

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

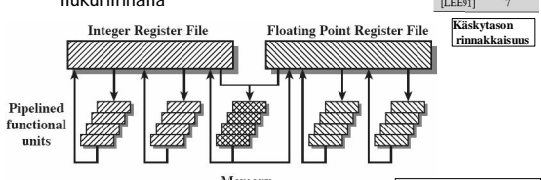
Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 4

Superskalaari-prosessointi

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

Reference	Speedup
[TIAD70]	1.8
[KUCK72]	8
[WEISS4]	1.58
[ACOS86]	2.7
[SOHB90]	1.8
[SMI89]	2.3
[JOU789b]	2.2
[LEE91]	7

Käskytason rinnakkaisuus



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

Tutut riippuvuudet

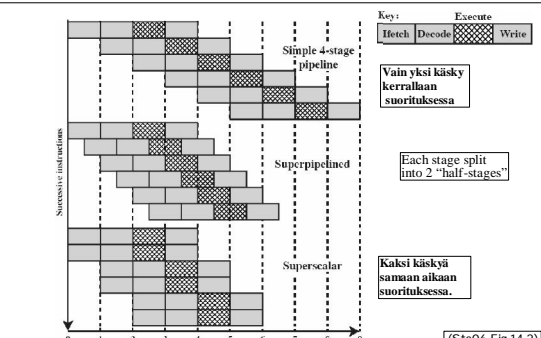
- n **Datariippuvuus** (True Data/Flow Dependency)
 - u write-read (Read after Write, RaW) riippuvuus
 - u Jäljempi käsky tarvitsee edeltävän tuottamaa dataa
- n **Kontrolliriippuvuus** (Procedural/Control Dependency)
 - u Hyppä seuraavat käskyt suoritetaan vain, jos hyppy ei toteudu
 - u Superskalaarihihnalla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

```
add r1,r2
move r3,r1
```

```
JNZ R2,100
ADD R1,-1
```

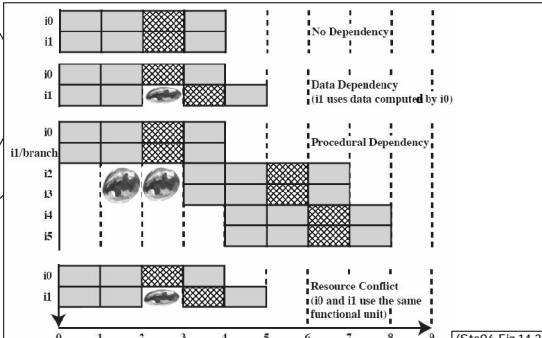
Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 5

Superskalaari-prosessointi



Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 3

Tutut riippuvuudet



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

Uudet riippuvuudet

Nimiriippuvuudet =

- Kirjoitusriippuvuus** (Output Dependency)
 - write-after-write (WaW) riippuvuus
 - Kun kaksi käskyä muuttaa samaa rekisteriä tai muistipaikan sisältöä, niin alkuperäisessä koodissa jäljempänä oleva talletus jäätävä voimaan
- Antirriippuvuus** (Antidependency, Read-write dependency)
 - Write-after-read (WaR) riippuvuus
 - Edeltävän käskyn ehdittävä noutaa rekisterin tai muistipaikan sisältö, ennen kuin jäljempi käsky tallettaa sinne uuden arvon
- Allakset?**
 - Kaksi rekisteriä osoittaa epäsuorasti samaan muistipaikkaan?
 - Eri virtuaaliosoitte, sama fyysinen osoite?
 - 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

Rinnakkaisuus

- Käskytason rinnakkaisuus** (Instruction-level parallelism)
 - Montako käskyä pystyttäisiin teoreettisesti suorittamaan rinnakkain
 - Riippuu suoritettavasta **koodista**
- Konetason rinnakkaisuus** (Machine parallelism)
 - Todellinen rinnakkaisuus - mitä todella saadaan irti?
 - Mitä tietty kone tai arkkitehtuuri todella voi tehdä rinnakkain
 - Montako käskyä voi hakea yhtäaikaan?
 - Montako käskyä voi suorittaa yhtäaikaan?
 - Montako liukuhinnua käytettävissä
 - 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

