

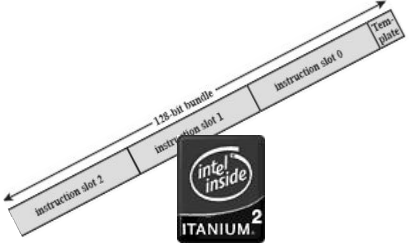
Luento 11

Tietokoneen rakenne

IA-64

Stallings: Ch 15

- n Yleistä IA-64:stä
- n Predikointi
- n Spekulointi
- n Ohjelmoitu liukuhinna
- n Itanium 2
- n Intel Multi-core ja STI Cell



The diagram shows a 128-bit bundle of instructions, labeled '128-bit bundle', with three instruction slots: 'instruction slot 2', 'instruction slot 1', and 'instruction slot 0'. An arrow labeled 'Tee päätös' (Make decision) points to the right. Below the bundle is an Intel Itanium 2 processor chip.

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 1

EPIC

EPIC - Explicit Parallel Instruction Computing

☞ Rinnakkaisuus esiin jo konekielen tasolla, ei näkymättömissä siellä jossain laitetasolla

- u Uutta semantiikkaa konekielen tasolle
- u Kääntäjä ratkoo riippuvuuteen liittyvät ongelmat, laitteisto(toteutus) luottaa siihen
- VLIW (Very Long Instruction Word)
 - u Käsittelee käskyjä nipuissa (bundle)
- Ž Hyppyjen predikointi, kontrollispekulointi
 - u Suorittaa useita haarautumispolkuja
- Spekulatiiviset muistinoudot myös datalle

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 2



IA-64 vs. Superskalaari

Superscalar	IA-64
RISC-like instructions, one per word	RISC-like instructions <u>bundled</u> into groups of three
Multiple parallel execution units	Multiple parallel execution units
Reorders and optimizes instruction stream at run time	Reorders and optimizes instruction stream at <u>compile</u> time
Branch prediction with speculative execution of one path	Speculative execution along <u>both</u> paths of a branch
Loads data from memory only when needed, and tries to find the data in the caches first	Speculatively loads data <u>before</u> its needed, and still tries to find data in the caches first

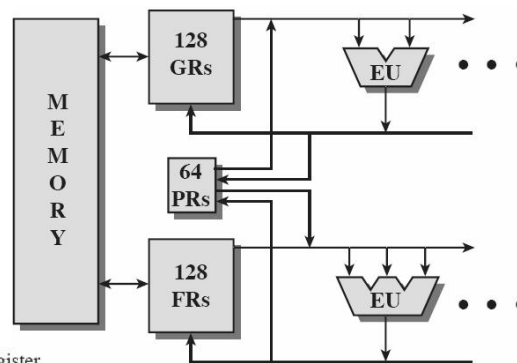
- n IA-64 liikkeelle puhtaalta pöydältä
 - u unohda historiallinen painolasti
- n HP ja Intel yhteistyössä

(Sta06 Table 15.1)



IA-64 Rakenne

- n Paljon rekistereitä, väh. 8 suoritusyksikköä
- n GR-rekistereissä NaT-bitti (Not a Thing) => "myrkkyä"



GR = General-purpose or integer register
 FR = Floating-point or graphics register
 PR = One-bit predicate register
 EU = Execution unit

(Sta06 Fig 15.1)

Käskyformaatti

← 128-bit bundle →

instruction slot 2		instruction slot 1			instruction slot 0		Tem- plate
41		41			41		5

n Nouto muistiin
 n Template
 u mi
 u

Major opcode	other modifying bits	GR3	GR2	GR1	PR
4	10	7	7	7	6

n PR: Käskyissä spekulointiin liittyvä predikaattirekisteri
 u 1-bittinen, tarkistetaan kommitointihetkellä
 n Käskyissä tavallisesti 3 rekisteriä
 n Load/Store arkkitehtuuri

(Sta06 Fig 15.2)

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 5

Toimintoyksiköt

Instruction Type	Description	Execution Unit Type
A	Integer ALU	I-unit or M-unit
I	Non-ALU integer	I-unit
M	Memory	M-unit
F	Floating-point	F-unit
B	Branch	B-unit
X	Extended	I-unit/B-unit

