahlen keine Neuigkeit, es gab sie spätestens seit Leibniz. Aber ihre Vorzüge gegenüber dezimalen fielen jetzt erst richtig ins Gewicht: Sie beanspruchten weniger Schaltelemente und vereinfachten Rechenelemente und Speicher, weil sie mit ihren Ziffern 0 und 1 dem An und Aus von Kippschaltern, Relais oder Elektronenröhren auf ganz natürliche Weise entsprechen. Im übrigen reduziert der Gebrauch von nur zwei statt zehn Ziffern ganz erheblich die Anforderungen an die Toleranz, an denen die brillanten Erfinder der Vergangenheit reihenweise gescheitert waren.

Mit seinem programmgesteuerten Rechner war Zuse nun keineswegs zufrieden, denn seine Z1, wie die Maschine später genannt werden sollte, funktionierte insgesamt nicht so, wie er sich das gedacht hatte. Da es in erste Linie die rein mechanischen Schaltelemente waren, die sich als Ursache der Störungen erwiesen, setzte er bei seinem Plan, eine zweite, bessere Maschine zu bauen, auf elektromechanische Bauteile. Der Speicher der Z2 blieb indes mechanisch. Diesmal traten Störungen ganz anderer Art auf: Es gab Krieg und Zuse wurde Soldat. Er wurde aber nach einiger Zeit vom Militärdienst

freigestellt und konnte sich dem Entwurf der Z3 widmen. 1941 war sie soweit, ein Relaisrechner und zugleich der erste programmgesteuerte Rechenautomat der Welt, der wirklich funktionierte, eine Pionierleistung, für die die technisch-wissenschaftliche Welt Konrad Zuse noch heute Anerkennung zollt.

Rechenwerk und Speicher der Z3 bestanden aus insgesamt 2600 Fernmelderelais. Der Speicher faßte 64 Zahlen, jede 22 Dualstellen, also etwa sieben Dezimalstellen lang. Die Zahlen wurden eingetastet, und die Resultate waren von einem Lampenfeld abzulesen. Das Programm war noch starr und somit ohne die Möglichkeit, bedingte Befehle zu erteilen. Es wurde achtspurig in einen Kinofilm eingelocht. Später klassifizierte man die Z3 als *Einadreßmaschine*, weil die Befehle des Programmierers jeweils nur eine Operandenadresse aufnehmen konnten. Die Z3 schaffte 15 bis 20 arithmetische Operationen in der Sekunde und konnte außer addieren, subtrahieren, multiplizieren und dividieren auch, mit eingebauten Unterprogrammen, wurzelziehen und mit bestimmten Faktoren multiplizieren. Für eine Multiplikation brauchte die Z3 zwischen vier und fünf Sekunden.



Zuse - Z3-System

Zuses Maschinen überstanden den Weltkrieg nicht. Es ist lediglich ein Nachbau, der heute im Deutschen Museum in München an die Z3 erinnert.

Aiken geht in den USA ähnliche Wege

Ganz unabhängig von Zuse bahnte sich einige Jahre später in den USA etwas an, was dafür spricht, daß bestimmte Erfindungen, wenn die Zeit reif ist, einfach fällig sind, von wem und wo auch immer. Ein Mathematikprofessor namens Howard Aiken experimentierte dort seit 1939 mit einem vorwiegend aus Teilen der Lochkartentechnik gebildeten Rechenautomaten. Aikens Konzeption war der Zuses nahezu gleich und führte die Babbageschen Ideen weiter, von denen beide freilich erst später erfuhren. Seinen *Automatic Sequence-controlled Computer* ASCC, später Harvard Mark I genannt, übergab Aiken 1944 seiner Universität. Die Mark I war riesig, 15 Meter lang, zweieinhalb Meter hoch, hatte 700 000 Einzelteile, 3 000 Kugellager, 80 000 Meter Leitungsdraht und mehr dekadische Zählräder als je zuvor verwendet wurden. Sie hatte 72 Addierzähler, jeder zu 23 Dezimalstellen, die speicherten und addierten, eine Einheit zum Multiplizieren und Dividieren, Relaisketten als Kurzzeitspeicher und Schaltersätze als Festspeicher. Mark I multiplizierte zwei zehnstellige Zahlen in sechs Sekunden, für eine Division benötigte sie elf Sekunden. Die Daten wurden mit Lochstreifen und -karten eingegeben und über Kartenlocher und Schreibmaschinen ausgegeben.

Die Mark I war eine elektromechanische Maschine, und ihr Programm steckte, dual codiert, in einem 24spurigen Lochstreifen, die ersten acht Spuren für die Von-Adresse, die zweiten acht für die Nach-Adresse, der Rest für den Operationscode. Endlosstreifen für Unterprogramme und später eingefügte weitere Streifenleser erhöhten noch die Brauchbarkeit der Mark I. Aikens folgende Maschinen, Mark II bis IV, rechneten zwar

um einiges schneller, speicherten auch mehr, fügten indes der Anerkennung für Aikens ursprüngliche Leistung und der in jahrelanger Tag- und Nachtarbeit erwiesenen Zuverlässigkeit der Mark I wenig hinzu.

Eckert und Mauchly gelingt der Einstieg in die Elektronik

Den Einstieg in die elektronische Datenverarbeitung vollbrachten Presper Eckert, John Mauchly und deren Mitarbeiter, als sie in der Pennsylvania-Universität 1946 ihren *Electronic Numerical Integrator and Computer* ENIAC fertigstellten. Ihm lag die schon 1919 gewonnene Erkenntnis zugrunde, daß sich zwei Trioden zu einem bistabilen Schaltelement, einem Flip-Flop, schalten lassen. ENIAC war imposant, nach welchem Maßstab auch immer. Er wog 30 Tonnen, füllte das Volumen eines Güterwagens, hatte 18 000 Elektronenröhren, 1 500 Relais und verbrauchte 150 Kilowatt Leistung, genug, um ohne die dazugehörige Ventilation seine Lötstellen weich werden zu lassen. Eckert kompensierte, indem er die Röhren mit nur einem Viertel der Nennleistung belastete und damit die damals sehr niedrige Ausfallrate von zwei bis drei Röhren wöchentlich erreichte.

Statt des Zählrads gab es jetzt zehn Flip-Flops. 20 Akkumulatoren zu je zehn Dezimalstellen addierten und subtrahierten. Besondere Röhrenschaltungen multiplizierten und dividierten. Die Daten transportierte ENIAC durch Kanäle mit je einer Ader für jede Stelle. Die Impulse dazu entsprachen der zu übertragenden Ziffer, für eine 7 also sieben Impulse. Eine elfte Ader übertrug das Vorzeichen.

ENIAC multiplizierte zwei zehnstellige Zahlen in 0.0028 Sekunden. Die Daten wurden mit Lochkarten oder über 300 Drehschalter eingegeben. Programmiert wurde mühselig an Tafeln mit Schaltschnüren, Steckern und Schaltern. Ein *Master programmer* lieferte Impulse an die am jeweiligen Programm beteiligten Maschinenelemente. Bedingte Befehle und Rückwärtsverzweigungen gab es noch nicht.

