Rechnerstrukturen

Vorlesung im Sommersemester 2007

Prof. Dr. Wolfgang Karl

Universität Karlsruhe (TH)

Fakultät für Informatik

Institut für Technische Informatik

- **Kapitel 5: Vektorprozessoren**

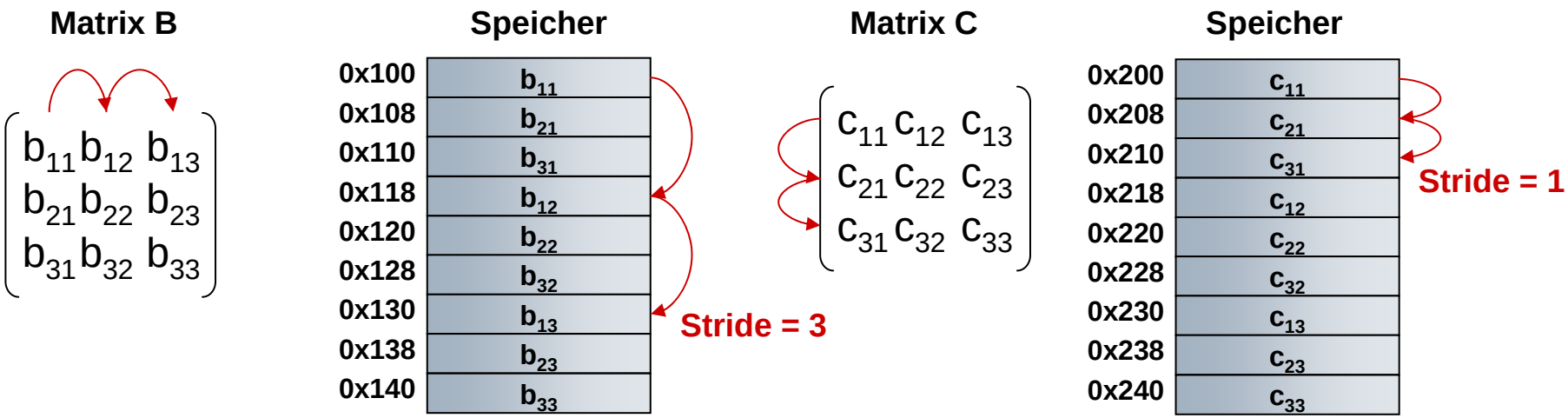
5.2: Eigenschaften

- Vektorprozessoren
  - Vektor Stride
    - Problem: die Elemente eines Vektors liegen nicht in aufeinander folgenden Speicherzellen
    - Beispiel: Matrix-Multiplikation

```
      do 10 i = 1,100
        do 10 J = 1,100
              A(i,j) = 0.0
              do 10 k = 1,100
10                  A(i,j) = A(i,j)+B(i,k)*C(k,j)
```

Anweisung mit der Marke 10: Vektorisierung der Multiplikation der Elemente jeder reihe von B mit den Elementen der Spalten von C.
Beachte dabei Adressierung der benachbarten Elemente in B und der benachbarten Elemente in C

- Vektorprozessoren
  - Vektor Stride
    - Problem: die Elemente eines Vektors liegen nicht in aufeinander folgenden Speicherzellen
    - Beispiel: Matrix-Multiplikation (k=3)



Berechnung: $b_{11}* c_{11}+ b_{12}* c_{21}+b_{13}*c_{31}$ …

- Vektorprozessoren
  - Vektor Stride
    - Abstand zwischen Elementen, die in einem Register abgelegt werden müssen
    - Beispiel:
      - bei spaltenweiser Ablage der Daten im Speicher ist die der Stride für die Matrix C gleich 1 (oder 1 Doppelwort) und für die Matrix B gleich 3 (oder 3 Doppelwörter)
    - Vektorprozessor mit Vektorregister können Strides größer 1 verarbeiten:
      - Vektorlade- und Vektrospeicherbefehle mit „Stride-Capability“
        - » Zugriff auf nicht sequentielle Speicherzellen und Umformen in dichte Struktur

