

Luento 10

## Tietokoneen rakenne

# Superskalaari- prosessointi



Stallings: Ch 14

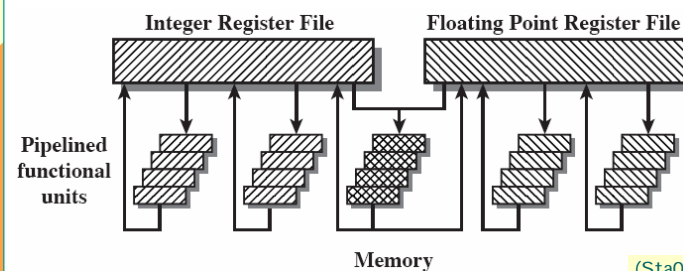
- n Käskyjen väliset riippuvuudet
- n Rekistereiden uudelleennimeäminen
- n Pentium / PowerPC

Tietokoneen rakenne / 2007 / Liisa Marttinen
30.11.2007
Luento 10 - 1

## Superskalaari-prosessointi

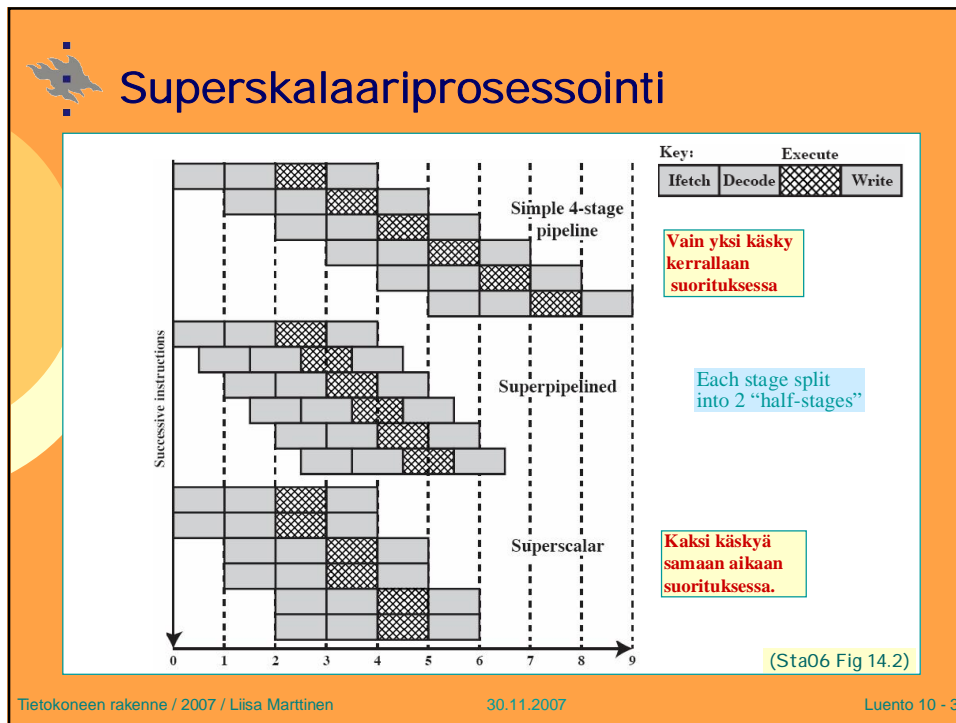
- n **Tavoite**
  - u Nopeuttaa skalaarikäskyjen prosessointia
- n **Useita itsenäisiä liukuhihnoja**
  - u Ei siis pelkästään enemmän vaiheita liukuhihnalla

Reference	Speedup
[TJAD70]	1.8
[KUCK72]	8
[WEIS84]	1.58
[ACOS86]	2.7
[SOHI90]	1.8
[SMIT89]	2.3
[JOUP89b]	2.2
[LEE91]	7



(Sta06 Fig 14.1, Tbl 14.1)

Tietokoneen rakenne / 2007 / Liisa Marttinen
30.11.2007
Luento 10 - 2



## Superskalaariprosessointi

- n Muistin käytön oltava tehokas
  - u Nouda useita käskyjä yhtäaikaan, ennaltanouto
  - u Datat nouto / talletus
  - u Rinnakkaisuus
- n Saman prosessin useita käskyjä yhtäaikaan suorituksessa eri liukuhihnoilla
  - u Valitse noudetuista sopivassa järjestyksessä suoritukseen (in-order issue/out-of-order issue)
- n Enemmän kuin yksi käsky valmistuu per sykli
  - u Saattavat valmistua eri järjestyksessä kuin aloitettu (out-of-order completion)
- n Milloin käsky saa valmistua ennen edeltävää käskyä?