Template	Slot 0	Slot 1	Slot 2
00	M-unit	I-unit	I-unit
01	M-unit	I-unit	I-unit
02	M-unit	I-unit	I-unit
03	M-unit	I-unit	I-unit
04	M-unit	L-unit	X-unit
05	M-unit	L-unit	X-unit
08	M-unit	M-unit	I-unit
09	M-unit	M-unit	I-unit
0A	M-unit	M-unit	I-unit
0B	M-unit	M-unit	I-unit
0C	M-unit	F-unit	I-unit
0D	M-unit	F-unit	I-unit
0E	M-unit	M-unit	F-unit
0F	M-unit	M-unit	F-unit
10	M-unit	I-unit	B-unit
11	M-unit	I-unit	B-unit
12	M-unit	B-unit	B-unit
13	M-unit	B-unit	B-unit
16	B-unit	B-unit	B-unit
17	B-unit	B-unit	B-unit
18	M-unit	M-unit	B-unit
19	M-unit	M-unit	B-unit
1C	M-unit	F-unit	B-unit
1D	M-unit	F-unit	B-unit

n Max 6 käskyä suoritukseen per sykli
 n Musta pystyviiva = stop = Käskyjen välillä riippuvuus

(Sta06 Table 15.2, 15.3)

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 6



Symbolisen konekielen formaatti

```
[qp] mnemonic[.comps] dests = srcs
```

qp qualifying predicate register
 - jos predikaattirekisterin arvo=1 (true), commit

mnemonic operaation mnemoninen nimi

comps completers, erottelu pilkuilla
 - jotkut käskyt muodostuvat kahdesta osasta

dests destination operands, erottelu pilkuilla


srcs source operands, erottelu pilkuilla



Symbolisen konekielen formaatti

- n** Käskyryhmän rajat merkitään ;;
- u Vihje: nämä konekäskyt voi suorittaa rinnakkain
 - u Konekielessä template, jossa "musta pystyviiva"
 - u Ryhmän sisällä ei data- tai kirjoitusriippuvuutta
 ts. no read after write (RaW) tai
 no write after write (WaW)

```
ld8 r1 = [r5]                    // ensimmäinen ryhmä
sub r6 = r8, r9 ;;
add r3 = r1, r4                 // toinen ryhmä
st8 [r6] = r12
```




Tietokoneen rakenne

Avainmekanismit

- n Predikointi
- n Kontrollispekulointi
- n Dataspekulointi
- n Ohjelmoitu liukuhihna

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 9



Predikoitu suoritus

Kääntäjä

- n **Muodosta käskyniput, aseta template**
 - u Kuvaa, mitkä käskyt voisi suorittaa samanaikaisesti
- n **Poista if-then-else rakenteen hyyt**
 - u Vertailu asettaa kaksi predikaattirekisteriä
 - u Kummankin haaran käskyihin mukaan oma predikaatti
 - u Kumpaakin haaraa tullaan suorittamaan

Intel kalvo 18

CPU

- n **Suorita molemmat haarat**
- n **Tarkista predikaatit, kun vertailukäsky valmistuu**
 - u Predikaatti on aina valmis kommitointivaiheessa?
 - u Hylkää väärä polku (käskyt), hyväksy oikea polku

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 10

Predikoitu suoritus

1. The branch has two possible outcomes.

2. The compiler assigns a predicate register to each following instruction, according to its path.

3. All instructions along this path point to predicate register P1.

4. All instructions along this path point to predicate register P2.

5. CPU begins executing instructions from both paths.

6. CPU can execute instructions from different paths in parallel because they have no mutual dependencies.

7. When CPU knows the compare outcome, it discards results from invalid path.

The compiler might rearrange instructions in this order, pairing instructions 4 and 7, 5 and 8, and 6 and 9 for parallel execution.

Instruction 1	Instruction 2	Instruction 3
Instruction 4	Instruction 7	Instruction 5
Instruction 8	Instruction 6	Instruction 9

(Sta06 Fig 15.3a)

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 11

Predikoitu suoritus

Source:

```

if (a&&b)
    j=j+1
else
    if (c)
        k=k+1
    else
        k=k-1
i=i+1;
                
```

Pentium:

```

cmp a,0
je L1
cmp b,0
je L1
add j,1
jmp L3
L1: cmp c,0
je L2
add k,1
jmp L3
L2: sub k,1
L3: add i,1
                
```

IA-64:

```

cmp.eq(p1),p2 = 0,a ;;
(p2) cmp.eq(p1),p3 = 0,b
(p3) add j = 1,j
(p1) cmp.ne p4,p5 = 0,c
(p4) add k = 1,k
(p5) add k = -1,k
add i = 1,i
                
```

(Sta06 Fig 15.4)

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 12

Kontrollispekulointi

n Aloita datan laataaminen muistista etukäteen
= spekulatiivinen load

- u Valmiina CPU:ssa kun tarvitaan, ei latenssia Intel kalvo 26
- u Yleensä helppoa, mutta ei, jos välissä haarautuminen/store

n Kontrollispekulointi = nosta (hoist) load-käskey aiemmaksi koodissa hyppykäskyn etupuolelle

- u Merkitse se kuitenkin spekulatiiviseksi (.s)
- u Jos spekulointi aiheuttaa poikkeuksen, sen käsittely viivästetään (NaT bitti)
 - § On mahdollista että kyseistä poikkeusta ei pitänyt tapahtua!
- u Lisää alkuperäiseen kohtaan chk-käskey (chk.s), joka tarkistaa poikkeuksen ja käynnistää recovery-rutiinin

je L2
ld8 r1=[r5]
use r1

→

ld8.s r1=[r5]
je L2
chk.s r1, recovery
use r1

← completer

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 13

Kontrollispekulointi

Intel kalvot 27-28

1. The compiler scans the source code and sees an upcoming load (instruction 8). It removes the load, inserts a speculative load here and a speculative check immediately before the operation that will use the data (instruction 9).

