Rechnerstrukturen

Vorlesung im Sommersemester 2006

Prof. Dr. Wolfgang Karl

Universität Karlsruhe (TH)

Fakultät für Informatik

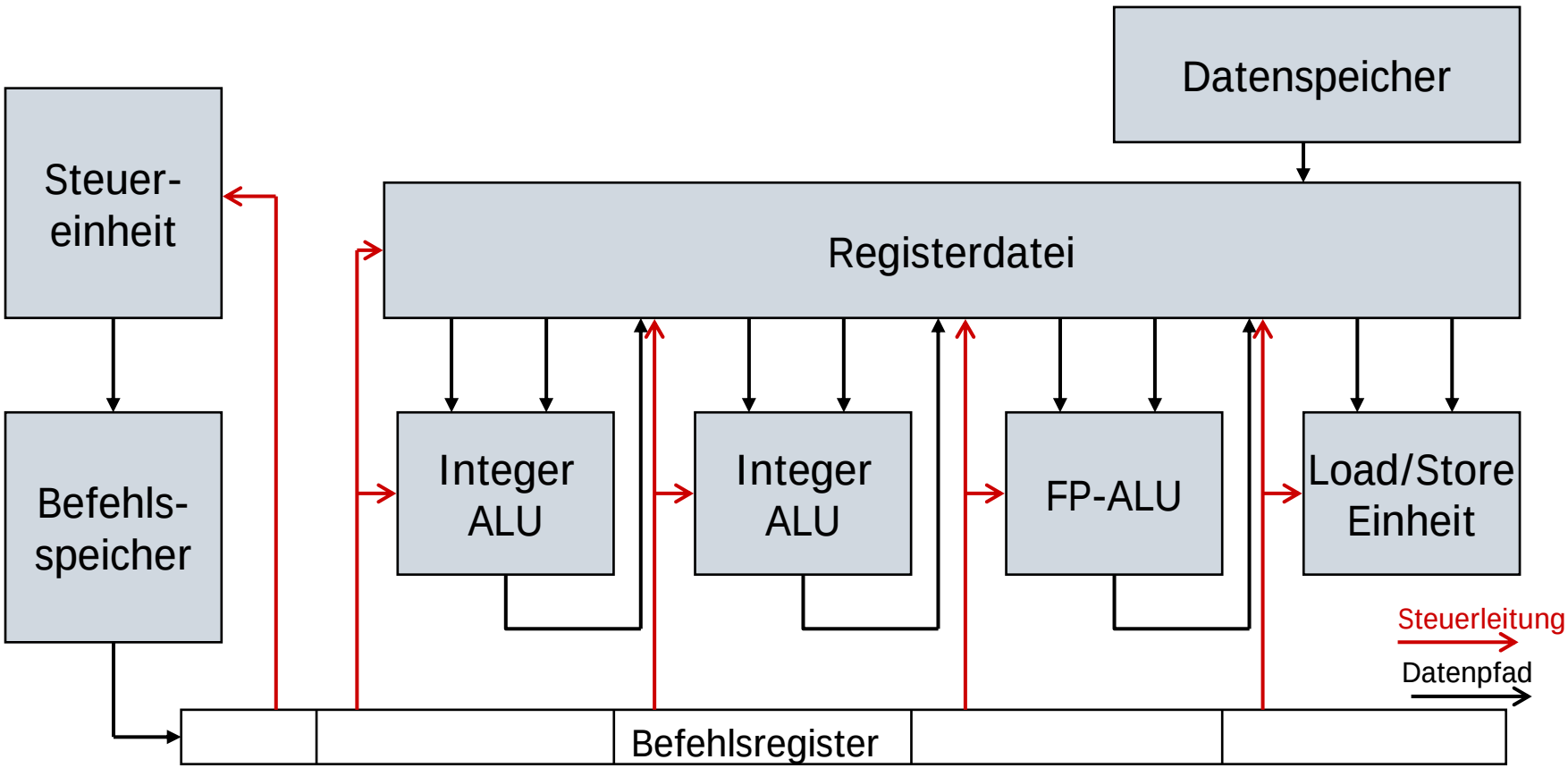
Institut für Technische Informatik

- **Kapitel 2: Parallelismus auf Befehlsebene**

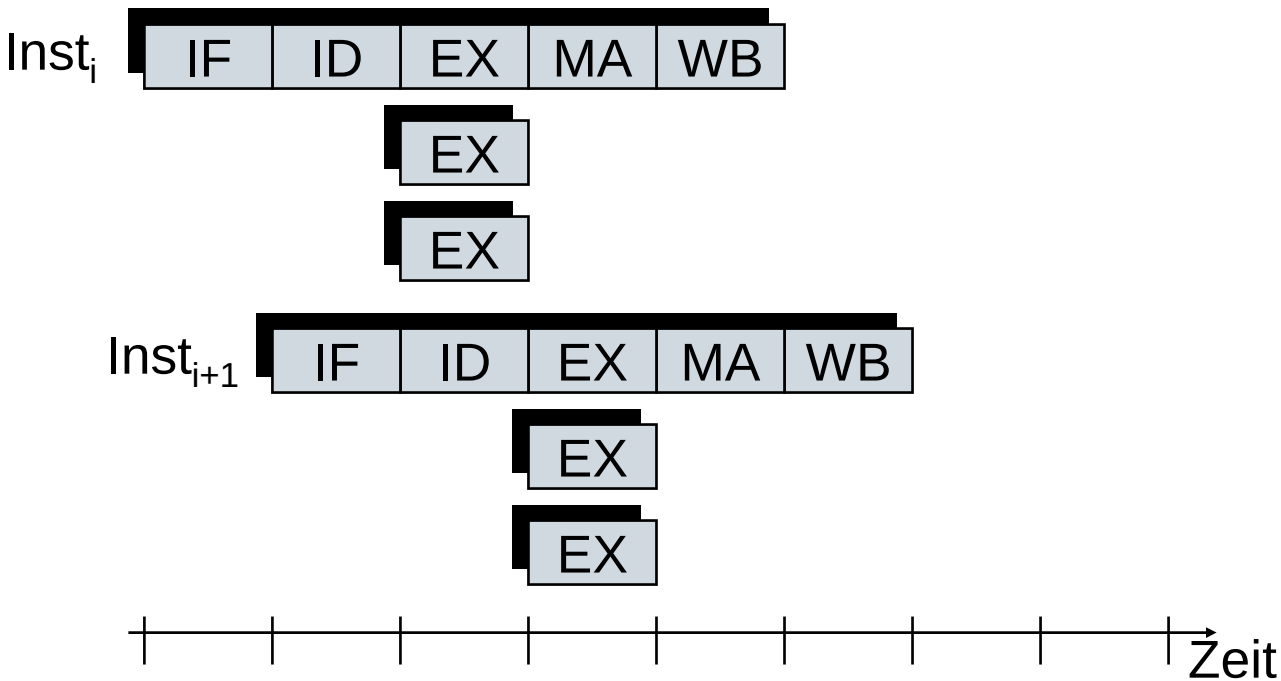
2.1: Nebenläufigkeit, Superskalartechnik

- Grundprinzip: VLIW
  - VLIW: Very Long Instruction Word
  - 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)

- Grundprinzip VLIW
  - Prinzipieller Aufbau



- Grundprinzip VLIW
  - Prinzipielle Pipeline-Struktur



- 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

- Statische Steuerung der parallelen Abarbeitung
  - 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
```

- Statische Steuerung der parallelen Abarbeitung
  - Scheduling (Fortsetzung Beispiel)

```
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			

