

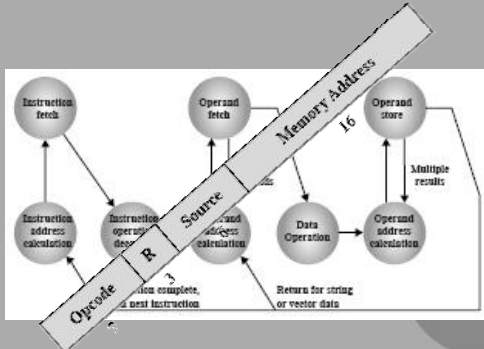
HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI


Lecture 7

Instruction sets (Käskykannat)

Ch 10-11 [Sta06]

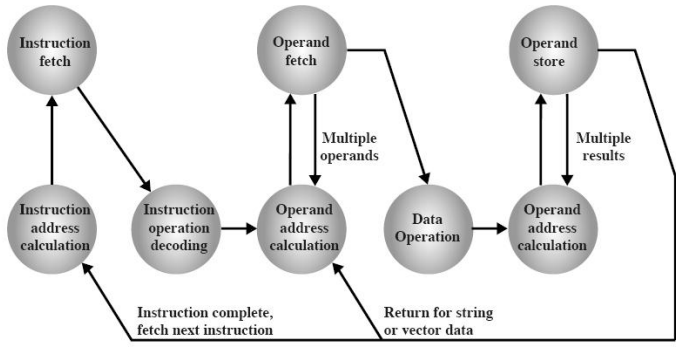
- Operations
- Operands
- Operand references (osoitustavat)
- Pentium / PowerPC





Instruction cycle

- CPU executes instructions “one after another”
- Execution of one instruction has several phases (see state diagram). The CPU repeats these phases



(Sta06 Fig 10.1)

Computer Organization II, Spring 2009, Tiina Niklander

30.3.2009 2

Computer Instructions (*konekäskyt*)

- Instruction set (*käskykanta*) =
 - Set of instructions CPU 'knows'
- Operation code (*käskykoodi*)
 - What does the instruction do?
- Data references (*viitteet*) – one, two, several?
 - Where does the data come for the instruction?
 - Registers, memory, disk, I/O
 - Where is the result stored?
 - Registers, memory, disk, I/O
- What instruction is executed next?
 - Implicit? Explicite?
- I/O?
 - Memory-mapped I/O → references as if in memory

Covered on
Comp. Org I

Access rate?

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 3

Instructions and data (*käskyt ja data*)

data instructions	Address		Contents		
		101	0010	0010	0000
	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

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

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)

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 4




Instruction types?

Sta06 Table 10.3

- Transfer between memory and registers
 - LOAD, STORE, MOVE, PUSH, POP, ...
- Controlling I/O
 - Memory-mapped I/O - same as
 - I/O not memory-mapped – own instructions to control
- Arithmetic and logical operations
 - ADD, MUL, CLR, SET, COMP, AND, SHR, NOP, ...
- Conversions (*esitystapamuunnokset*)
 - TRANS, CONV, 16bTo32b, IntToFloat, ...
- Transfer of control (*käskyjen suoritusjärjestyksen ohjaus*), conditional, unconditional
 - JUMP, BRANCH, JEQU, CALL, EXIT, HALT, ...
- Service requests (*palvelupyyntö*)
 - SVC, INT, IRET, SYSENTER, SYSEXIT, ...
- Privileged instructions (*etuoikeutetut käskyt*)
 - DIS, IEN, flush cache, invalidate TLB, ...

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 5




What happens during instruction execution?

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

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 6



What kind of data?


- Integers, floating-points
- Boolean (*totuusarvoja*)
- Characters, strings
 - IRA (aka ASCII), EBCDIC
- Vectors, tables
 - N elements in sequence
- Memory references

- Different sizes
 - 8 /16/32/ 64b, ...
 - Each type and size has its own operation code

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 (Short)	32
STD	Store (Long)	64

IBM S/390 (Sta06 Table 10.5)

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 7



Instruction representation (*käskyformaatti*)

- How many bits for each field in the instruction?
 - How many different instructions?
 - Maximum number of operands per instruction?
 - Operands in registers or in memory?
 - How many registers?
- Fixed or variable size (*vakio vai vaihteleva koko*)?

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)

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 8



How many registers?