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 4

## Tutut riippuvuudet

- n **Datariippuvuus** (True Data/Flow Dependency)
  - u write-read (Read after Write, RaW) riippuvuus
  - u Jäljempi käsky tarvitsee edeltävän tuottamaa dataa
- n **Kontrolliriippuvuus** (Procedural/Control Dependency)
  - u Hyppyä seuraavat käskyt suoritetaan vain, jos hyppy ei toteudu
  - u Superskalaarihinnalla hukatavana enemmän käskyjä
  - u Vaihtelevanpituisten käskyjen lisäosista tietoa vasta suoritettaessa
- n **Resurssiriippuvuus** (Resource Conflict)
  - u Yksi tai useampi liukuhinnan osa tarvitsee samoja resursseja
  - u Muistipuskuri, ALU, pääsy rekisterijoukkoon, ...

add r1,r2  
 ↙  
 move r3,r1

JNZ R2, 100  
 ADD R1, =1

Tietokoneen rakenne / 2007 / Liisa Marttinen      30.11.2007      Luento 10 - 5

## Tutut riippuvuudet

(Sta06 Fig 14.3)

Tietokoneen rakenne / 2007 / Liisa Marttinen      30.11.2007      Luento 10 - 6

## Uudet riippuvuudet

**Nimiriippuvuudet =**

- n **Kirjoitusriippuvuus** (Output Dependency)
  - u write-after-write (WaW) riippuvuus
  - u Kun kaksi käskyä muuttaa samaa rekisteriä tai muistipaikan sisältöä, niin alkuperäisessä koodissa jäljempänä oleva talletus jäättävä voimaan
- n **Antiriippuvuus** (Antidependency, Read-write dependency)
  - u Write-after-read (WaR) riippuvuus
  - u Edeltävän käskyn ehdittävä noutaa rekisterin tai muistipaikan sisältö, ennen kuin jäljempi käsky tallettaa sinne uuden arvon
- n **Aliakset?**
  - u Kaksi rekisteriä osoittaa epäsuorasti samaan muistipaikkaan?
  - u Eri virtuaaliosoite, sama fyysinen osoite?
  - u Mitä näkyy konekielen tasolla (ennen MMU'ta)?

```
load r1, X
add r2, r1, r3
add r1, r4, r5
```

```
move r2, r1
add r1, r4, r5
```

```
store R5, 40(R1)
load R6, 0(R2)
```

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 7

## Riippuvuuksista

- n i: "write" R1  
.....  
j: "read" R1
- n i: "read" R1  
.....  
j: "write" R1
- n i: "write" R1  
.....  
j: "write" R1

**datariippuvuus**

**Datariippuvuudessa**  
käskyä j ei voi suorittaa ennen käskyä i!

**antiriippuvuus**

Anti- ja kirjoitusriippuvuudessa käskyt i ja j voidaan kyllä suorittaa eri järjestyksessä, mutta on lopuksi varmistettava, että käytetään oikeaa arvoa ja oikea tulos jää voimaan

**kirjoitusriippuvuus**

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 8

## Kuinka käsitellä riippuvuudet?

- n Peruslähtökohta
  - u Kaikki riippuvuudet käsitellään jollain tavoin
- n Perusratkaisu (kuten ennenkin)
  - u Laitteistoa olemassa, joka huomaa riippuvuuden ja pysäyttää liukuhinnan tarvittaessa odottamaan (bubble)
- n Ratkaisu 2
  - u Kääntäjä generoi käskyt sellaisessa järjestyksessä, että riippuvuuksia ei tule
  - u Tällöin ei tarvita erityislaitteistoa
    - § Yksinkertaisempi suoritin, jonka ei tarvitse havaita riippuvuuksia
  - u Kääntäjän laatijan tunnettava kohdeprosessorin toiminta tarkoin