2. At run time, this instruction loads the data from memory before it is needed. If the load would trigger an exception, the CPU postpones reporting the exception.

```

graph TD
    I1[Instruction 1] --> I2[Instruction 2]
    I2 --> SL[Speculative Load]
    SL --> I3[Instruction 3 (branch)]
    I3 --> I4[Instruction 4 (P1)]
    I3 --> I7[Instruction 7 (P2)]
    I4 --> I5[Instruction 5 (P1)]
    I5 --> I6[Instruction 6 (P1)]
    I7 --> I8[Instruction 8 (load data)]
    I8 --> SC[Speculative check (P2)]
    SC --> I9[Instruction 9 (P2)]
    
```

3. The compiler replaced this load with the speculative load above, so instruction 8 does not actually appear in the program.

5. In effect, IA-64 has hoisted the load above the branch.

4. This instruction checks the validity of the data. If it is OK, the CPU does not report an exception.

(Sta06 Fig 15.3b)

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 14

Dataspekulointi

Intel kalvot 29-30

- n Nosta load-käskey aiemmaksi koodissa store-käskyn etupuolelle
 - u Merkitse se ennakkolataamiseksi (.a eli advanced load)
 - u Lisää alkuperäiseen paikkaan tarkistus (.c)
- n **ALAT** eli **Advanced Load Address Table** (laitteistoa) pitää kirjaa loadissa käytetyistä osoitteista
 - u Kukin load vie kohdeosoitteen ALAT-tauluun
 - u Kukin store poistaa kohteen ALAT-taulusta
 - u Tarkistus (.c): Jos kohde ei taulussa, lataa uudelleen

```
je L1
st8 [r3] = r13
ld8 r1 = [r5]
```

⇒

```
ld8.a r1 = [r5]
je L1
st8 [r3] = r13
ld8.c r1 = [r5]
```

Alias-ongelma:
r3 ja r5 voivat osoittaa samaan paikkaan

Intel kalvo 31

Tietokoneen rakenne / 2006 / Teemu Kerola
5.10.2006
Luento 11 - 15

Ohjelmoitu liukuhihna

Software pipelining

Why called software pipeline?

- n Laitteistotuki silmukan purkamiseksi s.e. voidaan suorittaa useita iteraatioita samanaikaisesti
- n Rinnakkaisuus syntyy suorittamalla eri iteraatiokierroksiin kuuluvia toimintoja yhtäaikaan
- n Kukin iteraatiokierros käyttää eri rekistereitä
 - u Automaattinen rekistereiden uudelleennimeäminen
- n Alku (prolog) ja loppu (epilog) erikoistapauksina rotatoivan predikaattirekisterin avulla
- n Silmukan hyppykäskey korvattu erityiskäskyllä, joka kontrolloi ohjelmoidun liukuhihnan käyttöä
 - u Rotatoi rekisterit, vähentää silmukkalaskuria

Tietokoneen rakenne / 2006 / Teemu Kerola
5.10.2006
Luento 11 - 16

Ohjelmoidun liukuhihnan idea

```

for i=5 to 1 do y[i] = x[i] + c
    mov lc = 5
L1: ld4 r4 = [r5],4 ;;
    add r7 = r4,r9 ;;
    st4 [r6] = r7,4
    br.cloop L1 ;;

ld4 r32 = [r5], 4 ;; // cycle 0
ld4 r33 = [r5], 4 ;; // cycle 1
ld4 r34 = [r5], 4 // cycle 2
add r36 = r32, r9 ;; // cycle 2
ld4 r35 = [r5], 4 // cycle 3
add r37 = r33, r9 // cycle 3
st4 [r6] = r36, 4 ;; // cycle 3
ld4 r36 = [r5], 4 // cycle 4
add r38 = r34, r9 // cycle 4
st4 [r6] = r37, 4 ;; // cycle 4
add r39 = r35, r9 // cycle 5
st4 [r6] = r38, 4 ;; // cycle 5
add r40 = r36, r9 // cycle 6
st4 [r6] = r39, 4 ;; // cycle 6
st4 [r6] = r40, 4 ;; // cycle 7
                
```

Vähän käskytason rinnakkaisuutta, pieni koodi

Enemmän rinnakkaisuutta suorituksessa!
Operoi eri iteraation registreillä samanaikaisesti

(Sta06 Fig 15.6)