- Minimum 16 to 32
 - Work data in registers
- Different register (sets) for different purpose?
 - Integers vs floating points, indices vs data, code vs. data
 - All sets can start register numbering from 0
 - Opcode determines the set that is used
- More registers than can be referenced?
 - CPU allocates them internally
 - Register window
 - Example subprogram parameters always in registers
 - Programmer thing that registers are always r8-r15,
 - CPU used register set of 8-132
 - (We'll come back to this later)



Käskyarkkitehtuureja

- Accumulator-based architecture (*akkukone*)
 - Just one register, accumulator, implicit reference to it
- Stack-based (*pinokone*) Ks. Sta06 Appendix 10A
 - Operands in stack, implicit reference
 - PUSH, POP Example: JVM
- Register-based (*yleisrekisterikone*)
 - All registers of the same size
 - Instructions have 2 or 3 operands
- Load/Store architecture
 - Only LOAD/STORE have memory refs
 - ALU-operations have 3 regs

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



Byte ordering (*tavujärjestys*): Big vs. Little Endian

Ks. Sta06 Appendix 10B

■ How to store a multibyte scalar value?

0x1200:
(*sanaosoite*) Word

0x1200	0x1201	0x1202	0x1203

Byte (*tavuosoitteet*)

"Isoimmassa lopputavu"

→

STORE 0x11223344, 0x1200 ???

Big-Endian:
Most significant byte
in lowest byte addr

0x11	0x22	0x33	0x44
0x1200	0x1201	0x1202	0x1203

Little-Endian:
Least significant byte
in lowest byte addr


0x44	0x33	0x22	0x11
0x1200	0x1201	0x1202	0x1203

"Pienimmässä lopputavu"

0x00000044 =

0x44	0x00	0x00	0x00
0x1200	0x1201	0x1202	0x1203

Computer Organization II, Spring 2009, Tiina Niklander
30.3.2009 11



Big vs. Little Endian


■ ALU uses only one of them

- Little-endian: x86, Pentium, VAX
- Big-endian: IBM 370/390, Motorola 680x0 (Mac),
most RISC-architectures
- Power-PC, a bi-endian machine, accepts both
 - machine status register, MSR, has 2 bits to indicate the
endian mode (one bit for kernel, one for the current mode)

■ Byte order must be known, when transferring data from one
machine to another

- Internet uses big-endian format
- Socket library (*pistokekirjasto*) has routines `htoi()` ja `itoh()`
(Host to Internet & Internet to Host)

Computer Organization II, Spring 2009, Tiina Niklander
30.3.2009 12



Data alignment (*kohdentaminen*)

```
0010...10010
0010...10100
0010...11000
```

- 16b data starts with even (*parillinen*) (byte)address
- 32b data starts with address divisible (*jaollinen*) by 4
- 64b data starts with address divisible by 8
- Aligned data is easier to access
 - 32b data can be loaded by one operation accessing the word address (*sanaosoite*)
- Unaligned data would contain no 'wasted' bytes, but
 - For example, loading 32b unaligned data requires two loads from memory (word address) and combining it

