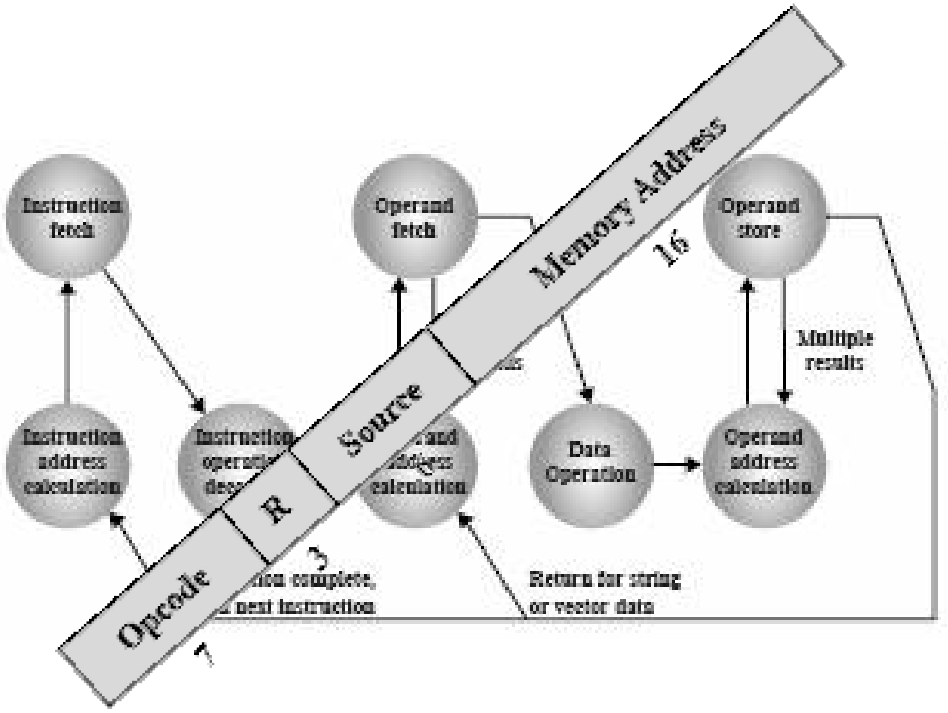
Tietokoneen rakenne

Käsky-kannat

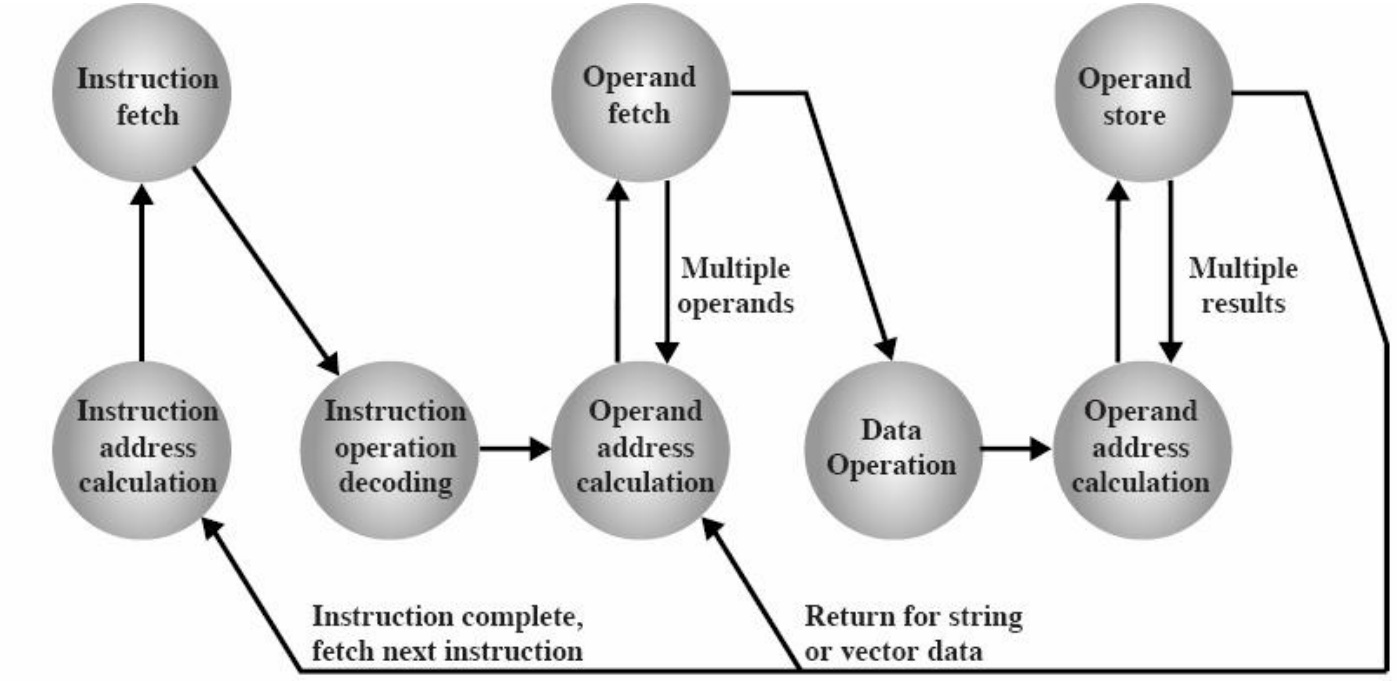


Ch 10-11 [Sta06]

- Operaatioista
- Operandeista
- Osoitustavoista
- Pentium / PowerPC

Käskysykli

- CPU suorittaa ohjelmaa konekielinen “käsky kerrallaan”
- Käskyn suoritus muodostuu vaiheista, joita CPU toistaa jokaiselle käskylle



(Sta06 Fig 10.1)

Konekäskyt

- Käskykanta =
  - CPU:n tunnistama konekielisten käskyjen kokoelma
- **Operaatiokoodi**
  - Mitä käsky tekee?
- **Viitteet operandeihin (yksi/useita)**
  - Mistä data, jolle operaatio tehdään?
    - Rekistereistä, muistista, I/O laitteelta
  - Minne tulokset talletetaan?
    - Rekistereihin, muistiin, I/O laitteelle
- **Mikä käsky seuraavaksi**
  - Implisiittisesti? Explisiittisesti?
- **I/O?**
  - Muistiinkuvattu I/O ž samat tavat kuin muistille

Pureskeltu TITO kurssilla

Nopeus?