- Statische Steuerung der parallelen Abarbeitung
  - Scheduling
    - 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
}
```

- Statische Steuerung der parallelen Abarbeitung
  - 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
```

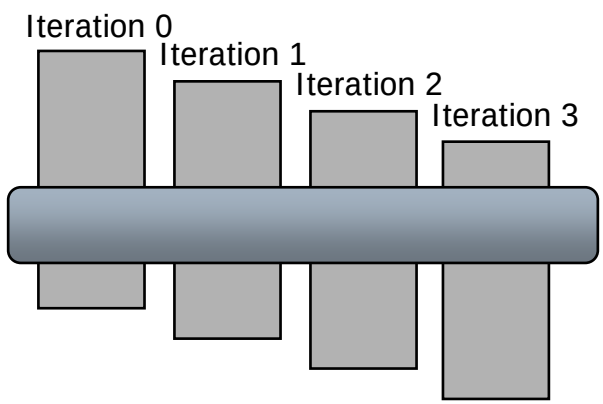
FE Int1	FE Int2	FE Mem1	FE Mem2	FE FP+	FE FPx
		ldf1			
		ldf2			
		ldf3			
add r1		ldf4		fadd f5	
				fadd f6	
				fadd f7	
				fadd f8	
		sd f5			
		sd f6			
		sd f7			
add r2	bne	sd f8			

- Statische Steuerung der parallelen Abarbeitung
  - 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		

- Statische Steuerung der parallelen Abarbeitung
  - Software-Pipelining
    - Technik zur Reorganisation von Schleifen
      - Jede Iteration in dem mit Software-Pipelining generierten Code enthält Befehle aus verschiedenen Iterationen der ursprünglichen Schleife