Intel kalvo 25

Tietokoneen rakenne / 2006 / Teemu Kerola

5.10.2006

Luento 11 - 17

Koodi

```

mov lc = 199 // set loop count register
mov ec = 4 // set epilog count register
mov pr.rot = 1<<16;; // pr16 = 1, rest = 0
L1: (p16) ld5 r32 = [r5], 4 // cycle 0
    (p17) --- // empty stage
    (p18) add r35 = r34, r9 // cycle 0
    (p19) st4 [r6] = r36, 4 // cycle 0
    br.ctop L1 ;; // cycle 0
                
```

(Sta06 Table 15.4)


Koodi s.555

Cycle	Execution Unit/Instruction				State before br.ctop					
	M	I	M	B	p16	p17	p18	p19	LC	EC
0	ld4			br.ctop	1	0	0	0	199	4
1	ld4			br.ctop	1	1	0	0	198	4
2	ld4	add		br.ctop	1	1	1	0	197	4
3	ld4	add	st4	br.ctop	1	1	1	1	196	4
...
100	ld4	add	st4	br.ctop	1	1	1	1	99	4
...
199	ld4	add	st4	br.ctop	1	1	1	1	0	4
200		add	st4	br.ctop	0	1	1	1	0	3
201		add	st4	br.ctop	0	0	1	1	0	2
202			st4	br.ctop	0	0	0	1	0	1
					0	0	0	0	0	0

Tietokoneen rakenne / 2006 / Teemu Kerola

5.10.2006

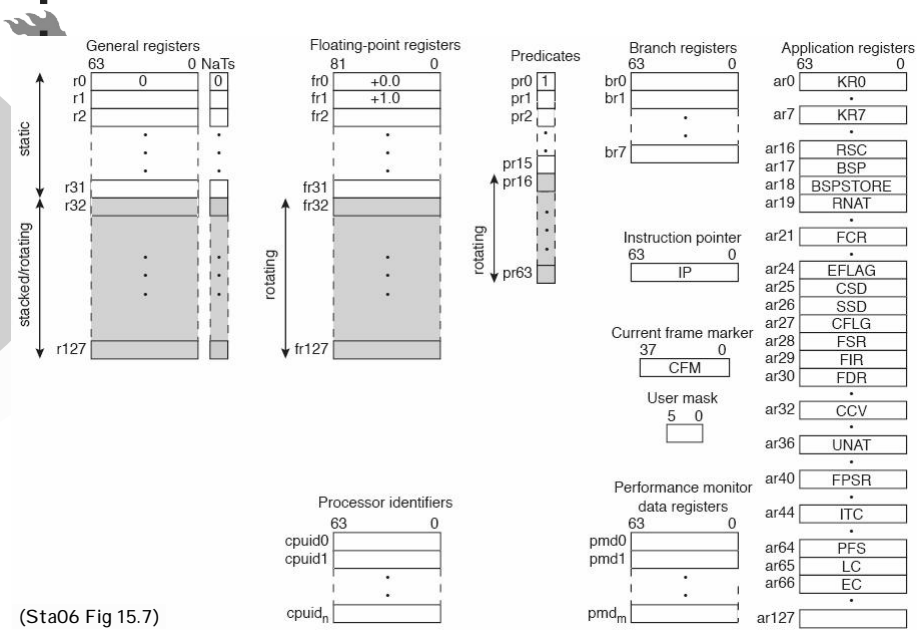
Luento 11 - 18



Tietokoneen rakenne


IA-64 Rekisterit

Tietokoneen rakenne / 2006 / Teemu Kerola
5.10.2006
Luento 11 - 19



(Sta06 Fig 15.7)

Tietokoneen rakenne / 2006 / Teemu Kerola
5.10.2006
Luento 11 - 20



Sovelluksen rekisterit Sta06 Fig 15.7

- n Yleisrekisterit (128), FP-rekisterit (128), Predik.rekisterit (64)
 - u Osa staattisia, osa rotatoivia (laitteisto uudelleennimeää)
 - u Osaa yleisrekistereistä käytetään pinona
- n Hyppyrekisterit, 8 kpl
 - u Kohdeosoite voi olla rekisterissä (siis epäsuora hyppy!)
 - u Aliohjelman paluuosoite tavallisesti rekisteriin br0
 - u Jos uusi kutsu, br0:n talteen rekisteripinon
- n Instruction pointer
 - u Nipun osoite, ei yksittäisen käskyn
- n User mask
 - u Lipukkeet poikkeuksia ja monitorointia varten
- n Performance monitor data registers
 - u Tietoa laitteiston käyttäytymisestä
 - u Esim. Hyppyennustuksista, rekisteripinon käytöstä, muistin odotusajoista, ..

