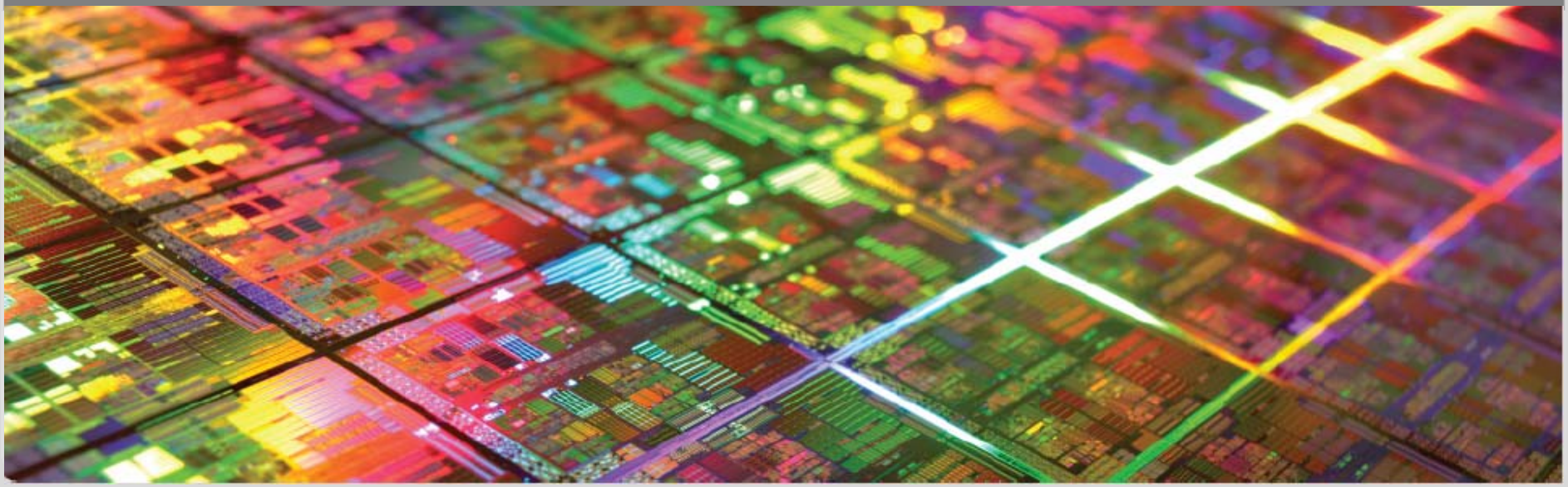
Karlsruher Institut für Technologie

# Rechnerstrukturen

## Vorlesung im Sommersemester 2010

**Prof. Dr. Wolfgang Karl**

Fakultät für Informatik – Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

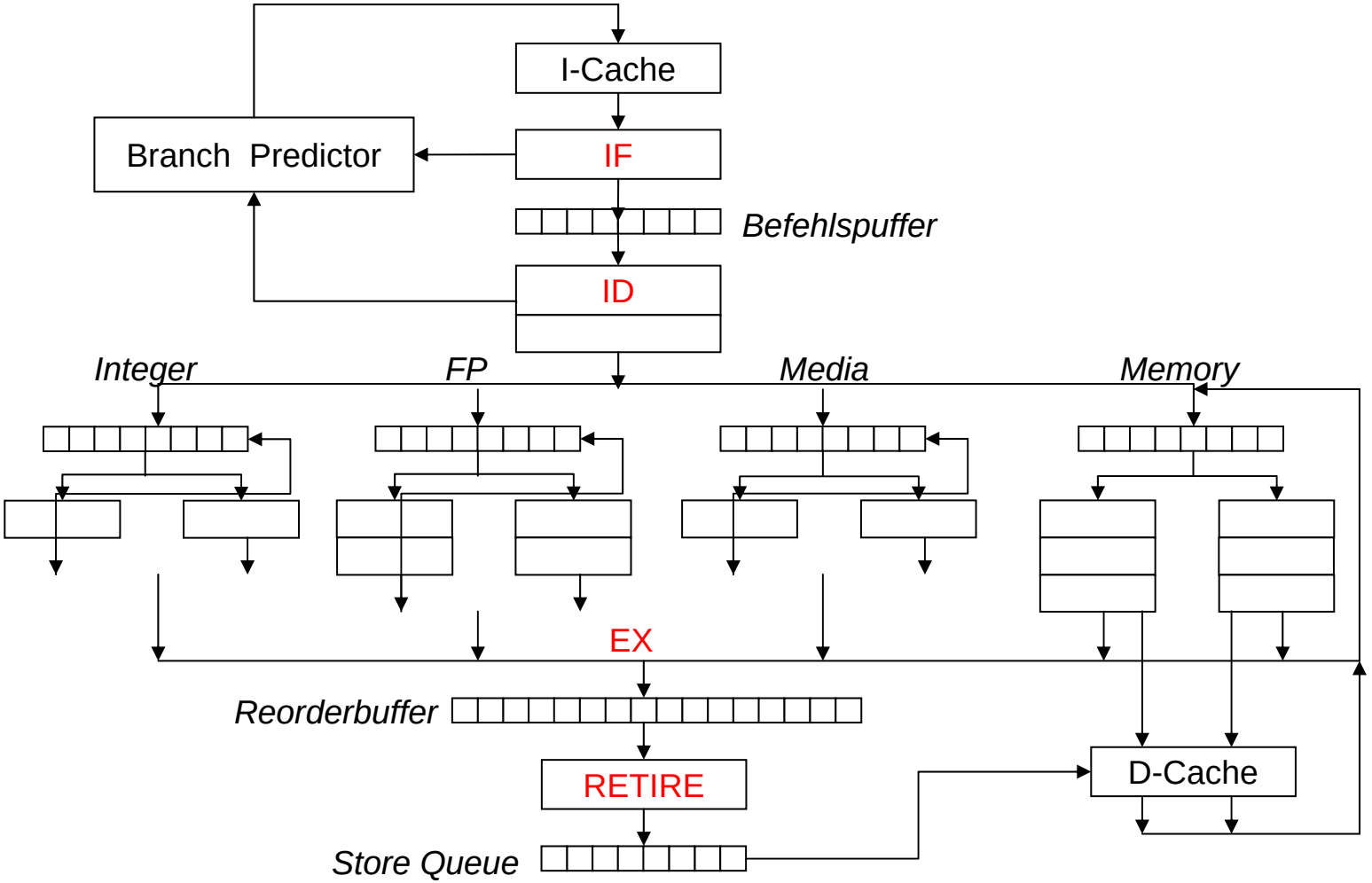


# Vorlesung Rechnerstrukturen

- **Kapitel 2: Parallelismus auf Maschinenbefehlsebene**
- 2.1 Nebenläufigkeit
- 2.1.1 Superskalartechnik