Käskyt ja Data

käskyt

data

Address	Contents			
101	0010	0010	0000	0001
102	0001	0010	0000	0010
103	0001	0010	0000	0011
104	0011	0010	0000	0100
201	0000	0000	0000	0010
202	0000	0000	0000	0011
203	0000	0000	0000	0100
204	0000	0000	0000	0000

(a) Binary program

Address	Contents
101	2201
102	1202
103	1203
104	3204
201	0002
202	0003
203	0004
204	0000

(b) Hexadecimal program

osoite

arvo

Address	Instruction	
101	LDA	201
102	ADD	202
103	ADD	203
104	STA	204
201	DAT	2
202	DAT	3
203	DAT	4
204	DAT	0

(c) Symbolic program

symbolinen tunnus

Label	Operation	Operand
FORMUL	LDA	I
	ADD	J
	ADD	K
	STA	N
I	DATA	2
J	DATA	3
K	DATA	4
N	DATA	0

(d) Assembly program

(Sta06 Fig 10.11)

Millaisia käskyjä tarvitaan?

Sta06 Table 10.3

- **Siirto muistista rekisteriin / rekisteristä muistiin**
  - LOAD, STORE, MOVE, PUSH, POP, ...
- **I/O-laitteen ohjaus**
  - Kuten yllä (muistiinkuvattu IO)
  - Omat I/O-ohjauskäskyt (ei muistiinkuvattu I/O)
- **Aritmeettiset ja loogiset operaatiot**
  - ADD, MUL, CLR, SET, COMP, AND, SHR, NOP, ...
- **Esitystapamuunnokset**
  - TRANS, CONV, 16bTo32b, IntToFloat, ...
- **Käskyjen suoritusjärjestyksen ohjaus, ehdoton/ehdollinen**
  - JUMP, BRANCH, JEQU, CALL, EXIT, HALT, ...
- **Palvelupyyntö**
  - SVC, INT, IRET, SYSENTER, SYSEXIT, ...
- **Etuoikeutetut käskyt**
  - DIS, IEN, flush cache, invalidate TLB, ...

Miltä operaatioissa tapahtuu?

(Sta06 Table 10.4)

Data Transfer	Transfer data from one location to another
	If memory is involved: Determine memory address Perform virtual-to-actual-memory address transformation Check cache Initiate memory read/write
Arithmetic	May involve data transfer, before and/or after
	Perform function in ALU
	Set condition codes and flags
Logical	Same as arithmetic
Conversion	Similar to arithmetic and logical. May involve special logic to perform conversion
Transfer of Control	Update program counter. For subroutine call/return, manage parameter passing and linkage
I/O	Issue command to I/O module
	If memory-mapped I/O, determine memory-mapped address

Millaista dataa käsitellään?

- Kokonaislukuja, liukulukuja totuusarvoja
- Merkkejä, merkkijonoja
  - IRA (aka ASCII), EBCDIC
- Vektoreita, taulukoita
  - N kpl alkioita pötkössä
- Muistiosoitteita

- Erikokoisia operandeja
  - 8 /16/32/ 64b, ...
  - Kutakin tyyppiä/kokoa varten omat operaatiokoodit

Operation Mnemonic	Name	Number of Bits Transferred
L	Load	32
LH	Load Halfword	16
LR	Load	32
LER	Load (Short)	32
LE	Load (Short)	32
LDR	Load (Long)	64
LD	Load (Long)	64
ST	Store	32
STH	Store Halfword	16
STC	Store Character	8
STE	Store (Sort)	32
STD	Store (Long)	64

(Sta06 Table 10.5)

Käskyformaatti

- **Paljonko kullekin käskyn osalle bittejä?**
  - Montako erilaista käskykoodia tarvitaan?
  - Montako operandia voi osoittaa yhdessä käskyssä?
  - Onko operandi rekisterissä vai muistissa?
  - Montako rekisteriä osoitettavissa?
- **Vakio- vs. vaihtelevanmittaiset käskyt?**

Number of Addresses	Symbolic Representation	Interpretation
3	OP A, B, C	A ← B OP C
2	OP A, B	A ← A OP B
1	OP A	AC ← AC OP A
0	OP	T ← (T – 1) OP T

AC = accumulator
A, B, C = memory or register locations
T = top of stack
(T – 1) = second element of stack

(Sta06 Table 10.1)

Montako rekisteriä?

- **Vähintään 16-32 kpl**
  - Työdata rekistereissä
- **Rekisterijoukot omiin tarkoituksiinsa?**
  - Esim. Kokonaisluvuille omansa ja liukuluvuille omansa, indeksoinnille omansa ja datalle omansa
  - Ne voi molemmat numeroida alkaen 0:stä
  - Käskykoodi määrää kumpaa käytettävä
- **Enemmän rekistereitä kuin käsky voi viitata?**
  - CPU huolehtii sisäisesti niiden allokoinnista
    - Rekisteri-ikkuna
  - Esim. Aliohjelman parametrit aina rekistereissä
    - Ohjelmoijan mielestä rekistereissä r8-r15, CPU sijoittelee rekisterijoukkoon välille 8-132 (palataan tähän myöhemmin)

Käskyarkkitehtuureja

- **Akkukone**
  - Vain yksi rekisteri, implisiittinen osoittaminen
- **Pinokone**
  - Operandit pinossa, implisiittinen osoittaminen
  - PUSH, POP