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 21



Rekisteripino, Register Stack Engine Intel kalvot 15-17

- n r0..r31 globaaleille muuttujille
- n r32..r127 (96 kpl) aliohjelmakutsuille
- n Kutsu varaa pinosta rekisteri-ikkunallisen (frame)
 - u parametrit (inputs/outputs) + paikalliset mjat (locals)
 - u Koko dynaamisesti määriteltävissä (alloc-käsky)
- n Kutsun jälkeen rekisterit uudelleennimetty
 - u Aliohjelman näkemät parametrit alkavat aina r32:sta
- n Allokointi renkaana
 - u Jos pino täyttyy, laitteisto tallettaa vanhoja ikkunoita muistiin (= pinon, backing store)
 - § Sijainti rekistereissä BSP, BSPSTORE

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 22

Rekisteripino

Intel kalvo 17

- n Allokointi ja palautus käyttää kahta rekisteriä
- n CFM, Current Frame Marker
 - u Rekisteripinosta kutsun yhteydessä varatun alueen koko
 - § sof=size of frame, sol=size of locals,
 - § sor=size of rotation portion (SW pipeline)
 - u GR/FP/PR-rekistereiden rotatointitietoa
 - § rrb=register rename base
- n PFS, Previous Function State
 - u CFM:n edellinen sisältö tänne, vanha PFS jonnekin toiseen rekisteriin (alloc voi määrätä minne)

PFS		CFM					
		6	7	7	4	7	7
ppl	pec	rrb.pr	rrb.fr	rrb.gr	sor	sol	sof
2	4	6	14				38

(Sta06 Fig 15.9)

Tietokoneen rakenne / 2006 / Teemu Kerola
5.10.2006
Luento 11 - 23

Procedure Call and Return

Intel kalvo 17

Instruction Execution

```

graph TD
    A[Caller's frame (procA)] -- call --> B[Callee's frame (procB) after call]
    B -- alloc --> C[Caller's frame (procB) after alloc]
    C -- return --> D[Caller's frame (procA) after return]
            
```

Stacked General Registers

```


graph TD
    subgraph Caller
        A[Local A | Output A]
        A -- sof_a = 21 --> B[Output B1]
        A -- sol_a = 14 --> B
    end
    subgraph Callee
        B[Output B1]
        B -- sof_b1 = 7 --> C[Local B | ]
        C -- sof_b2 = 19 --> D[Output B2]
        C -- sol_b2 = 16 --> D
    end
    subgraph Caller2
        D[Local A | Output A]
        D -- sof_a = 21 --> E[ ]
        D -- sol_a = 14 --> E
    end
            
```

Frame Markers

CFM		PFS(pfm)	
sol	sof	sol	sof
14	21	x	x
0	7	14	21
16	19	14	21
14	21	14	21

(Sta06 Fig 15.8)

Tietokoneen rakenne / 2006 / Teemu Kerola
5.10.2006
Luento 11 - 24




Sovelluksen rekisterit

Sta06 Fig 15.7

Kernel registers (KR0-7)	Convey information from the operating system to the application.
Register stack configuration (RSC)	Controls the operation of the register stack engine(RSE)
RSE Backing store pointer (BSP)	Holds the address in memory that is the save location for r32 in the current stack frame.
RSE Backing store pointer to memory stores (BSPSTORE)	Holds the address in memory to which the RSE will spill the next value.
RSE NaT collection register (RNAT)	Used by the RSE to temporarily hold NaT bits when it is spilling general registers.
Compare and exchange value (CCV)	Contains the compare value used as the third source operand in the cmpxchg instruction.
User NaT collection register (UNAT)	Used to temporarily hold NaT bits when saving and restoring general registers with the ld8.fill and st8.spill instructions.
Floating-point status register (FPSR)	Controls traps, rounding mode, precision control, flags, and other control bits for floating-point instructions.
Interval time counter (ITC)	Counts up at a fixed relationship to the processor clock frequency.
Previous function state (PFS)	Saves value in CFM register and related information.
Loop count (LC)	Used in counted loops and is decremented by counted-loop-type branches.
Epilog count (EC)	Used for counting the final (epilog) state in modulo-scheduled loops.

(Sta06 Table 15.5)


Tietokoneen rakenne / 2006 / Teemu Kerola
5.10.2006
Luento 11 - 25



Tietokoneen rakenne

Itanium 2


Tietokoneen rakenne / 2006 / Teemu Kerola
5.10.2006
Luento 11 - 26



Itanium 2

- n Toteutettu IA-64 arkkitehtuuri, 2002
- n Yksinkertaisempi kuin perint. Superskalaari CPU
 - u Ei resurssien "varausasemia"
 - u Ei uudelleenjärjestelypuskureita (ROB)
 - u Ei suuria määriä uudelleennimeämisrekistereitä
 - u Ei logiikkapiirejä riippuvuuksien selvittelyyn
 - u Kääntäjä ratkonut riippuvuudet eksplisiittisesti
- n **Suuri osoiteavaruus**
 - u Pienin yksikkö: 1, 2, 4, 8, 10, 16 tavua
 - u Suositus: kohdenna luonnollisille rajoille
- n **Tukee sekä Big-endian että Little-endian muotoja**