# Superskalartechnik

## Superskalarer Prozessor



# Superskalartechnik

- **Dynamische Methoden zur Erkennung und Auflösung von Datenkonflikten**
- Detaillierte Betrachtung der Zuordnungsphase
- Fallstudie: Tomasulo (IBM 360/91)
  - Konfliktauflösung und Ablaufsteuerung verteilt;
  - jede Funktionseinheit verfügt über eine Reservierungstabelle mit möglicherweise mehreren Zeilen (Einträgen);
  - Reservierungstabelle (Umordnungspuffer, Reservation Station)
    - übernimmt die Kontrolle über die Abarbeitung eines Maschinenbefehls, wenn dieser von der Decodiereinheit zur Ausführung angestoßen wird (Issue);
  - Ergebnisbus:
    - ein von einer Funktionseinheit i produziertes Ergebnis wird direkt an die Funktionseinheit j weitergegeben, wenn diese das Ergebnis als Operand benötigt;
    - alle Funktionseinheiten, die auf einen Operanden warten, werden gleichzeitig bedient;

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Umordnungspuffer (Reservierungstabelle, Reservation Station)
  - Jeder Eintrag enthält jeweils Felder für:
    - die zwei möglichen Quelloperanden (Src1, Src2);
    - die Nummern (Namen, Tags) der Reservierungstabellen derjenigen Funktionseinheiten, welche die Quelloperanden für die auszuführende Operation liefern werden (RS1, RS2),
    - jeweils ein Flag für jeden Operanden, das anzeigt, ob ein Operand verfügbar ist (Vld1,Vld2) und
    - einen Namen (destination tag) für das Ziel (Dest).

| | Quelloperand 1 | | | Quelloperand 2 | | | Ziel |
|---|---|---|---|---|---|---|---|
| | Vld1 | Src1 | RS1 | Vld2 | Src2 | RS2 | Dest |
| Befehl n | | | | | | | |
| Befehl n+1 | | | | | | | |
| Befehl n+2 | | | | | | | |

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Registerdatei
  - Jedes Register einer Registerdatei enthält
    - das Feld für den Wert (Registerinhalt)
    - einen Namen (destination tag, Dest), der mit dem Ergebnis assoziiert ist, und
    - ein Bit, das anzeigt, ob der Wert für das Register gerade berechnet wird.

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Befehlsausführung
  - Ein Maschinenbefehl kann zur Ausführung angestoßen werden, falls ein Eintrag in der Reservierungstabelle einer Funktionseinheit frei ist.
    - Falls alle Einträge belegt sind, dann ist ein Ressourcenkonflikt gegeben, und der Maschinenbefehl muss warten, bis ein Eintrag in der Reservierungstabelle frei ist.
    - Für einen von der Decodiereinheit zur Ausführung angestoßener Maschinenbefehl werden die Inhalte seiner Quellregister und die dazugehörigen Ready-Bits von der Registerdatei in die entsprechenden Felder des Umordnungspuffers kopiert.

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Phasen der Pipeline (vereinfacht)
  - Issue
    - Holen der Befehle aus Instruction Queue
    - Holen der Operanden
  - Ausführung
    - Wenn die Operanden verfügbar sind, dann Anstoßen der Ausführung auf Funktionseinheit (Dispatch)
    - Beobachten des Ergebnisbusses
    - Befehlsausführung nicht in Programmreihemfolge (out-of-order execution)
  - Rückschreibphase
    - Schreiben der Ergebnisse auf Ergebnisbus
    - Kennzeichnen des Umordnungspuffers als verfügbar

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Beispiel (T. Ungerer)

**registers**

| R | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Value | - | - | (R3) | (R4) | (R5) | - |
| Vld | 1 | 1 | 1 | 1 | 1 | 1 |
| RS | 0 | 0 | 0 | 0 | 0 | 0 |

**register status**

```
mul Reg1, Reg3, Reg5
sub Reg2, Reg4, Reg3
div Reg6, Reg1, Reg4
add Reg4, Reg2, Reg3
```

reservation stations

| | | Empty | InFU | Op | Dest | Src1 | Vld1 | RS1 | Src2 | Vld2 | RS2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_{add}$ | 1 | 1 | | | | | | | | | |
| | 2 | 1 | | | | | | | | | |
| $S_{mul}$ | 3 | 1 | | | | | | | | | |
| $S_{div}$ | 4 | 1 | | | | | | | | | |

**RS status**

cycle 0

token.tag
token.data

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Beispiel (T. Ungerer)

```
mul Reg1, Reg3, Reg5
sub Reg2, Reg4, Reg3
div Reg6, Reg1, Reg4
add Reg4, Reg2, Reg3
```

**registers**

| R | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Value | - | - | (R3) | (R4) | (R5) | - |
| Vld | 0 | 1 | 1 | 1 | 1 | 1 |
| RS | 3 | 0 | 0 | 0 | 0 | 0 |

**register status**

reservation stations

| | | Empty | InFU | Op | Dest | Src1 | Vld1 | RS1 | Src2 | Vld2 | RS2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_{add}$ | 1 | 1 | | | | | | | | | |
| $S_{add}$ | 2 | 1 | | | | | | | | | |
| $S_{mul}$ | 3 | 0 | 0 | mul | 1 | (R3) | 1 | 0 | (R5) | 1 | 0 |
| $S_{div}$ | 4 | 1 | | | | | | | | | |

