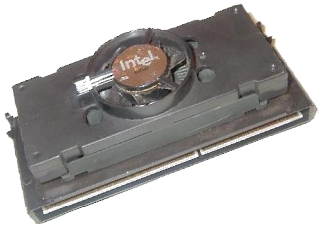
Luento 8

Tietokoneen rakenne

Suorittimen rakenne ja toiminta



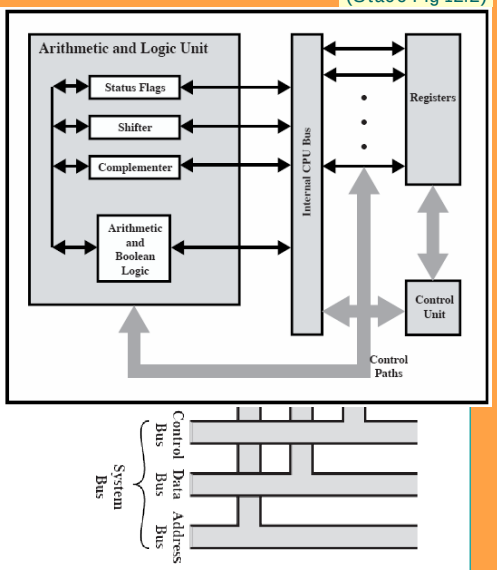
Ch 12.1-4 [Sta06]

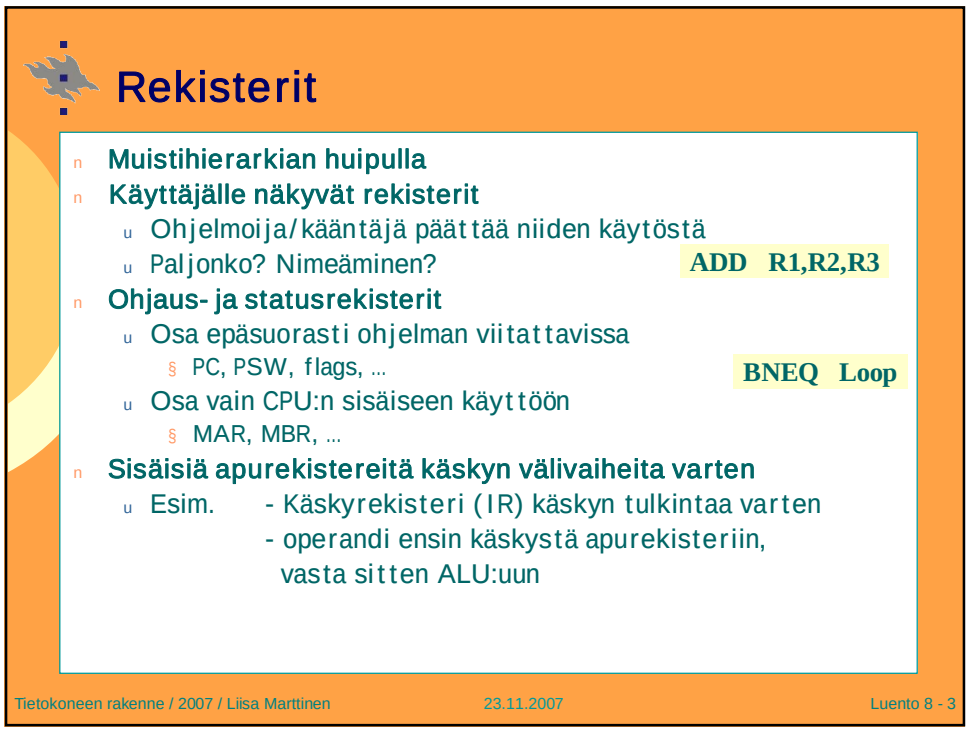
- Rekisterit
- Käskysykli
- Liukuhihna
- Riippuvuusongelmat
- Hyppyjen käsittely

CPU:n yleisrakenne

- **ALU**
  - Laskenta, vertailut
- **Rekisterit**
  - Nopeaa työtilaa
- **Prosessoriväylä**
  - Bittien siirto paikasta toiseen
- **Ohjausyksikkö (Ch 16-17)**
  - Kuka? Mitä? Milloin?
  - Kellopulssi
  - Generoi ohjaussignaalit
    - mitä seuraavalla kellopulssilla tapahtuu?
- **MMU?**
- **Välimuisti?**

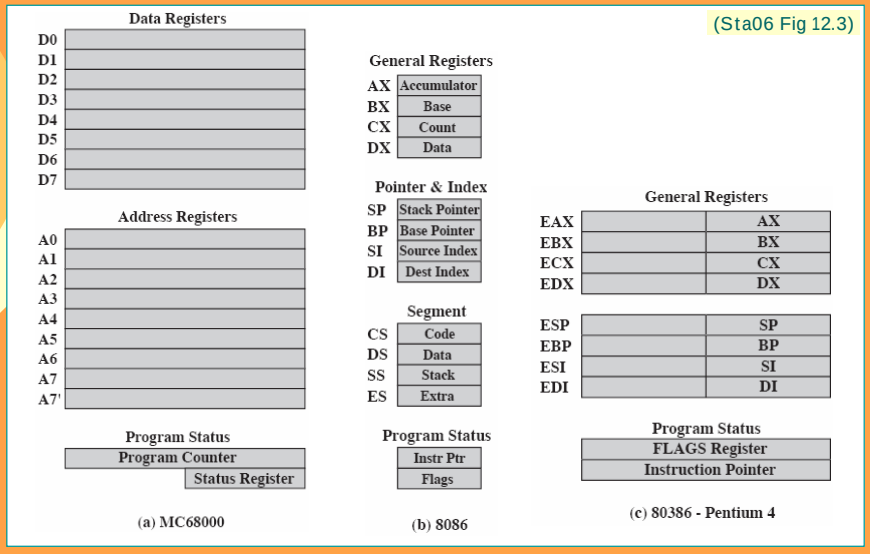
(Sta06 Fig 12.2)




Rekisterit
Muistihierarkian huipulla
Käyttäjälle näkyvät rekisterit
Ohjelmoija/kääntäjä päättää niiden käytöstä
Paljonko? Nimeäminen?
ADD R1,R2,R3
Ohjaus- ja statusrekisterit
Osa epäsuorasti ohjelman viitattavissa
PC, PSW, flags, ...
BNEQ Loop
Osa vain CPU:n sisäiseen käyttöön
MAR, MBR, ...
Sisäisiä apurekistereitä käskyn välivaiheita varten
Esim. - Käskyrekisteri (IR) käskyn tulkintaa varten
- operandi ensin käskystä apurekisteriin,
vasta sitten ALU:uun
Tietokoneen rakenne / 2007 / Liisa Marttinen
23.11.2007
Luento 8 - 3


Käyttäjälle näkyvät rekisterit
Eri prosessoriperhe ð
eri määrä erilaisia rekistereitä,
erilaiset nimeämistavat
erilaiset käyttötarkoitukset
Yleisrekisterit (general-purpose registers)
Datarekisterit (data registers)
Osoiterekisterit (address registers)
Segmenttirekisterit (segment registers)
Indeksirekisterit (index registers)
Pino-osoitin (stack pointer)
Ympäristöosoitin (frame pointer)
Tilarekisterit (condition code registers)
Tietokoneen rakenne / 2007 / Liisa Marttinen
23.11.2007
Luento 8 - 4