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 27



Itanium 2

- n **Leveä ja nopea väylä: 128b, 6.4 Gbps**
- n **Paranneltu välimuistihierarkia**
 - u L1: erilliset 16KB + 16KB, joukkoass. (4-way), 64B rivit
 - u L2: yhdistetty 256KB, joukkoass. (8-way), 128B rivit
 - u L3: yhdistetty, 3MB, joukkoass. (12-way), 64B rivit
 - u Kaikki on-chip, pienemmät latenssit
- n **TLB hierarkia**
 - u I-TLB L1: 32 alkiota, assosiatiivinen
L2: 128 alkiota, assosiatiivinen
 - u D-TLB L1: 32 alkiota, assosiatiivinen
L2: 128 alkiota, assosiatiivinen

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 28



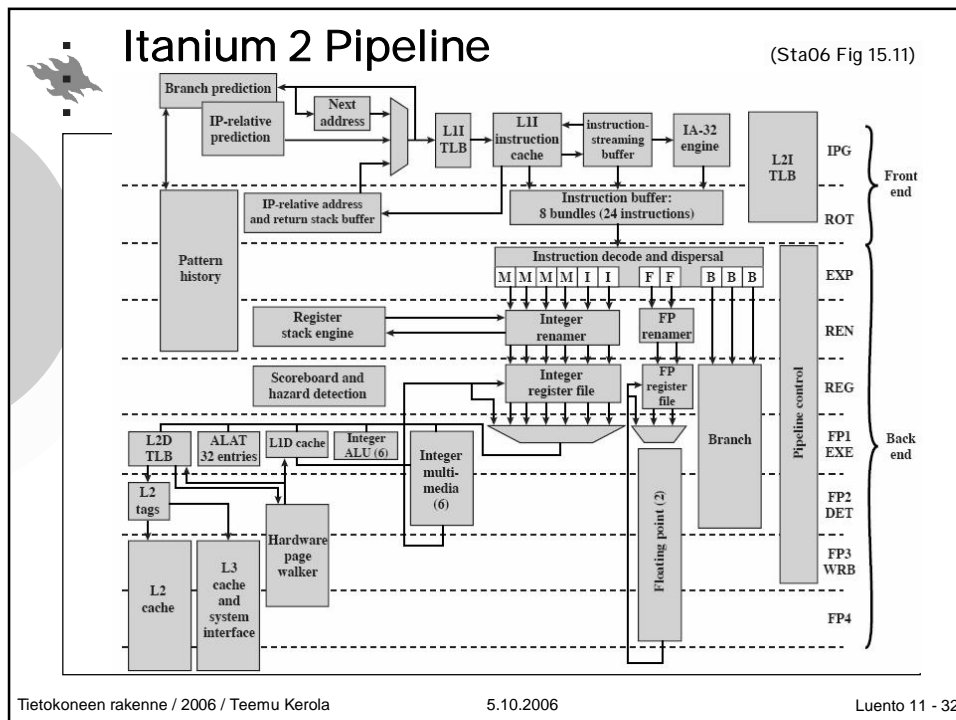
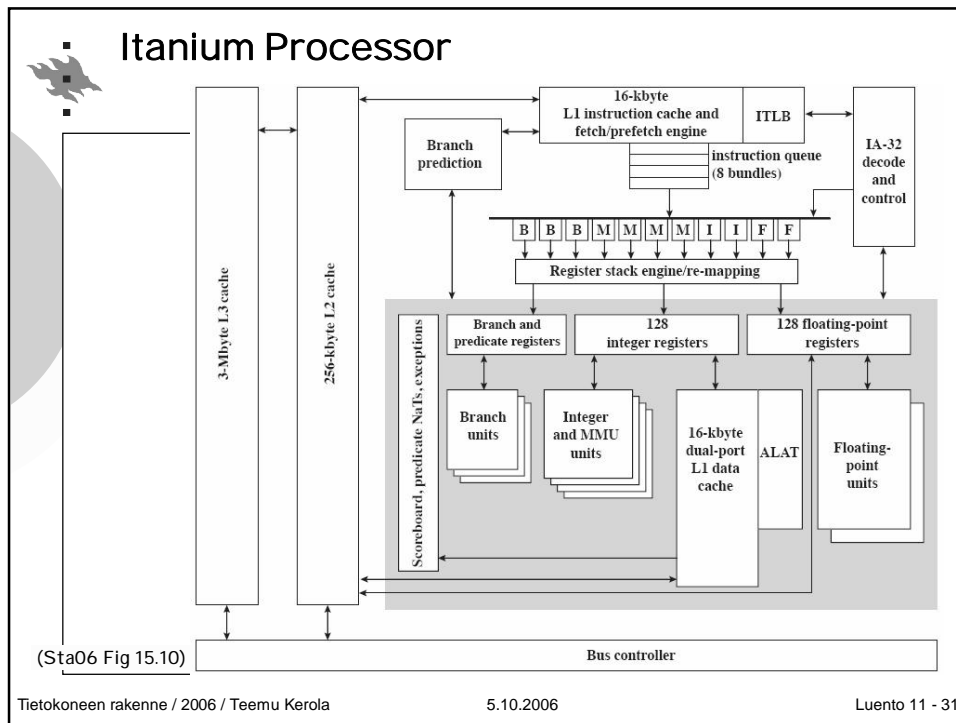
Muistinhallinta