Riippuvuuksista

- Datariippuvuus**
 - i: "write" R1
 -
 - j: "read" R1

Datariippuvuudessa käskyä j ei voi suorittaa ennen käskyä i!
- Antirriippuvuus**
 - i: "read" R1
 -
 - j: "write" R1

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**
 - i: "write" R1
 -
 - j: "write" R1

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

Superskalaari prosessointi

static program → instruction fetch and branch prediction → (odotusta?) → "check in" → (odotusta?) → "departure" instruction issue → (ei odotusta) instruction execution → (odotusta?) instruction reorder and commit

in-order issue vs. out-of-order issue

in-order complete vs. out-of-order complete

issue - laukaisu, liikkeellelaskeminen
dispatch - vuorottaminen, lähettää suorittamaan (Sta06 Fig 14.6)

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 11

Kuinka käsitellä riippuvuudet?

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

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

Superskalaari prosessointi

- Käskyjen nouto muistista**
 - Hyppyjen ennustus & ennaltanouto muistista CPU:hun
 - Valintaikkuna (window-of-execution)
 - Muistista noudetut käskyt
- Käskyn päästäminen liukuhinnalle** (dispatch/issue)
 - Selvitä data-, kontrolli- ja resurssiriippuvuudet
 - Uudelleenjärjestele, päästä sopivat liukuhinnoille
 - Valittujen päästävä etenemään ilman odotusjaksoja
 - Jos sopivaa ei löydy, odotuta tässä kohtaa
- Kun suoritus valmistuu** (complete, retire)
 - Hyväksy tai hylkää (commit/abort)
 - Selvitä kirjoitus- ja antiriippuvuudet
 - odota / järjestä uudelleen (reorder)

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 12

In-order issue, in-order complete

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

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

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

- Päästä käskyjä liikkeelle parhaaksi katsotussa järjestyksessä
 - Tarvitaan valintaikkuna
- Salli valmistuminen parhaaksi katsotussa järjestyksessä
 - Huolehdi kirjoitus- ja antiriippuvuudesta

Antiriippuvuus
 I1: R3 ~~B~~ R3 op R5
 I2: R4 ~~B~~ R3 + 1
 I3: R3 ~~B~~ R5 + 1
 I4: R7 ~~B~~ R3 op R4

Se oikea, aito superskalaari-prosessori

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

In-order issue, in-order complete

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

Decode
I1 I2
I3 I4
I3 I4
I4
I5 I6
I6

Execute
I1 I2
I1
I3 I4
I5 I6

Write
I1 I2
I3 I4
I5 I6

Cycle
1
2
3
4
5
6
7
8

(Sta06 Fig 14.4a)

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

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

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

Decode
I1 I2
I3 I4
I5 I6

Window
I1 I2
I3 I4
I5 I6

Execute
I1 I2
I1 I3
I6 I4
I5

Write
I1 I3
I4 I6
I5

Cycle
1
2
3
4
5
6

apuposkuri, ei lisävalhe

(Sta06 Fig 14.4c)

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

In-order Issue, out-of-order complete

- 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: resurssi riippuvuus
 I5 (käyttää) ja I4 (tuottaa): datariippuvuus
 I5 ja I6: resurssi riippuvuus

Decode
I1 I2
I3 I4
I3 I4
I4
I5 I6
I6

Execute
I1 I2
I1
I3 I4
I5 I6

Write
I2
I1 I3
I4
I5
I6

Cycle
1
2
3
4
5
6
7

Kirjoitusriippuvuus
 I1: R3 ~~B~~ R3 op R5
 I2: R4 ~~B~~ R3 + 1
 I3: R3 ~~B~~ R5 + 1
 I4: R7 ~~B~~ R3 op R4

(Sta06 Fig 14.4b)