- Vektorprozessoren
  - Vektor Stride
    - Problem:
      - Stride-Wert ist erst zur Laufzeit bekannt oder kann sich ändern
    - Lösung
      - Ablegen des Stride-Wertes in ein Allzweckregister
      - Vektorspeicherzugriffsbefehle greifen auf den Wert zu
    - Problem:
      - Zugriff auf eine Speicherbank erfolgt häufiger als es die Zugriffszeit der Bank erlaubt
      - Anhalten des Zugriffs, wenn:

$$\frac{\text{Anzahl der Speicherbänke}}{\text{kleinste gemeinsame Vielfache (Stride, Anzahl der Bänke)}} < \text{busy time der Bank}$$

    - In heutigen Vektorrechnern:
      - Verteilen der Zugriffe von jedem Prozessor über mehrere Hundert von Speicherbänken
      - Da Speicherbankkonflikte bei „nonunit strides“ grundsätzlich auftreten können, bevorzugen Programmierer „unit strides“

- Vektorprozessoren
  - Bedingt ausgeführte Anweisungen
    - Problem:
      - Programme mit if-Anweisungen in Schleifen können nicht vektorisiert werden:
        - » Kontrollflussabhängigkeiten!
    - Beispiel:

```
      do 100 i=1, 64
        if (A(i).ne. 0) then
              A(i) = A(i) –B(i)
         endif
100 continue
```

    - Lösung
      - Bedingt ausgeführte Anweisungen
      - Umwandlung von Kontrollflussabhängigkeiten in Datenabhängigkeiten

- Vektorprozessoren
  - Bedingt ausgeführte Anweisungen
    - Vektor-Maskierungssteuerung
      - verwendet einen Boole‘schen Vektor der Länge der festgelegten MVL (maximale Vektorlänge), um die Ausführung eines Vektorbefehls zu steuern, in ähnlicher Weise wie bedingt ausgeführte Befehle eine Boole‘sche Bedingung verwenden, um zu bestimmen, ob eine Instruktion ein gültiges Ergebnis liefert oder nicht
      - Vektor-Mask-Register
        - » jede ausgeführte Vektorinstruktion arbeitet nur auf den Vektorelementen, deren Einträge eine 1 haben. Die Einträge im Zielvektorregister, die eine 0 im entsprechenden Feld des VM Registers haben, werden nicht verändert

- Vektorprozessoren
  - Bedingt ausgeführte Anweisungen
    - Vektor-Maskierungssteuerung
      - Beispiel:

```
LV       V1,Ra     ;load vector A into V1
LV       V2,Rb     ;load vector B
L.D      F0,#0     ;load FP zero into F0
SNEVS.D  V1,F0     ;sets VM(i) to 1 if V1(i)!=F0
SUBV.D   V1,V1,V2  ;subtract under vector mask
CVM                ;set the vector mask to all 1s
SV Ra,V1           ;store the result in A
```

- Vektorprozessoren
  - Dünn besetzte Matrizen
    - Elemente eines Vektors werden in einer komprimierten Form im Speicher abgelegt
    - Beispiel:

```
      do   100 i = 1,n
100        A(K(i)) = A(K(i)) + C(M(i))
```

» implementiert die Summe der dünn besetzten Felder A und C mit Hilfe der Indexvektoren K und M. K und M zeigen jeweils die Elemente von A und C an, die nicht 0 sind.

» Alternative Darstellung ist die Verwendung von Bit-Vektoren

- Vektorprozessoren
  - Dünn besetzte Matrizen
    - SCATTER-GATHER Operationen mit Index-Vektoren
      - unterstützen den Transport zwischen der gepackten Darstellung und der normalen Darstellung dünn-besetzter Matrizen
      - GATHER-Operation
        - » verwendet Index-Vektor und holt den Vektor, dessen Elemente an den Adressen liegen, die durch Addition einer Basisadresse und den Offsets im Index-Register berechnet werden
        - » Nicht gepackte Darstellung im Vektorregister
      - SCATTER-Operation
        - » Speichern der gepackten Darstellung

