

10 Superskalare Architekturen

10.1 Einleitung

10.2 Überlappung von Ein/Ausgabe und Rechenoperationen

10.3 Überlappung von Befehlsaufbereitung und Befehlsausführung

10.4 Kennzeichen superskalarer Prozessoren

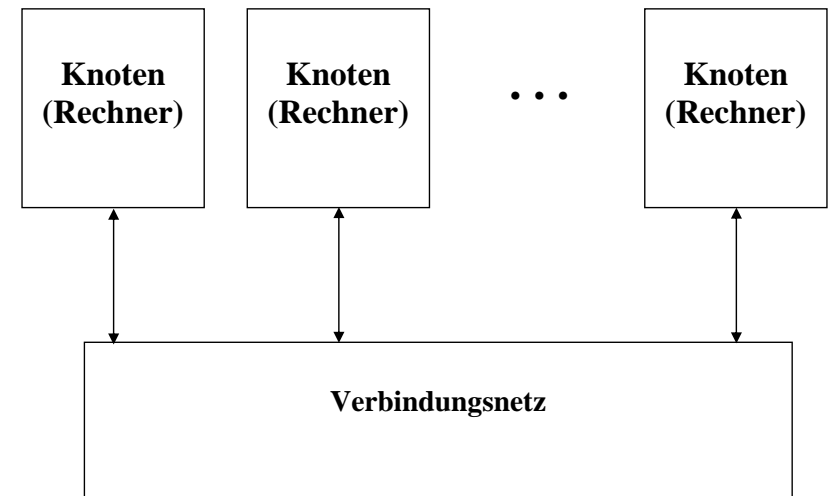
10.5 Hemmnisse bei superskalaren Berechnungen

10.6 Verfahrensvarianten bei superskalaren Prozessoren

10.7 CDC 6600

10.8 Scoreboard der CDC 6600

Bild eines Parallelrechners:



Bemerkungen:

- (i) **Hinter den Knoten können sich konventionelle Prozessoren, Spezialprozessoren oder Ein/Ausgabe-Prozessoren verbergen.**
- (ii) **Als Verbindungsnetz dienen systemspezifische Einzelanfertigungen oder auch kommerzielle lokale Netze wie Ethernet, FDDI, Myrinet oder ATM-Netz.**
- (iii) **Als man im Berkeley NOW-Projekt das Ethernet durch ein ATM-Netz ersetzte, erhielt man eine Leistungssteigerung des NOW-Systems um den Faktor 100.**
- (iv) **Das obige Bild ist auch deutbar als Monorechner mit autonom arbeitenden Einheiten.**

Superskalare Rechner:

Vorläufer:

IBM 7030 (Stretch)
CDC 6600
IBM 360/91

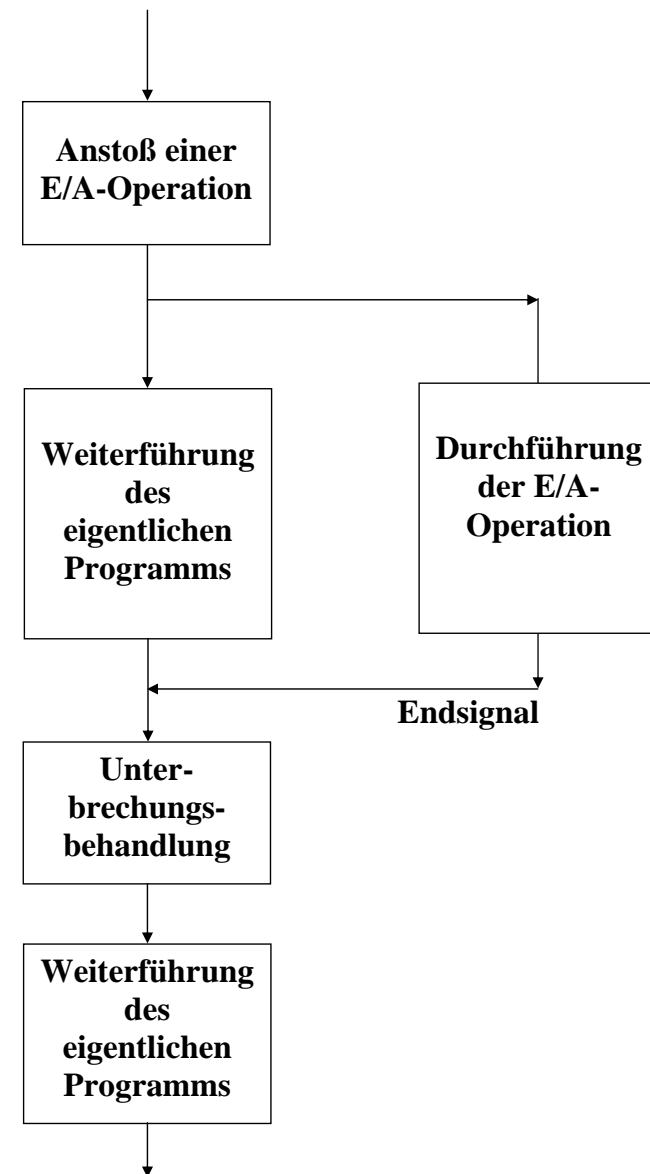
Gegenwärtige Prozessoren:

Hewlett-Packard PA-RISC
Intel x86: Pentium 3 und 4
MIPS
SPARC V8 und V9
DEC Alpha
IBM POWER und PowerPC
IBM/Motorola/Apple PowerPC
IBM Power4, Power5, Power6

Bemerkungen:

- (i) Die Entwicklung der IBM 7030 startete in 1955, eine fertige Maschine wurde 1961 nach Los Alamos geliefert, von 1961 bis 1964 galt sie als schnellste Rechenmaschine. Sie war die erste Maschine mit spekulativer Befehlsausführung.
- (ii) MIPS steht für Microprocessor without interlocked pipeline stages.
- (iii) Power steht für Performance optimized with enhanced RISC.

Überlappung von Ein/Ausgabe und Rechenoperationen:



Sei

Bf OP1, OP2, Ziel

ein generischer Befehl.

Die Befehlsdurchführung in einem Rechner läßt sich in zwei Phasen unterteilen: Befehlsaufbereitung und Befehlsausführung. Zur Befehlsaufbereitung gehören das Holen des Befehls, das Decodieren des Befehls, Adreßberechnungen und das Bereitstellen der Operanden, die Befehlsausführung betrifft dann die eigentliche Operation.

Phasen: Aufbereitung (A), Durchführung (D)

Für eine Befehlsfolge erhält man:

Bf1 = A1, D1

Bf2 = A2, D2

Bf3 = A3, D3

Bf4 = A4, D4

Bf5 = A5, D5

Bf6 = A6, D6

Bf7 = A7, D7

allgemein:

...

Bf(i) = A(i), D(i)

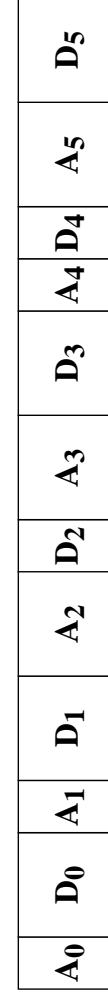
Bf(i+1) = A(i+1), D(i+1)

...

Die sequentielle Ausführungssemantik fordert:

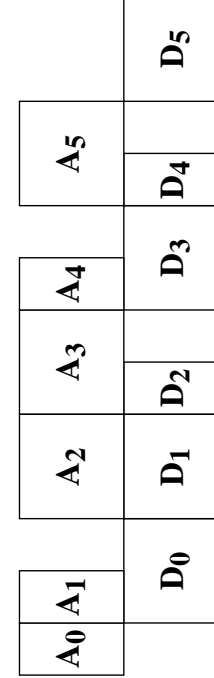
- (1) A(i) zeitlich vor D(i)
- (2) A(i) zeitlich vor A(i+1)
- (3) D(i) zeitlich vor A(i+1)