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

Rekistereiden uudelleennimeäminen

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

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

- Ratkaisu: Rekistereiden uudelleennimeäminen
 - Laitteistossa enemmän rekistereitä kuin ohjelmoijalle näkyy
 - Laitteisto alokoi todelliset rekisterit suoritusajana
 - Allokointi s.e. vältetään nimiriippuvuus
- Tarvitaan
 - Enemmän sisäisiä työrekistereitä (rekisterijoukot), esim. Pentium II:ssa 40 työrekisteriä
 - Laitteistoa, joka alokoi 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 arvon R3:sta

Uudelleennimeä R3 s.e. käytössä työreisterit 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

Tietokoneen rakenne

Pentium 4

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

Lisälaitteiston vaikutus

base: out-of-order issue (Sta06 Fig 14.5)

+ld/st: base ja lisäksi kaksi load/store yksikköä datavälimuistille

+alu: base ja lisäksi kaksi ALUa

Speedup Without renaming

Speedup With renaming

Window size (construction): 8, 16, 32

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 20

Pentium 4

AGU = address generation unit
BTB = branch target buffer
D-TLB = data translation lookaside buffer
I-TLB = instruction translation lookaside buffer

(Sta06 Fig 14.7)

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 23

Yhteenveto

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

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

Liukuhinna

- Ulospäin CISC-käskykanta (IA-32)
- Suoritus kuitenkin mikro-operaatioina kuten RISC
 - Nouda CISC-käsky ja muodosta siitä mikro-operaatiot (μops)
 - L1 tason välimuistiin (trace cache)
 - Hinnan loppuosa operoi vakiopituusilla μ-operaatioilla (118b)
- Pitkä liukuhinna
 - 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 Next IP	TC Fetch	Driv	Alloc	Renam	Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Driv			

TC Next IP = trace cache next instruction pointer Renam = 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 välimuisttiin (trace cache)

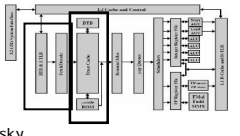
- Käskyille oma TLB, hyppyjen kohteille oma BTB
- Staatittinen hyppyjen ennustus
 - § taaksepäin "taken",
 - § eteenpäin "not taken"
- 1-4 μops (=118 bitin RI SC) per käsky, näitä monimutkaisemmat ROM-muistissa

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

- Dynaaminen hyppyjen ennustus (4-bit)
- 512 alkion joukkoassosiativinen 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

Integer ja FP yksiköt

Sta06 Fig 14.9i-j

i) Nouda data rekistereistä tai välimuistista

j) Suorita käsky, aseta lipukkeet

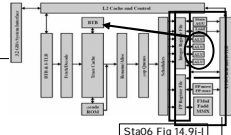
- Hae operandit rekistereistä / L1 välimuistista
- Useita liukuhinnoitettuja suoritusyksiköitä
 - § 2 * ALU, 2 * FPU, 2 * load/store
 - § Esim. nopea ALU helpoille, kertolaskuille oma ALU
- Tulosten talletus: in-order complete
- Päivitä ROB, salli uusien operaatioiden tulo hinnalle

k) Tarkista miten hyppykäskyssä kävi

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

l) Kirjaa hypyn tulos ennustuslogiikka varten

- Anna aikaa signaalien etenemiseksi



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

Resurssien allokointi

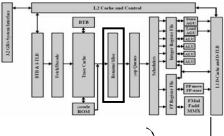
Sta06 Fig 14.9e

e) Allokoi resurssit, rekistereiden uudelleennimeäminen

- 3 operaatiota per sykli
- Varaa alkio uudelleenjärjestelypuskurista (126:sta) (reorder buffer, ROB)
- Varaa tulokselle yksi 128 sisäisestä työrekeristä ja mahd. yksi load ja yksi store puskuri (48:sta ja 24:stä)
- Poista nimiriippuvuusia rekistereiden uudelleennimeämisellä
- Allokoi alkio mikro-operaatioiden valintajonosta
- Jos ei vapaita resursseja, odota (z out-of-order)