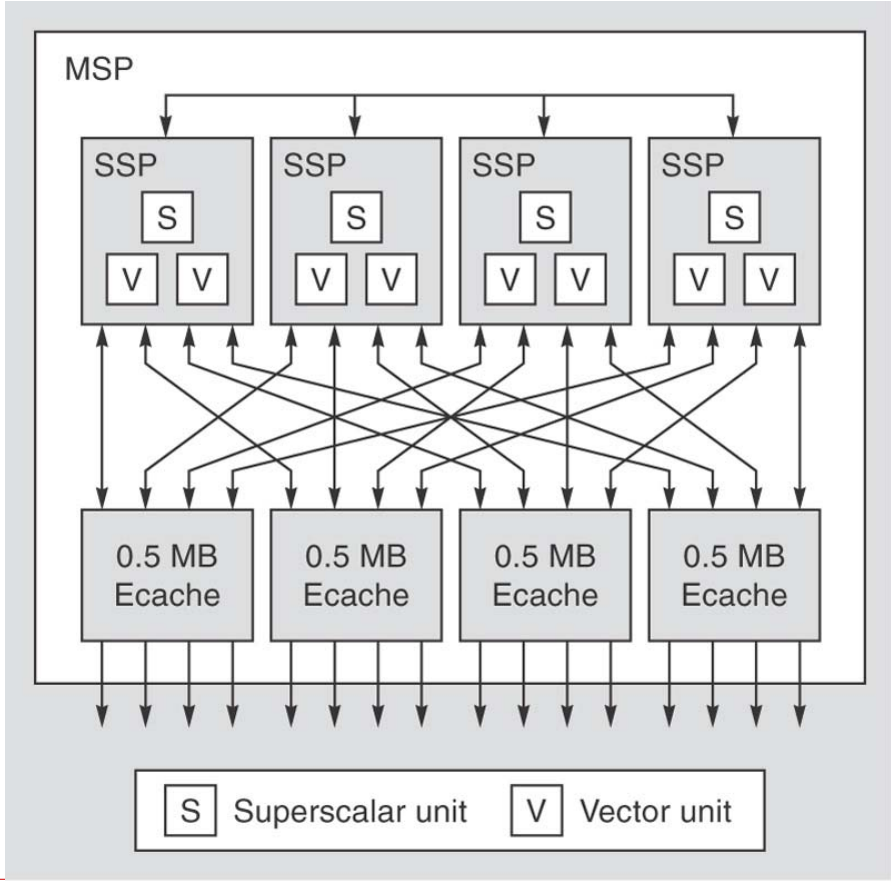
- Vektorprozessoren
  - Dünn besetzte Matrizen
    - SCATTER-GATHER Operationen mit Index-Vektoren

```
LV        Vk,Rk          ;load K
LVI       Va,(Ra+Vk)     ;load A(K(I))
LV        Vm,Rm          ;load M
LVI       Vc,(Rc+Vm)     ;load C(M(I))
ADDV.D    Va,Va,Vc       ;add them
SVI       (Ra+Vk),Va     ;store A(K(I))
```

    - Problem für vektorisierenden Compiler: konservative Annahmen wegen Speicherreferenzen
      - Verwendung einer Software-Hash Tabelle, ähnlich der ALAT im Intanium
        - » erkennt, wenn zwei Elemente innerhalb einer Interation auf dieselbe Adresse zeigen

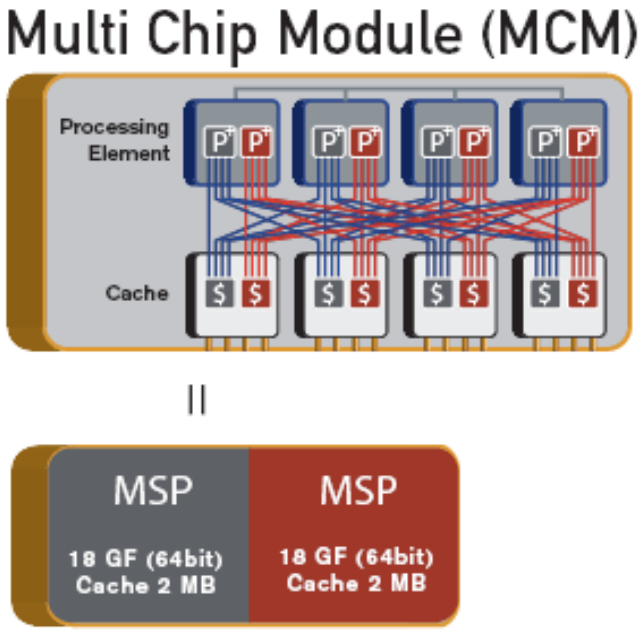
- Vektorprozessoren
  - Beispiel: CRAY X1 (200“
    - ausbarbar auf mehrere tausend Vektorprozessoren
    - Shared-Memory Architektur
    - Prozessorarchitektur: Multi-Streaming Prozessor (MSP) mit
      - 4 Single-Streaming Prozessoren (SSP)
        - » Jeder SSP ist ein Vektorprozessor mit einer skalaren Einheit.