**RS status**

cycle 1

token.tag

token.data

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Beispiel (T. Ungerer)

```
mul Reg1, Reg3, Reg5
sub Reg2, Reg4, Reg3
div Reg6, Reg1, Reg4
add Reg4, Reg2, Reg3
```

**registers**

| R | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Value | - | - | (R3) | (R4) | (R5) | - |
| Vld | 0 | 0 | 1 | 1 | 1 | 1 |
| RS | 3 | 1 | 0 | 0 | 0 | 0 |

**register status**

reservation stations

| | | Empty | InFU | Op | Dest | Src1 | Vld1 | RS1 | Src2 | Vld2 | RS2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_{add}$ | 1 | 0 | 0 | sub | 2 | (R4) | 1 | 0 | (R3) | 1 | 0 |
| | 2 | 1 | | | | | | | | | |
| $S_{mul}$ | 3 | 0 | 1 | mul | 1 | (R3) | 1 | 0 | (R5) | 1 | 0 |
| $S_{div}$ | 4 | 1 | | | | | | | | | |

**RS status**

cycle 2

token.tag

token.data

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Beispiel (T. Ungerer)

```
mul Reg1, Reg3, Reg5
sub Reg2, Reg4, Reg3
div Reg6, Reg1, Reg4
add Reg4, Reg2, Reg3
```

**registers**

| R | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Value | - | - | (R3) | (R4) | (R5) | - |
| Vld | 0 | 0 | 1 | 1 | 1 | 0 |
| RS | 3 | 1 | 0 | 0 | 0 | 4 |

**register status**

reservation stations

| | | Empty | InFU | Op | Dest | Src1 | Vld1 | RS1 | Src2 | Vld2 | RS2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_{add}$ | 1 | 0 | 1 | sub | 2 | (R4) | 1 | 0 | (R3) | 1 | 0 |
| | 2 | 1 | | | | | | | | | |
| $S_{mul}$ | 3 | 0 | 1 | mul | 1 | (R3) | 1 | 0 | (R5) | 1 | 0 |
| $S_{div}$ | 4 | 0 | 0 | div | 6 | | 0 | 3 | (R4) | 1 | 0 |

**RS status**

cycle 3

token.tag

token.data

3

remaining cycles in FU

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Beispiel (T. Ungerer)

```
mul Reg1, Reg3, Reg5
sub Reg2, Reg4, Reg3
div Reg6, Reg1, Reg4
add Reg4, Reg2, Reg3
```

**registers**

| R | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Value | - | (R4)-(R3) | (R3) | - | (R5) | - |
| Vld | 0 | 1 | 1 | 0 | 1 | 0 |
| RS | 3 | 0 | 0 | 2 | 0 | 4 |

**register status**

reservation stations

| | | Empty | InFU | Op | Dest | Src1 | Vld1 | RS1 | Src2 | Vld2 | RS2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_{add}$ | 1 | 1 | 1 | sub | 2 | (R4) | 1 | 0 | (R3) | 1 | 0 |
| | 2 | 0 | 0 | add | 4 | (R4)-(R3) | 1 | 0 | (R3) | 1 | 0 |
| $S_{mul}$ | 3 | 0 | 1 | mul | 1 | (R3) | 1 | 0 | (R5) | 1 | 0 |
| $S_{div}$ | 4 | 0 | 0 | div | 6 | | 0 | 3 | (R4) | 1 | 0 |

2

**RS status**

cycle 4

token.tag 1

token.data (R4)-(R3)

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Beispiel (T. Ungerer)

```
mul Reg1, Reg3, Reg5
sub Reg2, Reg4, Reg3
div Reg6, Reg1, Reg4
add Reg4, Reg2, Reg3
```

**registers**

| R | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Value | - | (R4)-(R3) | (R3) | - | (R5) | - |
| Vld | 0 | 1 | 1 | 0 | 1 | 0 |
| RS | 3 | 0 | 0 | 2 | 0 | 4 |

**register status**

reservation stations

| | | Empty | InFU | Op | Dest | Src1 | Vld1 | RS1 | Src2 | Vld2 | RS2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_{add}$ | 1 | 1 | 1 | sub | 2 | (R4) | 1 | 0 | (R3) | 1 | 0 | |
| $S_{add}$ | 2 | 0 | 1 | add | 4 | (R4)-(R3) | 1 | 0 | (R3) | 1 | 0 | |
| $S_{mul}$ | 3 | 0 | 1 | mul | 1 | (R3) | 1 | 0 | (R5) | 1 | 0 | 1 |
| $S_{div}$ | 4 | 0 | 0 | div | 6 | | 0 | 3 | (R4) | 1 | 0 | |

**RS status**

cycle 5

token.tag

token.data

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Beispiel (T. Ungerer)

```
mul Reg1, Reg3, Reg5
sub Reg2, Reg4, Reg3
div Reg6, Reg1, Reg4
add Reg4, Reg2, Reg3
```

**registers**

| R | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Value | - | (R4)-(R3) | (R3) | (R4)-(R3)+(R3) | (R5) | - |
| Vld | 0 | 1 | 1 | 1 | 1 | 0 |
| RS | 3 | 0 | 0 | 0 | 0 | 4 |

**register status**

reservation stations

| | | Empty | InFU | Op | Dest | Src1 | Vld1 | RS1 | Src2 | Vld2 | RS2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_{add}$ | 1 | 1 | 1 | sub | 2 | (R4) | 1 | 0 | (R3) | 1 | 0 | |
| $S_{add}$ | 2 | 1 | 1 | add | 4 | (R4)-(R3) | 1 | 0 | (R3) | 1 | 0 | |
| $S_{mul}$ | 3 | 0 | 1 | mul | 1 | (R3) | 1 | 0 | (R5) | 1 | 0 | 0 |
| $S_{div}$ | 4 | 0 | 0 | div | 6 | | 0 | 3 | (R4) | 1 | 0 | |

**RS status**

cycle 6

token.tag 2

token.data (R4)-(R3)+(R3)