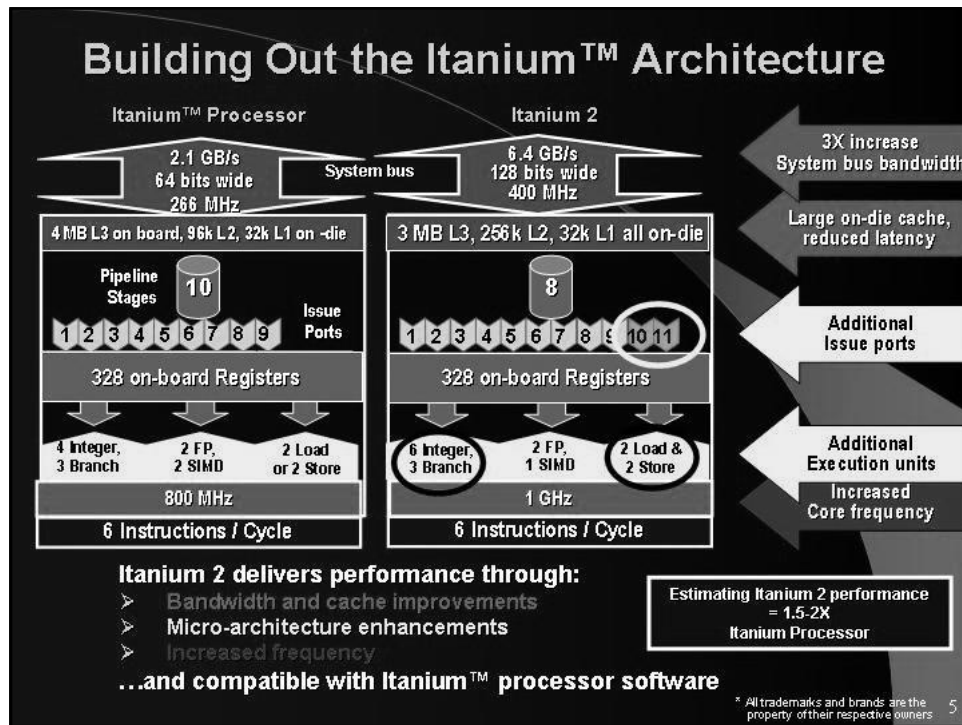
- n **Muistihierarkia näkyy myös sovellukselle**
= mahdollisuus antaa vihjeitä
 - u Noutojärjestys: varmista, että aiemmat operaatiot valmiita
 - u Paikallisuus: nouda paljon/vähän lohkoja välimuistiin
 - u Ennaltanouto: milloin siirtää lähemmäs CPU:ta
 - u Tyhjennys: rivin invalidointi, kirjoituspolitiikka
- n **Implisiittinen kontrolli (poissulkeminen)**
 - u Muistipaikan ja rekisterien sisältöjen vaihto
 - u Vakion lisääminen muistipaikkaan
- n **Mahdollisuus kerätä suorituskykydataa**
 - u Jotta voi antaa parempia vihjeitä...



Itanium 2

- n **11 käskyn suoritukseen valintaikkuna (issue ports)**
- n **Max 6 käskyä suoritettavaksi per sykli**
 - u in-order issue, out-of-order completion
- n **8-vaiheinen liukuhihna**
- n **Entistä enemmän suoritusyksiköjä (22 kpl)**
 - u 6 general purpose ALU's (1 cycle)
 - u 6 multimedia units (2 cycles)
 - u 3 FPU's (4 cycles)
 - u 3 branch units
 - u 4 data cache memory ports (L1: 1/2 cycle load)
- n **Paranneltu hyppyjen ennustuslogiikka**
 - u Myös sovellus voi antaa vihjeitä





Tietokoneen rakenne

Current State (2006)

- Intel hyper-thread and multi-core
- STI multi-core

Intel Pentium 4 HT (IA-32)