Abgeschaltet wurde ENIAC im Jahr 1955, nach fast einem Jahrzehnt meist wissenschaftlichen Rechnens. Von seinen auf mehrere Museen verstreuten Komponenten sind vier noch in Philadelphia zu sehen, am Ort ihres Entstehens.

1.3 Aufbruch in die Computertechnik - Die ersten IBM Rechner

Die eigentlichen Anstöße zur Entwicklung großer Rechenanlagen kamen aus dem wissenschaftlichen Bereich. Auch die IBM hatte bereits 1947, ein Jahr nach der Entwicklung des ENIAC, eine erste elektronische Anlage, den *Selective Sequence Electronic Calculator* SSEC herausgebracht, entwickelt von R.R. Seeber, F.E. Hamilton und anderen. Diese Maschine hatte bereits informationsgesteuert aufrufbare Unterprogramme mit Adreßänderung, war also dem gespeicherten Programm mit bedingten Befehlen sehr nahe.

Im Gegensatz zu den meisten Frühentwicklungen richtete sich in der IBM das Interesse bald auf die kommerziellen Anwendungen und die Serienfertigung von Computern. So kam 1948 ein elektronischer Rechenlocher (Typ 604), ein nachgeschalteter Rechner, auf den Markt, der ab 1954 auch in Deutschland in Serie produziert wurde.

Der erste echt speicherprogrammierte Computer der IBM, der in Serie gebaut wurde, hieß 701. Er wurde als duale Parallel-Maschine 1953 erstmalig ausgeliefert. Eine kommerzielle Version, die 702, folgte ein Jahr später. 1955 erschien dann der erste

Magnettrommelrechner der IBM, die 650. Die Serienfertigung der elektronischen Anlagen setzte sich in den nächsten Jahren mit den Modellen 704, 705 und 709 fort.

Ende der 50er Jahre war die Transistortechnik überall so weit, daß sie die Röhrenmaschinen insgesamt ablöste. Der IBM Rechner 7090 wurde zu einer weitverbreiteten Transistor-Großrechenanlage Anfang der 60er Jahre. Vielerorts auf der Welt entstanden um 1960 Transistor-Computer, und im selben Jahr erschien auch der erste integrierte Schaltkreis, ein Transistor-Flip-Flop, auf dem Bauelemente-Markt.

Im Jahr 1959 kündigte IBM das System 1401 für kommerzielle Anwendungen an, es wurde das für seine Zeit erfolgreichste System. Die 1401 hatte einen Magnetkern-Speicher von 4096 Zeichen (dieser war durch eine externe Speichereinheit auf die Größe von 16K (16×1024) Zeichen erweiterbar) und hatte eine variable Wortlänge, die vom Programmierer definiert werden konnte. Die Maschine lief mit einer Zykluszeit von 11,5 Mikrosekunden (0,087 MHz), für eine Addition wurden 300 Mikrosekunden benötigt, für Multiplikation und Division 1960 bzw. 2170 Mikrosekunden.

Die 1401 wurde bei den Kunden in zwei unterschiedlichen Arten eingesetzt: Einerseits als selbständiges System und andererseits als Front/End-System für die 7090. Das heißt, die Geräte zur Dateneingabe (Kartenleser und Bandeinheiten) wurden an die 1401 angeschlossen und dann an die 7090 weitergeleitet, die dadurch in ihrer Verarbeitungsgeschwindigkeit nicht vom Tempo eines Kartenlesers abhängig war. Nach der Verarbeitung wurden die Ergebnisse wieder an eine 1401 übertragen, an die Bandeinheiten, Kartenstanzer und Drucker angeschlossen waren. Hierfür gab es den ersten Kettendrucker (Modell 1403) mit 600 Zeilen pro Minute. Dieser Drucker blieb sehr lange auf dem Markt und war später auch noch Teil der Konfiguration eines Systems /360. Ein Basis-System kostete seinerzeit 146 000 $ (1401 Processing Unit 83 700 $, 1402 Kartenleser/Stanzer 30 000 $ sowie 1403 Drucker 32 900 $).

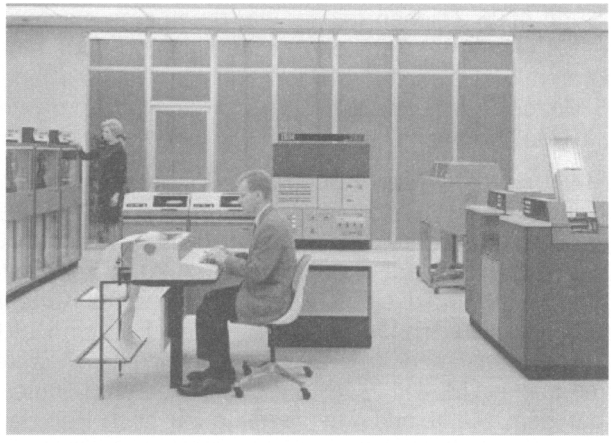
Es gab zu dieser Zeit verschiedene Computer Codes, also Darstellungsmöglichkeiten der Daten im System. Sie hatten so komplizierte Namen wie *Seven-Bit Alphameric Code*, *Two-out-of-Five Fixed Count Code* oder *Bi-quinary Code*. Bei der 1410 wurde der *Standard Binary Coded Decimal Interchange Code* benutzt. Dieser Code war entwickelt worden, um eine Kompatibilität beim Datenaustausch zwischen verschiedenen Systemen herzustellen. Die Struktur dieses Codes besteht aus 64 Bit-Kombinationen, mit denen man 69 verschiedene Zeichen darstellen kann (5 der 64 Kombinationen können in Abhängigkeit von der Druckerkette unterschiedliche Zeichen darstellen - entweder für kommerzielle Anwendungen oder bei der Benutzung der seinerzeit gängigen Programmiersprachen FORTRAN und COBOL). Die Weiterentwicklung dieser Codes ist EBCDIC (*Extended Binary Coded Decimal Interchange Code*) und wird auch heute noch in den Rechnern der zSeries benutzt.

Interessanterweise sei erwähnt, daß die 1401, obwohl seit mehr als 35 Jahren aus den Rechenzentren verschwunden, ihren Anteil am Jahr-2000-Problem hatte. Sie war so populär, daß ihre Programme (in Autocoder Assembler, FORTRAN, RPG oder COBOL geschrieben) noch über Jahrzehnte hinweg in den modernen Computern liefen, auf denen die 1401 emuliert wurde. Jetzt war endgültig die Zeit gekommen, die Programme auf den neuesten Stand zu bringen - oder sie ganz zu ersetzen.

1.4 IBM /360 als richtungsweisende Systemarchitektur

Um 1960 boten die Computer eine schillernde Vielfalt von Typen und Architekturen. Daher faßte man bei der IBM den Plan, in einem konzertierten Einsatz mehrerer Laboratorien der Welt eine neue einheitliche *Systemfamilie* zu entwickeln. Programme sollten innerhalb dieser Familie übertragbar sein. Das Ziel war, dem Kunden ein problemloses Wachstum von System zu System zu ermöglichen. Diese Systemfamilie mit der späteren Bezeichnung IBM /360 sollte für alle Anlagen mit einem gleichen Betriebssystem betrieben werden können, dem dafür entwickelten OS/360.