- Vektorprozessoren
  - Beispiel: CRAY X1 (2002)
    - Multi-Streaming-Prozessor (MSP)

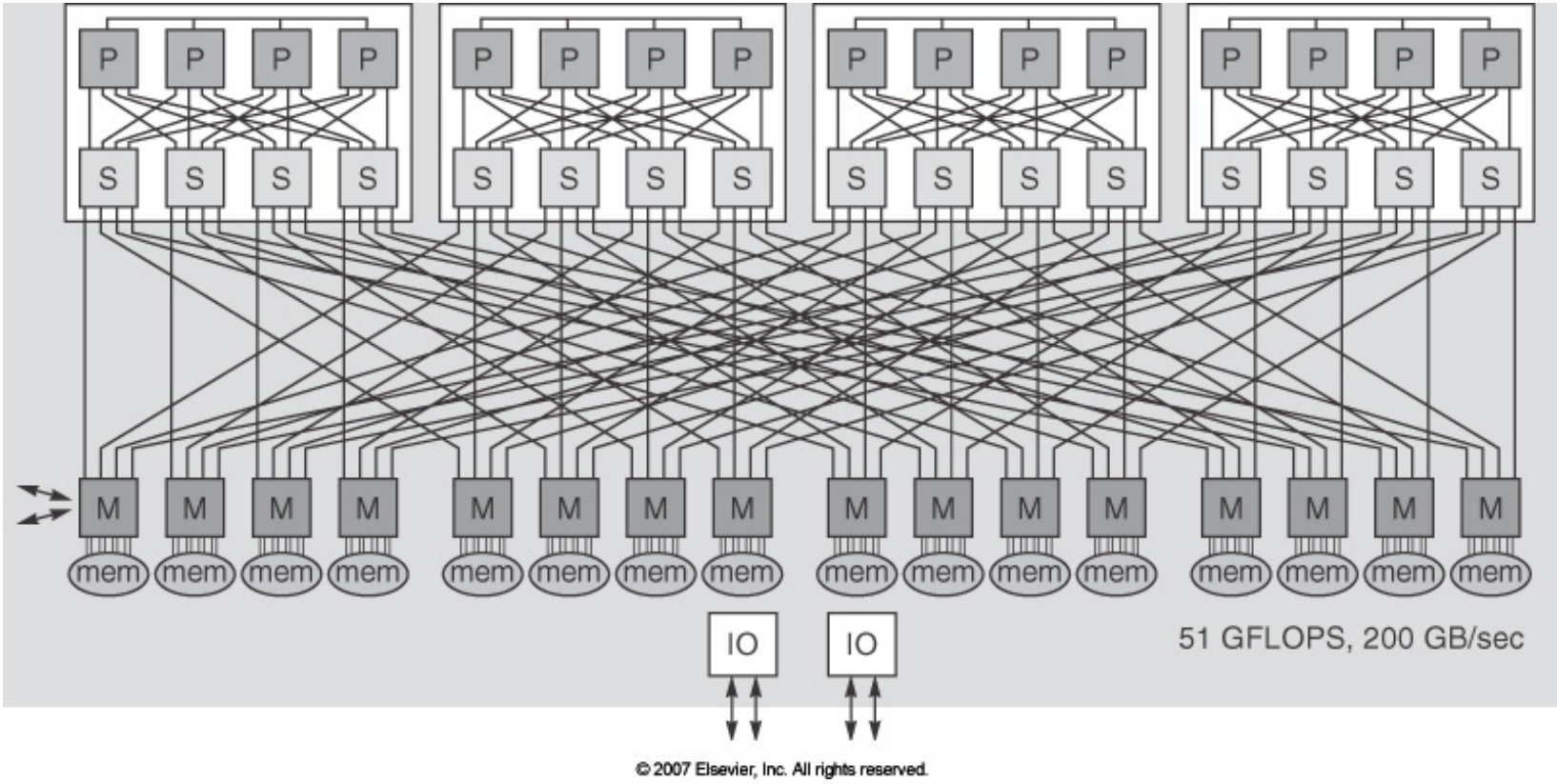


- Vektorprozessoren
  - Beispiel: CRAY X1 (2002)
    - Single-Streaming-Prozessor (MSP)
      - skalare Einheit
        - » Superskalarer Prozessor
        - » 16 KB Befehls-Cache und 16 KB Write-through Daten-Cache, zweifach-satz-assoziativ mit 32-Bytes langen Zéilen.
      - Vektoreinheit
        - » Vektorregisterdatei
        - » drei Vektor-Arithmeische Einheiten
        - » eine Vektor-Lade- und Speichereinheit

- Vektorprozessoren
  - Beispiel: CRAY X1 (2002)
    - Single-Streaming-Prozessor (MSP)



- Vektorprozessoren
  - Beispiel: CRAY X1 (2002)
    - Knoten:



- Vektorprozessoren
  - Beispiel: CRAY X1 (2002)
    - Konfigurationen

Cray X1E Supercomputer Sample Configurations

	1 AC* Cabinet	1 LC* Cabinet	4 LC Cabinets	8 LC Cabinets
MSPs	32	128	512	1,024
Peak Performance	576 GFLOPS	2.3 TFLOPS	9.2 TFLOPS	18.4 TFLOPS
Maximum Memory	128 GB	512 GB	2 TB	4 TB
Aggregate Peak Memory Bandwidth	800 GB/s	3.2 TB/s	12.8 TB/s	25.6 TB/s

*Air Cooled (AC) Liquid Cooled (LC)

- Vektorprozessoren
  - Beispiel: CRAY X1 (2002)
    - Merkmale

CPU	64-bit Cray X1E Multistreaming Processor (MSP); 8 per compute module
	8 vector pipes per MSP
	Scalar operations overlapped with vector operations
	IEEE floating point compatible
	Bit matrix multiply, pop count, and integer add/subtract with carry
Cache	2MB cache per MSP
FLOPS	18 GFLOPS theoretical peak performance per MSP (1.13 GHz vector clock speed)
SMP	4-way SMP node
Main Memory	16 GB or 32 GB RDRAM memory per compute module
Memory Bandwidth	200 GB/s per compute module peak bandwidth; 34 GB/s per processor peak bandwidth

- Vektorprozessoren
  - Beispiel: CRAY X1 (2002)
    - Merkmale

Interconnect	Custom high performance interconnect providing distributed shared memory access through entire system
	16 parallel 2D torus networks
	51 GB/s per compute module peak bandwidth
	5 microsecond MPI latency between processors
External I/O	Peak bandwidth of 4.8 GB/s through dedicated I/O channels from each compute module
	Shared memory allows any processor to perform I/O to any I/O channel
	Separate Cray Network Server (CNS) supports network I/O
File System	XFS (direct-attached disk)
	ADIC StorNext File System (SAN-attached disk)