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Beispiel (T. Ungerer)

```
mul Reg1, Reg3, Reg5
sub Reg2, Reg4, Reg3
div Reg6, Reg1, Reg4
add Reg4, Reg2, Reg3
```

**registers**

| R | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Value | (R3)*(R5) | (R4)-(R3) | (R3) | (R4)-(R3)+(R3) | (R5) | - |
| Vld | 1 | 1 | 1 | 1 | 1 | 0 |
| RS | 0 | 0 | 0 | 0 | 0 | 4 |

**register status**

reservation stations

| | | Empty | InFU | Op | Dest | Src1 | Vld1 | RS1 | Src2 | Vld2 | RS2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_{add}$ | 1 | 1 | 1 | sub | 2 | (R4) | 1 | 0 | (R3) | 1 | 0 |
| $S_{add}$ | 2 | 1 | 1 | add | 4 | (R4)-(R3) | 1 | 0 | (R3) | 1 | 0 |
| $S_{mul}$ | 3 | 1 | 1 | mul | 1 | (R3) | 1 | 0 | (R5) | 1 | 0 |
| $S_{div}$ | 4 | 0 | 0 | div | 6 | (R3)*(R5) | 1 | 0 | (R4) | 1 | 0 |

**RS status**

cycle 7

token.tag 3

token.data (R3)*(R5)

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Beispiel (T. Ungerer)

```
mul Reg1, Reg3, Reg5
sub Reg2, Reg4, Reg3
div Reg6, Reg1, Reg4
add Reg4, Reg2, Reg3
```

**registers**

| R | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Value | (R3)*(R5) | (R4)-(R3) | (R3) | (R4)-(R3)+(R3) | (R5) | - |
| Vld | 1 | 1 | 1 | 1 | 1 | 0 |
| RS | 0 | 0 | 0 | 0 | 0 | 4 |

**register status**

reservation stations

| | | Empty | InFU | Op | Dest | Src1 | Vld1 | RS1 | Src2 | Vld2 | RS2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_{add}$ | 1 | 1 | 1 | sub | 2 | (R4) | 1 | 0 | (R3) | 1 | 0 |
| $S_{add}$ | 2 | 1 | 1 | add | 4 | (R4)-(R3) | 1 | 0 | (R3) | 1 | 0 |
| $S_{mul}$ | 3 | 1 | 1 | mul | 1 | (R3) | 1 | 0 | (R5) | 1 | 0 |
| $S_{div}$ | 4 | 0 | 1 | div | 6 | (R3)*(R5) | 1 | 0 | (R4) | 1 | 0 |

**RS status**

cycle 8

token.tag
token.data

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Beispiel (T. Ungerer)

```
mul Reg1, Reg3, Reg5
sub Reg2, Reg4, Reg3
div Reg6, Reg1, Reg4
add Reg4, Reg2, Reg3
```

**registers**

| R | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Value | (R3)*(R5) | (R4)-(R3) | (R3) | (R4)-(R3)+(R3) | (R5) | - |
| Vld | 1 | 1 | 1 | 1 | 1 | 0 |
| RS | 0 | 0 | 0 | 0 | 0 | 4 |

**register status**

reservation stations

| | | Empty | InFU | Op | Dest | Src1 | Vld1 | RS1 | Src2 | Vld2 | RS2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_{add}$ | 1 | 1 | 1 | sub | 2 | (R4) | 1 | 0 | (R3) | 1 | 0 | |
| $S_{add}$ | 2 | 1 | 1 | add | 4 | (R4)-(R3) | 1 | 0 | (R3) | 1 | 0 | |
| $S_{mul}$ | 3 | 1 | 1 | mul | 1 | (R3) | 1 | 0 | (R5) | 1 | 0 | |
| $S_{div}$ | 4 | 0 | 1 | div | 6 | (R3)*(R5) | 1 | 0 | (R4) | 1 | 0 | 3 |

**RS status**

cycle 9

token.tag
token.data

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Beispiel (T. Ungerer)

```
mul Reg1, Reg3, Reg5
sub Reg2, Reg4, Reg3
div Reg6, Reg1, Reg4
add Reg4, Reg2, Reg3
```

**registers**

| R | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Value | (R3)*(R5) | (R4)-(R3) | (R3) | (R4)-(R3)+(R3) | (R5) | - |
| Vld | 1 | 1 | 1 | 1 | 1 | 0 |
| RS | 0 | 0 | 0 | 0 | 0 | 4 |

**register status**

reservation stations

| | | Empty | InFU | Op | Dest | Src1 | Vld1 | RS1 | Src2 | Vld2 | RS2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_{add}$ | 1 | 1 | 1 | sub | 2 | (R4) | 1 | 0 | (R3) | 1 | 0 | |
| $S_{add}$ | 2 | 1 | 1 | add | 4 | (R4)-(R3) | 1 | 0 | (R3) | 1 | 0 | |
| $S_{mul}$ | 3 | 1 | 1 | mul | 1 | (R3) | 1 | 0 | (R5) | 1 | 0 | |
| $S_{div}$ | 4 | 0 | 1 | div | 6 | (R3)*(R5) | 1 | 0 | (R4) | 1 | 0 | 2 |

**RS status**

cycle 10

token.tag
token.data

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Beispiel (T. Ungerer)

```
mul Reg1, Reg3, Reg5
sub Reg2, Reg4, Reg3
div Reg6, Reg1, Reg4
add Reg4, Reg2, Reg3
```

**registers**

| R | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Value | (R3)*(R5) | (R4)-(R3) | (R3) | (R4)-(R3)+(R3) | (R5) | - |
| Vld | 1 | 1 | 1 | 1 | 1 | 0 |
| RS | 0 | 0 | 0 | 0 | 0 | 4 |

**register status**

reservation stations

| | | Empty | InFU | Op | Dest | Src1 | Vld1 | RS1 | Src2 | Vld2 | RS2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_{add}$ | 1 | 1 | 1 | sub | 2 | (R4) | 1 | 0 | (R3) | 1 | 0 | |
| $S_{add}$ | 2 | 1 | 1 | add | 4 | (R4)-(R3) | 1 | 0 | (R3) | 1 | 0 | |
| $S_{mul}$ | 3 | 1 | 1 | mul | 1 | (R3) | 1 | 0 | (R5) | 1 | 0 | |
| $S_{div}$ | 4 | 0 | 1 | div | 6 | (R3)*(R5) | 1 | 0 | (R4) | 1 | 0 | 1 |