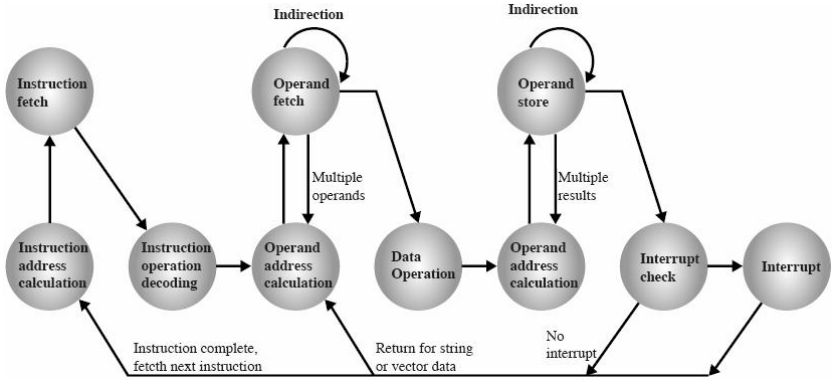
Esimerkkejä



PSW - Program Status Word

- Nimi vaihtelee eri arkkitehtuureissa
- CPU:n tila
  - Etuoikeutettu vs. käyttäjätila
- Vertailun tulos
  - Esim. Greater, Equal, Less, Zero, ...
- Sattuiko käskyn suorituksessa poikkeuksia?
  - Nollallajako, ylivuoto
  - Page fault, “memory violation”
- Tarvitseeko ulkoinen laite ohjausta?
  - Bitti keskeytyksen ilmaisemiseksi
- Keskeytyksen esto / salliminen
  - Kullekin luokalle omat bittinsä

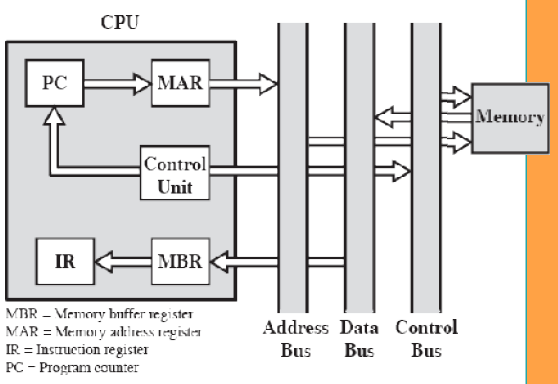
Käskysykli (Instruction cycle)



(Sta06 Fig 12.5)

Käskyn nouto (10)

- **MAR • PC**
- **MAR • MMU(MAR)**
- **Control Bus • Reserve**
- **Control Bus • Read**
- **PC • ALU(PC+1)**
- **MBR • MEM[MAR]**
- **Control Bus • Release**
- **IR • MBR**



Välimuisti! Ennaltanouto!

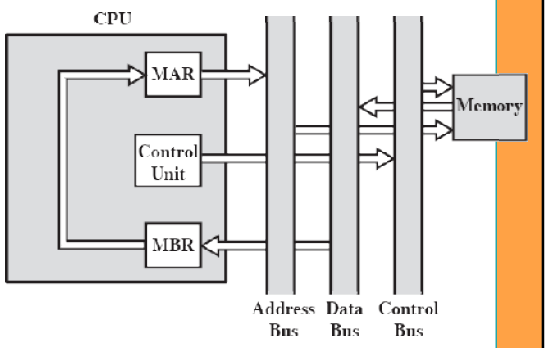
(Sta06 Fig 12.6)

Operandin nouto, Epäsuora osoitus (13)

MAR • Address
MAR • MMU(MAR)
Control Bus • Reserve
Control Bus • Read
MBR • MEM[MAR]

MAR • MBR
MAR • MMU(MAR)
Control Bus • Read
MBR • MEM[MAR]
Control Bus • Release
ALU? Regs? • MBR

Välimuisti!

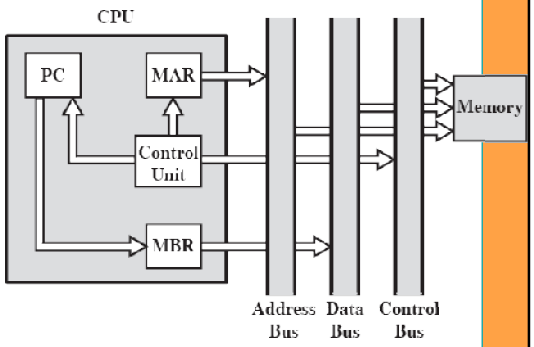


(Sta06 Fig 12.7)

Tietokoneen rakenne / 2007 / Liisa Marttinen 23.11.2007 Luento 8 - 9

Keskeytyskäsittelyn alkutoimet (esim.) (16)

MAR • SP
MAR • MMU(MAR)
Control Bus • Reserve
MBR • PC
Control Bus • Write
MAR • SP • ALU(SP+1)
MAR • MMU(MAR)
MBR • PSW
Control Bus • Write
SP • ALU(SP+1)
PSW • privileged & disable
MAR • Interrupt number
Control Bus • Read
PC • MBR • MEM[MAR]
Control Bus • Release