n) ROB-alkiossa kirjanpito operaation etenemisestä hinnalla

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



Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 26

Pentium 4 Hyperthreading

Sta06 Fig 14.9f

n Yksi fyysinen IA-32 CPU, mutta 2 loogista CPU:ta

n Näkyy KJ:lle kahden CPU:n SMP-järjestelmänä


- Molemmat suorittavat eri prosesseja, tai saman prosessin eri säikeitä (kuten SMP = Symmetric multiprocessing)
- Ei tarvitse huomioida kooditasolla
- KJ:n osattava SMP-temput (mm. vuorottaminen)

n Perustuu CPU:n odotussykliin hyötykäyttöön

- Muistiinviittaus (cache miss)
- Riippuvuudet, väärä hyppynnustus

n Jos toinen käyttää FP-yksikköä, toinen voi käyttää INT-yksikköä

- Hyödyt pitkälti sovellusriippuvia



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

Window of Execution

Sta06 Fig 14.9f-h

f) 2 FIFO Jonoa mikro-operaatioiden valitsemiseksi

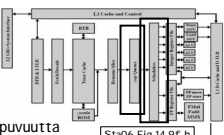
- Tarvittavat resurssit saatavilla, ei riippuvuutta
- Muistiinviittaaville oma, muille oma

g) Mikro-operaatioiden nouto jonosta (scheduling)

- Ne joiden operandit ovat jo kunnossa

h) Päästäminen liukuhinnalle (dispatching)

- Tutki FIFO-jonojen keulimmaisten ROB-alkioita
- Jos tarvittava suoritusyksikkö vapaa, operaation voi päästää suoritusliukuhinnalle
- Kaksi jonoa z 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



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

Pentium 4 Hyperthreading

Sta06 Fig 14.7

n Kahdennettu

- IP, EFLAGS ja muut kontrollirekisterit
- KäskyTLB
- Rekistereiden uudelleennimeämislogiikka


n Puollitettu

- Kristallinen tasajako, ei monopolia
- Uudelleenjärjestelypuskurit (ROB)
- Mikro-operaatioiden valintajonot (2 keulaa?)
- Load/store puskurit

n Yhteiskäytössä

- Rekisterijoukot (128 GPRs, 128 FPRs)
- Välimuistit: trace cache, L1, L2, L3
- Mikro-operaatioiden suorituksessa tarvittavat rekisterit
- Suoritusyksiköt: 2 ALUa, 2 FPUta, 2 ld/st-units

Nehalem: yhdellä lastulla 8 ydintä, 1-16 säätettä (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

Dispatch unit

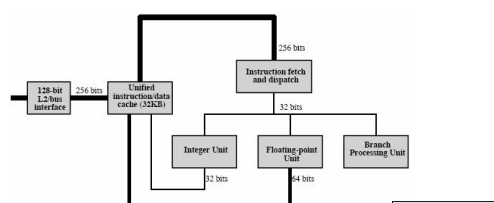
Sta06 Fig 14.11

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

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

PowerPC 601

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



(Sta06 Fig 14.10)

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

Suoritus

Sta06 Fig 14.11

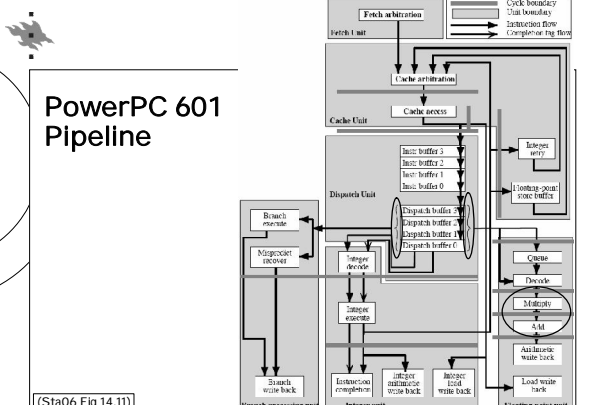
- Muutokset rekistereihin / tulosten vasta suorituksen lopuksi "Write Back" vaiheessa
- ALU-osaatit tallettavat tietoa CR-rekisteriin (condition register)
 - 8 kenttää a' 4 b, tallenna useita edellisiä vertailutuloksia
- Liukuluvut tarvitsevat useampia syklejä