Fall 1:



Ausführungszeit: 19 Zyklen

Fall2: Lockerung von Bedingung 3: D(i) zeitlich vor A(i+2)



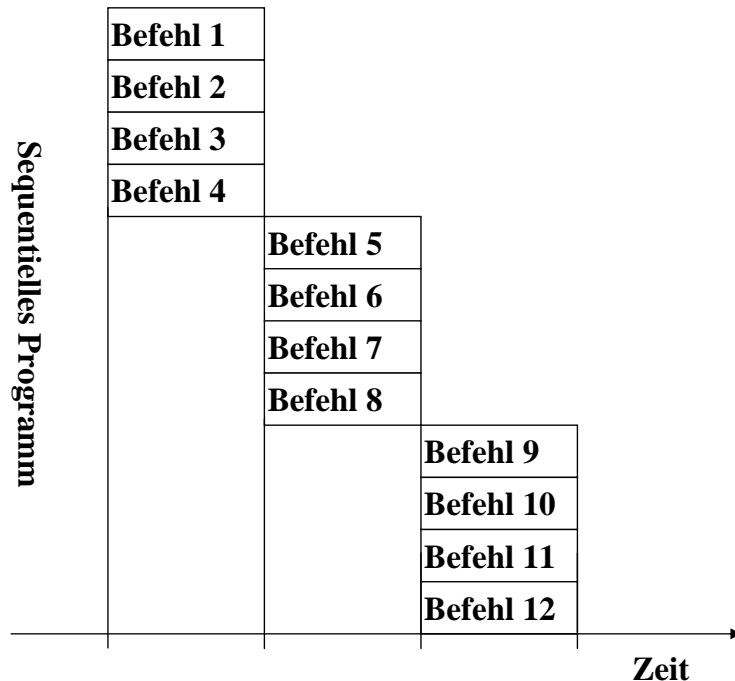
Ausführungszeit: 13 Zyklen

Bemerkung: In diesem Fall erhält man eine um den Faktor 1,46 bessere Ausführungszeit. Sind die Werke für Aufbereitung und Durchführung besser aufeinander abgestimmt, kann der Faktor bis 2,0 reichen.

Superskalarität:

Der Begriff "superskalar" wurde etwa 1987 von T. Agerwala und J. Cocke geprägt. Man nennt einen Rechner superskalar, falls er mehrere Befehle zum gleichen Zeitpunkt anstoßen kann, und diese dann unabhängig voneinander parallel ausgeführt werden. Die folgende Skizze möge dies veranschaulichen.

Wir haben eine Befehlsfolge von 12 Befehlen B1 bis B12, jeweils vier werden gleichzeitig gestartet.



Hemmnisse:

Datenabhängigkeit:

- (a) $R4 = R3 * R1$
- (b) $R5 = R4 + R2$

Die Befehle (a) und (b) können nicht gleichzeitig gestartet werden. Vielmehr muß der Befehl (b) warten, bis das Ergebnis von (a) zur Verfügung steht. Diese Wartezeit kann sinnvoll genutzt werden, falls man von (a) und (b) unabhängige Befehle vorzieht.

Ausgabeabhängigkeit:

- (a) $R4 = R3 * R1$
- (b) $R7 = R4 + R0$
- (c) $R4 = R5 + R2$

Starten die Befehle (a) bis (c) gleichzeitig und der Befehl (c) ist vor dem Befehl (a) beendet, dann nutzt der Befehl (b) eventuell die falschen Daten. Dies kann man umgehen durch Umbenennung von Registern, z. B.

- (a) $R4 = R3 * R1$
- (b) $R7 = R4 + R0$
- (c) $R401 = R5 + R2$

Gegenabhängigkeit:

- (a) $R4 = R3 * R1$
- (b) $R3 = R5 + R0$

In diesem Fall schreibt Befehl (b) in das Register R3, das einen Operanden von Befehl (a) darstellt. Auch diesen Konflikt kann man umgehen, in dem man die Register umbenennt, z. B.