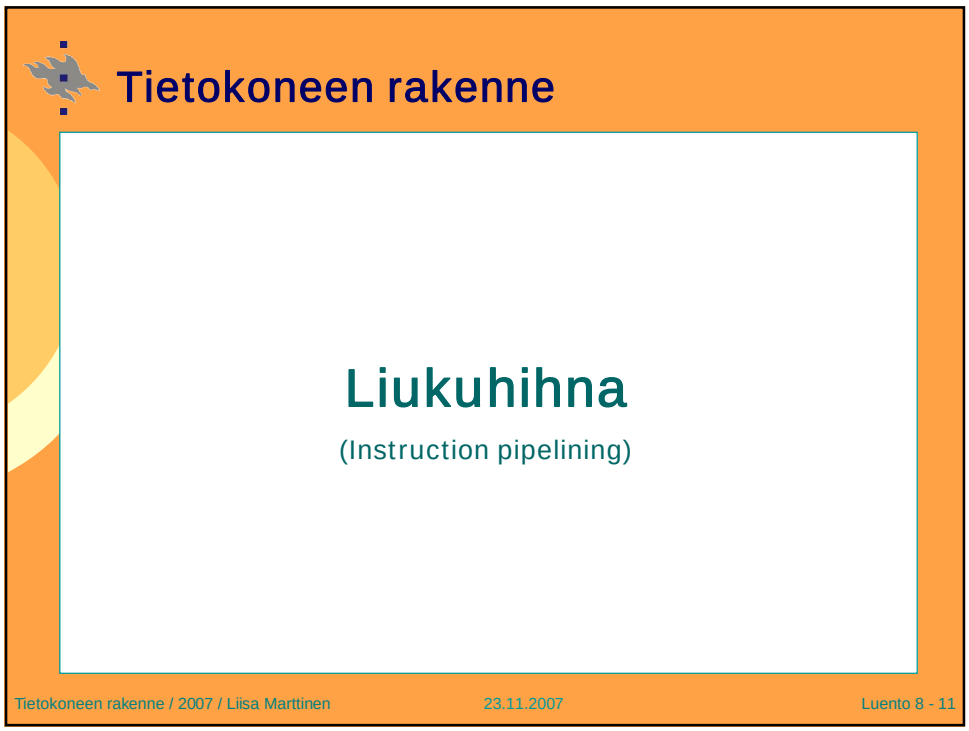


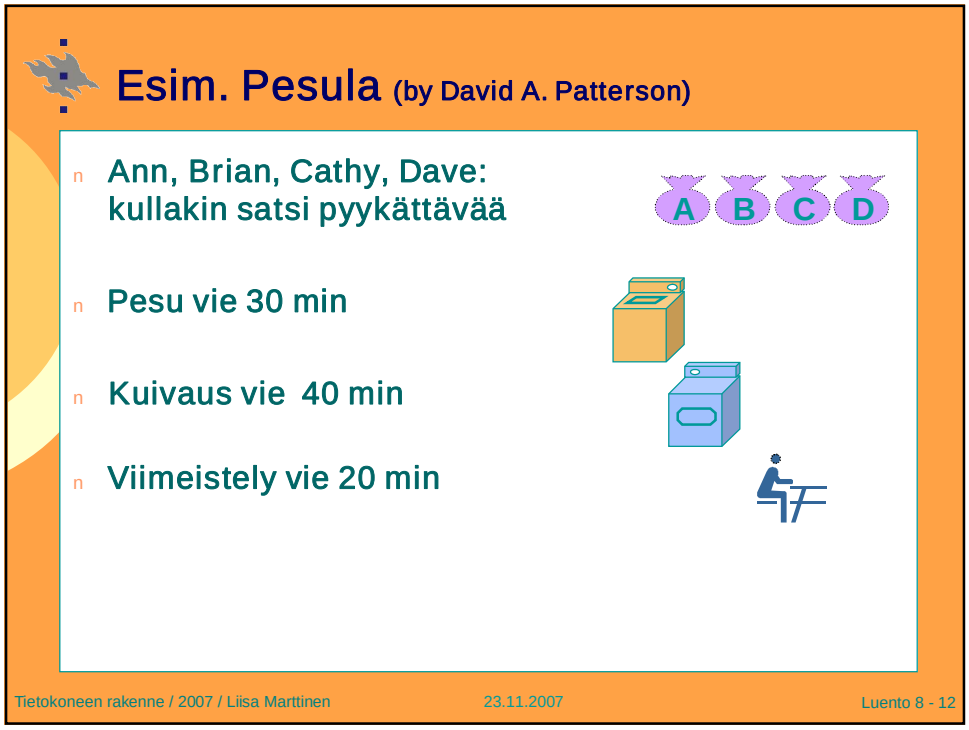
Ei osoitteenmuunnosta!

SP = Stack Pointer

(Sta06 Fig 12.8)

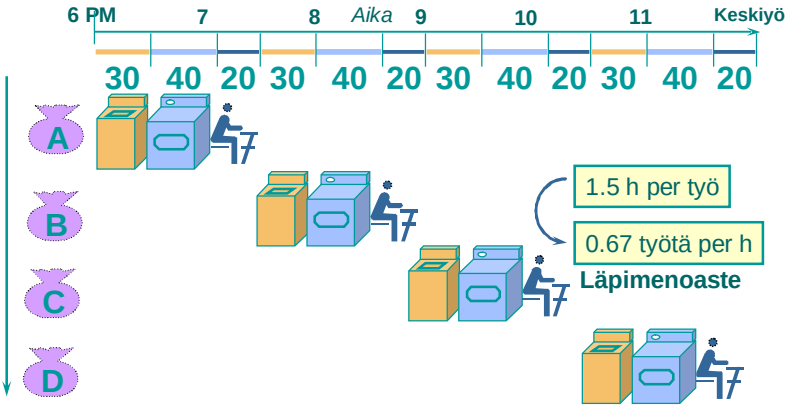
Tietokoneen rakenne / 2007 / Liisa Marttinen 23.11.2007 Luento 8 - 10


Tietokoneen rakenne
Liukuhihna
(Instruction pipelining)
Tietokoneen rakenne / 2007 / Liisa Marttinen
23.11.2007
Luento 8 - 11


Esim. Pesula (by David A. Patterson)
Ann, Brian, Cathy, Dave:
kullakin satsi pyykättävää
A
B
C
D
Pesu vie 30 min
Kuivaus vie 40 min
Viimeistely vie 20 min
Tietokoneen rakenne / 2007 / Liisa Marttinen
23.11.2007
Luento 8 - 12

Peräkkäin: Pyykkitupa varattu vain yhdelle (8)

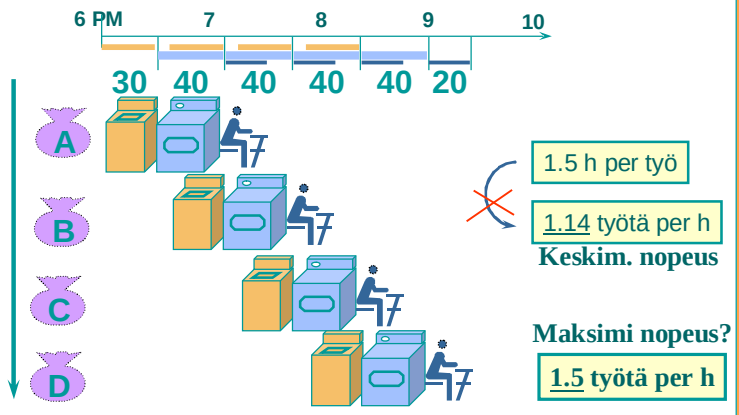
- Vie 6 tuntia, syntyy neljä satsia puhdasta
- Entäpä, jos toimisivat liukuhihnaperiatteella?



Tietokoneen rakenne / 2007 / Liisa Marttinen 23.11.2007 Luento 8 - 13

Liukuhihna: Varaus työvaiheittain (13)

- Vie 3,5 tuntia, syntyy 4 satsia puhdasta



Töitä valmistuu parhaimmillaan 40 min välein! (0.67 h / valmistunut työ)

Tietokoneen rakenne / 2007 / Liisa Marttinen 23.11.2007 Luento 8 - 14

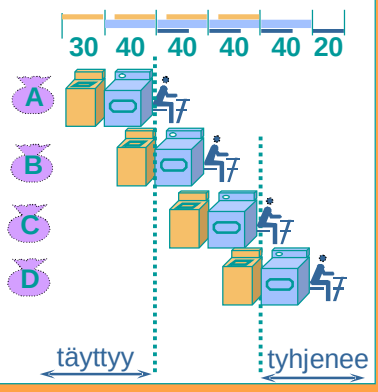
Huomioita

- **Liukuhihna ei nopeuta yhden työn aktiivisena oloaikaa järjestelmässä**
  - Mutta keskimääräinen läpimenoaika paranee
- **Liukuhihna voi aiheuttaa yhdelle työlle lisäviipeitä verrattuna siihen, jos se työ olisi yksin järjestelmässä**
  - Jos hihnan seuraava vaihe varattuna, ei voi edetä
- **Useita töitä etenee yhtä aikaa, mutta ne ovat eri vaiheissa**
- **Hitain vaihe määrää koko liukuhihnan nopeuden**
  - Liukuhihna nytkähtää, kun kaikki osat valmiita
  - Tehokkuus kärsii, jos kestot eripituisia
- **Yksi työ nopeutuu maksimissaan saman verran kuin liukuhihnassa vaiheita**

Tietokoneen rakenne / 2007 / Liisa Marttinen 23.11.2007 Luento 8 - 15

Huomioita

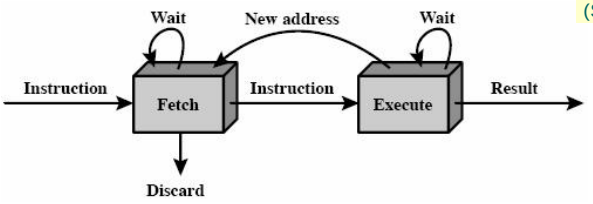
- **Toteutus monimutkaisempaa, lisäresurssien tarve**
  - Tuplapistorasia? ž pesukone ja kuivaaja voivat pörrätä yhtäaikaa
  - Kaksi tai kolme ihmistä yhtäaikaa pyykkituvassa
- **Hihna ei toimi alussa eikä lopussa täydellä teholla**
- **Hihna “yskii”**
  - Töitä ei tule tasaisella tahdilla



Tietokoneen rakenne / 2007 / Liisa Marttinen 23.11.2007 Luento 8 - 16

CPU ja 2-vaiheinen liukuhihna?