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 9

## Rinnakkaisuus

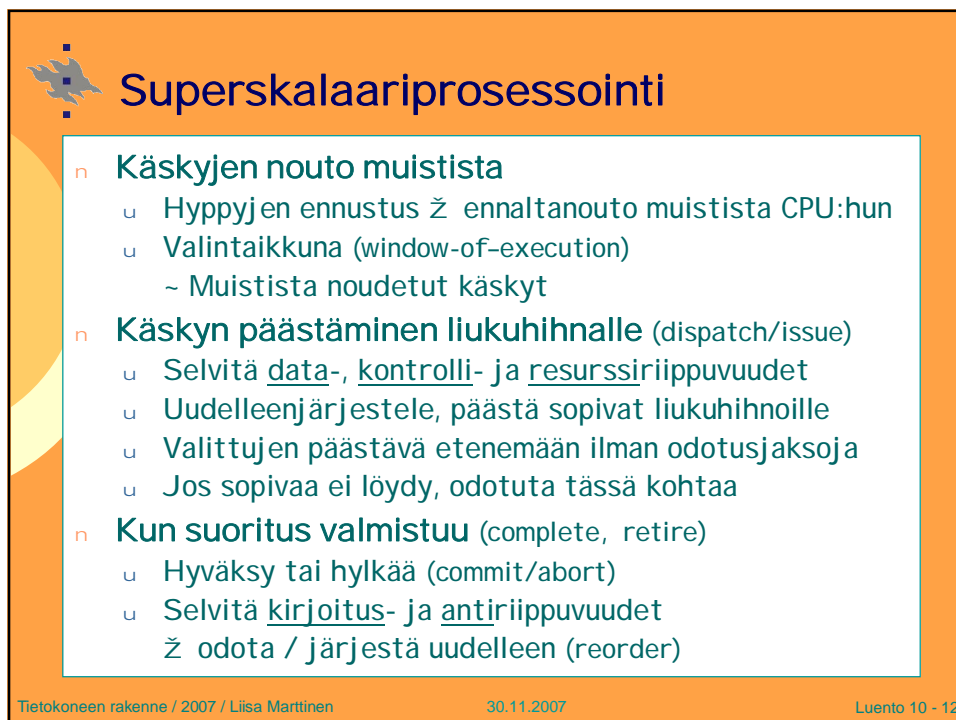
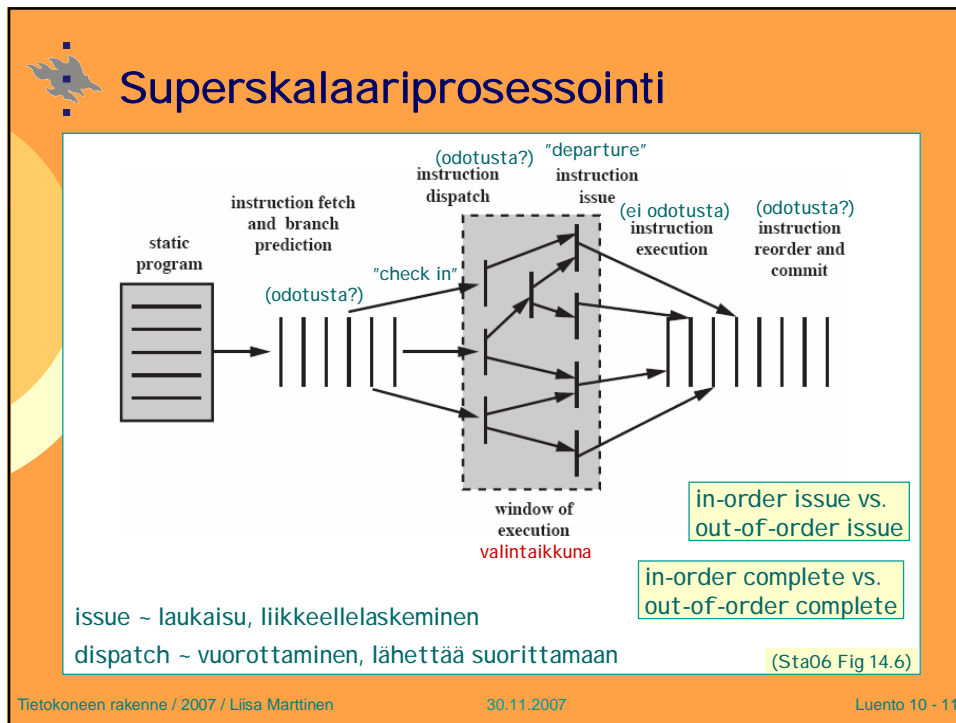
- n Käskytason rinnakkaisuus (Instruction-level parallelism)
  - u Montako käskyä pystyttäisiin teoreettisesti suorittamaan rinnakkain
    - § Riippuu suoritettavasta koodista
- n Konetason rinnakkaisuus (Machine parallelism)
  - u Todellinen rinnakkaisuus - mitä todella saadaan irti?
  - u Mitä tietty kone tai arkkitehtuuri todella voi tehdä rinnakkain
    - § Montako käskyä voi hakea yhtäaikaan?
    - § Montako käskyä voi suorittaa yhtäaikaan?
  - ~ Montako liukuhinnaa käytettävissä
  - u Aina pienempi kuin käskytason rinnakkaisuus
    - § Riippuvuudet?
    - § Huonosti optimoitu toteutus?

```

load r1 • r2
add r3 • r3+1
add r4 • r4, r2

add r3 • r3+1
add r4 • r3, r2
load r0 • r4
  
```

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 10



## In-order issue, in-order complete

- n Se perinteinen peräkkäisjärjestys
- n Ei käyttöä valintaikkunalle
- n Käskyjä hihnoille vain alkuperäisessä järjestyksessä
  - u Kääntäjä huolehtinut pääosin riippuvuuksista
  - u Tarkista silti riippuvuus edeltäjistä
  - u Tsekkaa etenemisvauhti, jätä tarvittaessa kuplia
  - u Voi päästää useita yhtäaikaakin hihnalle
- n Valmistuminen vain alkuperäisessä järjestyksessä
  - u Viereisellä hihnalla ei saa ohittaa
  - u Useita voi valmistua yhtäaikaan
  - u Commit/Abort

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 13

## In-order issue, in-order complete

Nouto 2 käskyä kerralla  
 I1 tarvitsee suoritukseen 2 sykliä  
 I3 ja I4: resurssiriippuvuus  
 I5 (käyttää) ja I4 (tuottaa): datariippuvuus  
 I5 ja I6: resurssiriippuvuus

Decode		Execute		Write		Cycle
I1	I2	I1	I2			1
I3	I4	I1				2
I3	I4					3
	I4		I3	I1	I2	4
I5	I6		I4			5
	I6	I5		I3	I4	6
		I6		I5	I6	7
						8

(Sta06 Fig 14.4a)

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 14

## In-order issue, out-of-order complete

**n** Kuten edellinen, mutta

- Salli valmistua eri järjestyksessä kuin käskyjä aloitettu
- Huolehdi kirjoitus- ja antiriippuvuudesta ennen tulosten kirjoittamista

Nouto 2 käskyä kerralla  
 I1 tarvitsee suoritukseen 2 sykliä  
 I3 ja I4: resurssiriippuvuus  
 I5 (käyttää) ja I4 (tuottaa): datariippuvuus  
 I5 ja I6: resurssiriippuvuus