Ks. Sta06 Appendix 10A

Esim. JVM

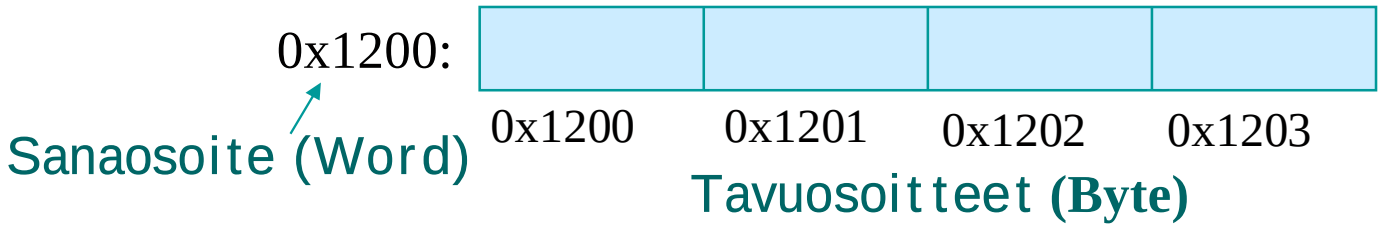
- **Yleisrekisterikone**
  - Vain yhdenkoon rekistereitä
  - Käskyssä 2 tai 3 operandia
- **Load/Store arkkitehtuuri**
  - Vain LOAD/STORE viittaavat muistiin
  - ALU-operaatioissa tav. 3 rekisteriä

```
LOAD  R3, C
LOAD  R2,B
ADD   R1,R2,R3
STORE R1,A
```

Tavujärjestys: Big vs. Little Endian

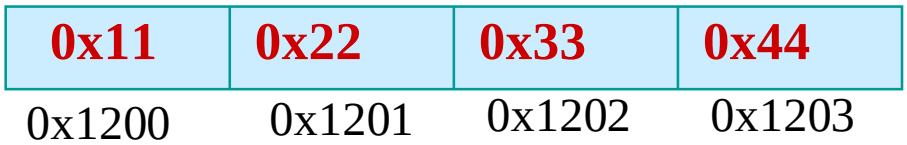
Ks. Sta06 Appendix 10B

- Kuinka usean tavun kokoinen luku talletetaan?

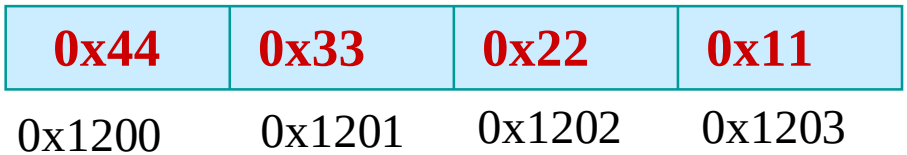


STORE 0x11223344 ,0x1200 ???

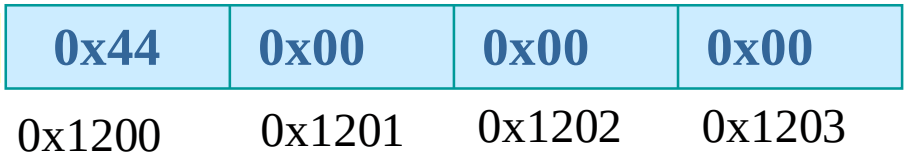
Big-Endian:
eniten merkitsevällä tavulla pienin osoite



Little-Endian:
vähiten merkitsevällä tavulla pienin osoite



0x00000044 =



Sama osoite tavuna ja sanana:

Big vs. Little Endian

- **ALU käyttää vain jompaakumpaa**
  - Little-endian: x86, Pentium, VAX
  - Big-endian: IBM 370/390, Motorola 680x0 (Mac), useimmat RISC-arkkitehtuurin koneet
  - Power-PC kelpuuttaa kummankin
    - Bitti ohjausrekisterissä (MSR, machine status register)
    - Järjestys vaihdetaan tarvittaessa ALUa ennen/jälkeen
- **Tavujärjestys huomioitava, kun tietoa siirretään koneesta toiseen**
  - Internet käyttää big-endian muotoa
  - Pistokekirjastossa rutiinit **`htoi()`** ja **`itoh()`** (Host to Internet & Internet to Host)

Datan kohdentaminen (alignment)

- 16b data alkamaan parillisista (tavu)osoitteista
- 32b data 4:llä jaollisista osoitteista
- 64b data 8:lla jaollisista osoitteista
- **Kohdennettu data helpompaa käsitellä**
  - esim. 32b data ladattavissa yhdellä muistinoudolla (sanaosoite)
- **Kohdentamaton data ei tuota väliin “hukkatavuja”**
  - Esim. 32b kohdentamaton data tarvitsee kaksi muistinoutoa (sanaosoite) ja yhdistämisen

0010…10010
0010…10100
0010…11000

11	22	33	44

```
load r1, 0(r4)
```

```
load r1, 0(r4)
shl  r1, =16
load r2, 1(r4)
shr  r2, =16
or   r1, r2
```