- (a) $R4 = R3 * R1$
- (b) $R301 = R5 + R0$

Ressourcenkonflikt:

- (a) $R4 = R3 + R1$
- (b) $R7 = R5 + R0$

Beide Operationen benutzen die gleiche Funktionseinheit, einer der beiden Befehle muß warten. Beim Entwurf einer Rechanlage sollte man darauf achten, daß die einzelnen Funktionseinheiten in der für die Aufgaben gebotenen Zahl zur Verfügung stehen.

Bemerkung: Die angeführten Hemmnisse lassen sich teilweise mildern durch Befehlsumstellungen und Registerumbenennungen. Diese Aufgaben kann teilweise die Software und teilweise die Hardware übernehmen.

Verfahrensvarianten:

Beispiel:

- B1: $F2 = F4 + F6$
- B2: $R1 = R8 + R9$
- B3: $R4 = R2 * R3$
- B4: $R5 = R6 * R7$
- B5: $R0 = R5 + R7$
- B6: $R10 = R2 + R3$

Bemerkungen:

- (i) Der Befehl B1 ist ein Gleitpunktbefehl, er dauert doppelt solange wie die anderen Befehle.
- (ii) Bei den Befehlen B3 und B4 liegt ein Ressourcenkonflikt vor.
- (iii) Bei den Befehlen B2, B5 und B6 liegt ein Ressourcenkonflikt vor.
- (iv) B5 benötigt das Ergebnis von B4.

Abarbeitung in Programmreihenfolge:

Decodieren

B1	B2
B3	B4
B5	B6

Ausführen

B1	B2	
B1		
	B3	
	B4	
	B5	
	B6	

Rückschreiben

F2	R1	
R4		
R5		
R0		
R10		

Zyklus

1
2
3
4
5
6
7
8

Befehlsstart in Programmreihenfolge, frühes Beenden erlaubt:

Decodieren

B1	B2
B3	B4
B5	B6

Ausführen

B1	B2	
B1		B3
		B4
	B5	
	B6	

Rückschreiben

	R1	
F2	R4	
R5		
R0		
R10		

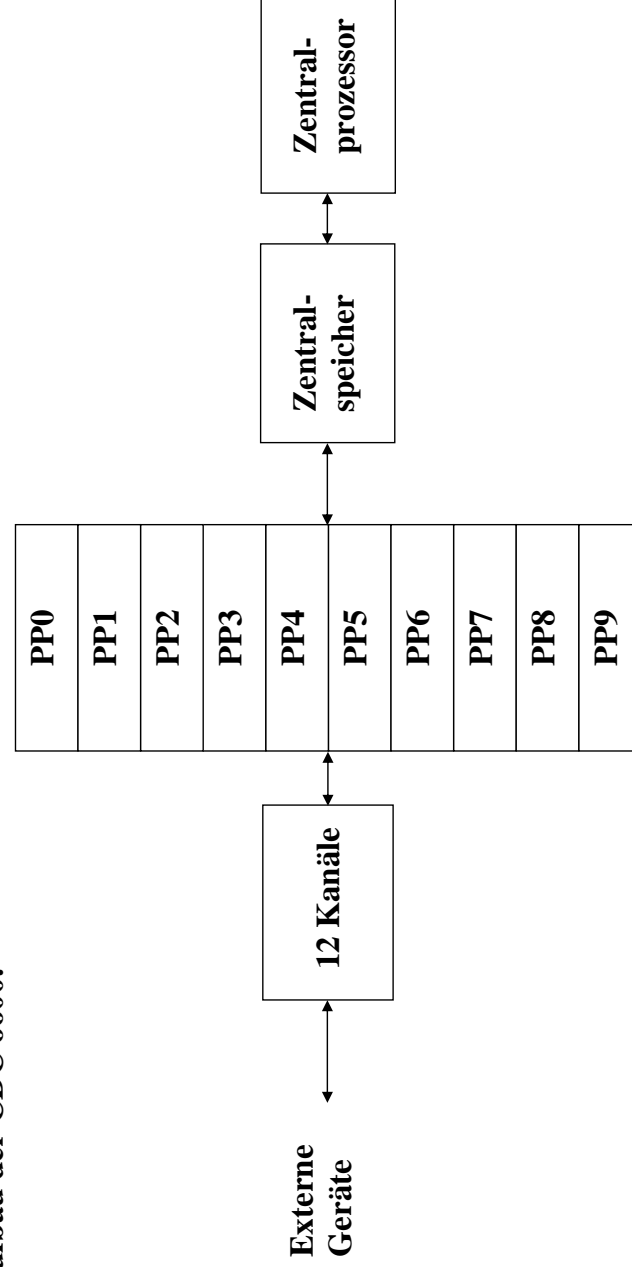
Zyklus

1
2
3
4
5
6
7

Vorziehen von Befehlen erlaubt, frühes Beenden erlaubt:

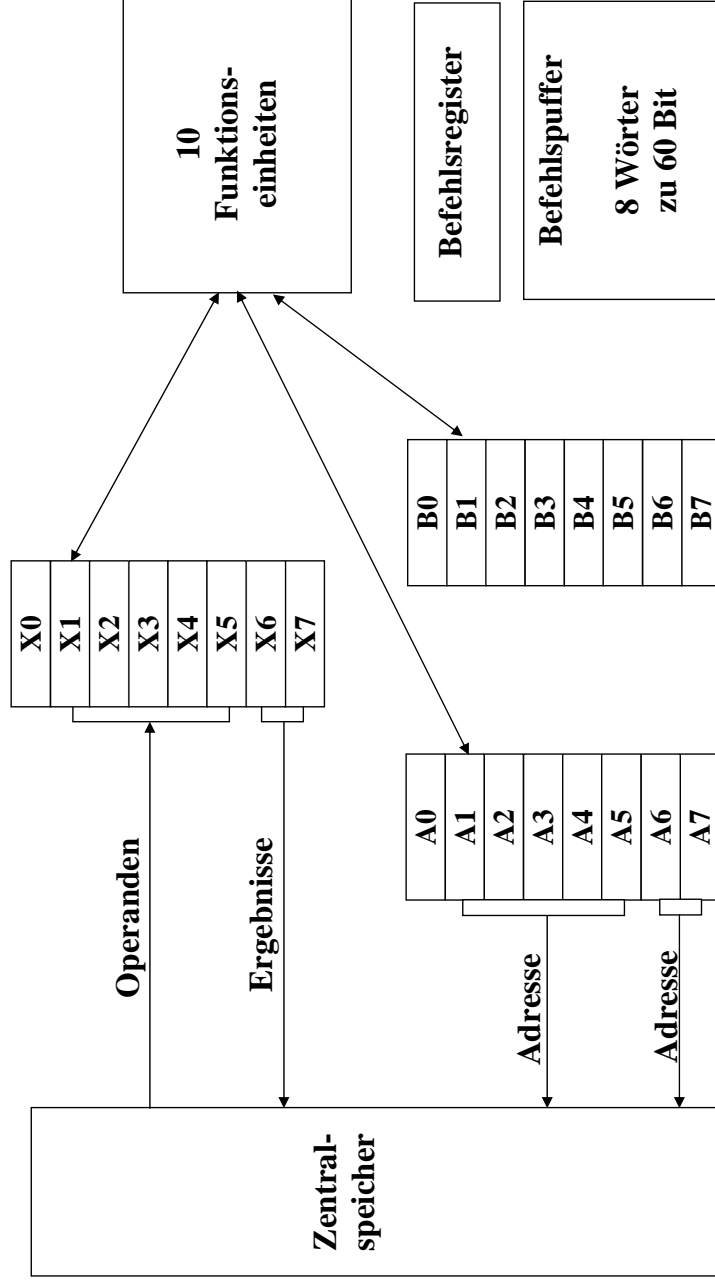
Decodieren	Ausführen	Rückschreiben	Zyklus
B1			1
B3	B2		2
B5	B1	R1	3
	B6	F2	4
	B5	R10	5
		R0	6