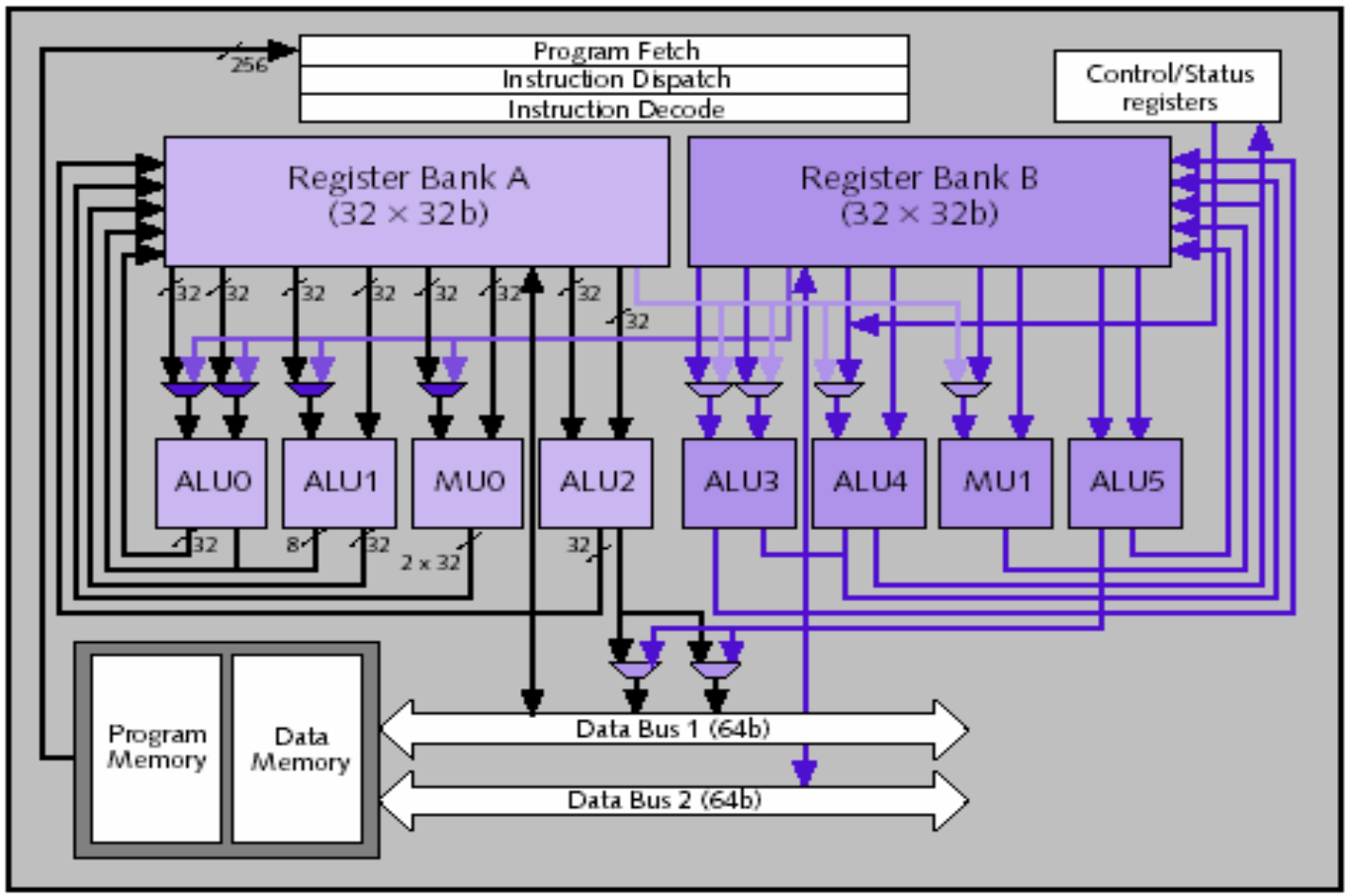


VLIW-Prinzip

- 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-Prinzip: Beispiel

- TI TMS320C6400

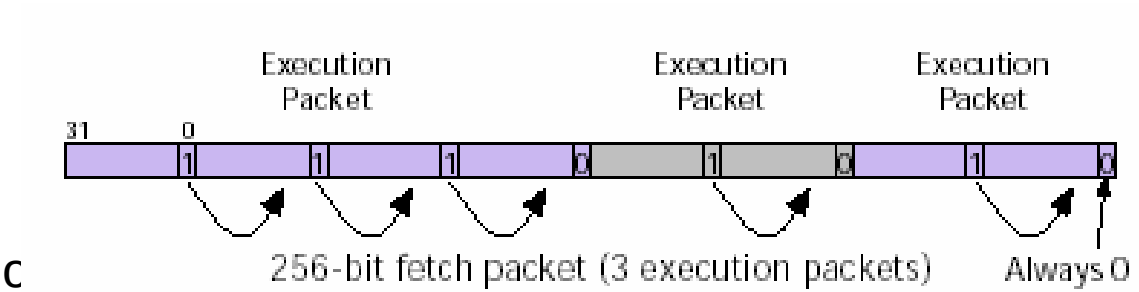


- TI TMS320C6000 Architekturmerkmale
  - 2 x 4 Funktionseinheiten (A und B Seite)
    - Jede Seite enthält 16 Register
      - Programmierer sieht 32 Register A0 – A15, B0 – B15
      - Ausgewählte Register für Boole‘sche Ergebnisse von bedingten Befehlen
      - 9 32 Bit Lese- und 6 32 Bit Schreibports
      - 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

- TI TMS320C6000 Architekturmerkmale
  - 2 x 4 Funktionseinheiten (A und B Seite)
    - 40 Bit Schiebeeinheit
      - Arithmetische Operationen, Verzweigung und Verzweigungsadressgenerierung
    - 32 Bit Addierer
      - Adressgenerierung

- TI TMS320C6000 Architekturmerkmale
  - VLIW-Prinzip
    - 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

- TI TMS320C6000 Architekturmerkmale
  - VLIW-Prinzip
    - Bis zu 8 Befehle können gleichzeitig ausgeführt werden: Gruppierung von gleichzeitig ausführbaren Befehlen (Execution Packets)
    - Werden im niedrigstwertigen Bit gekennzeichnet: alle nachfolgenden Befehle werden gleichzeitig ausgeführt
    - Prc
      ver