Am 7. April 1964 kündigte IBM das System /360 weltweit an, es hatte sechs verschiedene Modelle mit unterschiedlicher Leistung, von der S/360-30 mit 64 Kilobyte Hauptspeicher und einem knappen MegaHertz Takt bis zur S/360-75 mit 512 Kilobyte und 5 MegaHertz. Später kamen noch weitere Modelle wie S/360-20 als kleinster und die S/360-91 (eine Entwicklung für das amerikanische Raketenabwehr-System) als leistungsstärkster Rechner hinzu.



IBM S/360-25

Das Ereignis markierte das Ende der Pionierzeit der elektronischen Datenverarbeitung. Zuvor war jedes System eine Neuschöpfung gewesen, mit eigener Systemstruktur, Peripherie und Software. IBM allein hatte sechs unterschiedliche Produktfamilien gepflegt. Ein solcher Wildwuchs sollte nun ein Ende haben, denn bei jedem Systemwechsel mußten viele Anwendungen neu programmiert werden. Mit der Architektur des Systems /360 setzte IBM einen Industriestandard, dem schließlich 80 bis 90 Prozent aller Hersteller folgten. Ein Programm, das 1964 nach den Regeln der S/360-Architektur geschrieben wurde, läuft heute grundsätzlich genauso auf einem zSeries-Rechner. Der Anspruch, alle nur denkbaren Aufgaben mit einer einzigen Rechnerarchitektur zu lösen, wurde durch das Logo ausgedrückt: eine Windrose - die magische Zahl 360 stand für die 360 Winkelgrade des vollen Kreises.

Die S/360-Architektur

Der Begriff *Architektur* war neu im Rechnerbau. Darunter verstand man eine für alle Modelle gültige funktionelle Spezifikation, an die sich die Benutzer halten konnten, während *Design* und *Implementierung*, also die konkrete technische Ausführung, sich der aktuellen Technologie anpassen konnten. Eine klare Trennung von logischer und physischer Struktur war das Ziel.

Eine wesentliche Neuerung bei der Entwicklung der S/360 war die Einführung des *Bytes* als eine 8-Bit alphanumerische Zeichen-Codierung, obwohl es bereits einen 7-Bit ASCII-Code gab. Aber 8 Bit boten die Möglichkeit, 2×4 Bit für die dekadische Codierung zu verwenden und andererseits $2^8 = 256$ Möglichkeiten zur Befehlscodierung zu haben. Ferner wurde die relative Adressierung eingeführt, die sich auf Basisadressen

bezog und damit die Speicherverwaltung vom Programmierer zum Betriebssystem verlagerte. Damit war auch schon der Weg vorgezeichnet für eine leichte Verschiebbarkeit der Programme im Speicherraum, der in der nächsten Generation zur Speicherhierarchie mit virtuellen Speichern führen sollte.

Darüber hinaus wurde das Kanalkonzept durchgehend eingeführt, das heißt die Entkopplung des zentralen Rechners vom peripheren Datenverkehr. Die Maschinenbefehle wurden in den kleineren Modellen als Mikroprogramme in schnellen Festspeichern gespeichert, während sie bei den leistungsstärkeren Rechnern direkt in der Hardware implementiert waren.

Von entscheidender Bedeutung für das gewaltige Vorhaben der Systemfamilie /360 war die Wahl der geeigneten Technologie. Alle Welt diskutierte damals die großen künftigen Chancen der integrierten Halbleiter. Aber über denen schwebte immer noch das Fragezeichen der Ausbeute und der Toleranzen. So entschied sich die IBM für eine hybride Transistortechnologie, *Solid Logic Technology* (SLT) genannt. Diese bestand aus diskreten Transistoren und im klassischen Siebdruck aufgebrachten Leitern und Widerständen und hatte einen wichtigen Vorteil: Ihre Schaltkreise konnten 1964 in automatisierten Verfahren auf ein halbes Prozent genau wirtschaftlich hergestellt werden, während die integrierten Halbleiter-Schaltkreise damals noch Streuungen und Ausbeuten von etwa 20 Prozent aufwiesen, ihre Zeit war noch nicht reif.

Mit der S/360-Architektur wurde, wie bereits erwähnt, das Byte als eine 8-Bit-Codierung eingeführt. Es ist damit im Speicher die kleinste adressierbare Einheit. Die nächst größeren Einheiten sind: Halbwort (2 Bytes), Wort (4 Bytes) und Doppelwort (8 Bytes). Dazu ist allerdings zu vermerken, daß für alle Größen entsprechende Adreßgrenzen (*Boundaries*) beachtet werden müssen. So ist beispielsweise eine Wortgrenze im Speicher jeweils eine durch 4 teilbare Zahl. Das bedeutet, daß man 4 Bytes in den Speicheradressen 100-103 als Wort bezeichnen und in den Instruktionen entsprechend als Wort verarbeiten kann, während 4 Bytes ab Position 101 kein Wort sind.

Spezielle Speicherplätze in Wortlänge sind 16 sogenannte *General Purpose Register*, die eine besondere Bedeutung haben. Sie liegen auch nicht im allgemeinen Speicherbereich, sondern sind Teil des Prozessors und können somit extrem schnell angesprochen werden. Man kann sie als Zähler benutzen und mit ihnen rechnen, oder man kann sie als Basis- oder Index-Register bei der Adressierung verwenden. Auf Grund dieser Eigenschaft war es notwendig, einen speziellen Speicherungsmechanismus zu entwerfen, der den Inhalt der 16 Register bei jedem Task-Wechsel in einer Multi-Tasking-Umgebung so speicherte, daß die einzelnen Programme unabhängig voneinander laufen konnten, ohne sich gegenseitig zu beeinflussen. Denn jedes Programm hatte seinen eigenen Adreßbereich, auf den es über die Register zugreifen konnte, aber es gab nur einen Satz von 16 Registern - also mußten die jeweils bei einer Programm-Unterbrechung gespeichert und für die weitere Ausführung wieder geladen werden.

Es gibt drei Arten der Arithmetik:

- Festkomma (*Fixed Point*) - Festkommazahlen werden als Halbwort oder als Wort dargestellt, haben somit ein Größe von entweder 16 oder 32 Bit. Dabei wird das erste Bit als Vorzeichen interpretiert, so daß die größten Zahlen in einem Halbwort oder Wort die Werte 2^{15} beziehungsweise 2^{31} sind.
- Gleitkomma (*Floating Point*) - Bei den Gleitkommazahlen hat man die beiden Varianten von einfacher und doppelter Genauigkeit *Short / Long Floating Point*, die als Wort oder als Doppelwort gespeichert werden. Auch hier ist jeweils das erste Bit das Vorzeichen, die nächsten 7 Bit enthalten den Exponenten und die

restlichen 24 bzw. 56 Bit stellen die Mantisse dar. Damit kann ein Wertebereich von etwa 10^{-78} bis 10^{76} mit einer Genauigkeit von etwa 7 Dezimalstellen (Short) bis etwa 17 Dezimalstellen (Long) abgedeckt werden. Die Gleitkommazahlen wurden später noch durch die Variante *Extended Floating Point* mit einer 113-Bit-Mantisse und einem 15-Bit-Exponenten in zwei Doppelworten erweitert.