		11	22
33	44		

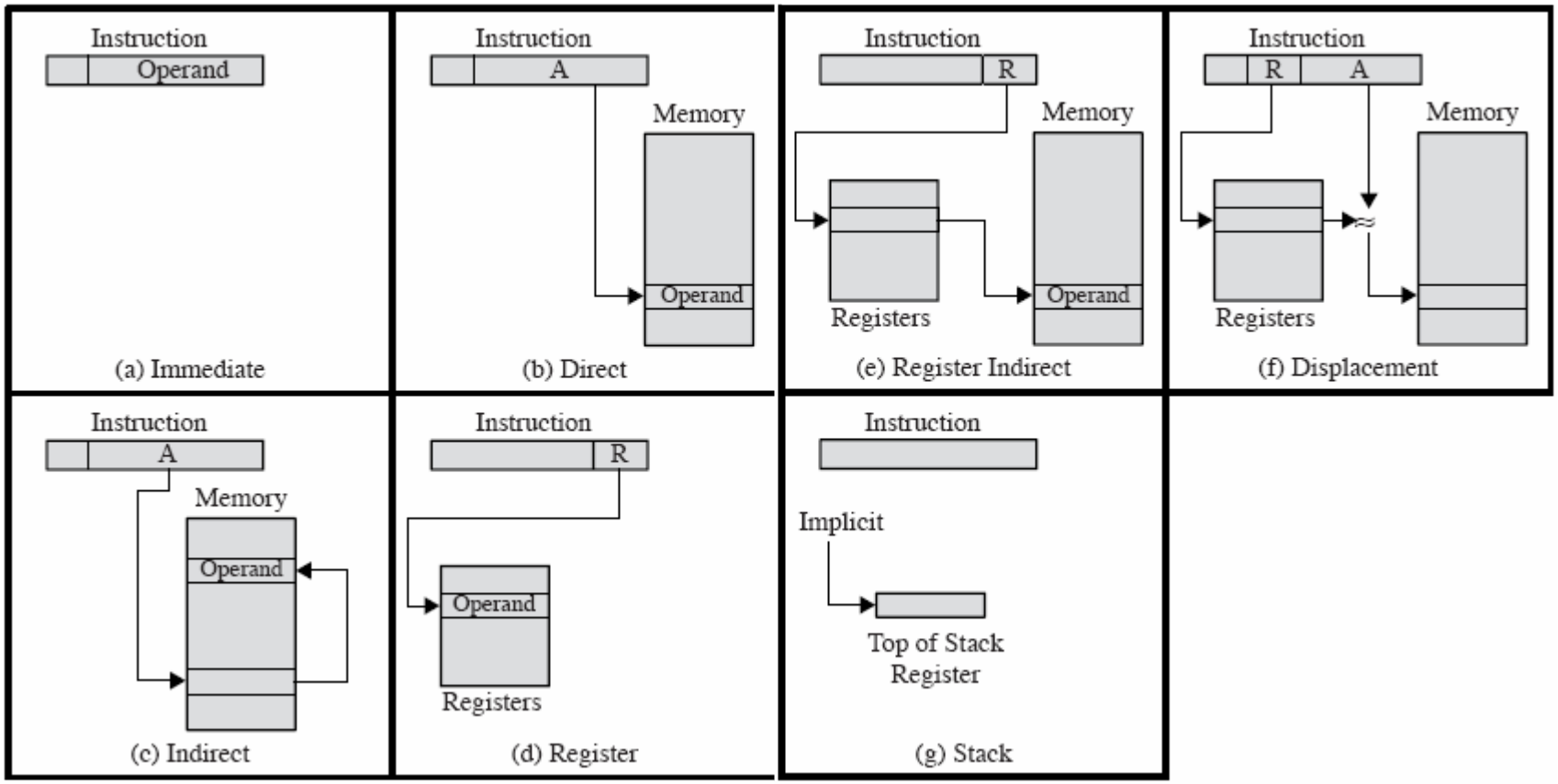
Muistin osoitustavat

Ch 11 [Sta06]

Missä käskyn operandit?

- **Muistissa**
  - Ohjelman muuttujat, pino, keko
- **Rekistereissä**
  - Käsittelyn aikana (nopeus)
- **Käskyn osana**
  - Pienet vakiot
- **Miten tuo kerrotaan CPU:lle?**
  - Bitit käskyformaatissa
  - Useita osoitusmuotoja

Tiedon osoitusmuodot



(Sta06 Fig 11.1)

Tiedon osoitusmuodot

Mode	Algorithm	Principal Advantage	Principal Disadvantage
Immediate	Operand = A	No memory reference	Limited operand magnitude
Direct	EA = A	Simple	Limited address space
Indirect	EA = (A)	Large address space	Multiple memory references
Register	**Operand = (R)**	No memory reference	Limited address space
Register indirect	EA = (R)	Large address space	Extra memory reference
Displacement	EA = A + (R)	Flexibility	Complexity
Stack	EA = top of stack	No memory reference	Limited applicability

- EA = Effective Address
- (A) = Muistipaikan A sisältö
- (R) = Rekisterin R sisältö
- Pinon päällimmäisen alkion osoitteelle oma rekisteri
- Pinon päällimmäinen alkio (tai 2) omassa rekisterissä

(Sta06 Table 11.1)

Siirtymä (Displacement Address)

- **Effective address = (R1) + A** Tehollinen muistiosoite

rekisterin sisältö + käskyssä annettu vakio

- **Vakio usein pieni (8 b, 16 b?)**
- **Käyttötapoja**
  - PC:n suhteen viittaus
  - Base rekisterin suhteen
  - Taulukon indeksointi
  - Tietueen kenttään viittaus
  - Pinoon viittaaminen (aktivointitietue)

```
JUMP   *+5
CALL   SP, Summation(BX)
ADDF   F2,F2, Table(R5)
MUL    F4,F6, Salary(R8)
STORE  F2, -4(FP)
```

Lisää tiedon osoitustapoja

- **Autoincrement** (before/after)
  - Esim. CurrIndex=i++;

$EA = (R),\ R \leftarrow (R) + S$

operandin koko

- **Autodecrement** (before/after)
  - Esim. CurrIndex=--i;

$R \leftarrow (R) - S,\ EA = (R)$

- **Autoincrement deferred**
  - Esim. Sum = Sum + (*ptrX++);

$EA = Mem(R),\ R \leftarrow (R) + S$

- **Autoscale**
  - Esim. Double X;
    X=Tbl[i][j];

