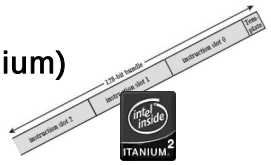
Luento 11

Tietokoneen rakenne

IA-64 (Itanium)



Stallings: Ch 15

- Yleistä IA-64:stä
- Predikointi
- Spekulointi
- Ohjelmoitu liukuhihna (software pipelining)
- Itanium 2

Intel Multi-core ja STI Cell

EPIC (Explicit Parallel Instruction Computing)

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

- Uutta semantiikkaa konekielen tasolle
- Kääntäjä ratkoo riippuvuuteen liittyvät ongelmat, laitteisto (toteutus) luottaa siihen

• VLIW (Very Long Instruction Word)

- Käsittelee käskyjä nipuissa (bundle)

Ž Hyppyjen predikointi, kontrollispekulointi

- Suorittaa useita haarautumispolkuja

• Spekulatiiviset muistinoudot myös datalle

Linuksen kommentti IA-64:sta (2005): http://www.realworldtech.com/forums/index.cfm?action=detail&id=60298&threadid=60123&roomid=2

IA-64 vs. Superskalaari

(Sta06 Table 15.1)

Superscalar	IA-64
RISC-like instructions, one per word	RISC-like instructions bundled 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 compile time
Branch prediction with speculative execution of one path	Speculative execution along both paths of a branch
Loads data from memory only when needed, and tries to find the data in the caches first	Speculatively loads data before its needed, and still tries to find data in the caches first

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

HOW to use?
Bigger cache ?
More processors?
More superscalar?

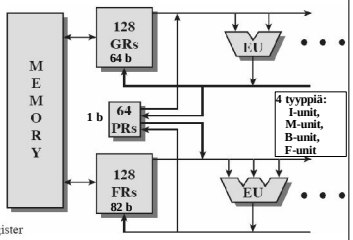
More transistors per chip

pipelining
+ RISC =>
superscalar
=> more and
=>more parallelism and speed

Billions?

IA-64 Rakenne

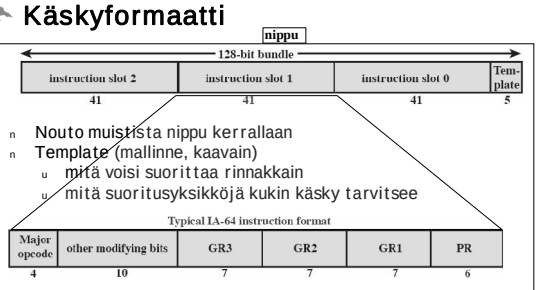
- Paljon rekistereitä => ei uudelleennimeämisiä ja riippuvuuksien analysointia
- vähintään 8 suoritusyksikköä (riippuu lastun transistorien määrästä ja niitä hyödynnetään niin paljon kuin pystytään)
- GR-rekistereissä NaT-bitti (Not a Thing) => "myrkkyä"



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



- Nouto muistista nippu kerrallaan
- Template (mallinne, kaavain)
  - mitä voisi suorittaa rinnakkain
  - mitä suoritusyksikköjä kukin käsky tarvitsee
- PR: Käskyissä spekulointiin liittyvä predikaattirekisteri
  - 1-bittinen, tarkistetaan kommitointihetkellä
- Käskyissä tavallisesti 3 rekisteriä
- Load/Store -arkkitehtuuri

(Sta06 Fig 15.2)

Toimintoyksiköt

(Sta06 Table 15.2, 15.3)

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

- Max 6 käskyä suoritukseen per sykli
- Musta pystyviiva = stop = Käskyjen välillä riippuvuus (ei tarvita NOP-käskyjä)

Template kertoo, mitkä käskyt voidaan suorittaa rinnakkain (parallel).
Useita nippuja voidaan sijoittaa peräkkäin ja asettaa template-kentät sopivasti => esim. 8 rinn. käskyä.