(Sta06 Fig 12.9)



- **Seuravan käskyn ennaltanouto samaan aikaan kuin edellistä suoritetaan**
  - Paikallisuus: oleta, että suoritus “esiintymisjärjestyksessä”
- **Ongelmat**
  - Suoritusvaihe voi kestää kauemmin ž noutovaihe jouten
  - Suoritusvaihe muuttaa PC:n arvoa ž haettu väärä käsky
    - Seuraavan käskyn osoite “ennustettiin” väärin!
- **Rinnakkaisuus vielä vähäistä ž Lisää vaiheita?**

6-vaiheinen liukuhihna

Time

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instruction 1	FI	DI	CO	FO	EI	WO								
Instruction 2		FI	DI	CO	FO	EI	WO							
Instruction 3			FI	DI	CO	FO	EI	WO						
Instruction 4				FI	DI	CO	FO	EI	WO					
Instruction 5					FI	DI	CO	FO	EI	WO				
Instruction 6						FI	DI	CO	FO	EI	WO			
Instruction 7							FI	DI	CO	FO	EI	WO		
Instruction 8								FI	DI	CO	FO	EI	WO	
Instruction 9									FI	DI	CO	FO	EI	WO

FE - Fetch instruction	FO - Fetch operands
DI - Decode instruction	EI - Execute instruction
CO - Calculate operand addresses	WO - Write operand

(Sta06 Fig 12.10)

Paljonko liukuhihna nopeuttaa?

- 6 vaiheinen liukuhihna ž 14 aikayksikköä
- Ei liukuhihnaa ž 9*6 = 54 aikayksikköä
- Nopeutus = 54/14 = 3.86 < 6 !
- Todellisuudessa nopeutus vieläkin pienempi

- Jokainen käsky ei tarvitse kaikkia vaiheita
  - Ei-liukuhihnoitettu voi edetä heti seuraavaan vaiheeseen
  - Liukuhihnoitettu joutuu odottamaan seuraavan vaiheen vapautumista, ko. CPU:n yksikkö jouten
    - Suorituksessa on “kupla” (bubble)

Yhden vaiheen kesto liukuhinalla?

$$\tau = \max_{i=1..k}[\tau_i] + d = \tau_m + d \quad >> d$$

- Hitain vaihe määrää etenemisvauhdin
  - Kauimmin kestävä vaihe on pullonkaula
- Kukin vaihe asetetaan kestämään yhtä kauan
  - Montako kellopulssia?

Paljonko nopeuttaa?

n käskyä, k vaihetta, τ =vaiheen kesto

Ei liukuhihnaa: $T_1 = nk\tau$

Pessimistinen: olettaa, että kaikki vaiheet kestäisivät yhtä kauan

Liukuhihna: $T_k = [k + (n-1)]\tau$

Ks. Sta06 Fig 12.10 ja tarkista itse!

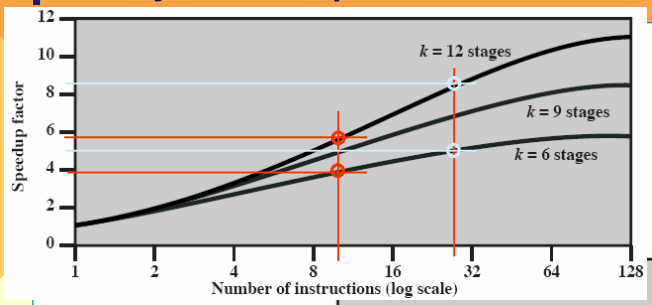
k vaihetta ennenkuin ensimmäinen työ valmistuu

seuraavat (n-1) valmistuvat yhden vaiheen välein

Nopeutus: $S_k = \frac{T_1}{T_k} = \frac{nk\tau}{[k + (n-1)]\tau} = \frac{nk}{[k + (n-1)]}$

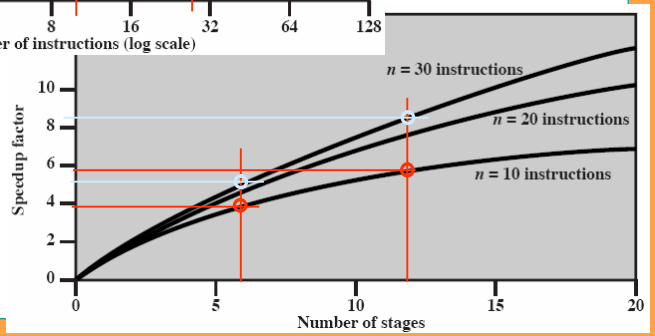
Paljonko nopeuttaa?

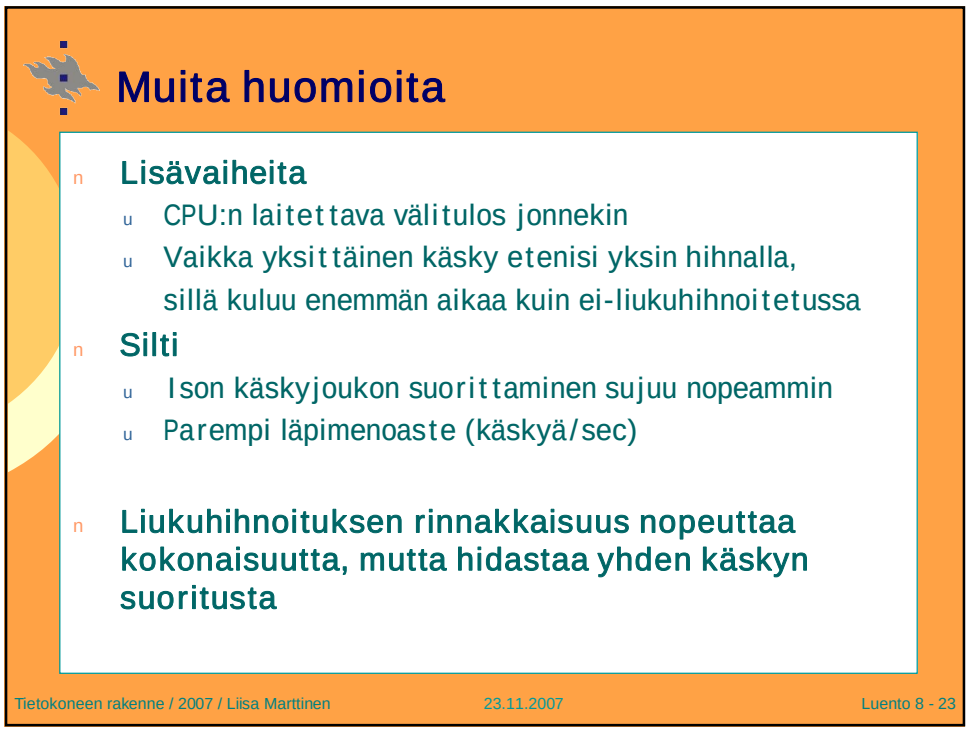
Oletuksella: ei hyppykäskyjä



(Sta06 Fig 12.14)

more gains from multiple stages when more instructions without jumps




Muita huomioita
Lisävaiheita
CPU:n laitettava välitulos jonnekin
Vaikka yksittäinen käsky etenisi yksin hihnalla, sillä kuluu enemmän aikaa kuin ei-liukuhihnoitetussa
Silti
Ison käskyjoukon suorittaminen sujuu nopeammin
Parempi läpimenoaste (käskyä/sec)
Liukuhihnoituksen rinnakkaisuus nopeuttaa kokonaisuutta, mutta hidastaa yhden käskyn suoritusta
Tietokoneen rakenne / 2007 / Liisa Marttinen
23.11.2007
Luento 8 - 23


Ongelmakohtia
Rakenteellinen riippuvuus (structural dependency)
Usea vaihe voi tarvita samaa HW-yksikköä
Muisti: FI, FO, WO
ALU: CO, EI
Kontrolliriippuvuus (control dependency)
Ehdollisen hyppykäskyn tulos selvillä vasta EI-vaiheen jälkeen
Ennaltanoudettu vääriä käskyjä
Datariippuvuus (data dependency)
Tarvitaan edeltävän käskyn tulosta, eikä se ole vielä valmistunut
STORE R1,VarX
ADD R2,R3,VarY
MUL R3,R4,R5
ADD R1,R7, R9
Jump There
ADD R2,R3,R4
MUL R1,R4,R5
MUL R1,R2,R3
LOAD R6, Arr(R1)
Tietokoneen rakenne / 2007 / Liisa Marttinen
23.11.2007
Luento 8 - 24

Ratkaisuja?

