

Tietokoneen rakenne

IA-64

Stallings: Ch 15

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

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 mulstinoudot 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)

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 3

IA-64 Rakenne

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

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

(Sta06 Fig 15.1)

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 4

Käskyformaatti

- n Nouto muistiin
- n Template
- u m...
- u m...

- n PR: Käskyissä spekulointiin liittyvä predikaattirekisteri
 - u 1-bitinen, 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/PR-unit

Template	Slot 0	Slot 1	Slot 2
00	M-unit	F-unit	I-unit
01	M-unit	F-unit	I-unit
02	M-unit	F-unit	I-unit
03	M-unit	F-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	F-unit	B-unit
11	M-unit	F-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

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 7

Symbolisen konekielen formaatti

Käskyryhmän rajat merkitään ;;

- Vihje: nämä konekäskyt voi suorittaa rinnakkain
- Konekielessä template, jossa "musta pystyviiva"
- 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 / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 8

Tietokoneen rakenne

Avainmekanismit

- Predikointi
- Kontrollispekulointi
- Dataspekulointi
- Ohjelmoitu liukuhinna

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

Predikoitu suoritus

Kääntäjä

- Muodosta käskynput, aseta template
- Kuvaa, mitkä käskyt voisi suorittaa samanaikaisesti
- Polsta if-then-else rakenteen hyppyt
- Vertailu asettaa kaksi predikaattirekisteriä
- Kummankin haaran käskyihin mukaan oma predikaatti
- Kumpaakin haaraa tullaan suorittamaan

CPU

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

Intel kalvo 18

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

Predikoitu suoritus

- The branch has two possible outcomes.
- The compiler assigns a predicate register to each following instruction, according to its path.
- All instructions along this path point to predicate register P1.
- All instructions along this path point to predicate register P2.
- CPU begins executing instructions from both paths.
- CPU can execute instructions from different paths in parallel because they have no mutual dependencies.
- 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
  k=k+1
  k=k-1
  i=i+1;
```

Pentium:

```
cmp a,0
je L1
cmp b,0
je L1
add j,1
jmp L3
k=k+1
k=k-1
i=i+1;
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
(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

Aloita datan laataaminen muistista etukäteen
 = spekulatiivinen load

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

Kontrollispekulointi = nosta (hoist) load-käskeytymäksi koodissa hyppykäskeytymän etupuolelle

- Merkitse se kuitenkin spekulatiiviseksi (.s)
- Jos spekulointi aiheuttaa poikkeuksen, sen käsittely viivästetään (NaT bitti)
 - On mahdollista että kyseistä poikkeusta ei pitänyt tapahtua!
- Lisää alkuperäiseen kohtaan chk-käskeytymä (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 kalvo 27-28

- 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).
- 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.
- The compiler replaced this load with the speculative load above, so instruction 8 does not actually appear in the program.
- This instruction checks the validity of the data. If it is OK, the CPU does not report an exception.
- In effect, IA 54 has hoisted the load above the branch.

(Sta06 Fig 15.3b)

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

Dataspekulointi

Intel kalvo 29-30

Nosta load-käskeytymä alemmaksi koodissa store-käskeytymän etupuolelle

- Merkitse se ennakkolataamiseksi (.a eli advanced load)
- Lisää alkuperäiseen paikkaan tarkistus (.c)

ALAT eli Advanced Load Address Table (laittelsto) pitää kirjaa loadissa käytetyistä osoitteista

- Kukin load vie kohdeosoitteen ALAT-tauluun
- Kukin store poistaa kohteen ALAT-taulusta
- 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 liukuhinna Software pipelining

Laitteistotuki silmukan purkamiseksi s.e. voidaan suorittaa useita iteraatioita samanaikaisesti

- Rinnakkaisuus syntyy suorittamalla eri iteraatiokierroksiin kuuluvia toimintoja yhtäaikaan
- Kukin iteraatiokierron käyttää eri registreitä
 - Automaattinen rekistereiden uudelleennimeäminen
- Alku (prolog) ja loppu (epilog) erikoistapauksina rotatoivan predikaattirekisterin avulla
- Silmukan hyppykäskeytymä korvattu erityiskäskeytymällä, joka kontrolloi ohjelmoidun liukuhinnan käyttöä
 - Rotatoi rekisterit, vähentää silmukkalaskuria

Why called software pipeline?