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

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

- **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)
  - Entä antiriippuvuus (WaR)???

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


Tietokoneen rakenne

Avainmekanismit

- Predikointi
- Kontrollispekulointi
- Dataspekulointi
- Ohjelmoitu liukuhihna

Intellin kalvot: http://www.cs.helsinki.fi/u/kerola/tikra/IA64-Architecture.pdf

Predikoitu suoritus

Kääntäjä

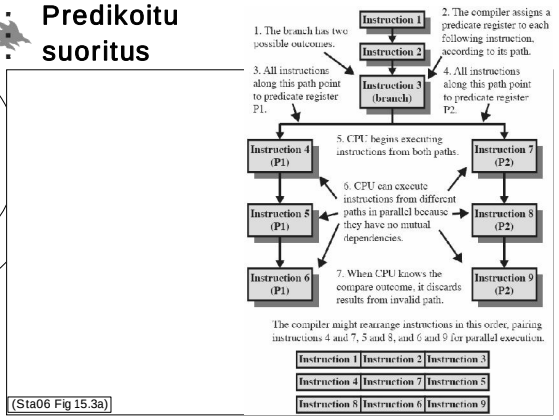
- **Muodosta käskyniput, aseta template**
  - Kuvaa, mitkä käskyt voisi suorittaa samanaikaisesti
  - Käskyjen todellinen suoritusjärjestys kimpun sisällä voi olla mikä vain
- **Poista if-then-else rakenteen hypyt**
  - Vertailu asettaa kaksi predikaattirekisteriä
  - Kummankin haaran käskyihin mukaan oma predikaatti
  - Kumpaakin haaraa tullaan suorittamaan

Intel kalvo 18

CPU

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

Predikoitu suoritus



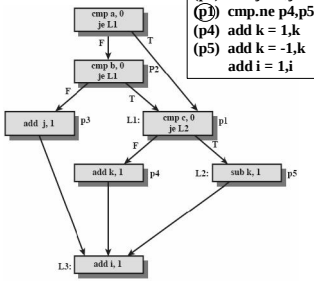
(Sta06 Fig 15.3a)

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)

Load-spekuloinnit

- **Aloita datan lataaminen muistista etukäteen**
  = spekulatiivinen load
  - Valmiina CPU:ssa kun tarvitaan, ei latenssia
  - Yleensä helppoa, mutta ei, jos välissä haarautuminen/store
- **Välissä haarautuminen?**
  - Spekuloitu load voi aiheuttaa keskeytyksen, jota ei olisi koskaan pitänyt tapahtua

```
…
Comp   R1, =Limit
JLE    Done
Load   R5, Table(R1)
…
```

- **Välissä store?**
  - Spekuloitu load voi kohdistua samaan muistipaikkaan, jota store on juuri muuttamassa

```
…
Store R1, (R3)
Load R5, (R4)
…
```


Kontrollispekulointi

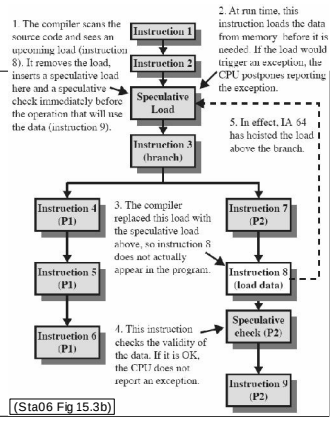
Intel kalvo 26

- **Kontrollispekulointi = nosta (hoist) load-käsky aiemmaksi koodissa <u>hyppykäskyn</u> 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-käsky (chk.s), joka tarkistaa poikkeuksen ja käynnistää recovery-rutiinin

```
je  L2                    ld8.s  r1=[r5]
ld8  r1=[r5]      ⇨       je  L2
use r1                    chk.s  r1, recovery   ← completer
                          use r1
```


Kontrolli-spekulointi

Intel kalvot 27-28



(Sta06 Fig 15.3b)

Dataspekulointi

Intel kalvot 29-30

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

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

<u>Alias-ongelma:</u>
r3 ja r5 voivat osoittaa samaan paikkaan

Intel kalvo 31

Ohjelmoitu liukuhihna

Software pipelining

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

Why called software pipeline?