Aufbau der CDC 6600:

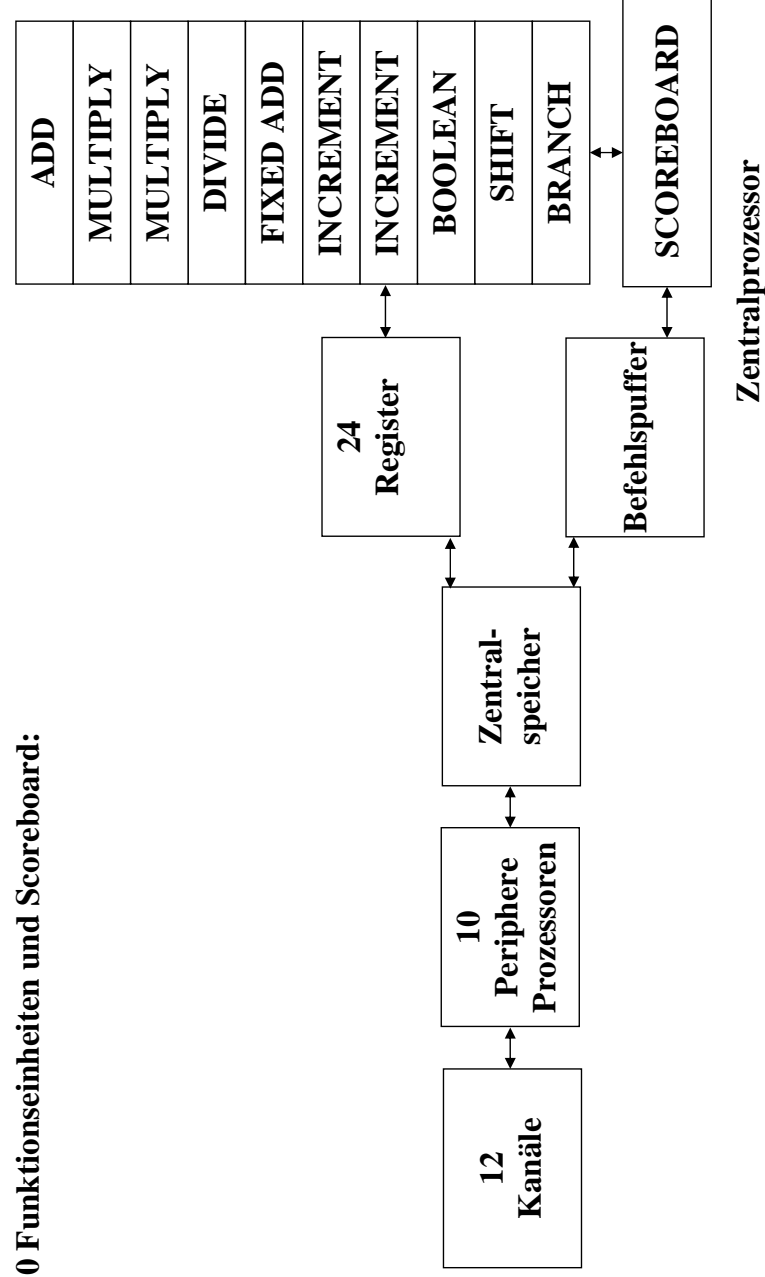


Bemerkung: PP steht für Peripheral Processor; jeder PP verfügt über lokalen Speicher. Das Betriebssystem residiert weitgehend in PP0 und PP1. Der Zentralprozessor hat keine direkte Verbindung zur Außenwelt.