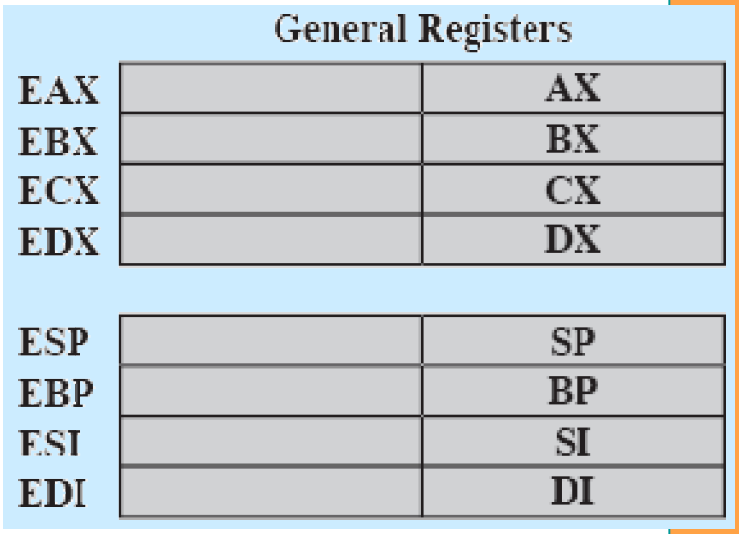
$EA = A + (R_j) + (R_i) * S$

Tietokoneen rakenne

Pentium

Pentium: Rekisterit

- Yleisrekisterit, 32-b
  - EAX, EBX, ECX, EDX työrekisterit
    accu, base, count, data
  - ESI, EDI source & destination index
  - ESP, EBP stack pointer, base pointer
- Niiden loppuosia voi käyttää 16-bittisinä
  - AX, BX, CX, DX, SI, DI, SP, BP
- Työrekistereiden loppuosia 8-bittisinä
  - AH, AL, BH, BL, CH, CL, DH, DL
- Segmenttirekisterit, 16b
  - CS, SS, DS, ES, FS, GS
    - code, stack, data, stack, extra data
- Käskynosoitin
  - EIP Extended Instruction Pointer
- Statusrekisteri
  - EFLAGS
    - overflow, sign, zero, parity, carry,...



Pentium: Datatyypit

- Ei kohdennettu
- Little Endian

Pentium Data Type	Description
General	Byte, word (16 bits), doubleword (32 bits), and quadword (64 bits) locations with arbitrary binary contents.
Integer	A signed binary value contained in a byte, word, or doubleword, using twos complement representation.
Ordinal	An unsigned integer contained in a byte, word, or doubleword.
Unpacked binary coded decimal (BCD)	A representation of a BCD digit in the range 0 through 9, with one digit in each byte.
Packed BCD	Packed byte representation of two BCD digits; value in the range 0 to 99.
Near pointer	A 32-bit effective address that represents the offset within a segment. Used for all pointers in a nonsegmented memory and for references within a segment in a segmented memory.
Bit field	A contiguous sequence of bits in which the position of each bit is considered as an independent unit. A bit string can begin at any bit position of any byte and can contain up to $2^{32} - 1$ bits.
Byte string	A contiguous sequence of bytes, words, or doublewords, containing form zero to $2^{32} - 1$ bytes.
Floating point	**Single / Double / Extended precision**

(Sta06 Table 10.2)

Pentium: Operaatiot (tässä vain osa)

High-Level Language Support	
ENTER	Creates a stack frame that can be used to implement the rules of a block-structured high-level language.
LEAVE	Reverses the action of the previous ENTER.
BOUND	Check array bounds. Verifies that the value in operand 1 is within lower and upper

Segment Register	
LDS	Load pointer into D segment register.
	System Control
HLT	Halt.
LOCK	Asserts a hold on shared memory so that the Pentium has exclusive use of it during the instruction that immediately follows the LOCK.
ESC	Processor extension escape. An escape code that indicates the succeeding instructions are to be executed by a numeric coprocessor that supports high-precision integer and floating-point calculations.
WAIT	Wait until BUSY# negated. Suspends Pentium program execution until the processor detects that the BUSY pin is inactive, indicating that the numeric coprocessor has finished execution.
Protection	
SGDT	Store global descriptor table.
LSL	Load segment limit. Loads a user-specified register with a segment limit.
VERR/VERW	Verify segment for reading/writing.
Cache Management	
INVD	Flushes the internal cache memory.
WBINVD	Flushes the internal cache memory after writing dirty lines to memory.
INVLPG	Invalidates a translation lookaside buffer (TLB) entry.

(Sta06 Table 10.8)

Pentium: MMX Operaatiot (tässä vain osa)

Category	Instruction	Description
Arithmetic	PADD [B, W, D]	Parallel add of packed eight bytes, four 16-bit words, or two 32-bit doublewords, with wraparound.
	PADDS [B, W]	Add with saturation.
	PADDUS [B, W]	Add unsigned with saturation
	PSUB [B, W, D]	Subtract with wraparound.
	PSUBS [B, W]	Subtract with saturation.
	PSUBUS [B, W]	Subtract unsigned with saturation
	PMULHW	Parallel multiply of four signed 16-bit words, with high-order 16 bits of 32-bit result chosen.
	PMULLW	Parallel multiply of four signed 16-bit words, with low-order 16 bits of 32-bit result chosen.
	PMADDWD	Parallel multiply of four signed 16-bit words; add together adjacent pairs of 32-bit results.
Conversion	PACKUSWB	Pack words into bytes with unsigned saturation.
	PACKSS [WB, DW]	Pack words into bytes, or doublewords into words, with signed saturation.
	PUNPCKH [BW, WD, DQ]	Parallel unpack (interleaved merge) high-order bytes, words, or doublewords from MMX register.
	PUNPCKL [BW, WD, DQ]	Parallel unpack (interleaved merge) low-order bytes, words, or doublewords from MMX register.

No under/overflow.
Use closest representation

(Sta06 Table 10.11)

Pentium: Osoitustavat

Pentium Addressing Mode	Algorithm
Immediate	Operand = A
Register Operand	**Operand = (R)**
Displacement	LA = (SR) + A
Base	LA = (SR) + (B)
Base with Displacement	LA = (SR) + (B) + A
Scaled Index with Displacement	LA = (SR) + (I) × S + A
Base with Index and Displacement	LA = (SR) + (B) + (I) + A
Base with Scaled Index and Displacement	LA = (SR) + (I) × S + (B) + A
Relative	LA = (PC) + A