**RS status**

cycle 11

token.tag
token.data

# Superskalartechnik

- **Fallstudie: Tomasulo (IBM 360/91)**
- Beispiel (T. Ungerer)

**registers**

| R | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Value | (R3)*(R5) | (R4)-(R3) | (R3) | (R4)-(R3)+(R3) | (R5) | (R3)*(R5) /(R4) |
| Vld | 1 | 1 | 1 | 1 | 1 | 1 |
| RS | 0 | 0 | 0 | 0 | 0 | 0 |

**register status**

```
mul Reg1, Reg3, Reg5
sub Reg2, Reg4, Reg3
div Reg6, Reg1, Reg4
add Reg4, Reg2, Reg3
```

reservation stations

| | | Empty | InFU | Op | Dest | Src1 | Vld1 | RS1 | Src2 | Vld2 | RS2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_{add}$ | 1 | 1 | 1 | sub | 2 | (R4) | 1 | 0 | (R3) | 1 | 0 | |
| $S_{add}$ | 2 | 1 | 1 | add | 4 | (R4)-(R3) | 1 | 0 | (R3) | 1 | 0 | |
| $S_{mul}$ | 3 | 1 | 1 | mul | 1 | (R3) | 1 | 0 | (R5) | 1 | 0 | |
| $S_{div}$ | 4 | 1 | 1 | div | 6 | (R3)*(R5) | 1 | 0 | (R4) | 1 | 0 | 0 |

**RS status**

cycle 12

token.tag 4

token.data (R3)*(R5) /(R4)

# Superskalartechnik

- **Zusammenfassung**
  - Aus einem sequentiellen Befehlsstrom werden Befehle zur Ausführung angestoßen (zugewiesen).
  - Die Zuweisung erfolgt dynamisch durch die Hardware
  - Es kann mehr als ein Befehl zugewiesen werden.
  - Die Anzahl der zugewiesenen Befehle pro Takt wird dynamisch von der Hardware bestimmt und liegt zwischen Null und der maximalen Zuweisungsbreite.
  - Komplexe Hardware-Logik für dynamische Zuweisung notwendig.
  - Mehrere von einander unabhängige Funktionsanweisungen sind verfügbar.
  - Mikroarchitektur bestimmt superskalare Eigenschaft.

# Superskalartechnik

- **Literatur:**
  - Brinkschulte/Ungerer: Mikrocontroller und Mikroprozessoren. Springer-Verlag, 2002: Kap. 6.1-6.4, Kap. 7
  - Hennessy/Patterson: Computer Architecture – A Quantative Approach. 3. Auflage: Kap. 3

# Vorlesung Rechnerstrukturen

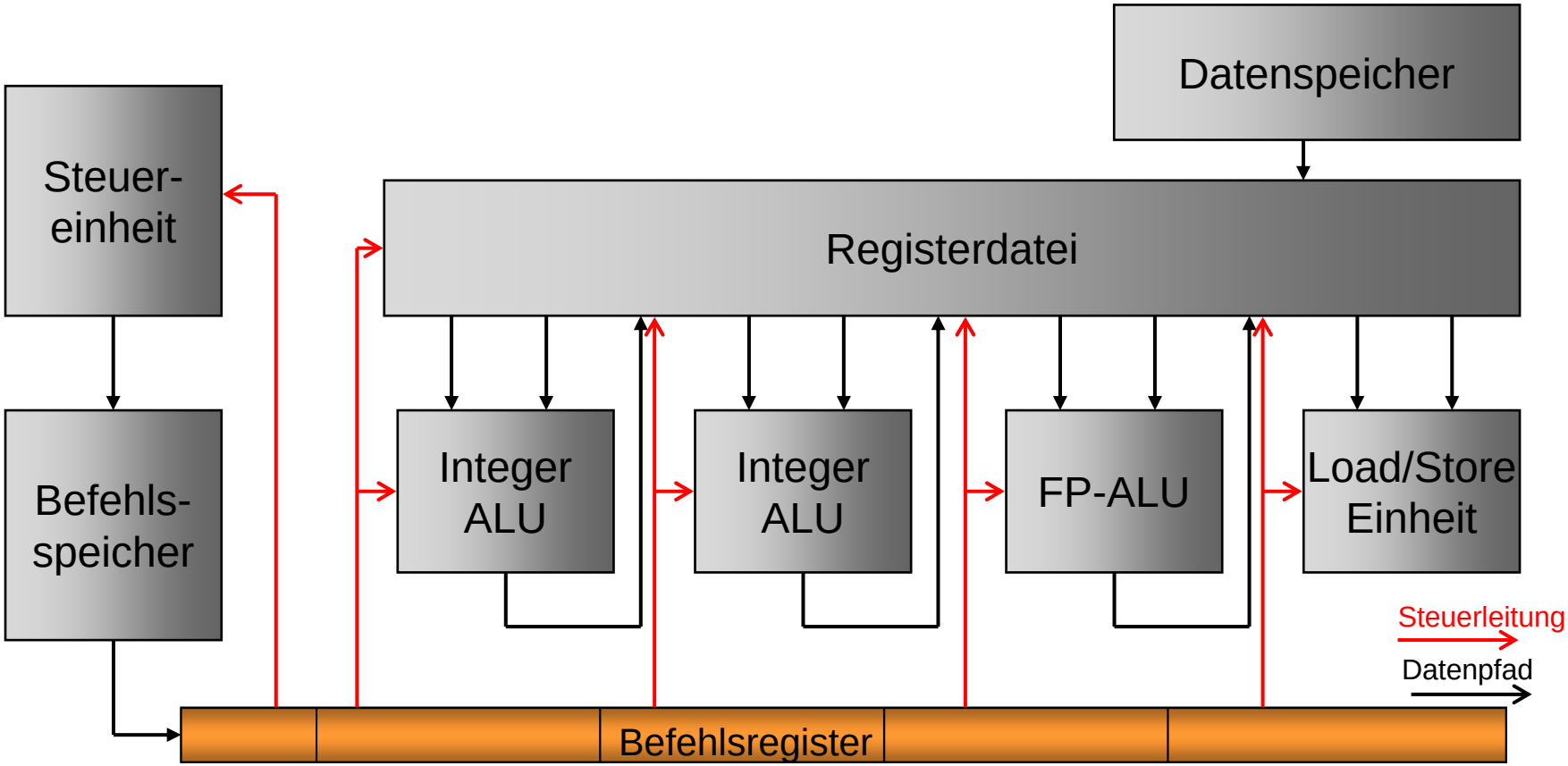
- **Kapitel 2: Parallelismus auf Maschinenbefehlsebene**
- 2.1 Nebenläufigkeit
- 2.1.1 VLIW-Architektur