- **Huomattava laitteistolla, odotettava, että riippuvuus poistettu**
  - Hihnalle ylimääräisiä odotusjaksoja eli kuplia, “bubbles”
  - Kupla viivästää kaikkia sen takana tulevia vaiheita
  - Yleisesti käytetty tapa
- **Rakenteellinen riippuvuus**
  - Lisää laitteistoa, esim. CO- ja EI-vaiheelle omat ALUt
  - Paljon rekistereitä, vähemmän muistioperandeja
- **Kontrolliriippuvuus**
  - Tyhjennä hihna, ja hae uudet käskyt
  - Hyppyjen ennustuslogiikka: haenko ennalta tästä vai tuolta?
- **Datariippuvuus**
  - Vaihda käskyjen suoritusjärjestystä
  - Oikopolut laitteistossa (by-pass): tulokseen pääsee käsiksi ennen WO-vaihetta

Tietokoneen rakenne / 2007 / Liisa Marttinen 23.11.2007 Luento 8 - 25

Datariippuvuudet ja niiden poisto

```
MUL  R1, R2, R3
ADD  R4, R5, R6
SUB  R7, R1, R8
ADD  R1, R1, R3
```

	1	2	3	4	5	6	7	8	9	10	11
MUL	FI	DI	CO	FO	EI	WO					
ADD		FI	DI	CO	FO	EI	WO				
SUB			FI	DI	CO		FO	EI	WO		
ADD				FI	DI	CO		FO	EI	WO	

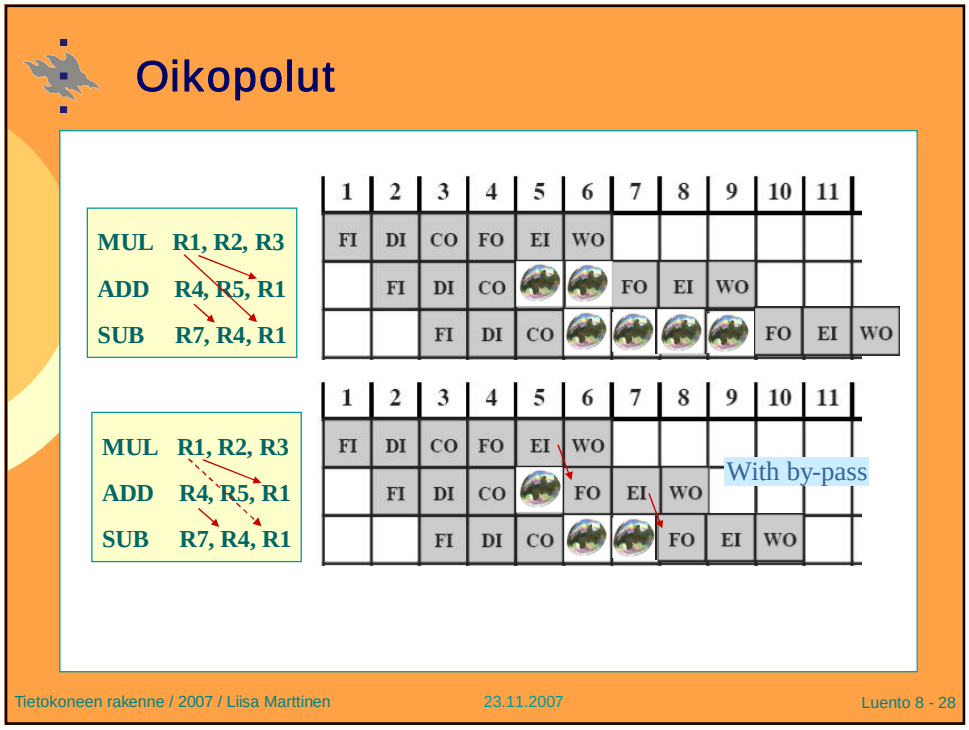
```
MUL  R1, R2, R3
ADD  R4, R5, R6
SUB  R7, R7, R8
ADD  R1, R1, R3
```

	1	2	3	4	5	6	7	8	9	10	11
MUL	FI	DI	CO	FO	EI	WO					
ADD		FI	DI	CO	FO	EI	WO				
SUB			FI	DI	CO	FO	EI	WO			
ADD				FI	DI	CO	FO	EI	WO		

too far, no effect

Tietokoneen rakenne / 2007 / Liisa Marttinen 23.11.2007 Luento 8 - 26


Käskyjen suoritusjärjestyksen muutos
MUL R1, R2, R3
ADD R4, R5, R6
SUB R7, R1, R8
ADD R9, R0, R8
1 2 3 4 5 6 7 8 9 10 11
FI DI CO FO EI WO
MUL R1, R2, R3
ADD R4, R5, R6
ADD R9, R0, R8
SUB R7, R1, R8
switched instructions
Tietokoneen rakenne / 2007 / Liisa Marttinen
23.11.2007
Luento 8 - 27


Oikopolut
MUL R1, R2, R3
ADD R4, R5, R1
SUB R7, R4, R1
1 2 3 4 5 6 7 8 9 10 11
FI DI CO FO EI WO
With by-pass
Tietokoneen rakenne / 2007 / Liisa Marttinen
23.11.2007
Luento 8 - 28

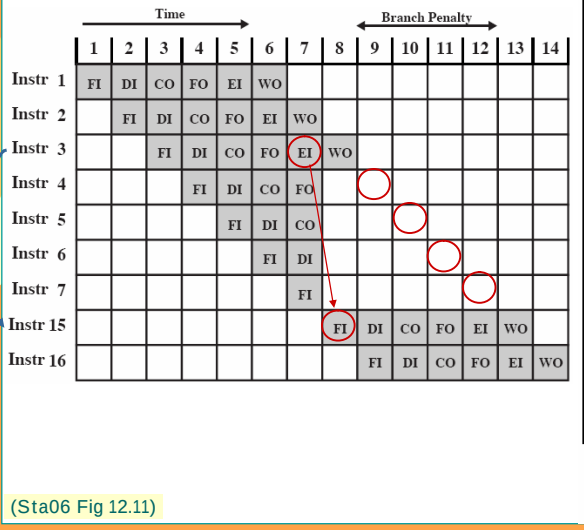
Tietokoneen rakenne

Hypyt ja liukuhihna

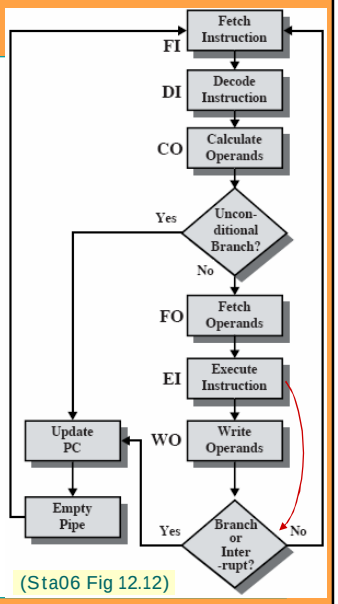
Jumps and pipelining

- Monta suorituspolkua (Multiple streams)
- Viivästetty hyppy (Delayed branch)
- Kohteen ennaltanouto (Prefetch branch target)
- Silmukkapuskuri (Loop buffer)
- Ennustuslogiikka (Branch prediction)