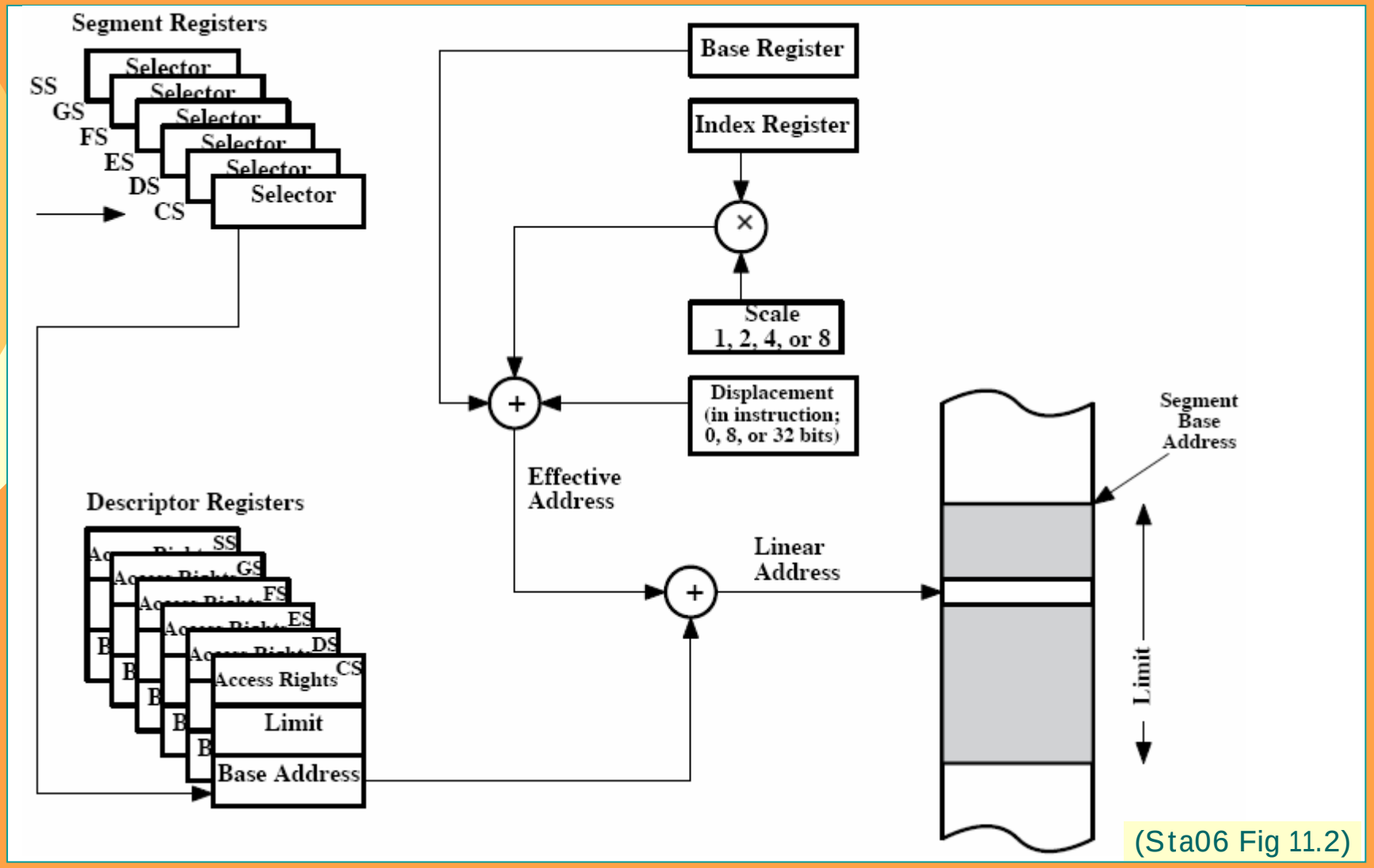
1, 2, 4, 8B

Rekisterit: 1, 2, 4, 8B

LA = linear address
(X) = contents of X
SR = segment register
PC = program counter
A = contents of an address field in the instruction
R = register
B = base register
I = index register
S = scaling factor

(Sta06 Table 11.2)