CDC 6600, Teil der Zentraleinheit:

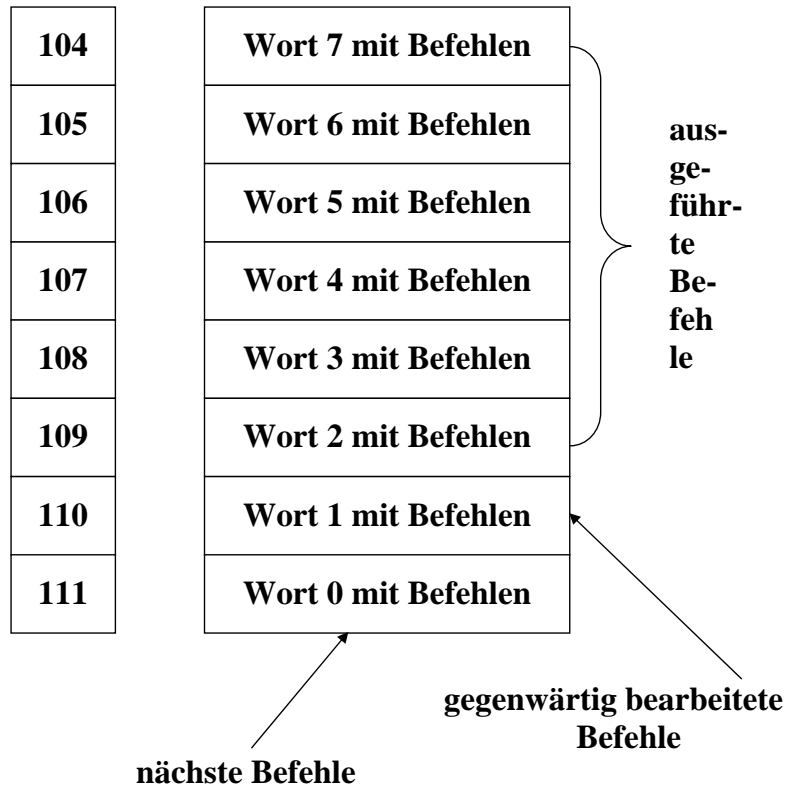


10 Funktionseinheiten und Scoreboard:



Befehlspeicher:

Haupt-
speicher-
adressen



Bemerkung: Kleine Schleifen passen vollständig in den Befehlspeicher. Der Befehlspeicher wird in Kellermanier verwaltet.

Bemerkungen zur CDC 6600:

- (i) Im Jahre 1964 wurde die erste CDC 6600 ausgeliefert. Sie ist der Archetyp des RISC-Rechners.
- (ii) Die Zentraleinheit besitzt drei Registersätze. Je acht Adreßregistern der Breite 18 Bit, je acht Indexregister der Breite 18 Bit, wobei das Indexregister 0 die Konstante 0 enthält, je acht Datenregister der Breite 60 Bit, wobei ein Datenregister eine Ganzzahl oder eine Gleitpunktzahl aufnimmt.
- (iii) Die Zentraleinheit besitzt zehn gleichzeitig arbeitsfähige Funktionseinheiten, diese sind: eine "Branch"-Einheit, eine Addiereinheit für Ganzzahlen, eine Addiereinheit für Gleitpunktzahlen, zwei Multiplikationseinheiten und eine Divisionseinheit für Gleitpunktzahlen, eine Einheit für Schiebeoperationen, eine für boolesche Operationen und zwei für Inkrementoperationen.
- (iv) Eine "Scoreboard" genannte Station sorgt für eine konfliktfreie Nutzung der unabhängigen Funktionseinheiten.
- (v) Der Zentralspeicher besteht aus 60-Bit-Speicherzellen; er umfaßt maximal 2^{17} Wörter.
- (vi) Die Befehle sind einheitlich aufgebaut, sie umfassen 15 oder 30 Bit, so daß ein Speicherwort zwei bis vier vollständige Befehle aufnimmt.