- Vektorprozessoren
  - Beispiel: CRAY X1 (2002)
    - Merkmale

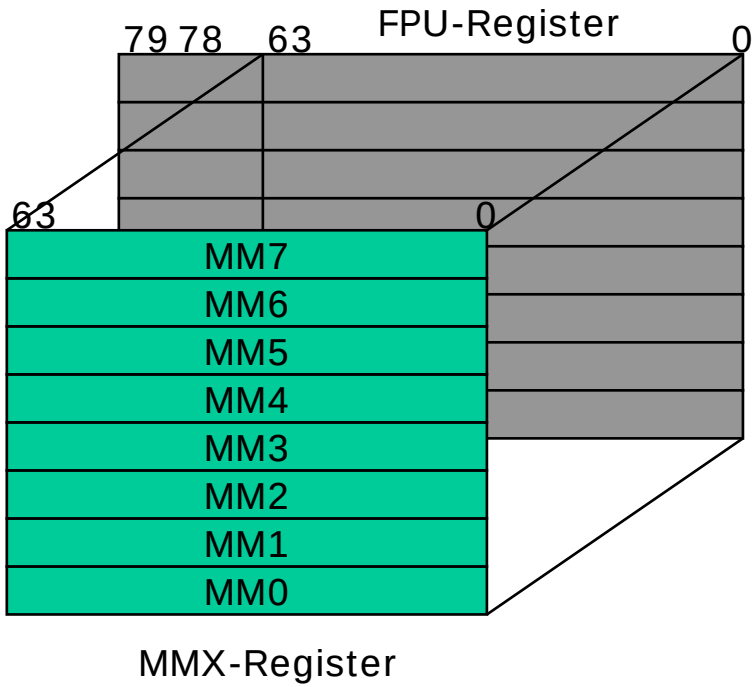
Reliability Features	System is resilient to hardware or software failures in application processors	
	System can be run in degraded mode if necessary due to hardware failures	
Operating System	UNICOS/mp	
Parallel Programming	MPI 1.2, SHMEM, OpenMP, Co-Array Fortran, UPC	
Cray Fortran Compiler	Fully adheres to Fortran 95 standard (ISO/IEC 1539-1:1997 Part 1); supports selected features from proposed Fortran 2003 standard	
Cray C & C++ Compilers	Adhere to industry standards for C (ISO/IEC 9899:1999 (C99)) and C++ (ISO/IEC 14882:1998)	
Cabinets	**Liquid-cooled (LC) cabinet** up to 16 compute modules (128 processors)	**Air-cooled (AC) cabinet** up to 4 compute modules (32 processors)
Maximum Configuration	Up to 64 LC cabinets (8,192 processors)	Up to 4 AC cabinets (128 processors)
Standard Configuration*	Up to 8 LC cabinets (1,024 processors)	1 AC cabinet (32 processors)
Power (cabinet)	65 kW, 200-208 VAC	17 kW, 200-208 VAC
Footprint (cabinet)	50.75 in. x 103 in. (1.3 m x 2.6 m)	35.5 in. x 59.75 in. (.9 m x 1.5 m)
Weight (cabinet)	5,754 lbs. (2,610 kg)	1,973 lbs. (895 kg)

* Configurations exceeding these limits, up to the design parameters of the system, are available by special order.