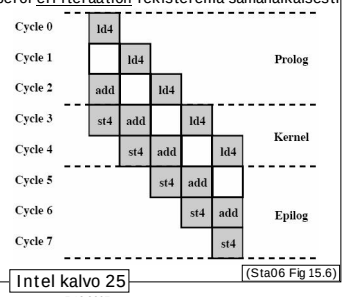
Ohjelmoidun liukuhihnan idea

```
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 ;;
```

Vähän käskytason rinnakkaisuutta, pieni koodi

Paljon suoritusaikaista rinnakkaisuutta!
Operoi <u>eri iteraation</u> rekistereillä samanaikaisesti

```
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
```



(Sta06 Fig 15.6)

Intel kalvo 25

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
```

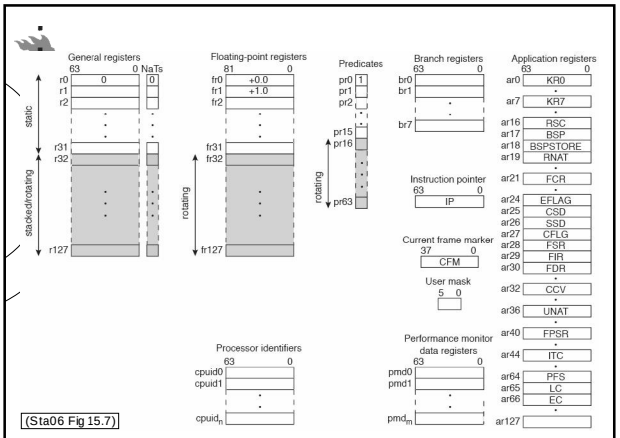
Koodi s.555

(Sta06 Table 15.4)

	Cycle	Execution Unit/Instruction				State before br.ctop					
		M	I	M	B	p16	p17	p18	p19	LC	EC
prolog	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
epilog	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

IA-64 Rekisterit



(Sta06 Fig 15.7)

Sovelluksen rekisterit

Sta06 Fig 15.7

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

Rekisteripino, Register Stack Engine

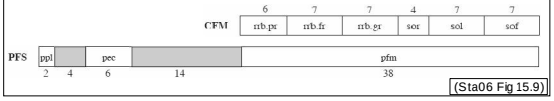
Intel kalvot 15-17

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

Rekisteripino

Intel kalvo 17

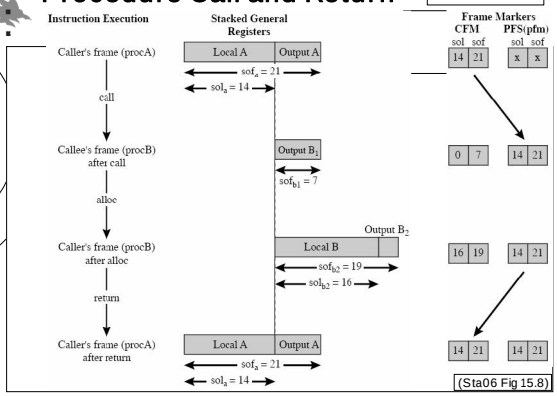
- **Allokointi ja palautus käyttää kahta rekisteriä**
- **CFM, Current Frame Marker**
  - Rekisteripinosta kutsun yhteydessä varatun alueen koko
    - sof=size of frame, sol=size of locals, sor=size of rotation portion (SW pipeline)
  - GR/FP/PR-rekistereiden rotatointitietoa
    - rrb=register rename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



(Sta06 Fig 15.9)

Procedure Call and Return

Intel kalvo 17



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

Itanium 2

Itanium 2

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

Itanium 2

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

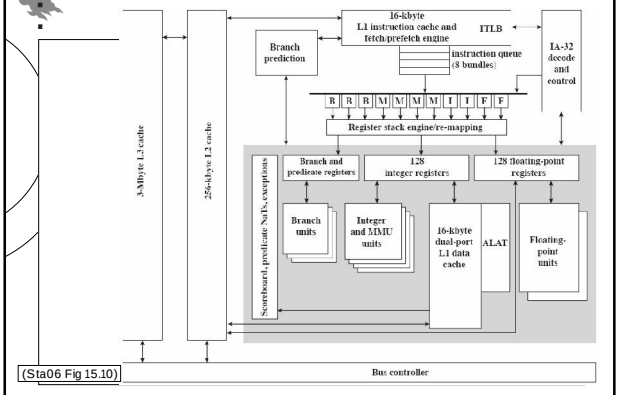
Muistinhallinta

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