Decode	Execute	Write	Cycle
I1 I2			1
I3 I4	I1 I2		2
	I1 I3 I4	I2	3
I5 I6	I5 I6	I1 I3	4
		I4	5
		I5	6
		I6	7

**Kirjoitusriippuvuus**

I1: R3 β R3 op R5

I2: R4 β R3 + 1

I3: R3 β R5 + 1

I4: R7 β R3 op R4

(Sta06 Fig 14.4b)

Tietokoneen rakenne / 2007 / Liisa Marttinen      30.11.2007      Luento 10 - 15

## Out-of-order issue, out-of-order complete

**n** Päästä käskyjä liikkeelle parhaaksi katsotussa järjestyksessä

- u Tarvitaan valintaikkuna

**n** Salli valmistuminen parhaaksi katsotussa järjestyksessä

- u Huolehdi kirjoitus- ja antiriippuvuudesta

**Antiriippuvuus**

I1: R3 β R3 op R5

I2: R4 β R3 + 1

I3: R3 β R5 + 1

I4: R7 β R3 op R4

**Se oikea, aito superskalaari-prosessori**

Tietokoneen rakenne / 2007 / Liisa Marttinen      30.11.2007      Luento 10 - 16



## Out-of-order issue, Out-of-order complete

Nouto 2 käskyä kerralla  
 I1 tarvitsee suoritukseen 2 sykliä  
 I3 ja I4: resurssiriippuvuus  
 I5 (käyttää) ja I4 (tuottaa): datariippuvuus  
 I5 ja I6: resurssiriippuvuus

Decode	Window	Execute	Write	Cycle
I1 I2				1
I3 I4	I1, I2	I1 I2		2
I5 I6	I3, I4	I1 I3	I2	3
	I4, I5, I6	I6 I4	I1 I3	4
	I5	I5	I4 I6	5
			I5	6

apupuskuri,  
ei lisävaihe

(Sta06 Fig 14.4c)

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 17

## Rekistereiden uudelleennimeäminen

- Nimiriippuvuusongelman syynä usein se, että toisistaan riippumattomille asioille käytetty samaa rekisteriä
  - Käskyjen välillä tarpeettomia anti- ja kirjoitusriippuvuuksia
  - Turhaa odottelua

$$\begin{aligned}
 R3 &\cdot R3 + R5 \\
 R4 &\cdot R3 + 1 \\
 R3 &\cdot R5 + 1 \\
 R7 &\cdot R3 + R4
 \end{aligned}$$

- Ratkaisu: Rekistereiden uudelleennimeäminen
  - Laitteistossa enemmän rekistereitä kuin ohjelmoijalle näkyy
  - Laitteisto allokoit todelliset rekisterit suoritusajana
  - Allokointi s.e. vältetään nimiriippuvuus
- Tarvitaan
  - Enemmän sisäisiä työrekistereitä (rekisterijoukot), esim. Pentium II:ssa 40 työrekisteriä
  - Laitteistoa, joka allokoit työrekistereitä ja pitää kirjaa ohjelmoijalle näkyvistä rekistereistä

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 18

## Rekistereiden uudelleennimeäminen

**Kirjoitusriippuvuus (WaW):**  
i3 ei saa valmistua ennen i1:stä

**Antiriippuvuus (RaW):**  
i3 ei saa valmistua ennenkuin i2 lukeut arvot R3:sta

Uudelleennimeä R3 s.e. käytössä työrekisterit R3a, R3b, R3c  
Muut rekisterit vastaavasti: R4b, R5a, R7b

**Ei enää nimiriippuvuuksia!**

$R3 \cdot R3 + R5$	(i1)
$R4 \cdot R3 + 1$	(i2)
$R3 \cdot R5 + 1$	(i3)
$R7 \cdot R3 + R4$	(i4)

$R3b \cdot R3a + R5a$	(i1)
$R4b \cdot R3b + 1$	(i2)
$R3c \cdot R5a + 1$	(i3)
$R7b \cdot R3c + R4b$	(i4)

Miksi R3a ja R3b?

Tietokoneen rakenne / 2007 / Liisa Marttinen

30.11.2007

Luento 10 - 19

## Lisälaitteiston vaikutus

- n base: out-of-order issue
- n +ld/st: base ja lisäksi kaksi load/store yksikköä datavälimuistille
- n +alu: base ja lisäksi kaksi ALUa

(Sta06 Fig 14.5)

Window size (construction)    8    16    32


Without renaming

With renaming

Tietokoneen rakenne / 2007 / Liisa Marttinen

30.11.2007


Luento 10 - 20



## Yhteenveto

- n Useita toiminnallisesti itsenäisiä yksiköjä Sta06 Fig 14.6
- n Muistihierarkian tehokas käyttö
  - u Sallii useita rinnakkaisia muistinoutoja/talletuksia
- n Käskyjen ennaltanouto
  - u Hyppyjen ennustuslogiikka
- n Laitetason logiikka riippuvuuksien huomaamiseksi
  - u Ohituspiirit, joilla tieto heti suoraan toiselle yksikölle samaan aikaan kuin tulos rekisteriin tai muistiin
- n Laitetason logiikka useiden riippumattomien käskyjen liikkeellesaattamiseksi (issue)
  - u Riippuvuudet ž järjestys
- n Laitetason logiikka huolehtii käskyjen oikeasta valmistumisjärjestyksestä (completion)
  - u Riippuvuudet ž commit-järjestys (tilanmuutokset)