- Dezimal - Hierbei werden die Zahlen in gepackter Form dezimal gespeichert, für eine Zahl sind somit 4 Bit notwendig, so daß man in einem Byte zwei Zahlen oder eine Zahl mit Vorzeichen speichern kann, das Vorzeichen ist immer im letzten Halbbyte einer Dezimalzahl enthalten, die nicht an eine Wortgrenze gebunden ist.

Es gibt in der S/360-Architektur verschiedene Arten von Instruktionen, die noch in der z/Architektur genauso gültig sind, wenn auch in wesentlich erweiterter Form. Auf Einzelheiten wird in "*2.0 Die z/Architektur und die zSeries*" eingegangen.

Die Ausführung eines Programms geschieht durch eine Serie von Instruktionen, die durch einen *Sequence Counter* kontrolliert werden. Dieser Counter ist Teil des sogenannten *Program Status Word* (PSW), das eines der wesentlichen Bestandteile des Systems darstellt. Die Rechner der S/360-Architektur sind 2-Adreß-Rechner, das bedeutet, daß jede Instruktion eine oder zwei Adressen beinhaltet, die entweder in maximal zwei Registern, in einem Register und einer Speicheradresse oder in bis zu zwei Speicheradressen dargestellt werden.

Zur Adressierung stehen in der S/360-Architektur 24 Bit zur Verfügung, das reicht für einen Hauptspeicher von 16 MB (16 Millionen Bytes). Die Speichergröße, die ein Rechner adressieren kann, bezeichnet man als Adreßraum, und sie ist durch die Anzahl von Bits festgelegt, die man zur Adressierung benutzen kann. Allerdings war der Speicher seinerzeit noch sehr teuer, und ein Rechner mit einem Hauptspeicher von 512 KB galt schon als recht groß.

Bei der Speicher-Adressierung in den Instruktionen gibt es ein interessantes Design: Um zu verhindern, daß in jeder Instruktion, die den Speicher anspricht, die volle 24-Bit-Adresse angegeben werden muß, setzt man auf das Lokalitätsprinzip, davon ausgehend, daß die nächste Speicher-Referenz nahe der vorhergehenden Speicher-Referenz liegen würde. Damit kann man ein Register mit einer sogenannten Basis-Adresse laden und kommt in den Instruktionen mit nur noch 12 Bit aus (man spricht hier von dem *Displacement*), um die genaue Adresse anzugeben. Mit 12 Bit kann man einen Bereich der Größe 2^{12} = 4096 = 4K adressieren. Solange also das Programm mit den in ihm definierten Speicherplätzen kleiner als 4 KB ist, kommt man mit einem Basis-Register aus, anderenfalls muß man ein zweites Basisregister definieren.

Das Betriebssystem OS/360

Das Betriebssystem OS/360, das für die Rechner der S/360-Architektur entwickelt wurde, konnte allerdings nicht für alle Modelle eingesetzt werden, sondern für die kleinen Rechner bis zur S/360-40 gab es noch das Betriebssystem DOS/360 - DOS steht dabei für *Disk Operating System* und hat nichts mit MS/DOS zu tun. Es gab sogar noch ein TOS/360 für Systeme, die keine Platten, sondern nur Bandeinheiten hatten.

OS/360 - Operating System for S/360: Der Name legt die Vermutung nahe, daß es sich um **ein** Betriebssystem handelt, in der Realität waren es anfangs allerdings drei Varianten:

- OS/360-PCP (*Primary Control Program*): Eine sehr einfache Version, bei der immer nur ein Programm zu einer Zeit laufen konnte. Es hatte aber keine große praktische Relevanz.
- OS/360-MFT (*Multiprogramming with a Fixed number of Tasks*): Hier konnten mehrere Programme gleichzeitig laufen. Dafür mußte der Speicher in eine feste Zahl von Bereichen (Partitionen) aufgeteilt werden, und in jeder Partition konnte ein Programm laufen. Falls in einer Partition kein Programm aktiv war, konnte dieser Speicherbereich nicht von anderen Programmen genutzt werden.
- OS/360-MVT (*Multiprogramming with a Variable number of Tasks*): Damit war es möglich, Partitionen (man spricht hier auch von Regionen, und sie hatten eine variable Größe) so zu starten und zu beenden, wie es die Workload im System notwendig machte. Falls einerseits genug Speicher vorhanden war und andererseits in der *Job Queue*, also der Warteschlange für Jobs, ein Job wartete, der mit dem verfügbaren Speicherplatz auskommen konnte, so wurde automatisch eine Region mit der erforderlichen Größe gestartet. Der Vorteil eines solchen Mechanismus ist klar, als Nachteil ergab sich aber, daß es im Lauf der Zeit eine starke Fragmentierung des Speichers gab, besonders dann, wenn langlaufende Jobs mit möglicherweise nur geringem Speicherbedarf in der Mitte des Speichers liefen. Dafür wurde dann noch eine neue Komponente HASP, ein *Job Scheduler*, entwickelt, der die Speicherbelegung und das Starten der Jobs zu optimieren versuchte.

  Neben dem Multiprogramming wurden weitere wesentliche Konzepte mit MVT eingeführt:

  Multitasking - Hierbei handelt es sich quasi um ein Multiprogramming innerhalb eines Programms. Durch einen ATTACH Supervisor Call wird eine zusätzliche Task erstellt, die relativ unabhängig von dem Hauptprogramm gesteuert und verarbeitet werden kann. Die Funktion *fork*() im UNIX arbeitet ähnlich, wobei ein *child* allerdings in einem separaten Adreßraum laufen kann.

  I/O-Management - Für den Programmierer besteht eine relativ große Unabhängigkeit von der I/O-Einheit (*Input/Output*), auf die im Programm zugegriffen werden soll. Er muß deshalb nicht im Programm definieren, von welcher Platte oder von welchem Band ein Record gelesen werden soll, sondern er legt nur durch eine in der Programmiersprache enthaltene Anweisung (beispielsweise READ oder GET) fest, **daß** ein Record gelesen werden soll. Erst bei der Programmausführung wird definiert, von welcher I/O-Einheit (Band, Platte, Kartenleser) gelesen werden soll.

  Resource Management für mehrere Benutzer - Ein wichtiger Aspekt in einem Multiprogramming-System ist die Datensicherheit, sowohl im Speicher als auch auf den I/O-Einheiten. Es wurden deshalb Konzepte wie *Storage Protection* eingeführt, die sicherstellten, daß kein Programm auf Daten im Speicher zugreifen konnte, die einem anderen Programm gehörten. Ebenso hatte im I/O-Bereich jeder Benutzer die Möglichkeit zu definieren, ob auf die Datasets, mit denen er gerade arbeitete, gleichzeitig von anderen Benutzern zugegriffen werden konnte. Hierbei handelte es sich allerdings um eine Frage der Datenintegrität, die noch nichts mit Datensicherheit zu tun hatte. Derartige Aspekte spielten erst später eine Rolle.

Mit der späteren Entwicklung des virtuellen Speichers wurden diese Betriebssystemvarianten von MFT zu OS/VS1 und MVT erst zu OS/VS2 und danach zu MVS weiterentwickelt. Allerdings war MVT ein sehr langlebiges Betriebssystem, von dem die letzte Version im Jahr 1978 auf den Markt kam, um die Prozessoren IBM 3031, 3032 und 3033 zu unterstützen, die bereits zur S/370-Architektur gehörten[4].