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

Ohjelmoidun liukuhinnan 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 ;;
```

Vähän käskeytymän rinnakkaisuutta, pieni koodi

Enemmän rinnakkaisuutta suoritukseen!
 Operoi eri iteraation rekistereillä samanaikaisesti

Cycle 0	ld4									
Cycle 1		ld4								
Cycle 2			ld4							
Cycle 3				ld4						
Cycle 4					ld4					
Cycle 5						ld4				
Cycle 6							ld4			
Cycle 7								ld4		

Prolog
Kernel
Epilog

Intel kalvo 25 (Sta06 Fig 15.6)

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
```

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	epilog		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 paluusoite tavallisesti rekisteriin br0
 - u Jos uusi kutsu, br0:n talteen rekisteripinon
- n **Instruction pointer**
 - u Nipun osoite, ei yksittäisen kaskyn
- n **User mask**
 - u Lipukkeet poikkeuksia ja monitorointia varten
- n **Performance monitor data registers**
 - u Tietoa laitteiston käyttäytymisestä
 - u Esim. Hyppynnustuksista, 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**
 - u Kutsu varaa pinosta rekisteri-ikkunallisen (frame)
 - u parametrit (inputs/outputs) + paikalliset mjat (locals)
 - u Koko dynaamisesti määriteltävissä (alloc-käskey)
- 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 rotointitietoa
 - § rrb=register rename base
- n **PFS, Previous Function State**
 - u CFM:n edellinen sisältö tänne, vanha PFS jonnekin toiseen rekisteriin (alloc voi määrätä minne)

(Sta06 Fig 15.9)

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

Procedure Call and Return

(Intel kalvo 17)

(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 compare instruction.
User NaT collection register (UNAT)	Used to temporarily hold NaT bits when saving and restoring general registers with the i38.hll and sts spill instructions.
Floating-point status register (FPSR)	Controls trap, 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 osoitevaraus
 - 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, assosiatiiivinen
 - L2: 128 alkiota, assosiatiiivinen
 - u D-TLB L1: 32 alkiota, assosiatiiivinen
 - L2: 128 alkiota, assosiatiiivinen

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

Muistinhallinta

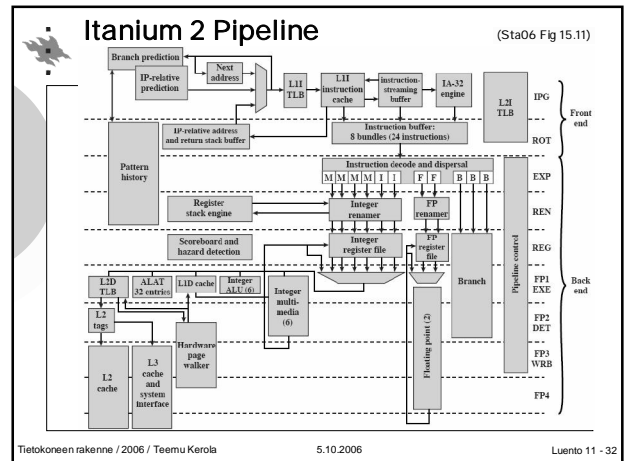
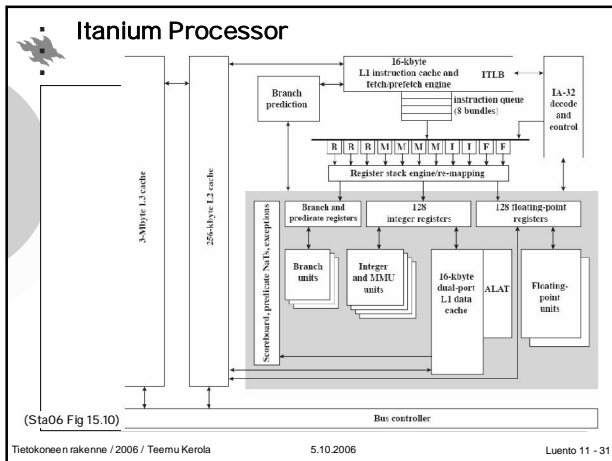
- n Muistihierarkia näkyy myös sovellukselle = mahdollisuus antaa vihjeitä
 - u Noutojärjestys: varmistaa, että aiemmat operaatiot valmiita
 - u Paikallisuus: nouda paljon/vähän lohkoja välimuistiin
 - u Ennaltanouto: milloin siirtää lähemmäs CPU:ta
 - u Tyhjennys: rivin invalidointi, kirjoituspolitiikka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