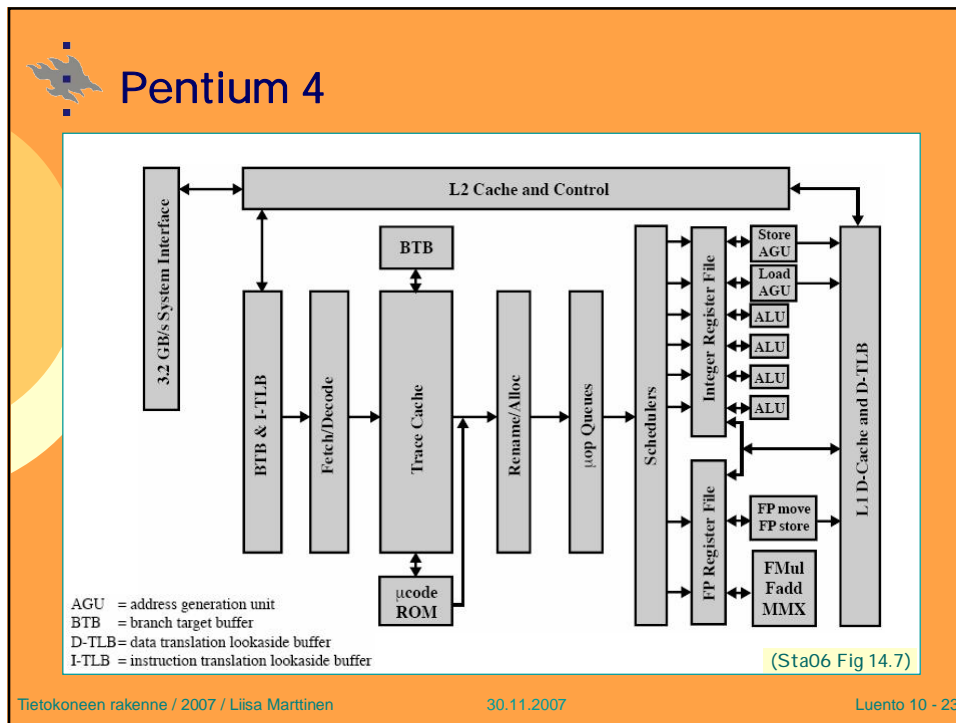
Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 21



## Tietokoneen rakenne

# Pentium 4

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 22



## Liukuhihna

- n Ulospäin CISC -käskykanta (IA-32)
- n Suoritus kuitenkin mikro-operaatioina kuten RISC
  - u Nouda CISC-käsky ja muodosta siitä mikro-operaatiot ( $\mu$ ops) L1 tason välimuistiin (trace cache)
  - u Hihnan loppuosa operoi vakiopituisilla  $\mu$ -operaatioilla (118b)
- n Pitkä liukuhihna
  - u Lisävaiheet (5 ja 20) signaalien etenemisviipeen vuoksi

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
TC	Nxt	IP	TC	Fetch	Drive	Alloc	Rename	Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br	Ck	Drive

TC Next IP = trace cache next instruction pointer    Rename = register renaming    RF = register file  
 TC Fetch = trace cache fetch    Que = micro-op queuing    Ex = execute  
 Alloc = allocate    Sch = micro-op scheduling    Flgs = flags  
 Disp = Dispatch    Br Ck = branch check

(Sta06 Fig 14.8)

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 24

## Mikro-operaatioiden generointi

Sta06 Fig 14.9a-d

a) Nouda IA-32 käsky L2 välimuistista ja generoi mikro-operaatiot L1 tason välimuistiin (trace cache)

- u Käskyille oma TLB, hyppyjen kohteille oma BTB
- u Staattinen hyppyjen ennustus
  - § taaksepäin "taken",
  - § eteenpäin "not taken"
- u 1-4 μops (=118 bitin RI SC) per käsky, näitä monimutkaisemmat ROM-muistissa

b) Määritä Trace-Cache-IP:n arvo mikro-operaatiolle

- u Dynaaminen hyppyjen ennustus (4-bit)
- u 512 alkion joukkoassosiatiivinen kohdepuskuri BTB (branch target buffer), joukon koko 4

c) Nouda operaatio L1 tason välimuistista

d) Drive – odotusjakso



Tietokoneen rakenne / 2007 / Liisa Marttinen

30.11.2007

Luento 10 - 25

## Resurssien allokointi

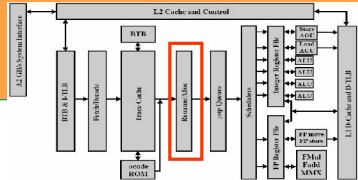
Sta06 Fig 14.9e

e) Allokoi resurssit, rekistereiden uudelleennimeäminen

- u 3 operaatiota per sykli
- u Varaa alkio uudelleenjärjestelypuskurista (126:sta) (reorder buffer, ROB)
- u Varaa tulokselle yksi 128 sisäisestä työrekisteristä ja mahdoll. yksi load ja yksi store puskurit (48:sta ja 24:stä)
- u Poista nimiriippuvuuksia rekistereiden uudelleennimeämisellä
- u Allokoi alkio mikro-operaatioiden valintajonosta
- u Jos ei vapaita resursseja, odota (ž out-of-order)

n **ROB-alkiossa kirjanpito operaation etenemisestä hinnalla**

- u Mikro-operaatio, ja alkuperäisen IA-32 käskyn osoite
- u State: scheduled, dispatched, completed, ready
- u Register Alias Table (RAT):  
mikä IA-32 rekisteri ž mikä työrekisteri