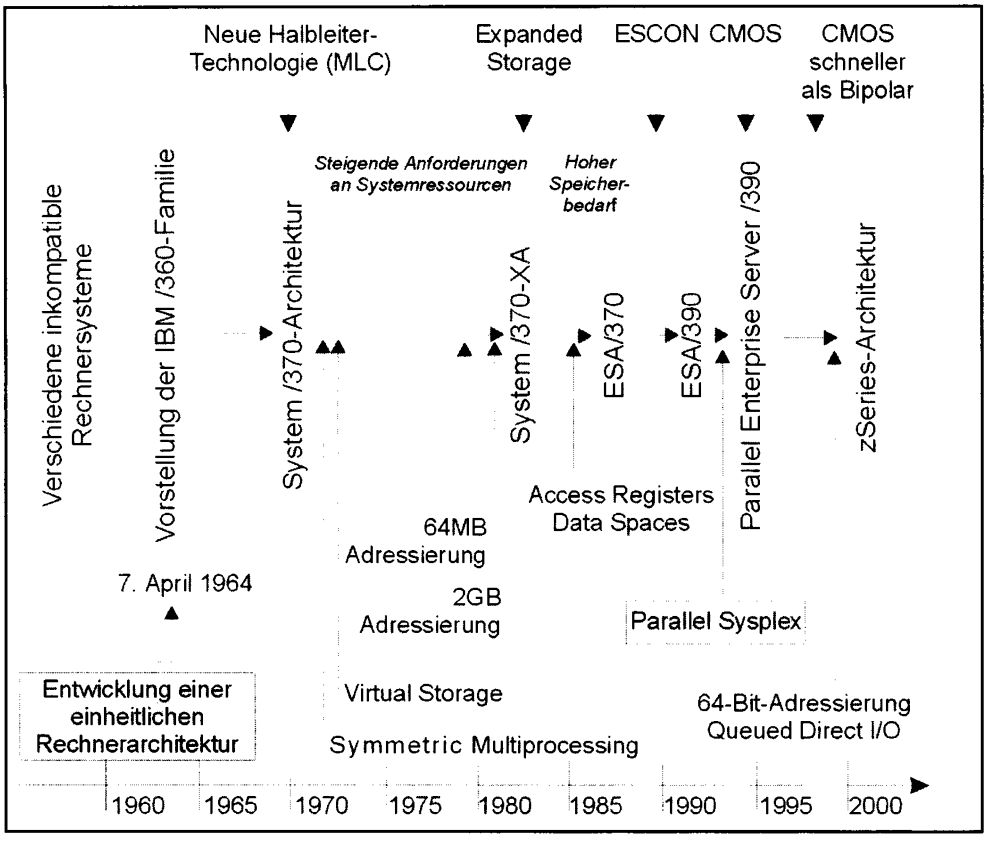
Datenverarbeitung war Stapelverarbeitung (*Batch Processing*), das heißt, die Jobs wurden als Lochkartenstapel in das System eingelesen und dann parallel zu einander, man hatte ja ein Multiprogramming-System, verarbeitet, entweder in den fest definierten Bereichen im MFT oder in den variablen Bereichen im MVT. Die Zeit war jetzt aber reif für Neues: Interaktive Datenverarbeitung. Neben OS/360 und DOS/360 wurde noch ein drittes Betriebssystem entwickelt: TSS/360 (*Time Sharing Multi-User System*). Diesem System war aber kein großer Erfolg beschieden, es war zu instabil und konnte keinen vernünftigen Betrieb anbieten. Deshalb ging die Entwicklung weiter und brachte TSO (*Time Sharing Option*) als Teil von OS/360. Time Sharing bedeutet, daß die zur Verfügung stehende CPU-Zeit in einzelnen Zeitscheiben (*Time Slicing*) an die Regionen verteilt wurde.

Um mit dem TSO-System arbeiten zu können, mußte man zwischen seinem Terminal, dabei handelte es sich zu dieser Zeit um ein schreibmaschinen-ähnliches Gerät, und dem Rechner eine Verbindung herstellen, üblicherweise über eine Telefonleitung. Damit begann ein Logon-Vorgang, der nach Überprüfung der Autorisation letztendlich ähnlich ablief wie die Systemaktivitäten nach dem Einlesen eines Lochkartenjobs - und so war dann schließlich auch ein TSO-Benutzer ein Job im OS/360. Der Unterschied zum Batch Processing war, daß dem TSO-Benutzer, wie oben beschrieben, eine Zeitscheibe zugewiesen wurde, in der seine vom Terminal gestartete Transaktion abgearbeitet werden konnte. Abhängig von dem dafür benötigten CPU-Verbrauch wurde die Transaktion in der ersten Zeitscheibe fertig oder mußte warten, bis sie wieder an der Reihe war.

Um die Systeme optimal auszulasten waren interaktive und Stapelverarbeitung parallel im System möglich, man bezeichnet die beiden unterschiedlichen Arten der Datenverarbeitung auch als *Foreground*- und *Background*-Aktivität.

Und dann gab es noch etwas: CP-67, das spätere VM/370 (*Virtual Machine /370*). Hierbei handelte es sich auch um ein *Multi-User* System, das aber komplett anders als TSO war. Es wurde nämlich so konzipiert und entwickelt, daß jeder Benutzer seinen eigenen (virtuellen) Rechner hatte, den er starten und auf dem er arbeiten konnte. Damit war VM das ideale Vehikel für alle Systemprogrammierer zum Testen von Betriebssystemen, an derer Entwicklung sie arbeiteten. VM war ursprünglich für IBM nur zum internen Gebrauch geplant, erfreute sich aber bald so großer Beliebtheit, daß es auch den Kunden zur Verfügung gestellt wurde - und es existiert heute noch als z/VM auf den zSeries-Rechnern.

[4] OS/360 MVT gibt es auch heute noch, und zwar als Simulation, die man auf Linux-Systemen laufen lassen kann - sicherlich recht ausgefallen, aber man kann im Internet darauf zugreifen, durch etwas Surfen (Stichwörter *OS/360 MVT* und *Hercules-390*) findet man bestimmt die aktuellen Adressen.



System /360 als Basis der Entwicklung

1.5 Von S/370 zur zSeries - Ein rasanter Fortschritt

Im Rahmen der S/360-Architektur hat die Entwicklung der Rechner und ihrer Betriebssysteme große Fortschritte gemacht. Aus dem System, in dem nur ein Programm laufen konnte, wurde ein Multi-User-System, in dem viele Programme und Online-Benutzer laufen und damit um die zur Verfügung stehenden Ressourcen ringen konnten.

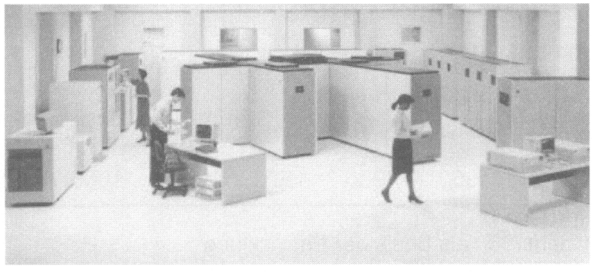
Um ein solches System optimal nutzen zu können, muß einerseits genügend Prozessor-Kapazität vorhanden sein, aber andererseits wächst auch der Bedarf nach immer mehr Speicher und nach schnellerem Zugriff auf die Daten.