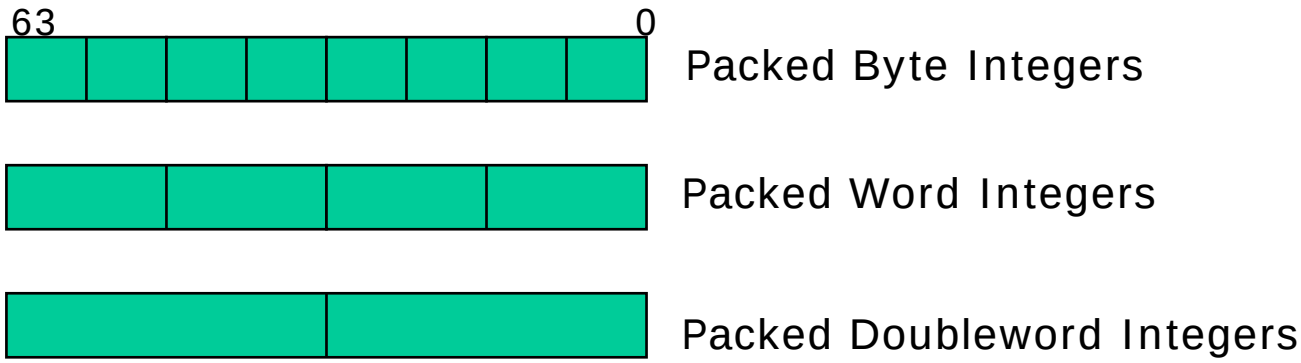
- **Kapitel 5: SIMD-Verarbeitung**

5.3: SIMD-Verarbeitung in Mikroprozessoren

- Beispiel Intel MMX-Technologie
  - MMX-Register, abgebildet auf FPU-Register

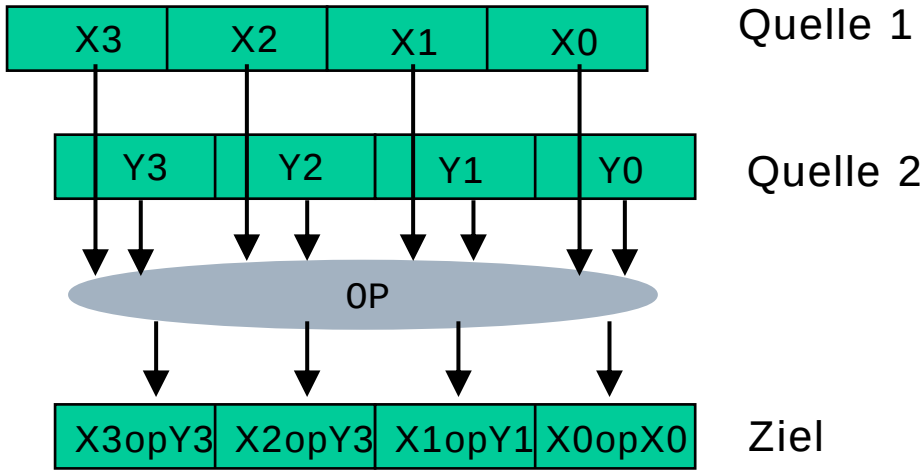


- Beispiel Intel MMX-Technologie
  - MMX-Datentypen und Formate