# VLIW-Architektur

- **Grundprinzip VLIW**
  - Eine VLIW-Architektur (Very Long Instruction Word) ist gekennzeichnet durch ein breites Befehlsformat, das in mehrere Felder aufgeteilt ist, aus denen die Funktionseinheiten gesteuert werden;
  - Eine VLIW-Architektur mit *n* voneinander unabhängigen Funktionseinheiten kann bis zu *n* Operationen gleichzeitig ausführen;
  - Operationen: RISC-Architektur
  - Steuerung der parallelen Abarbeitung zur Übersetzungszeit (automatisch parallelisierender Compiler)

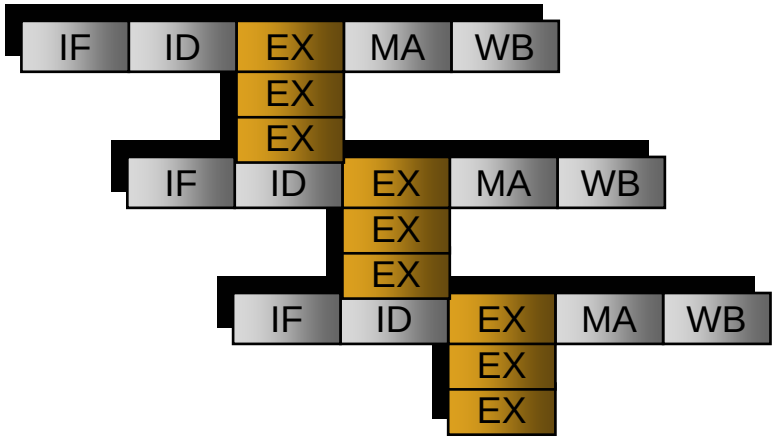
# VLIW-Architektur

- **Grundprinzip VLIW**
- Prinzipieller Aufbau



# VLIW-Architektur

- **Grundprinzip VLIW**
- Prinzipielle Pipelinestruktur



# VLIW-Architektur

- **Grundprinzip VLIW**
- Statische Steuerung der parallelen Abarbeitung
  - Aufgaben des Compilers
  - Frontend:
    - Lexikalische, syntaktische und semantische Analyse
  - Code-Generierung / Parallelisierung
    - Kontrollflussanalyse
    - Datenflussanalyse
    - Datenabhängigkeitsanalyse
  - Schleifenparallelisierung
    - Loop Unrolling
    - Software-Pipelining
  - Scheduling
    - Packen der voneinander unabhängigen Befehle in breite Befehlswörter

# VLIW-Architektur

- **Scheduling**
  - Beispiel (Quelle: K. Asanovic, MIT)

```
for (i=0;i<N;i++)
B[i]=A[i]+C
```

Übersetzung ↓

```
Loop:  ld f1,0(r1)
       add r1,8
       fadd f2,f0,f1
       sd f2, 0(r2)
       add r2,8
       bne r1,r3,loop
```

# VLIW-Architektur

- **Scheduling**
  - Beispiel (Quelle: K. Asanovic, MIT)

```
for (i=0;i<N;i++)
B[i]=A[i]+C
```

Übersetzung

```
Loop:  ld f1,0(r1)
       add r1,8
       fadd f2,f0,f1
       sd f2, 0(r2)
       add r2,8
       bne r1,r3,loop
```

Scheduling

| FE Int1 | FE Int2 | FE Mem1 | FE Mem2 | FE FP+ | FE FPx |
|---|---|---|---|---|---|
| add r1 | | ld | | | |
| | | | | | |
| | | | | | |
| | | | | fadd | |
| | | | | | |
| | | | | | |
| | | | | | |
| add r2 | bne | sd | | | |
| | | | | | |

# VLIW-Architektur

- **Scheduling**
  - Beispiel (Quelle: K. Asanovic, MIT)
  - Loop unrolling

```
for (i=0;i<N;i++)
B[i]=A[i]+C
```