Diese Forderungen wurden in den folgenden drei Jahrzehnten kontinuierlich durch immer neue Entwicklungen auf allen wesentlichen Gebieten erfüllt, und es ist auch heute noch kein Ende der Entwicklung abzusehen. Zwar gab es zwischenzeitlich in den 80er und 90er Jahren viele Propheten, die ein schnelles Ende der Dinosaurier, wie man die Großrechner gerne bezeichnete, vorhersagten. Das war ein Ergebnis der Euphorie, die weltweit mit der Entwicklung des PCs einsetzte. Das Wachstum dieser kleinen Wunderwerke geht zwar unvermindert weiter, und die Leistung der neuesten Laptops hat die der Großrechner aus den ersten Jahren schon weit übertroffen, aber dennoch hat sich inzwischen die Erkenntnis durchgesetzt, daß es kein Gegeneinander, sondern ein

Miteinander der Kleinen und der Großen geben sollte - jeder hat seine Vorteile, die man nutzen sollte.

Im folgenden sollen wichtige Neuerungen und Entwicklungen gezeigt werden, die bei der Entwicklung von S/360 über S/370 zur zSeries und z/OS von Bedeutung waren und wesentlich zu der Leistungsexplosion beigetragen haben, die die Großrechner geprägt hat. Dabei kommt es hier weniger auf die genaue Darstellung aller Architekturen, Betriebssystem-Varianten und Prozessoren-Familien an, sondern die Entwicklung soll anhand einiger wichtiger Aspekte dargestellt werden.

In mehr als drei Jahrzehnten wurden verschiedene Systemfamilien, Architekturen und Betriebssysteme entwickelt, wobei immer sehr viel Wert auf Kompatibilität gelegt wurde, durch die sichergestellt werden konnte, daß die Programme von gestern auch noch auf den Rechnern von morgen laufen konnte - das war für die hohen Investitionen der Kunden von großer Bedeutung. Die wesentlichen Architekturen, die sich aus S/360 heraus entwickelten, waren System/370 mit den dazugehörenden Betriebssystemen OS/VS2, aus dem sich das auch heute noch (als Kern im z/OS) existierende MVS (*Multiple Virtual Storage*) entwickelte. Die erste Version davon war etwa im Jahr 1973 MVS/370, dann ging es 1978 mit MVS/SE weiter. Zu dieser Zeit wurde damit begonnen, verschiedene Funktionen, die bisher komplett in der Software zur Verfügung standen, als Mikrocode zu implementieren, wodurch eine sehr starke Hardware-Abhängigkeit entstand: Das neue Betriebssystem konnte also nicht mehr auf jedem S/370- Prozessor laufen. Die nächsten Schritte brachten MVS/SP (1980), MVS/XA (1983) und MVS/ESA (1988) - alles Entwicklungen der S/370-Architektur. Im Jahr 1990 wurde das System/390 angekündigt, die dazugehörenden Rechner waren die der ES/9000-Familie.



IBM ES/9000-System

Die wesentlichen Komponenten eines Systems sind, wie bereits erwähnt, die Processing Unit, das I/O-Subsystem und der Speicher. Für die optimale Nutzung eines Systems ist es notwendig, dem Kunden ein *Balanced System* zur Verfügung zu stellen. Das bedeutet, daß Entwicklungen auf allen Gebieten notwendig sind - und auch stattgefunden haben. Darüber soll hier berichtet werden. Dabei wird es sich interessanterweise zeigen, daß gerade im Speicher-Bereich (ob nun real oder virtuell) die wesentlichen Entwicklungen stattgefunden haben, mit denen man die verschiedenen Architekturen und Betriebssystem-Varianten charakterisieren kann.

Die Processing Unit - So geht es schneller

Wie schnell ist ein Rechner? In "*8.0 Performance - Ein Problem der Datenverarbeitung*" werden wir uns ausführlich mit dem Thema beschäftigen, wie man die Leistung und Geschwindigkeit eines Rechners darstellen kann. Es gibt dafür viele Möglichkeiten, eine - und zwar zugegebenermaßen die geläufigste - ist die Bewertung über die Zahl der Instruktionen, die pro Sekunde verarbeitet werden können: MIPS (*Millions of instructions per second*). Deshalb werden wir in diesem Zusammenhang die Prozessoren auch über ihren MIPS-Wert charakterisieren, bei allen Bedenken, die es dazu gibt, aber das wird später in aller Ausführlichkeit betrachtet.

Die Gesamtleistung eines System ergibt sich immer aus dem Zusammenspiel aller Komponenten, und so berücksichtigt die folgende Tabelle, die einen Überblick für fast vier Jahrzehnte Großrechner-Entwicklung gibt, nicht nur die Prozessoren, sondern implizit auch den Fortschritt in den Bereichen Speicher, I/O und Software.

Zeitraum	Systemfamilie	UP-Performance	# Prozessoren	System-Performance
1965-1970	S/360	<0,5	1	<0,5
1970-1976	S/370 Mod 155 - 168	0,8 - 2,9	1	2,9
1977-1980	3031/3032/3033	1,2 - 5,1	1	5,1
1979-1984	4361	1,0 - 1,3	1	1,3
1979-1981	4341	0,9 - 1,7	1	1,7
1983-1989	4381	2,4 - 6,0	2	10
1981-1984	3083/3081/3084	3,2 - 8,2	4	29
1985-1989	3090	7,1 - 23	6	110
1986-1989	9373/9375/9377	0,4 - 2,3	1	2,3
1990-1992	ES/9000 - 9021	20 - 64	10	490
1990-1992	ES/9000 - 9121	6 - 31	4	108
1992	ES/9000 - 9221	2 - 14	2	25
1993	Parallel Enterprise Server S/390 - 9672-R1	15	6	65
1994	S/390 - 9672-R2	22	7	114
1995	S/390 - 9672-R3	23	10	180
1996	S/390 - 9672-G3	49 - 53	10	389
1997	S/390 - 9672-G4	63 - 71	10	473
1998	S/390 - 9672-G5	118 - 156	10	1110
1999	S/390 - 9672-G6	176 - 205	10	1644
2000	zSeries 900 101-116	249 - 261	16	2800
2002	zSeries 900 2C1-216	314	16	3300
2003	zSeries 990 301-316	500	32	>9000

UP-Performance: Minimale und maximale Performance der Uniprozessoren einer Systemfamilie (in MIPS)
Prozessoren: Maximal-Zahl von Prozessoren in einem SMP-System
System-Performance: Maximale Performance eines SMP-Systems (in MIPS)

Bei den ersten Rechnern der S/360 von MIPS zu sprechen, ist fast schon überzogen, denn anfangs war man noch im Bereich von KIPS - also Tausenden und nicht Millionen von Instruktionen. Doch man wurde kontinuierlich schneller, es gab eine Faustformel, nach der sich alle 18 bis 24 Monate die Rechnergeschwindigkeit verdoppelte. Es ist ungewiß, ob es sich hierbei um eine Naturkonstante handelte oder ob die Ingenieure angehalten waren, diesen Rhythmus einzuhalten - es hat über Jahrzehnte hinweg annähernd gestimmt.