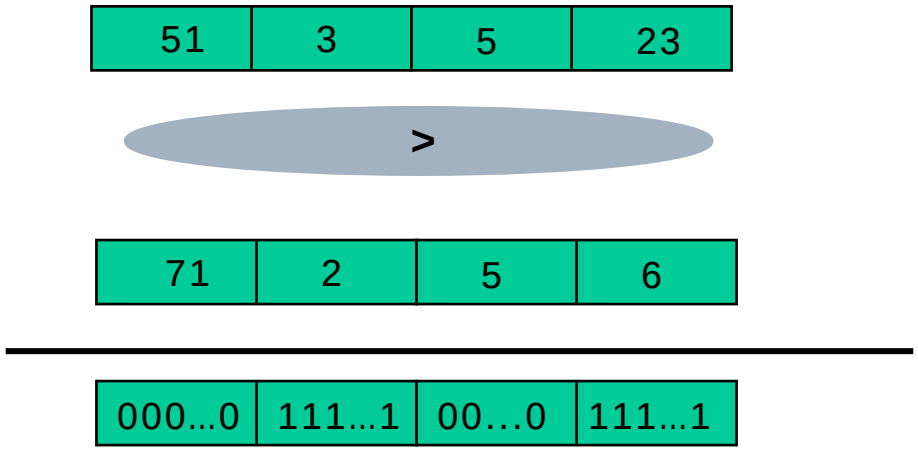


- Beispiel Intel MMX-Technologie
  - MMX-Befehle
    - Datentransport
    - Konvertierung
    - Gepackte arithmetische Befehle
    - Vergleichsbefehle
    - Logische Befehle
    - Zustandsverwaltung

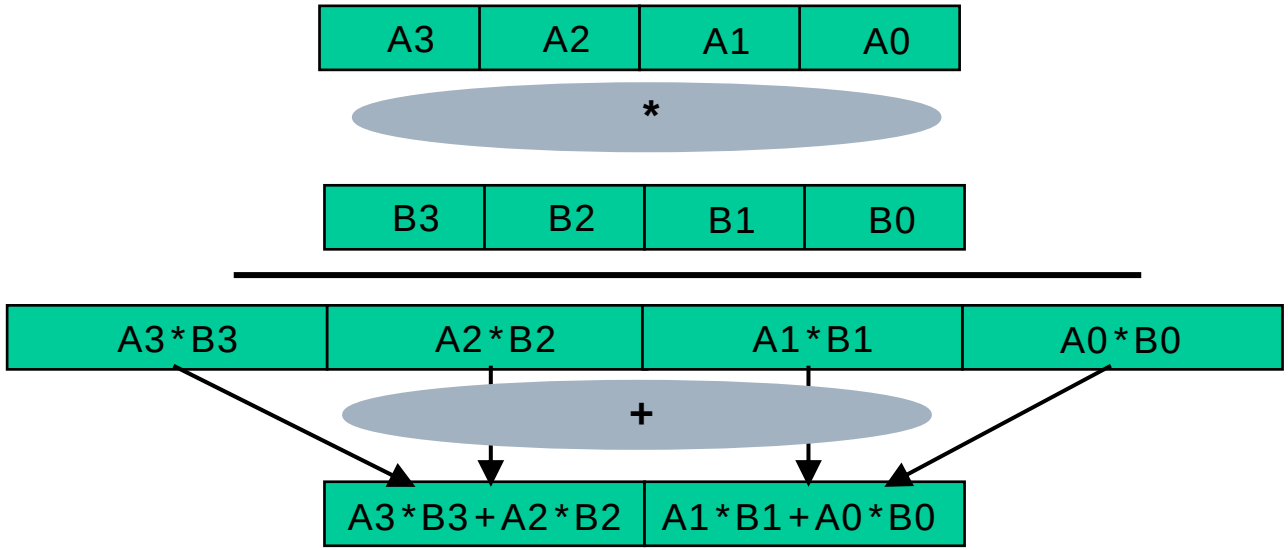
- Beispiel Intel MMX-Technologie
  - SIMD-Verarbeitung