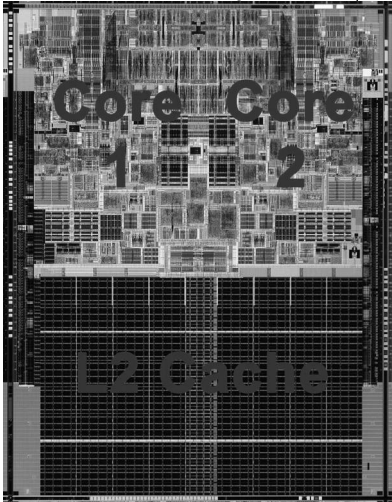
- n HT – Hyper-threading
- n 2 logical processors in one physical processor
- n OS sees it as symmetric 2-processor system
- n Use wait cycles to run the other thread
 - u memory accesses (cache miss)
 - u dependencies, branch miss-predictions
- n Utilize usually idle int-unit, when float unit in use
- n 3.06 GHz + 24%(?)
 - u GHz numbers alone are not so important
- n 20 stage pipeline
- n Dual-core hyper-thread processor
 - u Dual-core Itanium-2 with Hyper-threading

[Click for demos](#)

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 35

Intel Multi-Core Core-Architecture

- n 2 or more (> 100?) complete cores in one chip
 - u No more hyper-threading
 - u Simpler structure, less power
 - u Private L1 cache
 - u Private or shared L2 cache?
- n Intel Core 2 Duo E6700
 - u 128-bit data path
 - u Private 32 KB L1 data cache
 - u Private 32 KB L1 instr. Cache (for micro-ops)
 - u Shared/private 4 MB L2 data cache



[Click 1 or 2 for Torres articles](#) [Click for Pawlowski article](#) <http://www.hardwaresecrets.com/article/366>

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 36

Tietokoneen rakenne

STI Cell Broadband Engine

Tietokoneen rakenne / 2006 / Teemu Kerola
5.10.2006
Luento 11 - 37

STI Cell Broadband Engine

- n **Sony-Toshiba-IBM (STI)**
 - u James Kahle, IBM
- n **1 PowerPC PPE**
 - u Power Processing Element
 - u 32 KB L1 data and instr. caches
 - u 256KB L2 cache
 - u MMU with virtual memory
 - u 2 hyper-threads
 - u "normal programs"
- n **8 SPE's**
 - u Synergistic Processor Elements
 - u 256KB local data/instr memory
 - u Receive code/data packets from off-chip main memory

<http://researchweb.watson.ibm.com/journal/rd/494/kahle.html>

Tietokoneen rakenne / 2006 / Teemu Kerola
5.10.2006
Luento 11 - 38

STI Cell Broadband Engine

- n Programming Models for SPE use
 - u Function offload Model
 - § Run some functions at SPE's
 - u Device Extension Model
 - § SPE as front-end for some device
 - u Computational Acceleration Model
 - § SPE's do most of computation
 - u Streaming Models
 - § Data flow from SPE to SPE
 - u Shared-mem multiprocessor Model
 - § Local store as cache
 - § Cache coherent shared memory
 - u Asymmetric Thread Runtime Model

(a)

[Click for Kahle et al article](http://researchweb.watson.ibm.com/journal/rd/494/kahle.html) <http://researchweb.watson.ibm.com/journal/rd/494/kahle.html>

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 39

STI Cell (Cell B.E.)

- n Sony
 - u Playstation 3 (4 cells)
- n IBM [click](#)
 - u Roadrunner supercomputer (2006-2008)
 - § \$110M, 1100 m², Linux
 - § Peak 1.6 petaflops (1.6 * 10¹⁵ flops)
 - Sustained 1 petaflops
 - § Over 16000 AMD Opterons for file ops and communication (e.g.)
 - Normal servers
 - § Over 16000 Cells for number crunching
 - Blade centers

BlueGene/L, 131072 p5 processors, 225 m²

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 40

STI Cell (Cell B.E.)

- n **Toshiba**
 - u All TV's in 2006?
 - § 1 cell, 2006?
- n **Mercury Computer Systems**
 - u Cell accelerator board (CAB) for PC's
 - u 180 GFlops boost, Linux
- n **Blade servers**
 - u Mercury CTES
 - § Cell Technology Evaluation System
 - § 1-2 Dual-Cell Blades, Linux
 - u IBM Blade Server
 - § 7 boards, 2 Cells each
 - § 2.8 TFlops, Linux

Mercury Dual-Cell Blade

IBM Blade Server prototype w/ 2 cells (2005)

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 41

Kertauskysymyksiä

- n EPI C?
- n Miksi käskynipun yhteydessä on template?
- n Mitä tarkoitetaan predikoinnilla?
Kuinka se toimii?
- n Mitä tarkoittaa kontrollispekulointi?
Entä dataspekulointi?
- n Miten rekistereitä käytetään aliohjelmakutsuissa?
- n Mikä ero hyper-threadeillä ja multi-corella?

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 42