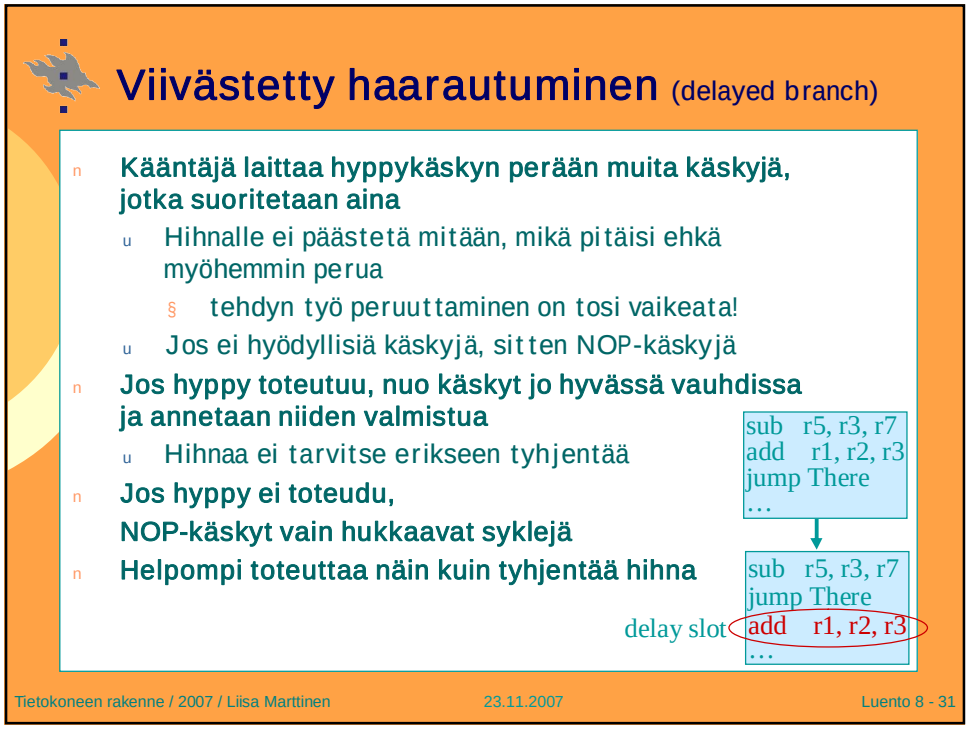
Hypyn vaikutus liukuhihnaan

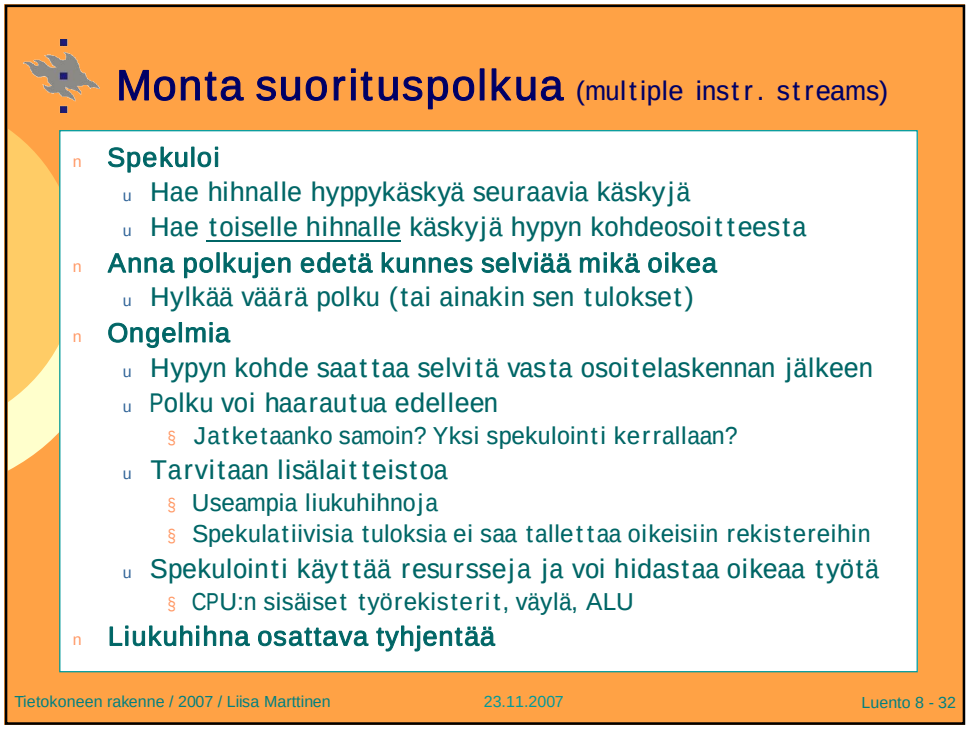


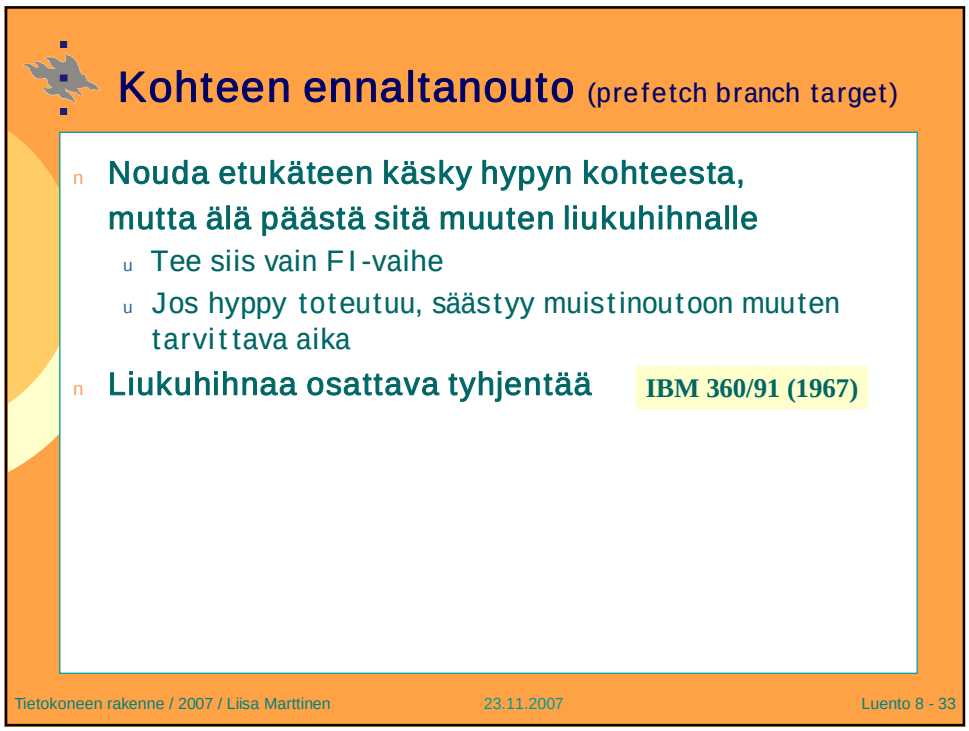
(Sta06 Fig 12.11)