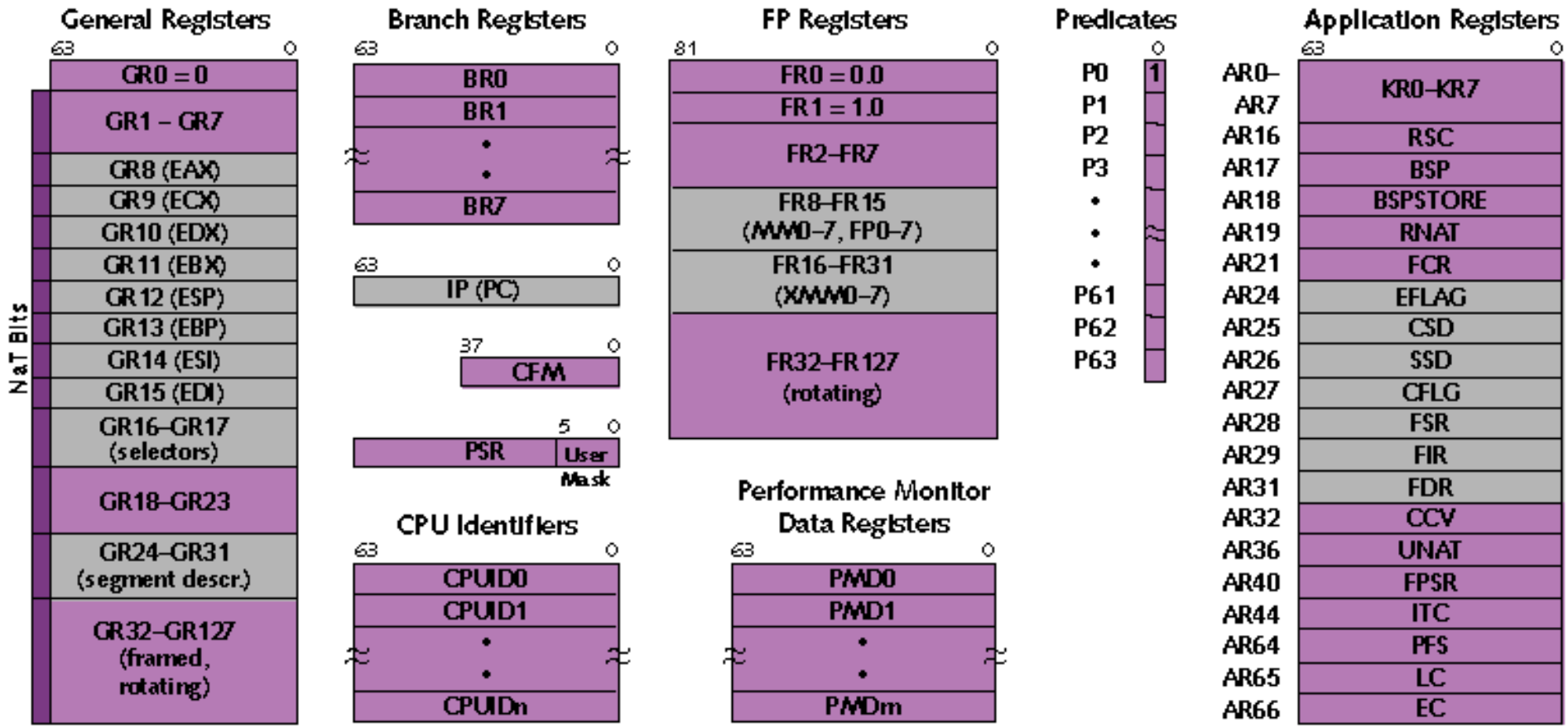


- Literatur:
  - Hennessy/Patterson: Computer Architecture A Quantative Approach. Kap. 4.1-4.4, 4.8

- EPIC: Explicitly Parallel Instruction Computing
  - Gemeinsames Projekt von Hewlett-Packard und Intel (1994 angekündigt)
  - Ziele:
    - 64 Bit Architektur: IA-64
    - Explizite Spezifikation des Parallelismus im Maschinencode: EPIC-Format, (entspricht VLIW-Prinzip)
    - Bedingte Ausführung von Befehlen (Predication)
    - Spekulative Ausführung von Ladeoperationen (Data Speculation)
    - Großer Registersatz
    - Skalierbarer Befehlssatz
    - Sinnvolles Zusammenwirken zwischen Compiler und Hardware
  - Itanium: erster Prozessor der P7-Generation (Code-Name Merced)