Pentium: Osoitteen laskenta

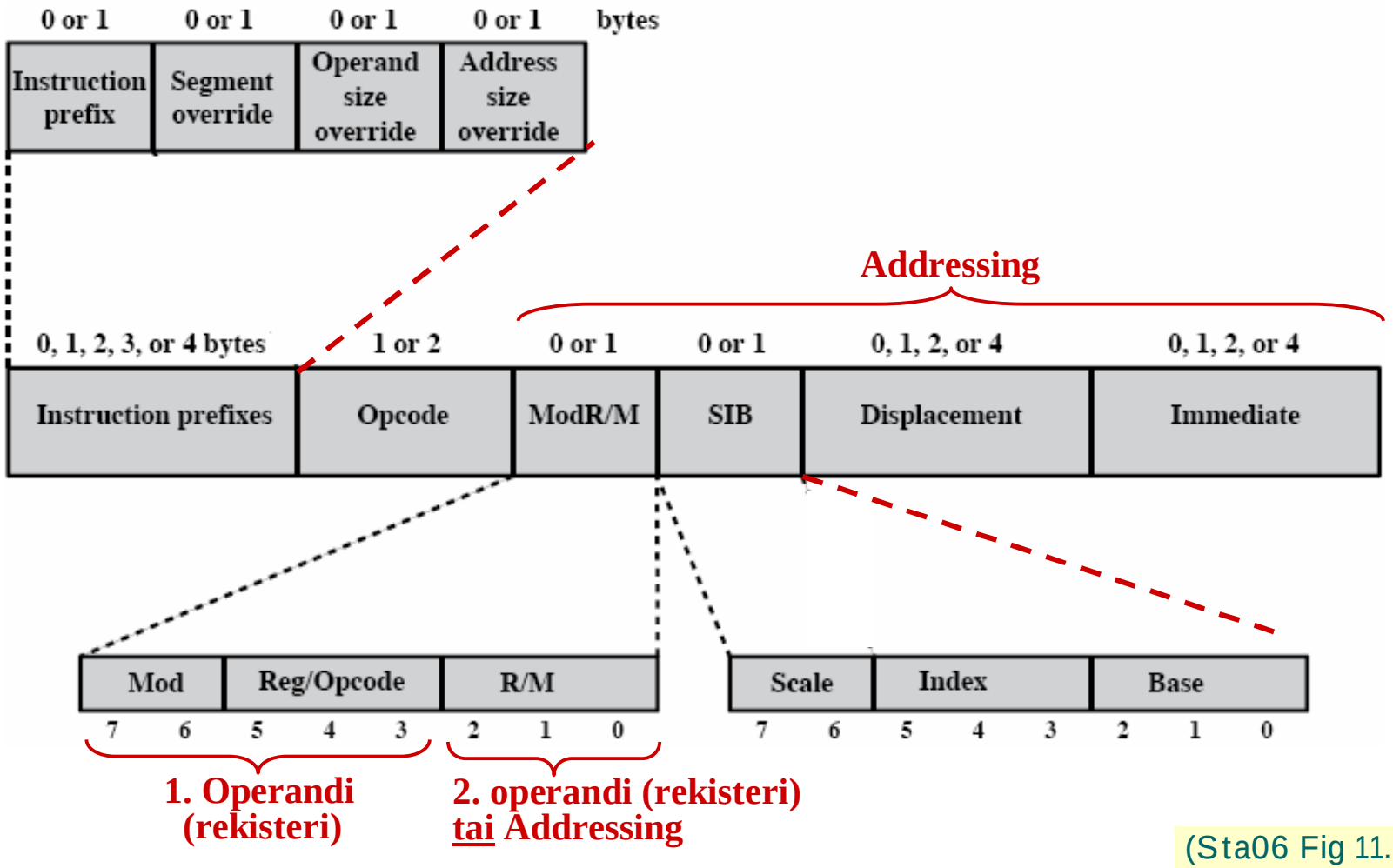


(Sta06 Fig 11.2)

Pentium: Käskyformaatti

- **CISC**
  - Complex Instruction Set Computer
- **Paljon valinnaisia osia**
  - Osio esiintyy käskyn bittijonossa tai ei
  - Alussa 0-4 prefix tavua
  - Loppuosan tulkinta riippuu aiempien kenttien sisällöistä
- **Monipuoliset osoitustavat**
  - Käskyn operandeista korkeintaan yksi muistissa
  - 24 erilaista
- **Taaksepäin yhteensopivuus**
  - Icuwanhojen 16-bittisten 8086-ohjelmien pitää toimia
    - Vanhat käskyt emuloimalla vai simuloimalla?

Pentium: Käskyformaatti



(Sta06 Fig 11.8)

Pentium: Käskyformaatti

Sta06 Fig 11.8

- **Instruction prefix** (optional)
  - LOCK – jakamaton operaatio moniCPU-järjestelmässä
  - REP – toista samaa käskyä merkkijonon kaikille merkeille
- **Segment override** (optional)
  - Käytä tässä käskyssä eksplisiittisesti annettua segmenttirekisteriä
  - Käytä muulloin implisiittistä oletusta
- **Operand size override** (optional)
  - Onko operandi 16 vai 32 bittinen (toinen oletuksena)
- **Address size override** (optional)
  - Onko osoite 16 vai 32 bittinen (toinen oletuksena)

Pentium: Käskyformaatti

Sta06 Fig 11.8

- **Opcode**
  - Kullekin käskylle oma bittikombinaatio
  - Bitit kertovat myös operandin koosta (8/16/32b)
- **ModR/M (optional)**
  - Onko operandi rekisterissä vai muistissa
  - Mitä osoitusmuotoa käytetään
  - Joissakin tapauksissa täsmentää operaatiokoodia
- **SIB = Scale/Index/Base (optional)**
  - Joissakin osoitusmuodoissa tarvitaan lisätarkennuksia
  - Scale: alkion koko indeksointia käytettäessä
  - Index: indeksirekisterin numero
  - Base: kantarekisterin numero

Pentium: Käskyformaatti

Sta06 Fig 11.8

- **Displacement** (optional)
  - Tarvitaan eräissä osoitustavoissa
  - 0, 1, 2 tai 4 tavua
- **Immediate** (optional)
  - Tarvitaan eräissä osoitustavoissa
  - 0, 1, 2 tai 4 tavua

Tietokoneen rakenne

PowerPC

PowerPC: Käskykanta

- RISC
  - Reduced Instruction Set Computer
- Kiinteä käskynpituus (32b), vähän formaatteja
  - Käskyissä tavallisesti 3 operandia
- Minimijoukko erilaisia käskyjä
  - Helpompi laitteistototeutus, nopeampi suorittaa
  - Pitemmät ohjelmat?
- Vain 2 osoitusmuotoa
  - Load/Store-arkkitehtuuri
- 32 yleisrekisteriä
- Kiinteä datan koko (32/64)
- Ei esim. merkkijono-operaatioita
  - Kirjastoina

PowerPC: Osoitustavat

Mode	Algorithm
Load/Store Addressing	
Indirect	EA = (BR) + D
Indirect Indexed	EA = (BR) + (IR)
Branch Addressing	
Absolute	EA = I
Relative	EA = (PC) + I
Indirect	EA = (L/CR)
Fixed-Point Computation	
Register	EA = GPR
Immediate	Operand = I
Floating-Point Computation	
Register	EA = FPR

EA	=	effective address
(X)	=	contents of X
BR	=	base register
IR	=	index register
L/CR	=	link or count register
GPR	=	general-purpose register
FPR	=	floating-point register
D	=	displacement
I	=	immediate value
PC	=	program counter

(Sta06 Table 11.3)

PowerPC: Käskyformaatti

XO=Opcode extension, R=Record condition in CR1, O=Record overflow in XER
S = Part of Shift Amount field, *=64 bitin arkkitehtuureissa

← 6 bits →	← 5 bits →	← 5 bits →	← 16 bits →				
Ld/St Indirect	Dest Register	Base Register	Displacement				
Ld/St Indirect	Dest Register	Base Register	Index Register	Size, Sign, Update	/		
Ld/St Indirect	Dest Register	Base Register	Displacement		XO		*