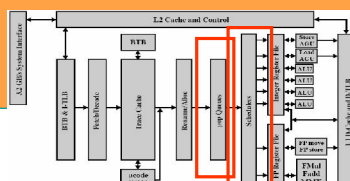


Tietokoneen rakenne / 2007 / Liisa Marttinen

30.11.2007

Luento 10 - 26

## Window of Execution



f) 2 FIFO jonoa mikro-operaatioiden valitsemiseksi

- Tarvittavat resurssit saatavilla, ei riippuvuutta
- Muistiin viittaaville oma, muille oma

g) Mikro-operaatioiden nouto jonoista (scheduling)

- Ne joiden operandit ovat jo kunnossa

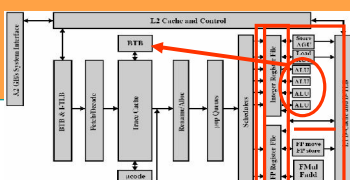
h) Päästäminen liukuhihnalle (dispatching)

- Tutki FIFO-jonojen keuhkimmaisten ROB-alkioita
- Jos tarvittava suoritusyksikkö vapaa, operaation voi päästää suoritusliukuhihnalle
- Kaksi jonoa  $\rightarrow$  out-of-order issue
- max 6 mikro-op liikkeelle yhden syklin aikana
  - ALU:ille tai FPU:ille kummallekin max 2 per sykli
  - Load ja store -yksiköille kummallekin max 1 per sykli

Sta06 Fig 14.9f-h

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 27

## Integer ja FP yksiköt



i) Nouda data rekistereistä tai välimuistista

j) Suorita käsky, aseta lipukkeet

- Hae operandit rekistereistä / L1 välimuistista
- Useita liukuhihnoitettuja suoritusyksiköitä
  - 2 \* Alu, 2 \* FPU, 2 \* load/store
  - Esim. nopea ALU helpoille, kertolaskuille oma ALU
- Tulosten talletus: in-order complete
- Päivitä ROB, salli uusien operaatioiden tulo hihnalle

k) Tarkista miten hypykyssä kävi


- Menikö niinkuin ennustettiin?
- Abortoi väärät käskyt hihnalta (estä tulosten talletus)

l) Kirjaa hypyn tulos ennustuslogiikkaa varten

- Anna aikaa signaalien etenemiseksi

Sta06 Fig 14.9i-l


Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 28



## Pentium 4 Hyperthreading

- n Yksi fyysinen IA-32 CPU, mutta 2 loogista CPU:ta
- n Näkyy KJ:lle kahden CPU:n SMP-järjestelmänä
  - u Molemmat suorittavat eri prosesseja, tai saman prosessin eri säikeitä (kuten SMP = Symmetric multiprocessing)
  - u Ei tarvitse huomioida kooditasolla
  - u KJ:n osattava SMP-temput (mm. vuorottaminen)
- n Perustuu CPU:n odotussykliin hyötykäyttöön
  - u Muistiinviittaus (cache miss)
  - u Riippuvuudet, väärä hyppyennustus
- n Jos toinen käyttää FP-yksikköä, toinen voi käyttää INT-yksikköä
  - u Hyödyt pitkälti sovellusriippuvia

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 29



## Pentium 4 Hyperthreading

- n **Kahdennettu**
  - u IP, EFLAGS ja muut kontrollirekisterit
  - u KäskyTLB Sta06 Fig 14.7
  - u Rekistereiden uudelleennimeämislogiikka Sta06 Fig 14.8
- n **Puolitettu**
  - u Kristillinen tasajako, ei monopolia
  - u Uudelleenjärjestelypuskurit (ROB)
  - u Mikro-operaatioiden valintajonot (2 keulaa?)
  - u Load/store puskurit
- n **Yhteiskäytössä**
  - u Rekisterijoukot (128 GPRs, 128 FPRs)
  - u Välimuistit: trace cache, L1, L2, L3
  - u Mikro-operaatioiden suorituksessa tarvittavat rekisterit
  - u Suoritusyksiköt: 2 ALUa, 2 FPUa, 2 ld/st-units

Nehalem: yhdellä lastulla 8 ydintä, 1-16 säiettä (820 miljoonaa transistoria)