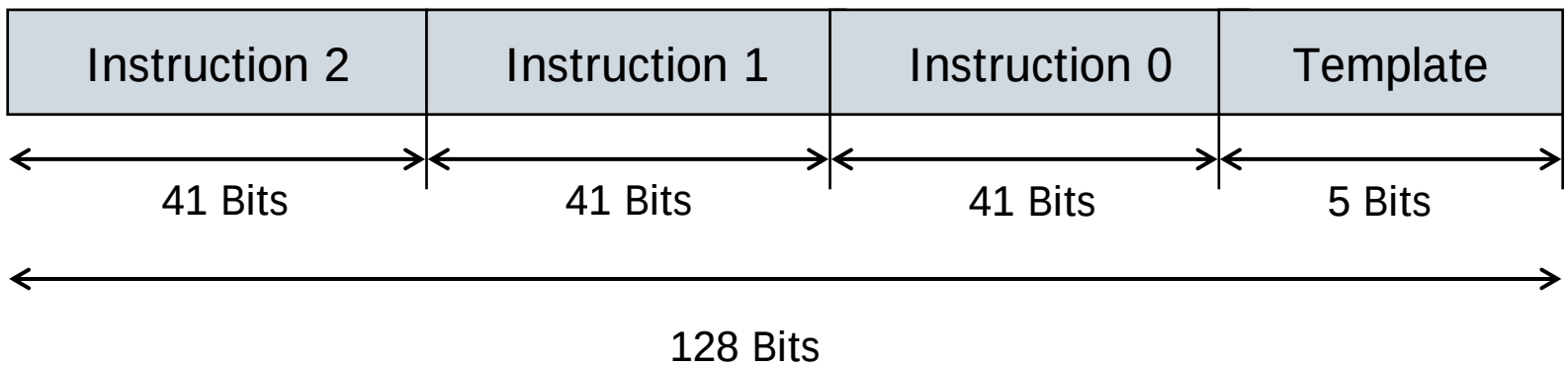
- Intel IA-64
  - Registersatz

Quelle: Microprocessor Report, Vol.13, Nr.7, 1999



- IA-64 ISA: Befehlsformat
  - Opcode
  - Predicate
  - 2 Felder für Quelloperandenregister
  - 1 Feld für Zielregister

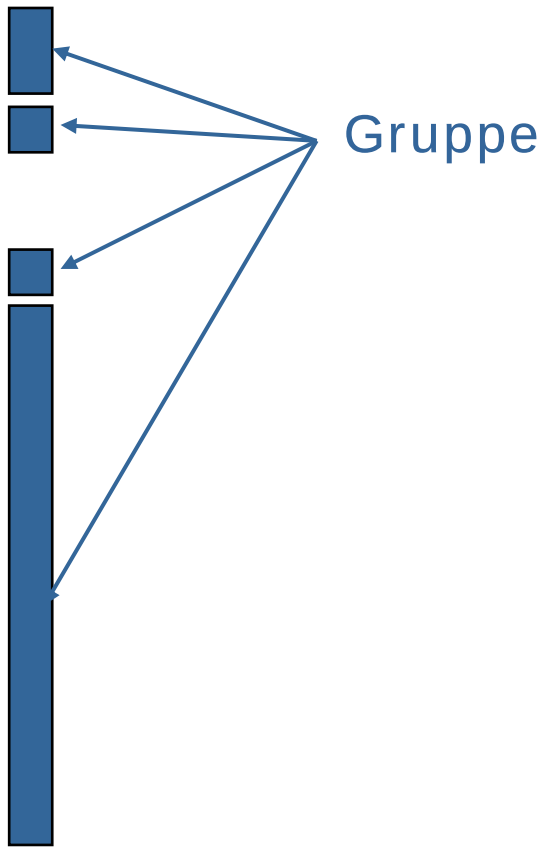
- IA-64 ISA
  - IA-64 Instruktionen werden vom Compiler in so genannte Bundles gepackt



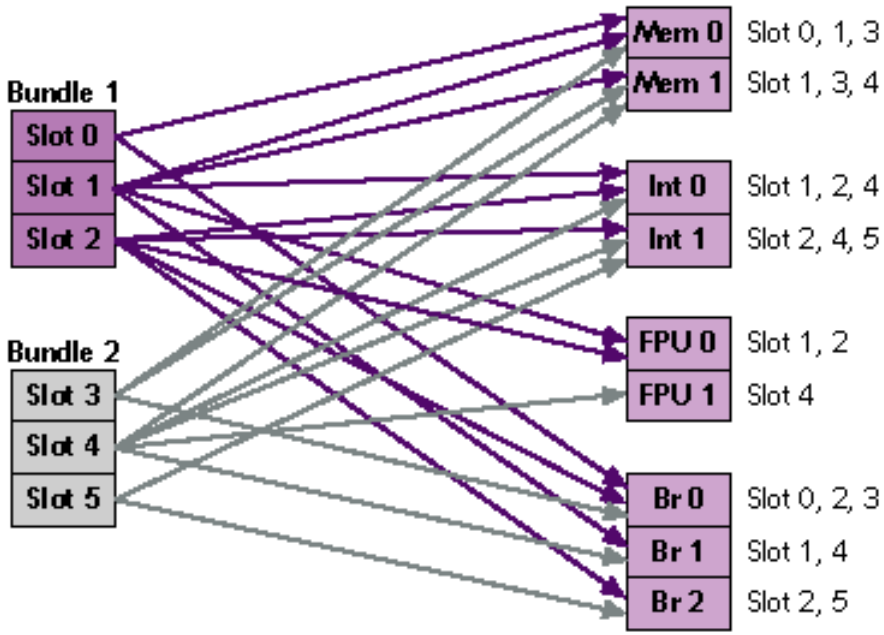
- IA-64 ISA
  - Bundles:
    - Template: zeigt explizit an, ob
      - die Instruktionen im Bundle gleichzeitig ausgeführt werden dürfen, oder
      - eine oder mehrere Instruktionen sequentiell auszuführen sind, oder
      - benachbarte Bundles parallel ausgeführt werden können.