Itanium 2

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

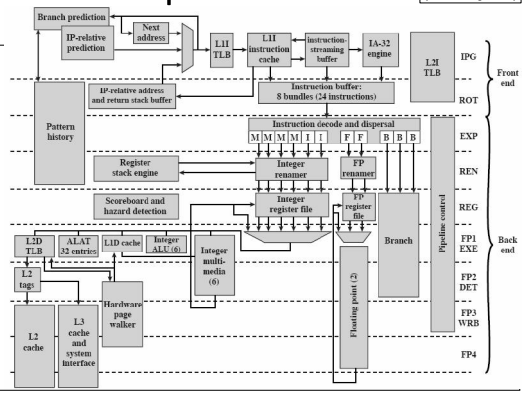
Itanium Processor



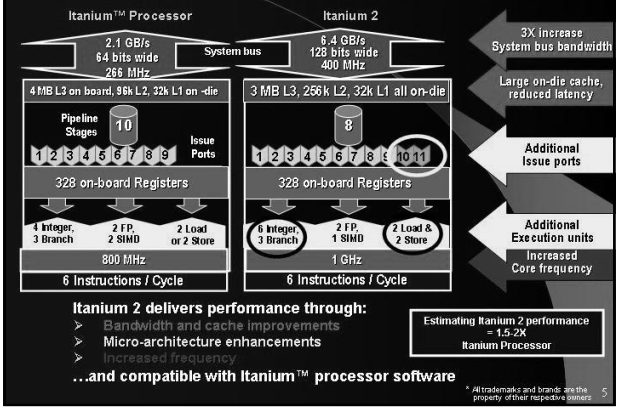
(Sta06 Fig 15.10)

Itanium 2 Pipeline

(Sta06 Fig 15.11)



Building Out the Itanium™ Architecture



Tietokoneen rakenne

Current State (2006-7)

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

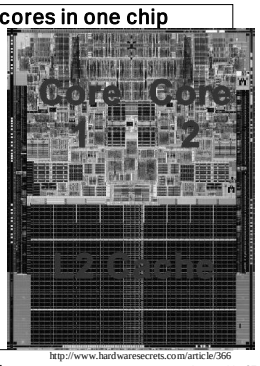
Intel Pentium 4 HT (IA-32)

- **HT – Hyper-threading**
- **2 logical processors in one physical prosessor**
- **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

http://www.intel.com/multi-core/index.htm

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

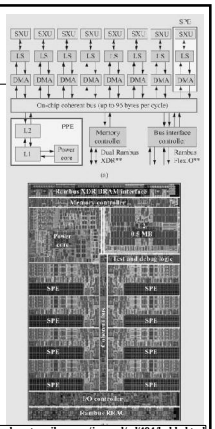
STI Cell Broadband Engine

Tausta

- **Yksinkertaistetaan CPU:n toimintaa**
  - Yksinkertaisia , mutta suorituskykyisiä yksiköitä
    - Useita tehokkaita vektoriprosessointiin erikoistuneita 'työjuhtia' (SPE), joiden toimintaa säätelee "työnjohtaja" (PPE)
  - PPE on tavallinen 64 b PowerPC with VMX
    - RISC arkkitehtuuri, kaksisäikeinen, in-order , yksinkertainen ennustuslogiikka => Kääntäjän tulee huolehti järjestelystä
  - SPE saa tehtävät kokonaisina
    - Data + koodi
    - 256 KB:n oma muisti, ei välimuistia
    - 128 rekisteria a' 128 bittiä, 64 GB/s
    - 2 liukuhihnaa: even, odd
    - Ei mitään hyppyennustuslogiikkaa, "branch hint"-käsky

STI Cell Broadband Engine

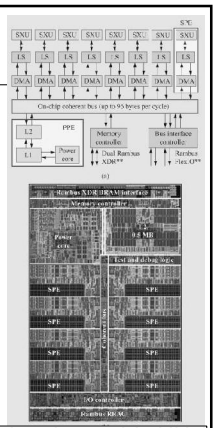
- **Sony-Toshiba-IBM (STI)**
  - James Kahle, IBM
- **1 PowerPC PPE**
  - Power Processing Element
  - 32 KB L1 data and instr. caches
  - 256KB L2 cache
  - MMU with virtual memory
  - 2 hyper-threads
  - "normal programs"
- **8 SPE's**
  - Synergistic Processor Elements
  - 256KB local data/instr memory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