Beispiel zur überlappten Ausführung von Befehlen:

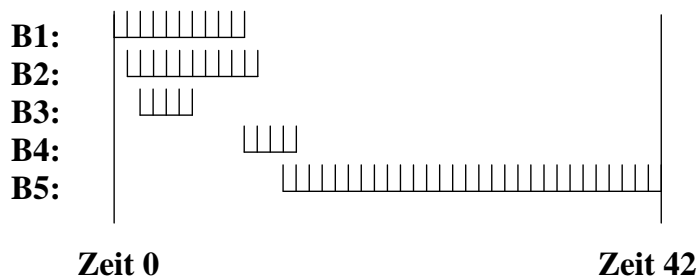
Rechnung: $(n * a - b * b) / (n - 1)$

Registerplan: x0: 1
 x1: n
 x2: a
 x3: b

Programm:

(B1) x4 := x1 * x2 ; 10 Zeiteinheiten
(B2) x5 := x3 * x3 ; 10 Zeiteinheiten
(B3) x6 := x1 - x0 ; 4 Zeiteinheiten
(B4) x0 := x4 - x5 ; 4 Zeiteinheiten
(B5) x5 := x0 / x6 ; 29 Zeiteinheiten

Zeitplan:



Bemerkung: Der Zeitverbrauch wird durch die Parallelausführung von Befehlen von 57 auf 42 Zeiteinheiten reduziert.

Scoreboarding:

Befehlsausführung:

1. Decodiere Befehl, bestimme Funktionseinheit, Ergebnisregister und Operandenregister.
2. Falls die Funktionseinheit beschäftigt ist, warte bis die Funktionseinheit frei ist.
3. Falls das Ergebnisregister von einer anderen Einheit belegt ist, warte bis die andere Einheit ihr Ergebnis abliefern.
4. Übergib Befehl an Funktionseinheit.
5. Reserviere das Ergebnisregister für Funktionseinheit.
6. Funktionseinheit wartet bis Operanden zur Verfügung stehen.
7. Befehl wird ausgeführt.
8. Funktionseinheit übergibt Ergebnis an Ergebnisregister und löscht Reservierungen.

Das Scoreboard verwaltet die zehn Funktionseinheiten und fünf Lesekanäle. Diese Einheiten werden durch einen eindeutigen Designator identifiziert. Zu jeder Funktionseinheit werden, falls sie belegt ist, folgende Informationen geführt: das Ergebnisregister, die Operandenregister, die Quellen für die Operandenregister und Anzeigen, ob die Operanden berechnet sind. Zusätzlich wird zu jedem Register vermerkt von welcher Einheit es reserviert ist.

Beispiel zur Reservierung:

$$X6 = X1 + X2$$

$$X7 = X5 / X6$$

Hier wird vermerkt, daß der zweite Operand für die Division von der Addiereinheit geliefert wird.

Beispiel zur Freigabe:

$$X3 = X1 / X2$$

$$X5 = X4 * X3$$

$$X4 = X0 + X6$$

Eine Division dauert bedeutend länger als eine Addition, daher darf das Ergebnis der Addition erst freigegeben werden, nachdem die Division ihr Ergebnis abgeliefert hat und der alte Wert von Register X4 von der Multiplikationseinheit entgegengenommen wurde.

Beispiel: Berechnung eines Polynomwertes: $Y = A * X * X + B * X + C$

Zeitangaben in Zyklen.

Assemblerprogramm:

```

A1 = A1 + K1; hole X
A2 = A2 + K2; hole A
X0 = X1 * X1; bilde X*X
X6 = X0 * X2; bilde A*X*X
A3 = A3 + K3; hole B
A4 = A4 + K4; hole C
X3 = X3 * X1; bilde B*X
X5 = X6 + X3; bilde A*X*X+B*X
X7 = X5 + X4; bilde Y
A7 = A7 + K5; speichere Y

```

S	T	A	R	T		1	3	9	19	11	17	20	30	35	39			
E	R	G	E	B	N	I	S		4	6	19	29	14	20	30	34	39	42
O	P	E	R	A	N	D			9	11		19	25					47