- Beispiele von Befehlsgruppen:

```
{ .mii
        add r1 = r2,r3
        sub r4 = r4, r5;;
        shr r7 = r4, r12;;
}
{ .mmi
        ld8 r2 = [r1];;
        st8 [r1] = r23
        tbit p1,p2 = r4,5
}
{ .mbb
        ld8 r45 = [r55]
(p3)    br.call b1  =func1
(p4)    br.cond Label1
}
{ .mfi
        st4 [r45] = r6
        fmac f1 = f2,f3
        add r3 = r3,8;;
```



- Anstoßen der Befehle zur Ausführung
  - Beispiel: Itanium



  - Es können bis zu sechs Instruktionen pro Takt zur Ausführung angestoßen werden. Die Instruktionen kommen von zwei Bundles. Jeder Bundle hat 3 Slots.

- IA-64: Skalierbarkeit
  - Jedes Bundle enthält drei Instruktionen für eine Menge von drei Funktionseinheiten.
  - Ein IA-64 Prozessor kann *n* Mengen von jeweils drei Funktionseinheiten haben, welche die Informationen im Template nutzen, dann können mehrere Bundles in ein Instruktionswort mit der Länge *n* Bundles gepackt werden.

  → Skalierbarkeit bezüglich der Anzahl der Funktionseinheiten

- Predication, bedingte Befehlsausführung
  - Beispiel:

```
if (r1 == r2)
        r9 = r10 - r11;
else
        r5 = r6 + r7;
```

Bedingte Befehlsfolge

```
        cmp.eq p1,p2 = r1, r2;;
(p1)    sub r9 = r10, r11
(p2)    add r5 = r6, r7
```

- Evaluierung der bedingten Ausdrücke mit Hilfe von Compare-Operationen.
- Jeder Befehl hat ein 6 Bit breites Predicate Feld zur Angabe des Predicate-Registers
- Elimination von Sprüngen

- Predication (Bedingte Befehlsausführung)
  - Zur Laufzeit werden die voneinander unabhängigen Befehle zur Ausführung angestoßen.
  - Der Prozessor führt die Befehle auf den möglichen Programmverzweigungen aus, aber speichert die Ergebnisse nicht endgültig.
  - Überprüfen der Predicate Register
    - Falls das Register eine 1 enthält, dann ist wird die Ausführung der Instruktion abgeschlossen.
    - Falls das Register eine 0 enthält, dann wird das Ergebnis verworfen.

- IA-64 Control Speculation
  - Problem: Verzweigungen schränken Code-Verschiebungen ein

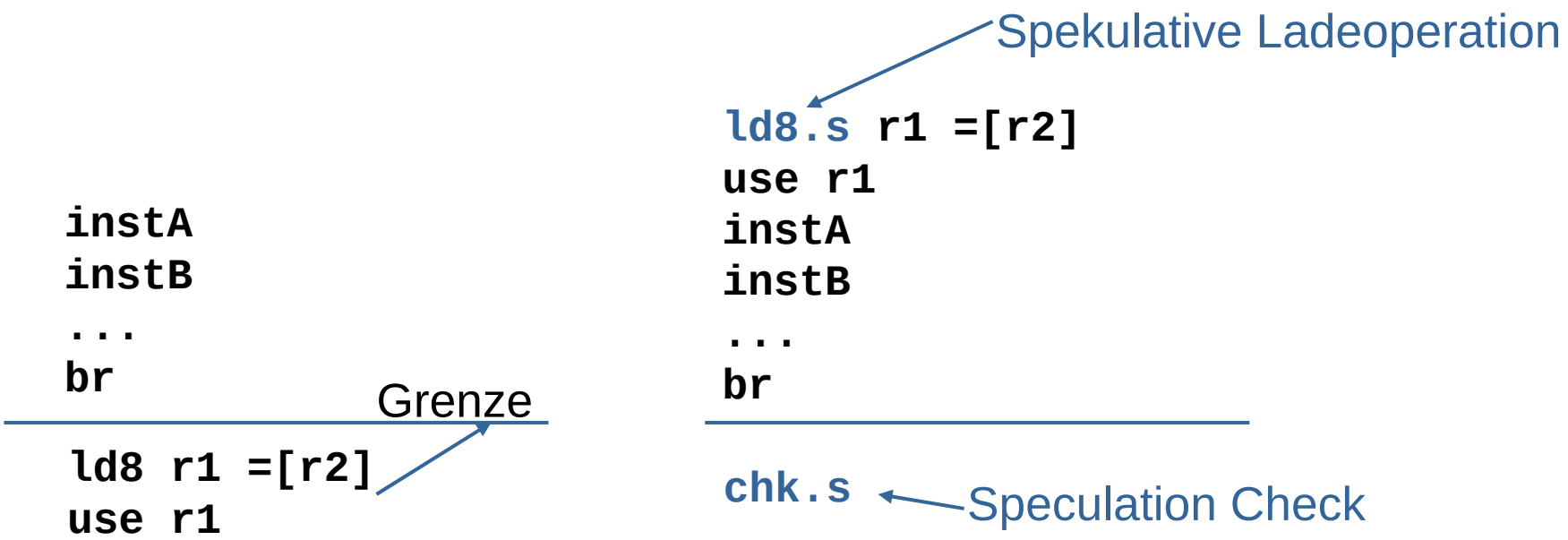
```
instA
instB
...
br
```

Grenze

```
ld8 r1 =[r2]
use r1
```

Ladeoperation kann nicht über den Sprung verschoben werden, denn es könnten Alarme ausgelöst werden.