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

(Sta06 Fig 14.12)

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

PowerPC 601 Pipeline



(Sta06 Fig 14.11)

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 33

Hyppyyden käsittely

- Zero cycle branches
 - Hyppy ei vaikuta muiden yksiköiden toimintaan
 - Yleensä ei tarvetta tyhjentää, tai hylätä tuloksia
 - Hyppyn kohdeosoite selvillä heti, kun hyppykäsky tulee käskyikkunaan (ennen suoritusta!)
- Yhden hyppyn spekulointi: Hyppääkö vai ei?
 - Jos ehdoton \checkmark taken
 - Jos ehdollinen ja CR-rekisteri asetettu aiemmin
 - tutki \checkmark taken / not taken
 - muuten jos taaksepäin \checkmark taken
 - jos eteenpäin \checkmark not taken
- Jos spekulointi meni pieleen
 - 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=r8,r12  #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

64bnärrkitechtuuri

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, Pages: 285 - 291

- 6 suoritusyksikköä
 - Instruction-yksikkö (dispatcher)
 - 3 integer-yksikköä
 - Load/Store yksikkö
 - FP-yksikkö
- Max 4 käskyä suoritukseen yhtäaikaan
- Reservation stations
 - Kullakin yksiköllä kaksi tai useampia
 - Jos käsky ei voi edetä (riippuvuudet) se odottaa tässä, eikä estä jäljempänä olevien etenemistä
- Uudelleennimeäminen: 8 Integer ja 12 FP Ilsärekisteriä
 - Vähentää riippuvuuksia
 - Välikaistulosten tallettamiseksi
- In-order-complete
 - max 4 käskyä yhtäaikaan

[HePa96 Fig. 4.49]

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 40

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	W								
bc ELSE,cr0/gt=false	F	-	-	-	-	-	-	W								
IF: add r12=r8,r12	F	-	-	-	-	-	-	D	W							
add r12=r12,r9	F	-	-	-	-	-	-	-	D	E	W					
add r12=r12,r10	F	-	-	-	-	-	-	-	-	D	E	W				
add r4=r12,r11	F	-	-	-	-	-	-	-	-	-	D	E	W			
stw a(r1)=r4	F	-	-	-	-	-	-	-	-	-	-	D	E	C		
b OUT	F	-	-	-	-	-	-	-	-	-	-	-	D	E	C	
ELSE: subf r12=r8,r12	F	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
subf r12=r12,r9	F	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
subf r12=r12,r10	F	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
subf r4=r12,r11	F	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
stw a(r1)=r4	F	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT:																

(a) Correct prediction: Branch was not taken

[Sta06 Fig 14.14a]

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

PowerPC 620

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

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

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	W								
bc ELSE,cr0/gt=false	F	-	-	-	-	-	-	W								
IF: add r12=r8,r12	F	-	-	-	-	-	-	-	D	W						
add r12=r12,r9	F	-	-	-	-	-	-	-	-	D	E	W				
add r12=r12,r10	F	-	-	-	-	-	-	-	-	-	D	E	W			
add r4=r12,r11	F	-	-	-	-	-	-	-	-	-	-	D	E	W		
stw a(r1)=r4	F	-	-	-	-	-	-	-	-	-	-	-	D	E	W	
b OUT	F	-	-	-	-	-	-	-	-	-	-	-	-	D	E	C
ELSE: subf r12=r8,r12	F	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
subf r12=r12,r9	F	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
subf r12=r12,r10	F	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
subf r4=r12,r11	F	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
stw a(r1)=r4	F	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT:																

(b) Incorrect prediction: Branch was taken

[Sta06 Fig 14.14b]

Tietokoneen rakenne / 2007 / Liisa Marttinen 30.11.2007 Luento 10 - 39

Kertauskysymyksiä

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

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