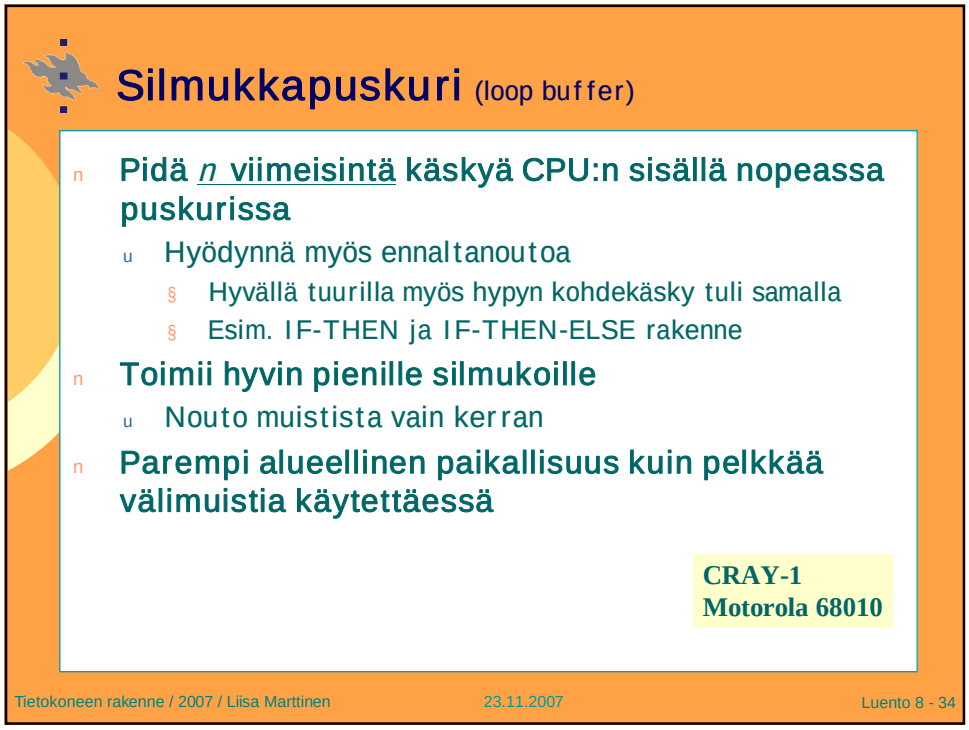


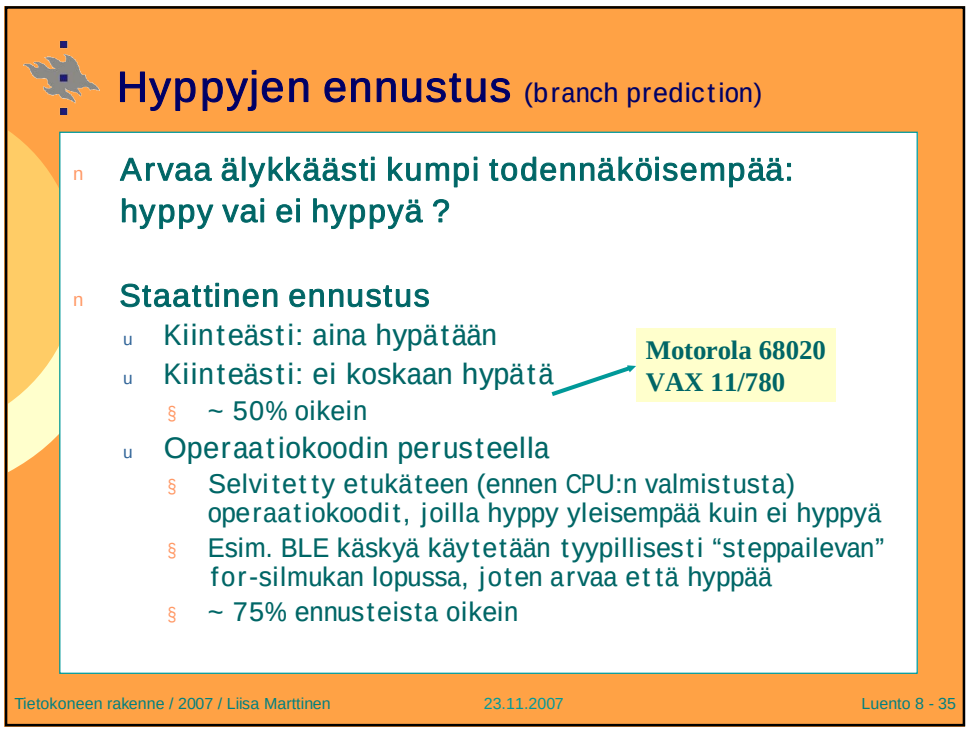
(Sta06 Fig 12.12)

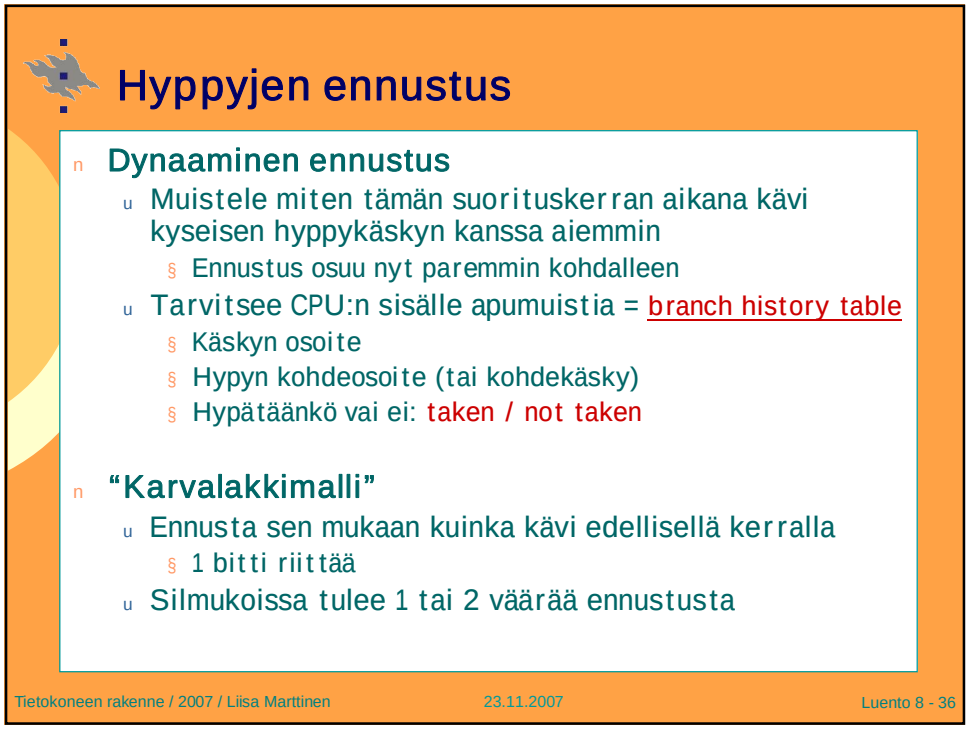

Viivästetty haarautuminen (delayed branch)
Kääntäjä laittaa hyppykäskyn perään muita käskyjä, jotka suoritetaan aina
Hihnalle ei päästetä mitään, mikä pitäisi ehkä myöhemmin perua
tehdyn työ peruuttaminen on tosi vaikeata!
Jos ei hyödyllisiä käskyjä, sitten NOP-käskyjä
Jos hyppy toteutuu, nuo käskyt jo hyvässä vauhdissa ja annetaan niiden valmistua
Hihnaa ei tarvitse erikseen tyhjentää
Jos hyppy ei toteudu,
NOP-käskyt vain hukkaavat syklejä
Helpompi toteuttaa näin kuin tyhjentää hihna
sub r5, r3, r7
add r1, r2, r3
jump There
…
sub r5, r3, r7
jump There
delay slot
add r1, r2, r3
…
Tietokoneen rakenne / 2007 / Liisa Marttinen
23.11.2007
Luento 8 - 31


Monta suorituspolkua (multiple instr. streams)
Spekuloi
Hae hihnalle hyppykäskyä seuraavia käskyjä
Hae toiselle hihnalle käskyjä hypyn kohdeosoitteesta
Anna polkujen edetä kunnes selviää mikä oikea
Hylkää väärä polku (tai ainakin sen tulokset)
Ongelmia
Hypyn kohde saattaa selvitä vasta osoitelaskennan jälkeen
Polku voi haarautua edelleen
Jatketaanko samoin? Yksi spekulointi kerrallaan?
Tarvitaan lisälaitteistoa
Useampia liukuhihnoja
Spekulatiivisia tuloksia ei saa tallettaa oikeisiin rekistereihin
Spekulointi käyttää resursseja ja voi hidastaa oikeaa työtä
CPU:n sisäiset työrekisterit, väylä, ALU
Liukuhihna osattava tyhjentää
Tietokoneen rakenne / 2007 / Liisa Marttinen
23.11.2007
Luento 8 - 32