- IA-64 Control Speculation
  - Lösung: Spekulative Operationen, die keine Alarme auslösen



- IA-64 Control Speculation
  - Einführung von spekulativen Ladeoperationen
    - Verursachen keinen Alarm:
      - Falls zur Laufzeit die Operation einen Alarm auslöst, dann wird dieser Alarm verzögert
      - Setzen von NaT-Bit (Deferred Exception Token, Not-a-Thing Bit) in Zielregister
    - Spekulative Ladeoperation kann über Verzweigungen hinweg verschoben werden
    - Einfügen von Speculation Check Instruktion (chk.s) anstelle der Ladeoperation
      - Zur Laufzeit überprüft die Check-Operation das Zielregister, ob NaT gesetzt ist.Wenn ja, dann erfolgt eine Verzweigung zu einem speziellen Fix-up-Code.

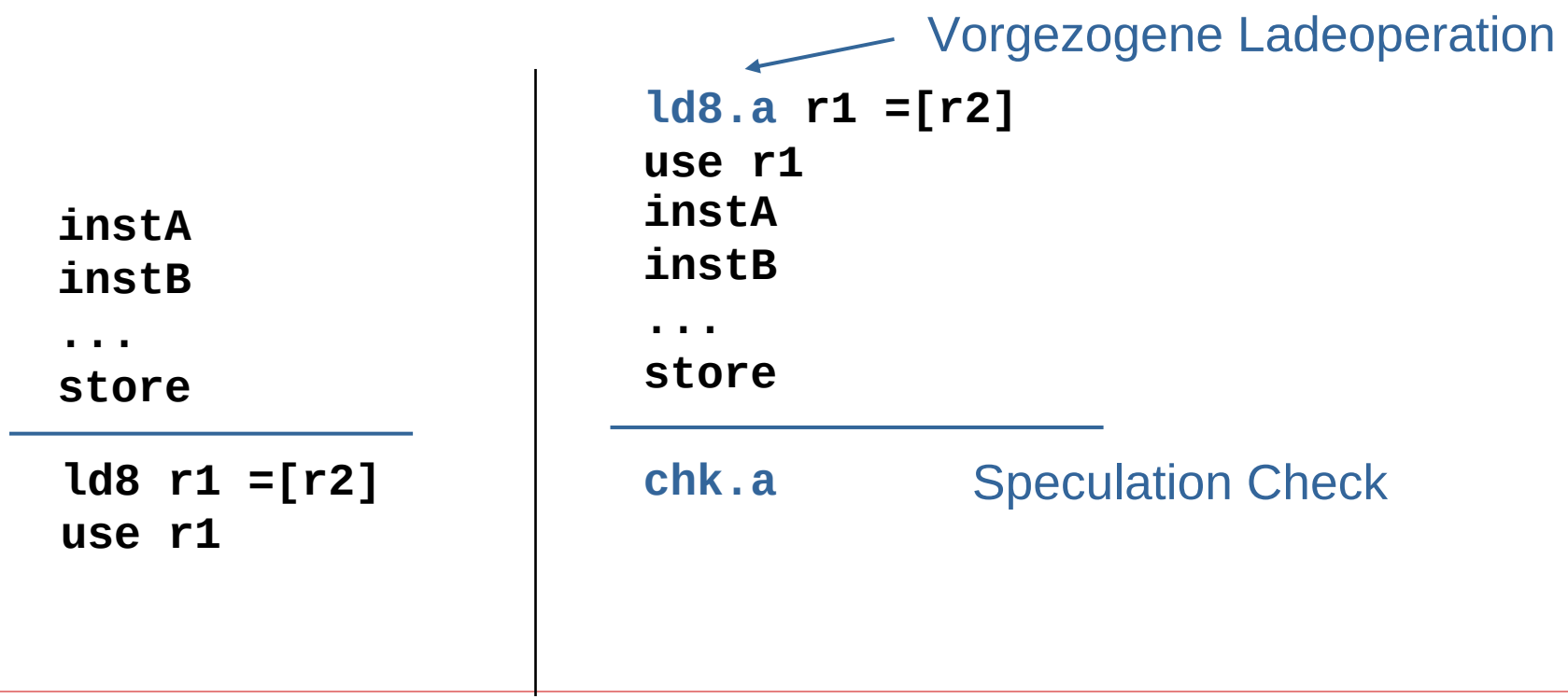
- IA-64 Data Speculation
  - Problem: Zeiger können Compiler zu konservativen Annahmen über die Referenzen zwingen, was eine Code-Verschiebung verhindert.

```
instA
instB
...
store
```

```
ld8 r1 =[r2]
use r1
```

Ladeoperation kann nicht über den die Speicheroperation verschoben werden, da beide dieselbe Adresse referenzieren können.