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 29

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 suoritusyksiköitä (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 / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 30



Building Out the Itanium™ Architecture

Itanium™ Processor	Itanium 2
2.1 GB/s 64 bits wide 265 MHz	6.4 GB/s 128 bits wide 400 MHz
4 MB L3 on board, 8k L2, 32k L1 on -die	3 MB L3, 256k L2, 32k L1 all on -die
Pipeline Stages: 10	Pipeline Stages: 8
Issue Ports: 10	Issue Ports: 11
328 on-board Registers	328 on-board Registers
4 Integer, 3 Branch, 2 FP, 2 SIMD, 2 Load or 2 Store	6 Integer, 3 Branch, 2 FP, 1 SIMD, 2 Load & 2 Store
800 MHz	1 GHz
6 Instructions / Cycle	6 Instructions / Cycle

Itanium 2 delivers performance through:

- Bandwidth and cache improvements
- Micro-architecture enhancements
- Increased frequency

...and compatible with Itanium™ processor software

Estimating Itanium 2 performance = 1.5-2x Itanium Processor

** All trademarks and brands are the property of their respective owners.*

Tietokoneen rakenne

Current State (2006)

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

Tietokoneen rakenne / 2006 / Teemu Kerola 5.10.2006 Luento 11 - 34

Intel Pentium 4 HT (IA-32)

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

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

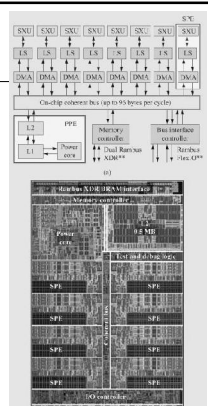
Tietokoneen rakenne

STI Cell Broadband Engine

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

STI Cell Broadband Engine

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

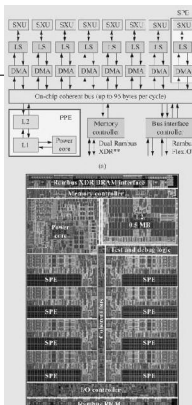


<http://research.watson.ibm.com/journal/rd494/kahle.html>

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

STI Cell Broadband Engine

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




[Click for Kahle et al article](http://research.watson.ibm.com/journal/rd494/kahle.html)

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

STI Cell (Cell B.E.)

- Sony**
 - Playstation 3 (4 cells)
- IBM**
 - 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

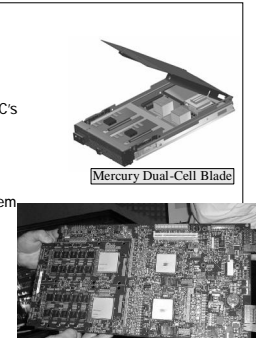


BlueGene/L, 131072 p5 processors, 225 m²

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

STI Cell (Cell B.E.)

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



IBM Blade Server prototype w/ 2 cells (2005)

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

Kertauskysymyksiä

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

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