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 30

## Tietokoneen rakenne

# PowerPC

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 31

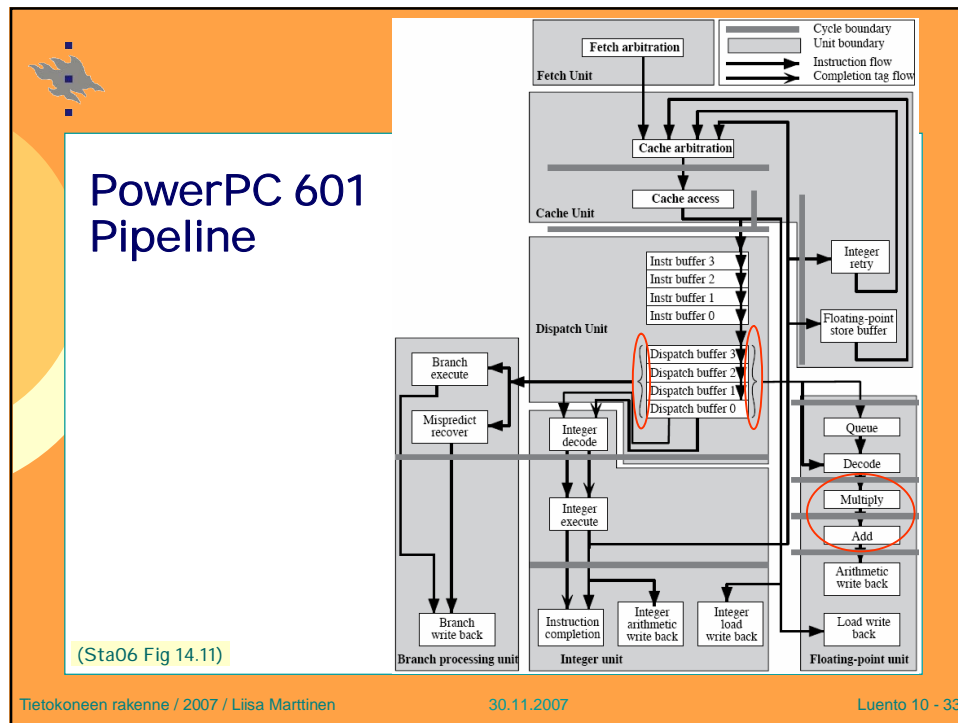
## PowerPC 601

- n Käskyjen noutoyksikkö
  - u Voi noutaa 8 käskyä (a' 32b) kerralla välimuistista
- n Käskyjen valinta suoritukseen (dispatch)
- n 3 käskyjä suorittavaa yksikköä
  - u Kokonaisluvut, liukuluvut, hyppyt

(Sta06 Fig 14.10)

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 32





## Dispatch unit

Sta06 Fig 14.11

- = Käskyjen suoritukseen valinta
- n 4:n alkion apupuskuri + 4:n alkion valintaikkuna
  - u dispatch buffer = window of execution
- n Käsky ikkunasta suoritettavaksi, kun hihnalla tilaa
  - u Integer-yksikölle vain jonon alusta
  - u Muille se, joka lähinnä jonon keulaa
  - u Max 3 käskyä yhtäaikaan (out-of-order issue)
- n Kun käsky suoritukseen, muita jonoissa eteenpäin
- n Jos riippuvuus, ei päästä hihnalle (stall, bubble)
- n Laitetason logiikka hyppyosoitteiden laskemiseksi
  - u Nopeasti jo ennen varsinaista käskyn suoritusta

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 34

## Suoritus

Sta06 Fig 14.11

- n Muutokset rekistereihin / muistiin vasta suorituksen lopuksi "Write Back" vaiheessa
- n ALU-operaatiot tallettavat tietoa CR-rekisteriin (condition register)
  - u 8 kenttää a' 4 b, tallessa useita edellisiä vertailutuloksia
- n Liukuluvut tarvitsevat useampia syklejä

Branch Instructions	Fetch	Dispatch Decode Execute Predict				
Integer Instructions	Fetch	Dispatch Decode	Execute	Writeback		
Load/store Instructions	Fetch	Dispatch Decode	Addr gen	Cache	Writeback	
Floating-point Instructions	Fetch	Dispatch	Decode	Execute1	Execute2	Writeback

(Sta06 Fig 14.12)

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 35

## Hyppyjen käsittely

- n Zero cycle branches
  - u Hyppy ei vaikuta muiden yksikköjen toimintaan
    - § Yleensä ei tarvetta tyhjentää, tai hylätä tuloksia
  - u Hypyn kohdeosoite selvillä heti, kun hyppykäsky tulee käskyikkunaan (ennen suoritusta!)
- n Yhden hypyn spekulointi: Hyppääkö vai ei?
  - u Jos ehdoton ž taken
  - u Jos ehdollinen ja CR-rekisteri asetettu aiemmin
    - tutki ž taken / not taken
    - muuten jos taaksepäin ž taken
    - jos eteenpäin ž not taken
- n Jos spekulointi meni pieleen
  - u abortoi spekuloidut käskyt ennen write-back -vaihetta

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 36

## Hyppyjen käsittely

```

if (a > 0)
    a = a + b + c + d + e;
else
    a = a - b - c - d - e;
    
```

```

#r1 points to a,
#r1+4 points to b,
#r1+8 points to c,
#r1+12 points to d,
#r1+16 points to e.

lwz  r8=a(r1)      #load a
lwz  r12=b(r1,4)   #load b
lwz  r9=c(r1,8)    #load c
lwz  r10=d(r1,12)  #load d
lwz  r11=e(r1,16)  #load e
cmpi  cr0=r8,0     #compare immediate
bc    ELSE,cr0/gt=false #branch if bit false

IF:
    add  r12=r8,r12  #add
    add  r12=r12,r9  #add
    add  r12=r12,r10 #add
    add  r4=r12,r11  #add
    stw  a(r1)=r4    #store
    b    OUT         #unconditional branch

ELSE:
    subf r12=r12,r8  #subtract
    subf r12=r9,r12  #subtract
    subf r12=r10,r12 #subtract
    subf r4=r12,r11  #subtract
    stw  a(r1)=r4    #store

OUT:
    
```