Wenn wir uns zuerst damit beschäftigen wollen, wie man als Ingenieur oder Designer Einfluß auf die Geschwindigkeit des Rechners nehmen kann, so spielen immer viele Komponenten eine Rolle, ein MIPS-Wert hängt beispielsweise auch von der Zugriffsgeschwindigkeit auf den Speicher ab, aber hier geht es zuerst um den Rechner, die CPU (*Central Processing Unit*). Die originäre Geschwindigkeitskomponente ist natürlich die Taktfrequenz, mit der der Rechner läuft, sie lag bei den S/360-Rechnern zwischen 1 und 5 MegaHertz, was einer Zykluszeit von 0,2 bis 1 Millisekunden entspricht. Damit liegt die erste Aufgabe darin, diese Frequenz zu erhöhen. Das hört sich einfacher an als es ist - es war aber möglich, indem die physikalischen Komponenten im Rechner, die Schaltungen und Schaltkreisdichten mit allem was dazu gehört verbessert werden konnten. Diese Verbesserungen konnten kontinuierlich über die Jahre

hinweg erreicht werden, wobei von der Physik natürliche Grenzen vorgegeben waren. Heute ist man bei den schnellsten Rechnern im Bereich von einer Nanosekunde als Zykluszeit angekommen.

Die zur Ausführung einer Instruktion benötigte Zeit hängt einerseits von der Zykluszeit und andererseits von der Zahl der notwendigen Zyklen ab. Es gibt viele Instruktionen, die innerhalb eines Zyklus beendet werden können, oftmals sind aber doch mehrere Zyklen notwendig. Das ist durch die Komplexität und Implementierung jeder Instruktion bedingt. Es geht sicherlich schneller, den Inhalt zweier Register zu addieren als ein Datenfeld von 1000 Zeichen innerhalb des Speichers zu verschieben - soviel zur Komplexität. Was hat es nun mit der Implementierung zu tun? Es gibt zwei Möglichkeiten: Eine Instruktion ist direkt in der Hardware, also als logische Kombination verschiedener Schaltungen, implementiert, oder sie steht über Mikrocode zur Verfügung. Dabei handelt es sich um einen speziellen Assembler-Code, den man sich als Zwischenebene zwischen der Hardware und dem auszuführenden Programm vorstellen kann. Diese zweite Art der Implementierung ist billiger und flexibler als die Hardware-Implementierung, aber auch langsamer. Die Schnelligkeit ist also unter diesem Aspekt eine Kostenfrage, und so hat es sich ergeben, daß häufig in einer Prozessoren-Familie die Rechner im unteren Leistungsspektrum aus Kostengründen für viele Instruktionen eine Mikrocode-Implementierung hatten, während man für die schnelleren (und damit teureren) Maschinen vieles oder alles *in Hardware goß*, wie so die gängige Ausdrucksweise war.

Wenn die Processing Unit eine Instruktion zur Ausführung bringen soll, so muß diese zuerst aus dem Speicher, in dem sich das auszuführende Programm befindet, geladen werden, und das gilt genauso für die Daten, die in der Instruktion verarbeitet werden sollen. Das Laden aus dem Speicher ist ein relativ zeitaufwändiger Vorgang, und deshalb versucht man, ihn zu optimieren. Aus diesem Grund hat man den Cache eingeführt, einen sehr schnellen (aber im Vergleich zum Hauptspeicher kleinen) Zwischenspeicher, in den die Daten aus dem Hauptspeicher asynchron geladen werden. Die CPU kann damit wesentlich schneller auf die Daten zugreifen, sofern sie im Cache vorhanden sind, man spricht hier von einem *Cache Hit*. Naturgemäß wird das aber nicht immer der Fall sein, es kommt dann zu einem *Cache Miss*, und der Cache muß erst neu geladen werden, bevor die nächste Instruktion ausgeführt werden kann. Daraus folgt, daß der MIPS-Wert eines Rechners auch sehr stark von der Zahl der Cache Misses abhängt, die man einerseits durch die Größe des Caches beeinflussen kann, die aber auch von vielen anderen Faktoren abhängt.

Besonders stark tritt dieses Verhalten auf, wenn mehrere Rechner im Verbund arbeiten und einen gemeinsamen Cache haben. Denn in einem solchen Fall muß der Cache jedesmal neu geladen werden, wenn ein anderer Rechner auf ihn zugreifen will. Das ist einer der Gründe, warum sich im Verbund von zwei oder drei Rechnern die Gesamtleistung nicht um den Faktor 2 oder 3 im Vergleich zur Einzelrechner-Performance verbessert, sondern deutlich niedriger ist.

Die Idee, ein System mit mehreren Prozessoren auszustatten, kam schon sehr früh auf und ist aus den heutigen Rechner-Familien nicht mehr wegzudenken. Eine solche Möglichkeit wurde bei der Entwicklung der S/370-Architektur berücksichtigt: Der *Symmetric Multi Processor* (SMP). Dabei konnten mehrere CPUs auf einen gemeinsamen Hauptspeicher zugreifen und wurden von einem gemeinsamen Betriebssystem gesteuert. Im I/O-Bereich gab es allerdings noch eine Trennung, jede CPU hatte ihren eigenen *Channel Set*, an den die Platten angeschlossen waren. Diese Einschränkung wurde erst später mit der XA-Architektur aufgehoben, in der jede CPU auf jeden Kanal zugreifen konnte. Der bereits erwähnte Performance-Verlust war anfangs recht groß, so daß man

über viele Jahre hinweg nur bis zu sechs Prozessoren zu einem Multiprozessor-System kombinierte, der absolute Leistungsgewinn durch mehr Prozessoren stand zu den damit verbundenen Kosten in keiner vernünftigen Relation. Hier lag einer der Entwicklungs-Schwerpunkte für die Designer, nach Wegen zu suchen, die ein größeres SMP-System erlaubten - sie wurden gefunden, und in der zSeries-Familie gibt es neuerdings Systeme mit bis zu 32 Rechnern, die optimal miteinander arbeiten können. Einige der dafür wesentlichen Faktoren werden in "*2.0 Die z/Architektur und die zSeries*" erklärt.

Wie bereits erwähnt dauert das Laden der nächsten auszuführenden Instruktion einige Zeit, selbst wenn sie sich im schnellen Cache befindet. Deshalb kam man auf die Idee, die Instruktion *N+1* schon zu laden, solange noch die Instruktion *N* in der Ausführung ist. Das macht den ganzen Prozeß schneller, funktioniert aber nur dann, wenn die im Programm sequentiell aufeinander folgenden Instruktionen *N* und *N+1* wirklich nacheinander ausgeführt werden sollen, was in den meisten Fällen auch stimmt, und wenn das Ergebnis von *N* nicht für *N+1* benötigt wird (das wäre beispielsweise dann der Fall, wenn in *N* die Adresse berechnet wird, von der in *N+1* etwas geladen werden soll). Meistens wird es also funktionieren, aber nicht immer. Man nennt dieses Verfahren *Pipelining*. Und wenn die *Pipe* aus oben genannten Gründen unterbrochen wird, muß sie eben neu aufgesetzt werden, und es gibt wieder eine Verzögerung.