- Beispiel Intel MMX-Technologie
  - SIMD-Verarbeitung
    - **PCMPGTW**-Befehl auf Wortoperanden (Vergleich größer als)



- Beispiel Intel MMX-Technologie
  - SIMD-Verarbeitung
    - **PMADDWD**-Befehl auf Wortoperanden, Ergebnis im Doppelwort (Multiply and add)



- Beispiel Intel MMX-Technologie
  - Saturation Arithmetik
    - Algorithmen in der graphischen Datenverarbeitung vermeiden Überlauf und Unterlauf bei der Addition und Subtraktion von nicht vorzeichenbehafteten Pixeln
    - Übergang zum größten oder kleinsten darstellbaren Wert

```
  10101010
+ 11001100
  11111111
```

    - Keine Überprüfung, ob Über- oder Unterlauf des Zahlenbereichs, keine Ausnahmeverarbeitung

- Intel MMX-Technologie
  - Beispiel: Chroma keying („bluebox“ Technik)
    - Überlagerung des Bildes einer Frau auf ein Bild mit einer Blume mit Hilfe eines blauen Hintergrundes
      - x: Bild der Frau auf dem blauen Hintergrund
      - y: Blumenbild
      - new_image: neues Bild

```
for (i=0; i<image_size; i++) {
   if (x[i]==blue) new_image[i]=y[i]
           else    new_image[i]=x[i]
}
```

- Intel MMX-Technologie
  - Beispiel: Chroma keying („bluebox“ Technik)
    - MMX Code-Folge

```
movq      mm3,mem1      #load 8 pixels from woman's image (addr. mem1)
movq      mm4,mem2      #load 8 pixels from flower's image (addr. mem2)
pcmpeqb   mm1,mm3       #gen. Selection bit mask into mm1 (orig. „blue“)
pand      mm4,mm1       #AND; use the bit mask for cond. select into mm4.
pandn     mm1,mm3       #(NOT mm1) AND mm3; -“- into mm1
por       mm4,mm1       #OR; result into mm4
```

pcmpeqb mm1,mm3

mm1	blue	blue	blue	blue	blue	blue	blue	blue
mm3	x7!=blue	x6!=blue	x5=blue	x4=blue	x3!=blue	x2!=blue	x1=blue	x0=blue
mm1	0x0000	0x0000	0xFFFF	0xFFFF	0x0000	0x0000	0xFFFF	0xFFFF

pand mm4,mm1

mm4	y7	y6	y5	y4	y3	y2	y1	y0
mm1	0x0000	0x0000	0xFFFF	0xFFFF	0x0000	0x0000	0xFFFF	0xFFFF
mm4	0x0000	0x0000	y5	y4	0x0000	0x0000	y1	y0

pandn mm1,mm3

mm1	0x0000	0x0000	0xFFFF	0xFFFF	0x0000	0x0000	0xFFFF	0xFFFF
mm3	x7	x6	x5	x4	x3	x2	x1	x0
mm1	x7	x6	0x0000	0x0000	x3	x2	0x0000	0x0000

por mm4,mm1

mm1	x7	x6	y5	y4	x3	x2	y1	y0