(Sta06 Fig 14.13)

Tietokoneen rakenne / 2007 / Liisa Marttinen      30.11.2007      Luento 10 - 37

## Hyppyjen käsittely

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
lwz r8=a(r1)	F	D	E	C	W											
lwz r12=b(r1,4)	F	.	D	E	C	W										
lwz r9=c(r1,8)	F	.	.	D	E	C	W									
lwz r10=d(r1,12)	F	.	.	.	D	E	C	W								
lwz r11=e(r1,16)	F	.	.	.	.	D	E	C	W							
cmpi cr0=r8,0	F	.	.	.	.	.	D	<b>E</b>								
bc ELSE,cr0/gt=false	F	.	.	.	<b>S</b>	.	.	<b>E</b>	W							
IF: add r12=r8,r12	F	.	.	.	.	.	.	D'	<b>E</b>	W						
add r12=r12,r9			F	.	.	.	.	.	D	E	W					
add r12=r12,r10			F	.	.	.	.	.	.	D	E	W				
add r4=r12,r11			F	.	.	.	.	.	.	F	.	D	E	W		
stw a(r1)=r4			F	.	.	.	.	.	.	F	.	.	D	E	C	
b OUT																
ELSE: subf r12=r8,r12																
subf r12=r12,r9																
subf r12=r12,r10																
subf r4=r12,r11																
stw a(r1)=r4																
OUT:																

(a) Correct prediction: Branch was not taken

(Sta06 Fig 14.14a)

Tietokoneen rakenne / 2007 / Liisa Marttinen      30.11.2007      Luento 10 - 38

## Hyppyjen käsittely

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
lwz r8=a(r1)	F	D	E	C	W											
lwz r12=b(r1,4)	F	.	D	E	C	W										
lwz r9=c(r1,8)	F	.	.	D	E	C	W									
lwz r10=d(r1,12)	F	.	.	.	D	E	C	W								
lwz r11=e(r1,16)	F	.	.	.	.	D	E	C	W							
cmpi cr0=r8,0	F	.	.	.	.	.	.	D	<b>E</b>							
<b>bc</b> ELSE,cr0/gt=false	F	.	.	.	S											
IF: add r12=r8,r12	F	.	.	.	.	.	.	D'	E' (no W)							
add r12=r12,r9									D'	E' (no W)						
add r12=r12,r10																
add r4=r12,r11																
stw a(r1)=r4																
b OUT																
ELSE: subf r12=r8,r12									F	D	<b>E</b>	W				
subf r12=r12,r9									F	.	D	E	W			
subf r12=r12,r10									F	.	.	D	E	W		
subf r4=r12,r11									F	.	.	.	D	E	W	
stw a(r1)=r4									F	.	.	.	.	D	E	C
OUT:																

(b) Incorrect prediction: Branch was taken

(Sta06 Fig 14.14b)

Tietokoneen rakenne / 2007 / Liisa Marttinen

30.11.2007

Luento 10 - 39

## 64b:n arkkitehtuuri

# PowerPC 620

The PowerPC 620 microprocessor: a high performance superscalar RISC microprocessor  
Levitan, D.; Thomas, T.; Tu, P.;


Compton '95. 'Technologies for the Information Superhighway',  
Digest of Papers. 5-9 March 1995 Page(s):285 - 291

- n **6 suoritusyksikköä**
  - u 1 instruction-yksikkö (dispatcher) HePa96 Fig. 4.49
  - u 3 integer-yksikköä
  - u Load/Store yksikkö
  - u FP-yksikkö
- n **Max 4 käskyä suoritukseen yhtäaikaan**
- n **Reservation stations**
  - u Kullakin yksiköllä kaksi tai useampia
  - u Jos käsky ei voi edetä (riippuvuudet) se odottaa tässä, eikä estä jäljempänä olevien etenemistä
- n **Uudelleennimeäminen: 8 Integer ja 12 FP lisärekisteriä**
  - u Vähentää riippuvuuksia
  - u Väli aikaistulosten tallettamiseksi
- n **In-order-complete**
  - u max 4 käskyä yhtäaikaan

Tietokoneen rakenne / 2007 / Liisa Marttinen

30.11.2007


Luento 10 - 40



## PowerPC 620

- n Hyppyjen spekulointi
  - u 256:n alkion branch target buffer (BTB)
    - § Joukkoassosiatiivinen, joukon koko 2
  - u 2048:n alkion branch history table
    - § Käyttää, jos kohde ei löydy BTB:stä
- n Spekuloitavana max 4 ratkaisematonta hyppyä
- n Tulokset uudelleennimeämisrekistereissä
  - u Commit: kopioi spekuloidut tulokset todellisiin rekistereihin
  - u Abort: vapauttaa rekisterit muuhun käyttöön

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 41



## Kertauskysymyksiä

- n Miten superskalaaritoteutus eroaa tavallisesta liukuhihnoitetusta toteutuksesta?
- n Mitä uusia rakenteesta johtuvia ongelmia tulee ratkottavaksi?
- n Miten niitä ongelmia ratkotaan?
- n Mitä tarkoittaa rekistereiden uudelleennimeäminen ja mitä hyötyä siitä on?

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 42