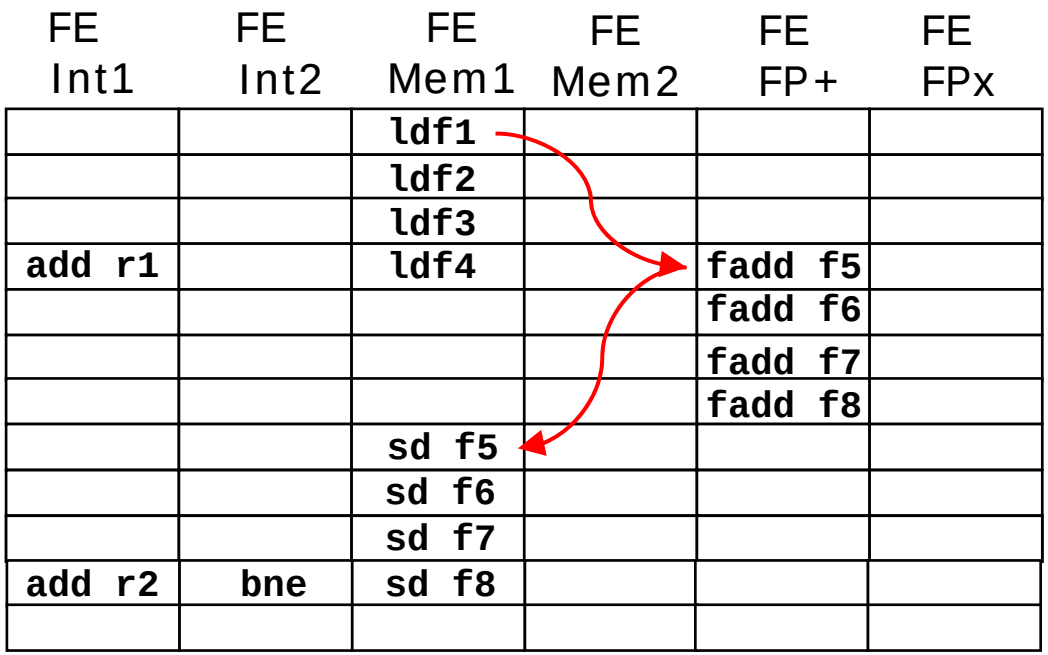
Abrollen der inneren Schleife

```
for (i=0;i<N-3;i+4)
{
B[i]=A[i]+C
B[i+1]=A[i+1]+C
B[i+2]=A[i+2]+C
B[i+3]=A[i+3]+C
}
```

# VLIW-Architektur

- **Scheduling**
  - Beispiel (Quelle: K. Asanovic, MIT)
  - Loop unrolling

```
Loop:   ld f1,0(r1)
        ld f2,8(r1)
        ld f3,16(r1)
        ld f4,24(r1)
        add r1,32
        fadd f5,f0,f1
        fadd f6,f0,f2
        fadd f7,f0,f3
        fadd f8,f0,f4
        sd f5,0(r2)
        sd f6,8(r2)
        sd f7,16(r2)
        sd f8,24(r2)
        add r2,32
        bne r1,r3,loop
```



# VLIW-Architektur

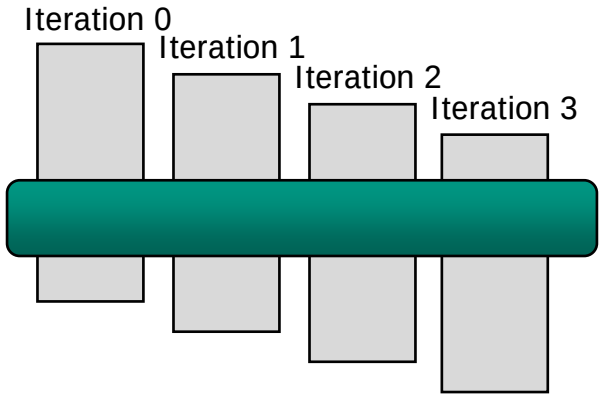
- **Scheduling**
  - Beispiel (Quelle: K. Asanovic, MIT)
  - Software-Pipelining

```
Loop:   ld f1,0(r1)
        ld f2,8(r1)
        ld f3,16(r1)
        ld f4,24(r1)
        add r1,32
        fadd f5,f0,f1
        fadd f6,f0,f2
        fadd f7,f0,f3
        fadd f8,f0,f4
        sd f5,0(r2)
        sd f6,8(r2)
        sd f7,16(r2)
        add r2,32
        sd f8,-8(r2)
        bne r1,r3,loop
```

| | FE Int1 | FE Int2 | FE Mem1 | FE Mem2 | FE FP+ | FE FPx |
|---|---|---|---|---|---|---|
| Prolog | | | ldf1 | | | |
| | | | ldf2 | | | |
| | | | ldf3 | | | |
| | add r1 | | ldf4 | | | |
| | | | ldf1 | | fadd f5 | |
| | | | ldf2 | | fadd f6 | |
| | | | ldf3 | | fadd f7 | |
| | add r1 | | ldf4 | | fadd f8 | |
| Schleife | | | ldf1 | sd f5 | fadd f5 | |
| | | | ldf2 | sd f6 | fadd f6 | |
| | | add r2 | ldf3 | sd f7 | fadd f7 | |
| | add r1 | bne | ldf4 | sd f8 | fadd f8 | |
| Epilog | | | | sd f5 | fadd f5 | |
| | | | | sd f6 | fadd f6 | |
| | | | | sd f7 | fadd f7 | |
| | | | | sd f8 | fadd f8 | |
| | | | | sd f5 | | |

# VLIW-Architektur

- **Scheduling**
  - Software-Pipelining
    - Technik zur Reorganisation von Schleifen
    - Jede Iteration in dem mit Software-Pipelining generierten Code enthält Befehle aus verschiedenen Iterationen der ursprünglichen Schleife