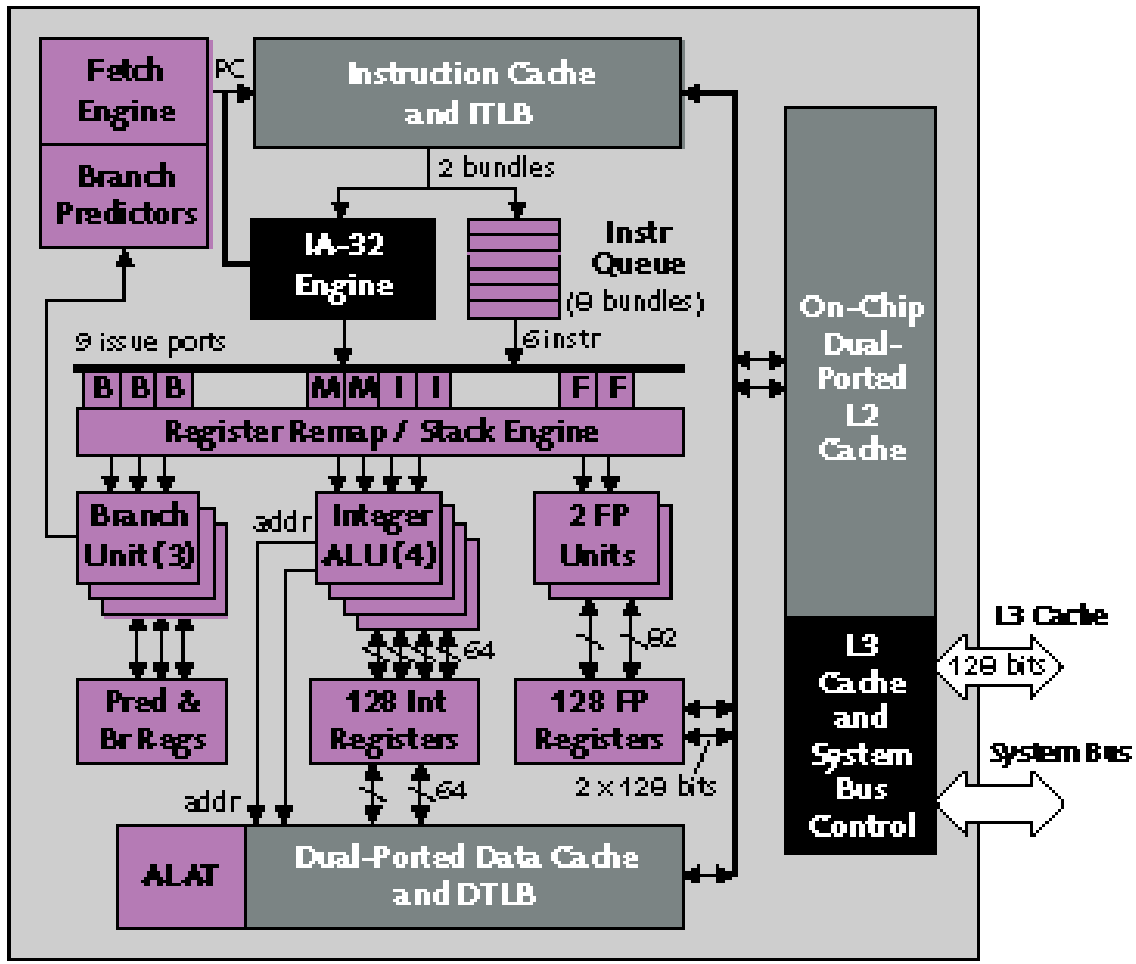
- IA-64 Data Speculation
  - Lösung: Vorgezogene Ladeoperationen



- IA-64 Data Speculation
  - Lösung: Vorgezogene Ladeoperationen
    - Verschieben von Lade-Operationen auch vor Speicher-Operationen.
      - Vorgezogene Lade-Operation (ld.a)
      - Zur Laufzeit werden Informationen (Zielregister, Speicheradresse, auf die zugegriffen wird, Zugriffsgröße) in Advanced Load Address Table (ALAT) festgehalten.
      - Zur Laufzeit prüft die Hardware, wenn eine Speicheroperation ausgeführt wird, ob eine Adresse in der ALAT mit der Zugriffsadresse übereinstimmt. Wenn ja, dann wird der Eintrag in ALAT gelöscht

- IA-64 Data Speculation
  - Lösung: Vorgezogene Ladeoperationen
    - Einfügen der Check-Operation (`chk.a`)
      - Prüft, ob Eintrag von der entsprechenden vorgezogenen Ladeoperation in ALAT steht.
      - Wenn nicht, dann hat es eine Kollission mit einer Speicheroperation gegeben und es erfolgt eine Verzweigung zu einem Fix-up-Code.

- Intel Itanium



Quelle: Microprocessor Report, Vol.13, Nr.13, October 5, 1999

- Intel Itanium
  - 64-Bit Prozessor der P7 Generation
  - EPIC
  - 10-stufige Pipeline mit 6 Befehlen, die gleichzeitig zur Ausführung angestoßen werden können.
  - 128 GP- und 128 FP-Register - keine Registerumbenennung: rotierende Registerfenster
  - Bedingte Befehlsausführung: 64 1 Bit Predicate Register
  - 9 Funktionseinheiten
  - Misprediction Penalty: 9 Zyklen

- Intel Itanium
  - Spekulative Ladeopationen
  - 4-fach mengenassoziative L1 Befehls- und Daten-Cache-Speicher (Write-through)
  - 6-fach mengenassoziativer L2 Cache (Copy-back)
  - Informationen
    - Intel: http://www.intel.com/
      - Intel Technology Journal: http://developer.intel.com/technology/itj
    - IEEE Micro: Sonderheft im Spe./Okt. 2000
    - Henn./Patt.: Computer Architecture: Kap. 4.7

- 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