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

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

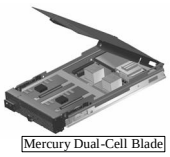
STI Cell (Cell B.E.)

- **Sony**
  - Playstation 3 (4 cells)
- **IBM** click
  - Roadrunner supercomputer (2006-2008)
    - $110M, 1100 m², Linux
    - Peak 1.6 petaflops (1.6 * 10^{15} 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

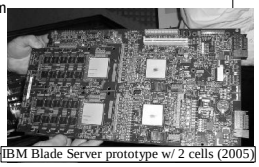
BlueGene/L, 131072 p5 processors, 225 m²

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



Mercury Dual-Cell Blade



IBM Blade Server prototype w/ 2 cells (2005)

Kehitys kulkee ...

- **X86 => Pentium => Core => *Nehalem***
  - Superscalar
    - Yhä tehokkaampaa liukuhihnatekniikan hyödyntämistä
      - Rinnakkaisia liukuhihnoja
      - Haarautumisten ennustaminen
      - Out-of order -suoritus
      - CISC => RICS muunnos
      - Hyperthreading = monistetaan osia suorittimesta
  - Chip –level multiprocessing
    - Yhä useampi suorittimia samalla lastulla
  - Vektorikäskykanta
    - Rinnakkaista datan käsittelyä
  - Välimuisti: useita tasoja, yhä suurempi välimuisti
    - OX9650: 12 MB L2

Hieman eri suuntaan ...

- **Virrankulutus**
  - Kannettavat laitteet
  - Pakkaustiheys => kuumeneminen
- **Superskalaaripolku jo kuljettu loppuun?**
  - Ennustuslogiikan parantaminen tuo yhä pienemmän hyödyn => yksinkertaisempi CPU
    - => Ohjelmallinen toteutus (Transmetan Crusoe, mutta muistin käyttö on hyvin hidasta!)
    - => kääntäjä hoitaa ja CPU saa käskyt paremmin järjestettynä (IA-64, Itanium2, CELL, ..)
- **Yhä useampia prosessoreita yhdellä lastulla**
  - Eri tehtäviä eri prosessoreilla
  - Prosessoreiden toiminnan koordinointi

Kertauskysymyksiä

- EPIC?
- Miksi käskynipun 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?