Kohteen ennaltanouto (prefetch branch target)
Nouda etukäteen käsky hypyn kohteesta, mutta älä päästä sitä muuten liukuhihnalle
Tee siis vain FI-vaihe
Jos hyppy toteutuu, säästyy muistinoutoon muuten tarvittava aika
Liukuhihnaa osattava tyhjentää
IBM 360/91 (1967)
Tietokoneen rakenne / 2007 / Liisa Marttinen
23.11.2007
Luento 8 - 33


Silmukkapuskuri (loop buffer)
Pidä n viimeisintä käskyä CPU:n sisällä nopeassa puskurissa
Hyödynnä myös ennaltanoutoa
Hyvällä tuurilla myös hypyn kohdekäsky tuli samalla
Esim. IF-THEN ja IF-THEN-ELSE rakenne
Toimii hyvin pienille silmukoille
Nouto muistista vain kerran
Parempi alueellinen paikallisuus kuin pelkkää välimuistia käytettäessä
CRAY-1
Motorola 68010
Tietokoneen rakenne / 2007 / Liisa Marttinen
23.11.2007
Luento 8 - 34

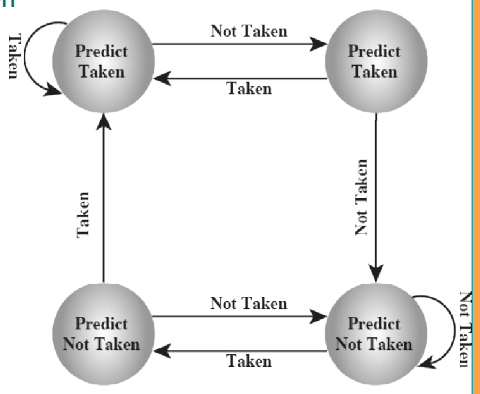

Hyppyjen ennustus (branch prediction)
Arvaa älykkäästi kumpi todennäköisempää: hyppy vai ei hyppyä ?
Staattinen ennustus
Kiinteästi: aina hypätään
Kiinteästi: ei koskaan hypätä
~ 50% oikein
Motorola 68020
VAX 11/780
Operaatiokoodin perusteella
Selvitetty etukäteen (ennen CPU:n valmistusta) operaatiokoodit, joilla hyppy yleisempää kuin ei hyppyä
Esim. BLE käskyä käytetään tyypillisesti “steppailevan” for-silmukan lopussa, joten arvaa että hyppää
~ 75% ennusteista oikein
Tietokoneen rakenne / 2007 / Liisa Marttinen
23.11.2007
Luento 8 - 35


Hyppyjen ennustus
Dynaaminen ennustus
Muistele miten tämän suorituskerran aikana kävi kyseisen hyppykäskyn kanssa aiemmin
Ennustus osuu nyt paremmin kohdalleen
Tarvitsee CPU:n sisälle apumuistia = branch history table
Käskyn osoite
Hypyn kohdeosoite (tai kohdekäsky)
Hypätäänkö vai ei: taken / not taken
“Karvalakkimalli”
Ennusta sen mukaan kuinka kävi edellisellä kerralla
1 bitti riittää
Silmukoissa tulee 1 tai 2 väärää ennustusta
Tietokoneen rakenne / 2007 / Liisa Marttinen
23.11.2007
Luento 8 - 36

Hyppyjen ennustus

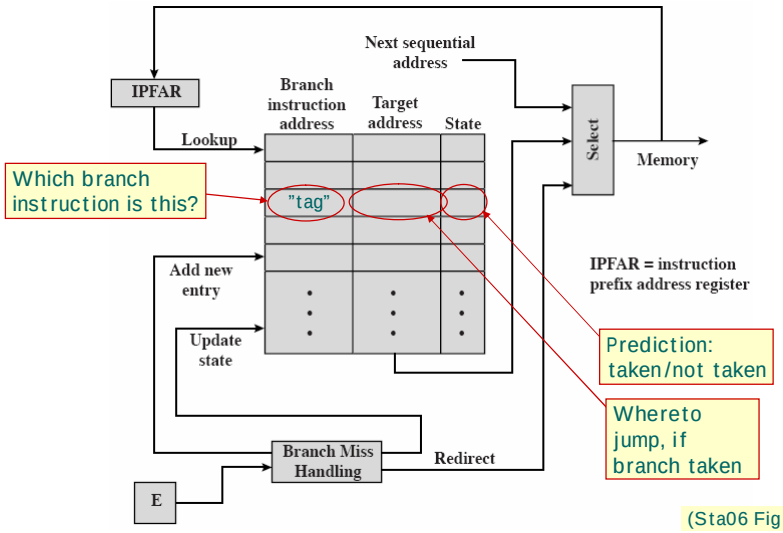
- Parannettu malli
  - Älä muuta ennustetta liian helposti
  - Ennusta kahden edellisen kerran mukaan
  - 2 bittiä riittää

PowerPC 620

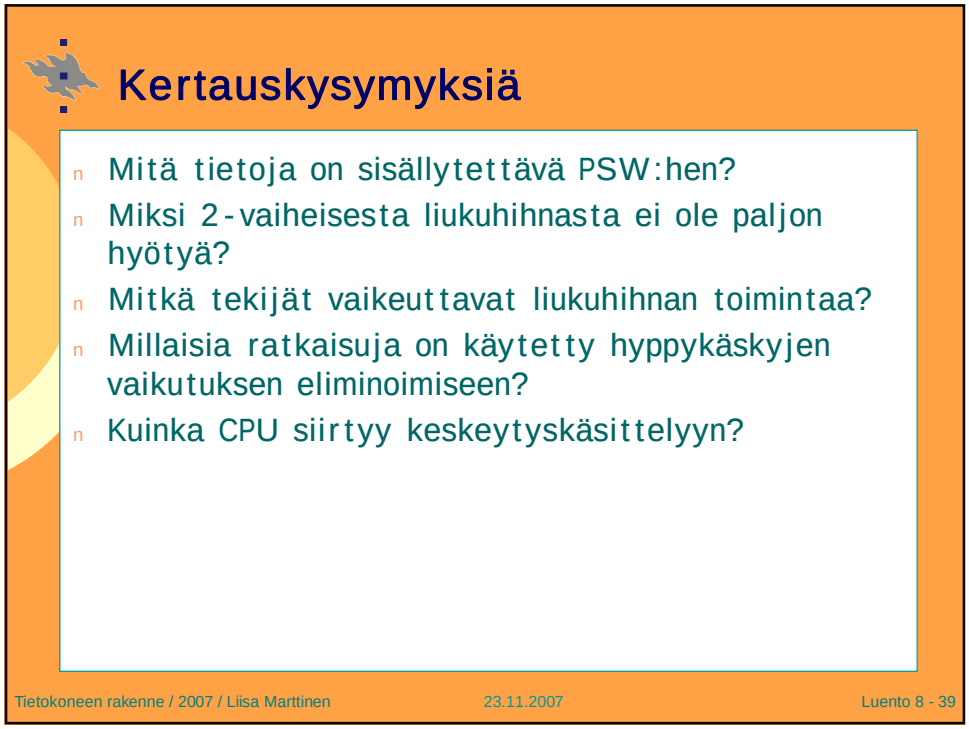


(Sta06 Fig 12.17)

Branch history table strategy
(ennustaminen hyppyhistorian avulla)



(Sta06 Fig 12.18)


Kertauskysymyksiä
Mitä tietoja on sisällytettävä PSW:hen?
Miksi 2-vaiheisesta liukuhihnasta ei ole paljon hyötyä?
Mitkä tekijät vaikeuttavat liukuhihnan toimintaa?
Millaisia ratkaisuja on käytetty hyppykäskyjen vaikutuksen eliminoimiseen?
Kuinka CPU siirtyy keskeytyskäsittelyyn?
Tietokoneen rakenne / 2007 / Liisa Marttinen
23.11.2007
Luento 8 - 39