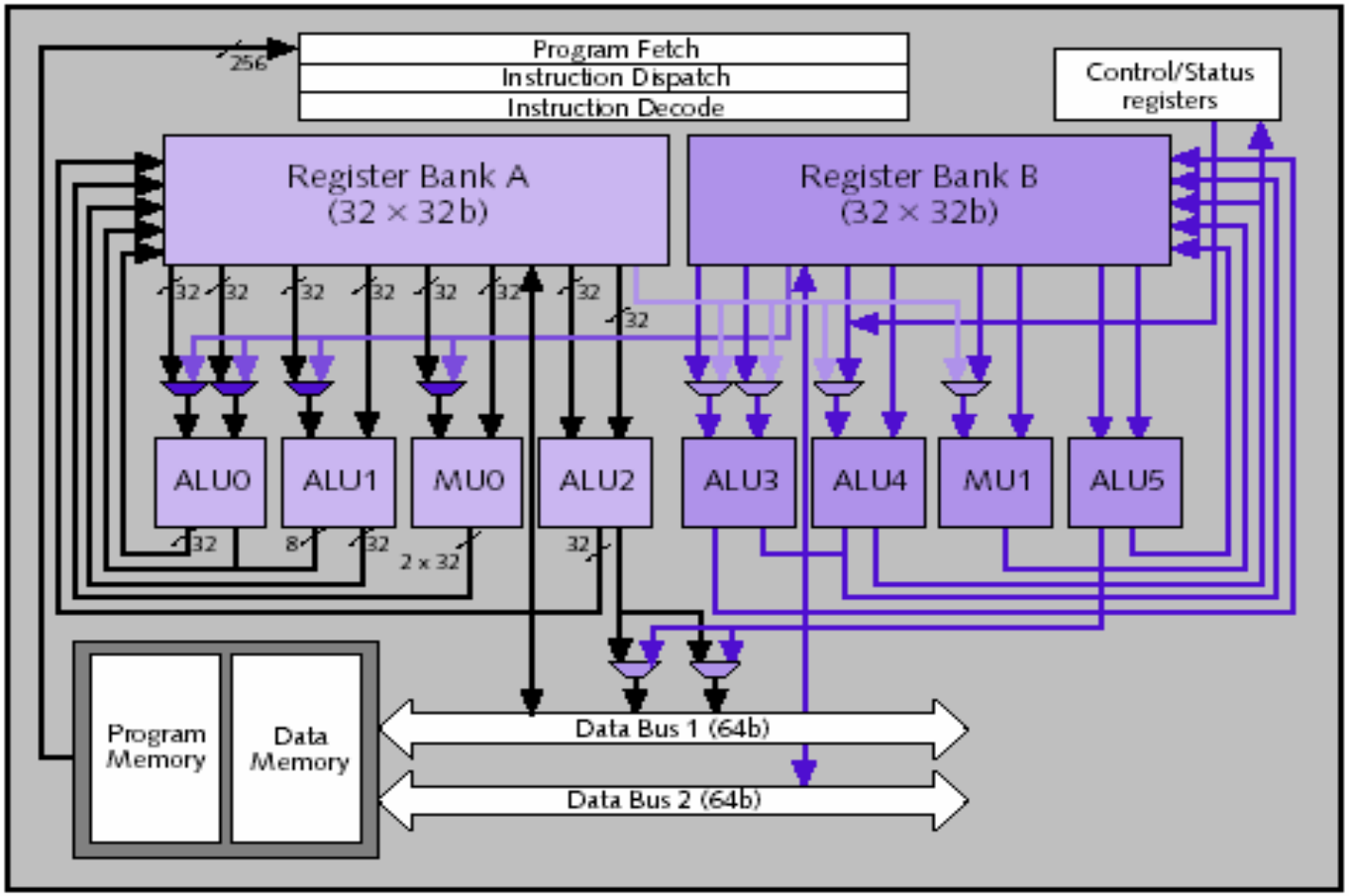


# VLIW-Architektur

- **Frühe VLIW-Rechner**
- Multiflow Trace (J. Fisher)
  - Rechnerkonfigurationen mit 7, 14 und 28 Operationen/VLIW-Instruktion
  - 28 Operationen in einem 1024 Bit Wort
  - Numerische Anwendungen
  - Trace-Scheduling
- Cydrome Cydra-5 (B. Rau)
  - 7 Operationen/256Bit Wort
  - Rotating Register File
- **Einsatz VLIW-Technik heute:**
  - Allzweck-Mikroprozessoren
    - Transmeta Crusoe
  - Bereich eingebetteter Prozessoren (DSP)
    - Beispiel: Trimedia TM32 Architecture
    - Agere/Motorola Star Core

# VLIW-Architektur

- **Fallstudie TI TMS320C6400**

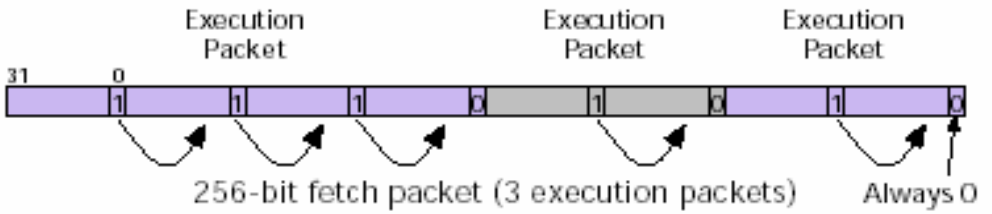


# VLIW-Architektur

- **Fallstudie TI TMS320C6400**
- Architekturmerkmale
  - 2 x 4 Funktionseinheiten (A und B Seite)
    - Jede Seite enthält 16 Register
    - Programmierer sieht 32 Register A0 – A15, B0 – B15
    - Ausgewählte Register für Boole‘sche Ergebnisse von bedingten Befehlen
    - 9 32 Bit Lese- und 6 32 Bit Schreibports
    - Jeweils ein Crossover-Pfad: Beschränkter wechselseitiger Zugriff
  - Jede Seite enthält
    - Eine 40 Bit Integer-ALU (L Unit)
      - Arithmetische und logische Operationen, Vergleiche, Normalisierung, Bit-Count,
    - 16 Bit Multiplizierer
      - 16 x 16 → 32 Bit Multiplikation
    - 40 Bit Schiebeeinheit
    - 32 Bit Addierer
      - Adressgenerierung

# VLIW-Architektur

- **Fallstudie TI TMS320C6400**
- Operationsprinzip
  - Holen von 8 32 Bit Befehlen über 256 Bit Befehlsbus (Fetch Packet)
    - Geholte Befehle müssen nicht unbedingt gleichzeitig ausgeführt werden
    - Ein Befehl in einem Fetch Packet ist nicht auf eine Ausführungseinheit beschränkt, jeder Befehl enthält Kodierung, mit der spezifiziert wird, auf welche Einheit der Befehl ausgeführt werden soll
    - Befehle sind nicht positionsabhängig
    - Programmierer / Compiler bestimmt Bindung
  - Bis zu 8 Befehle können gleichzeitig ausgeführt werden: Gruppierung von gleichzeitig ausführbaren Befehlen (Execution Packets)
    - Wird im niedrigstwertigen Bit eines Feldes gekennzeichnet: alle nachfolgenden Befehle werden gleichzeitig ausgeführt



# VLIW-Prinzip

- Literatur:
  - Hennessy/Patterson: Computer Architecture A Quantative Approach. Kap. 4.1-4.4, 4.8