Der häufigste Grund für eine Unterbrechung der Pipe ist eine bedingte Verzweigung im Programm, bei der auf Grund einer logischen Überprüfung entweder die sequentiell nächste Instruktion ausgeführt oder zu einer anderen Adresse verzweigt wird. Gibt es hier irgendwelche Wahrscheinlichkeiten für die eine oder andere Variante? Man mag meinen, daß es relativ zufällig ist, welches Ergebnis die Überprüfung ergibt, das stimmt aber nicht. Das kann man an einem einfachen Beispiel erkennen. Es kommt in Programmen häufig vor, daß ein bestimmter Algorithmus innerhalb einer Programmschleife solange ausgeführt wird, bis eine wohldefinierte Bedingung erfüllt ist, etwa Zähler = 100. Die Schleife wird also 100mal durchlaufen, und 99mal ergibt die Verzweigung, daß an den Beginn der Schleife gesprungen werden soll. Auf derartige Situationen kann man sich nun im Pipelining einstellen, und dafür wurde eine *Branch History Table* (BHT) entwickelt. Die BHT enthält die vorherigen Sprung-Adressen, und das Pipelining basiert auf der Annahme, daß jede Instruktion zur selben Adresse verzweigen wird wie zuvor. Damit wird man in den meisten Fällen richtig liegen und das Pipelining und somit die Performance wieder um einen deutlichen Schritt verbessern.

Ein ganz spezielles Design war die Entwicklung der *Vector Facility* in einigen Rechnern. In vielen technisch-wissenschaftlichen Anwendungen spielt die Verarbeitung von Vektoren und Matrizen eine zentrale Rolle. Will man beispielsweise zwei Vektoren der Länge 100 addieren, so kodiert man traditionell eine Programmschleife, die 100mal zwei Elemente der Vektoren addiert. Das mag zwar in höheren oder in objektorientierten Programmiersprachen etwas eleganter machbar sein, läuft aber auf der Ebene der Maschinensprache, die der Programmierer nicht mehr sieht, doch wieder auf obige Schleife hinaus. Für derartige Aufgaben wurde mit der Vector Facility eine neue Hardware entwickelt, mit der man mit einer Instruktion die beiden Vektoren addieren kann, und weil dabei die Addition der 100 Elemente parallel durchgeführt wird, gewinnt man natürlich deutlich an Performance. Insgesamt gab es für die Vector Facility 180 spezielle Instruktionen, darunter auch sogenannte *Compound Instructions,* wie etwa Multiplizieren mit anschließender Addition oder Subtraktion (a=b×c+d) oder auch die Quadratwurzel aus den Elementen eines Vektors. Diese Funktion war eine spezielle Eigenschaft der Rechner der 3090-Familie in den 90er Jahren, und sie existiert in den Rechnern der zSeries nicht mehr.

Anhand der oben aufgeführten Beispiele kann man erkennen, welche Anstrengungen im Lauf der Jahre unternommen wurden, um die Leistung der Rechner zu verbessern. Der Bedarf nach mehr *Compute Power* war immer vorhanden, die Anforderungen der Kunden und ihrer Programme und Anwendungen wurden immer größer - ständig sahen sich die Ingenieure vor neue Herausforderungen gestellt. Manche der Verbesserungen waren technologie-unabhängig, andere nicht. Speziell die erste und wohl auch einleuchtendste Verbesserung durch Erhöhung der Taktfrequenz stieß irgendwann auf Grenzen. Immer wieder war ein Technologie-Wechsel notwendig, um zu schnelleren Prozessoren zu kommen. Die ersten Rechner der S/360 basierten auf der *Solid Logic Technology* (SLT), die 1969 durch die *Monolithic System Technology* (MST) mit den ersten integrierten Schaltkreisen abgelöst wurde. Man konnte mehrere komplette Schaltkreise auf einem Chip haben, mit einem Höchstwert von 40 bei dem Prozessor 3033 im Jahr 1978. Der nächste wesentliche Schritt in der Technologie-Entwicklung war 1981 die Einführung des *Thermal Conduction Module* (TCM) mit der 308X-Serie. Logik-Chips bestanden dabei aus mehr als 700 Schaltkreisen und wurden auf Keramik-Modulen montiert, die Platz für 100 Chips hatten. Das bedeutete eine deutliche Verbesserung der Schaltkreisdichte, und damit letztendlich auch der Rechnerleistung. Die nächste Rechnerfamilie der 3090 konnte 1985 mit einer verbesserten TCM-Technologie und noch schnelleren Schaltkreisen aufwarten.

Anfang der 90er Jahre waren zwei Technologien zur Herstellung von Halbleiter-Logikschaltkreisen Stand der Technik: *Emitter Coupled Logic* (ECL) mit bipolaren Transistoren und *Complementary Metal Oxide Semiconductor* (CMOS) mit komplementären Feldeffekttransistoren. Dabei wurde die bipolare Technologie hauptsächlich für Großrechner verwendet, während die CMOS-Technologie ein wesentlich breiteres Anwendungsspektrum hatte. Bipolar und CMOS zeichnen sich durch grundverschiedene Eigenschaften aus.

Bipolar ist durch extrem kurze Signallaufzeiten, also hohe Schaltgeschwindigkeiten, gekennzeichnet. Die bipolaren Transistoren waren etwa zehnmal schneller als Feldeffekttransistoren. Die schnellsten Bipolar-Prozessoren erreichten dadurch eine Zykluszeit von unter sechs Nanosekunden (milliardstel Sekunden). Ihre Schaltkreise erfordern jedoch einen ständigen Stromdurchfluß. Dies bewirkt einen hohen Energieverbrauch, der zusammen mit der notwendigen und aufwendigen Kühlungstechnik (Luftkühlung bei kleineren Systemen und Wasserkühlung bei den Performance-Spitzenreitern) hohe Betriebskosten verursacht. Hinzu kommt der komplexe und mehrschichtige Aufbau des Bipolar-Transistors, der auch die Herstellung solcher Chips verteuert. Bipolar-Technologie kam daher vorzugsweise in Großrechnern wie der 9021-Serie zum Einsatz. Diese Rechner waren 1990 die leistungsstärksten Systeme und kamen auf über 230 MIPS, welch eine Leistungsexplosion beispielsweise zum kleinsten S/370-Rechner im Jahr 1970 mit 0,7 MIPS.

Wegen des einfacheren Transistoraufbaus in der CMOS-Technologie sind Prozessoren vergleichsweise kostengünstig herzustellen. Dank der geringen Energieaufnahme - Strom benötigt ein CMOS-Schaltkreis nur für den eigentlichen Schaltvorgang - brauchen sie zudem wesentlich weniger Kühlung (das hatte den großen Vorteil, daß sie nicht mehr in klimatisierten, doppelbödigen Rechenzentren stehen mußten, sondern in jedem beliebigen Büroraum aufgestellt werden konnten). Sowohl die Schaltungen auf dem Chip als auch die Chips untereinander können dichter gepackt werden. Allerdings bleiben CMOS-Prozessoren mit einer Schaltgeschwindigkeit von 15 Nanosekunden Zykluszeit deutlich hinter Bipolar-Prozessoren zurück. Die Folge: Bipolar-Prozessoren sind im Vergleich zu CMOS-Prozessoren um etwa den Faktor 2,5 leistungsfähiger -