(c) Load/store instructions

Arithmetic	Dest Register	Src Register	Src Register	O	Add, Sub, etc.		R	
Add, Sub, etc.	Dest Register	Src Register	Signed Immediate Value					
Logical	Src Register	Dest Register	Src Register	ADD, OR, XOR, etc.			R	
AND, OR, etc.	Src Register	Dest Register	Unsigned Immediate Value					
Rotate	Src Register	Dest Register	Shift Amt	Mask Begin	Mask End		R	
Rotate or Shift	Src Register	Dest Register	Src Register	Shift Type or Mask			R	
Rotate	Src Register	Dest Register	Shift Amt	Mask	XO	S	R	*
Rotate	Src Register	Dest Register	Src Register	Mask	XO		R	*
Shift	Src Register	Dest Register	Shift Type or Mask			S	R	*

(d) Integer arithmetic, logical, and shift/rotate instructions

Flt sgl/dbl	Dest Register	Src Register	Src Register	Src Register	Fadd, etc.	R

(e) Floating-point arithmetic instructions

(Sta06 Fig 11.9)

PowerPC: Käskyformaatti

- **Yksinkertaiset hyppykäskyt**
  - CR: mitä CR-rekisterin bittejä tutkitaan
  - L (Link): onko kyseessä aliohjelmaan siirtyminen (Aktivointitietue kuntoon!)
  - A (Absolute): onko annettu hypyn kohdeosoite, vai onko hyppy suhteellinen PC:n suhteen

Branch	Long Immediate			A	L
Br Conditional	Options	CR Bit	Branch Displacement	A	L
Br Conditional	Options	CR Bit	Indirect through Link or Count Register		L

(a) Branch instructions

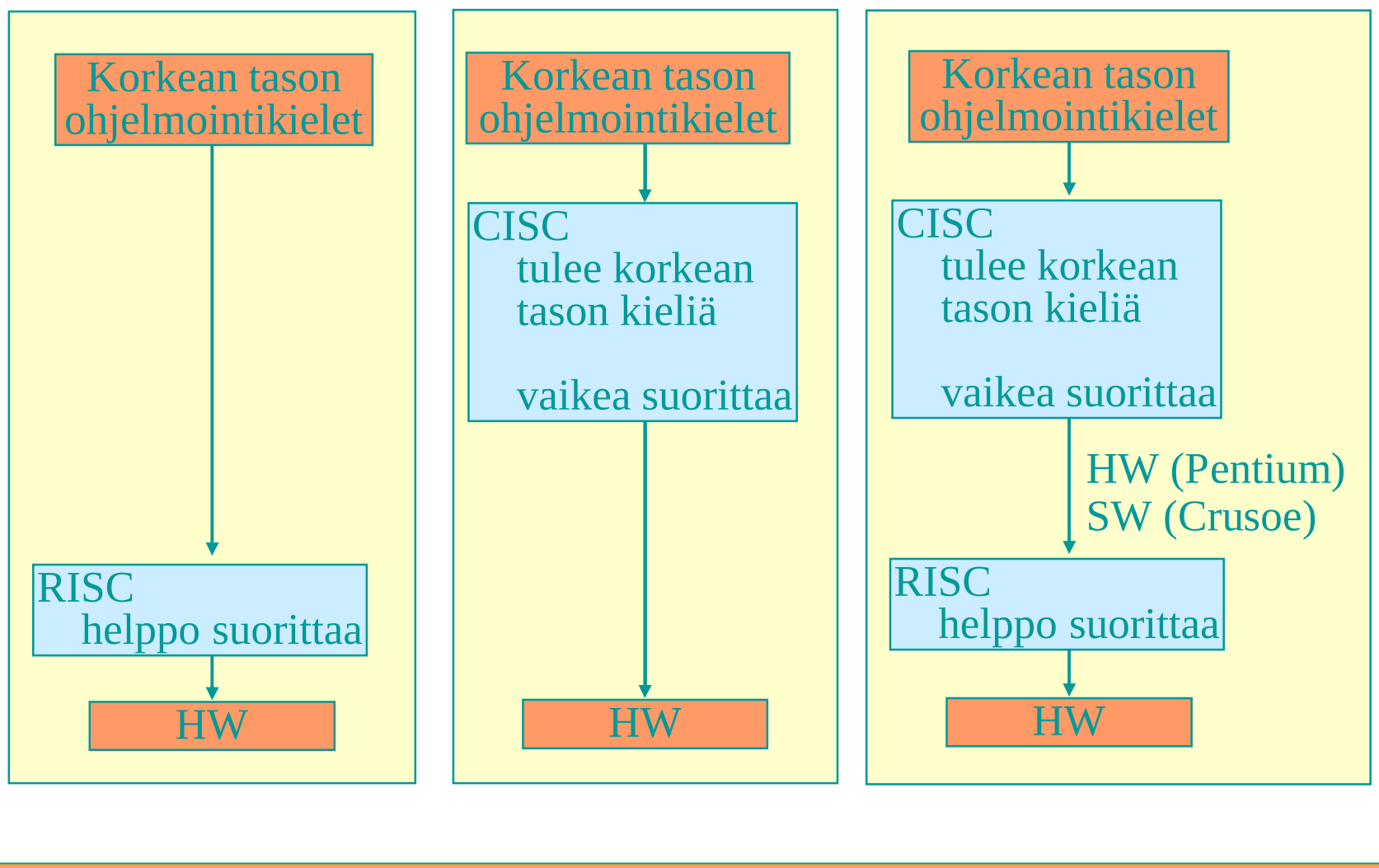
CR	Dest Bit	Source Bit	Source Bit	Add, OR, XOR, etc.	/

(b) Condition register logical instructions

(Sta06 Fig 11.9)

RISC vs. CISC

lisää myöhemmin (Luento 9)



Kertauskysymyksiä

- Millaisista osista konekielinen käsky muodostuu?
- Miten CPU tietää onko sen käsittelemä kokonaisluku 16 bittinen vai 32 bittinen?
- Mitä tarkoittaa Big-Endian?
- Mitä hyötyä on kiinteästä käskyformaatista verrattuna vaihtelevanpituiseen formattiin?