11	22	33	44
----	----	----	----

		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
```

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 13



Computer Organization II

Memory references

(*Muistin osoitustavat*)

Ch 11 [Sta06]

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 14

Where are the operands?

- In the memory
 - Variable of the program, stack (*pino*), heap (*keko*)
- In the registers
 - During the instruction execution, for speed
- Directly in the instruction
 - Small constant values
- How does CPU know the specific location?
 - Bits in the operation code
 - Several alternative addressing modes allowed


Computer Organization II, Spring 2009, Tiina Niklander
30.3.2009 15

Addressing modes (*osoitusmuodot*)

<p>(a) Immediate</p>	<p>(b) Direct</p>	<p>(e) Register Indirect</p>	<p>(f) Displacement</p>
<p>(c) Indirect</p>	<p>(d) Register</p>	<p>(g) Stack</p>	

(Sta06 Fig 11.1)

Computer Organization II, Spring 2009, Tiina Niklander
30.3.2009 16




Addressing modes

(Sta06 Table 11.1)

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

- EA = Effective Address
- (A) = content of memory location A
- (R) = content of register R
- One register for the top-most stack item's address
- Register (or two) for the top stack item (or two)

Computer Organization II, Spring 2009, Tiina Niklander
30.3.2009 17



Displacement Address (*siirtymä*)


(tehollinen muistiosoite)

- Effective address = (R1) + A

\nearrow
 register content + constant in the instruction

\nwarrow
- Constant relative small (8 b, 16 b?)
- Usage
 - Relational to PC JUMP *+5
 - Relational to Base CALL SP, Summation(BX)
 - Indexing a table ADDF F2,F2, Table(R5)
 - Ref to record field MUL F4,F6, Salary(R8)
 - Stack content (aktivointitietue) STORE F2, -4(FP)

Computer Organization II, Spring 2009, Tiina Niklander
30.3.2009 18




More addressing modes

- Autoincrement (before/after) $EA = (R), R \leftarrow (R) + S$
 - Example `CurrIndex=i++;`
- Autodecrement (before/after) $R \leftarrow (R) - S, EA = (R)$
 - Example `CurrIndex=--i;`
- Autoincrement deferred $EA = Mem(R), R \leftarrow (R) + S$
 - Example `Sum = Sum + (*ptrX++);`
- Autoscale $EA = A + (R_j) + (R_i) * S$
 - Example `Double X;`
`X=Tbl[i][j];`

Size of operand

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 19



Computer Organization II

Pentium

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 20

Pentium: Registers

- General registers (*yleisrekisterit*), 32-b
 - EAX, EBX, ECX, EDX accu, base, count, data
 - ESI, EDI source & destination index
 - ESP, EBP stack pointer, base pointer
- Part of them can be used as 16-bit registers
 - AX, BX, CX, DX, SI, DI, SP, BP
- Or even as 8-bit registers
 - AH, AL, BH, BL, CH, CL, DH, DL
- Segment registers 16b
 - CS, SS, DS, ES, FS, GS
 - code, stack, data, stack, extra data
- Program counter (*käskynosoitin*)
 - EIP Extended Instruction Pointer
- Status register
 - EFLAGS
 - overflow, sign, zero, parity, carry,...

General Registers	
EAX	AX
EBX	BX
ECX	CX
EDX	DX
ESP	SP
EBP	BP
ESI	SI
EDI	DI


Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 21

Pentium: Data types

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 from zero to $2^{32} - 1$ bytes.
Floating point	Single / Double / Extended precision IEEE 754 standard

(Sta06 Table 10.2)

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 22




Data transfers, arithmetics, moves, jumps, stricts, etc

Pentium: Operations (just part of)

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	Invalidate a translation lookaside buffer (TLB) entry.

(Sta06 Table 10.8)

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 23



SIMD

Pentium: MMX Operations (just part of)

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. No under/overflow.
	PSUB [B, W, D]	Subtract with wraparound. Use closest representation
	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.

(Sta06 Table 10.11)

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 24

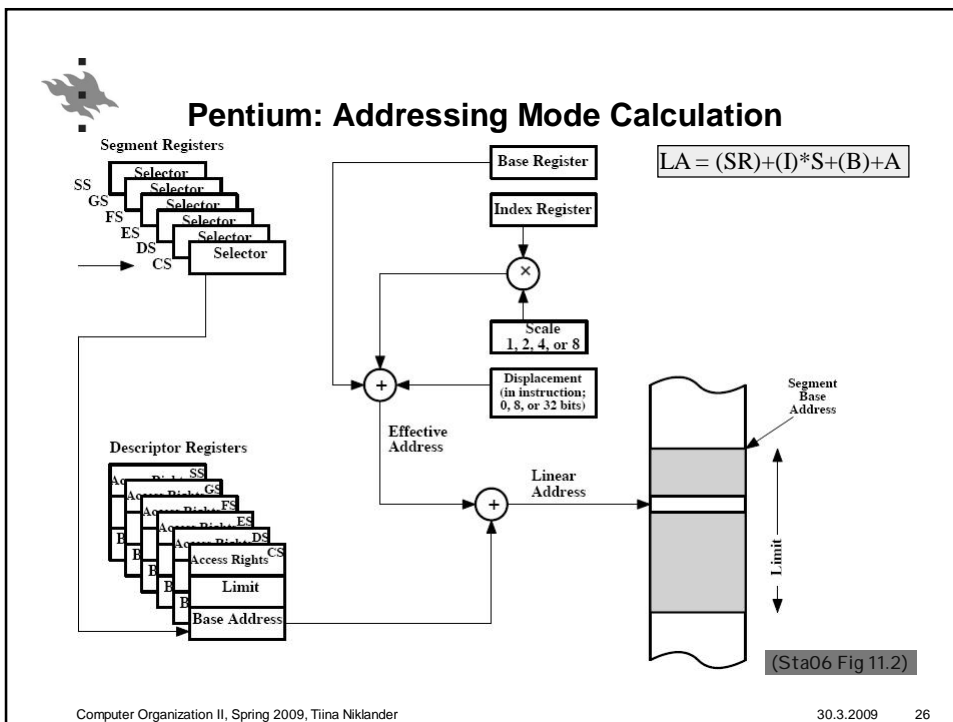
Pentium: Addressing modes (osoitustavat)

Pentium Addressing Mode	Algorithm	
Immediate	Operand = A	1, 2, 4, 8B
Register Operand	Operand = (R)	Registers: 1, 2, 4, 8B
Displacement	LA = (SR) + A	
Base	LA = (SR) + (B)	
Base with Displacement	LA = (SR) + (B) + A	For indexing arrays
Scaled Index with Displacement	LA = (SR) + (I) × S + A	For arrays in stack or for two dimensional arrays
Base with Index and Displacement	LA = (SR) + (B) + (I) + A	
Base with Scaled Index and Displacement	LA = (SR) + (I) × S + (B) + A	different element size
Relative	LA = (PC) + A	

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

(Sta06 Table 11.2)

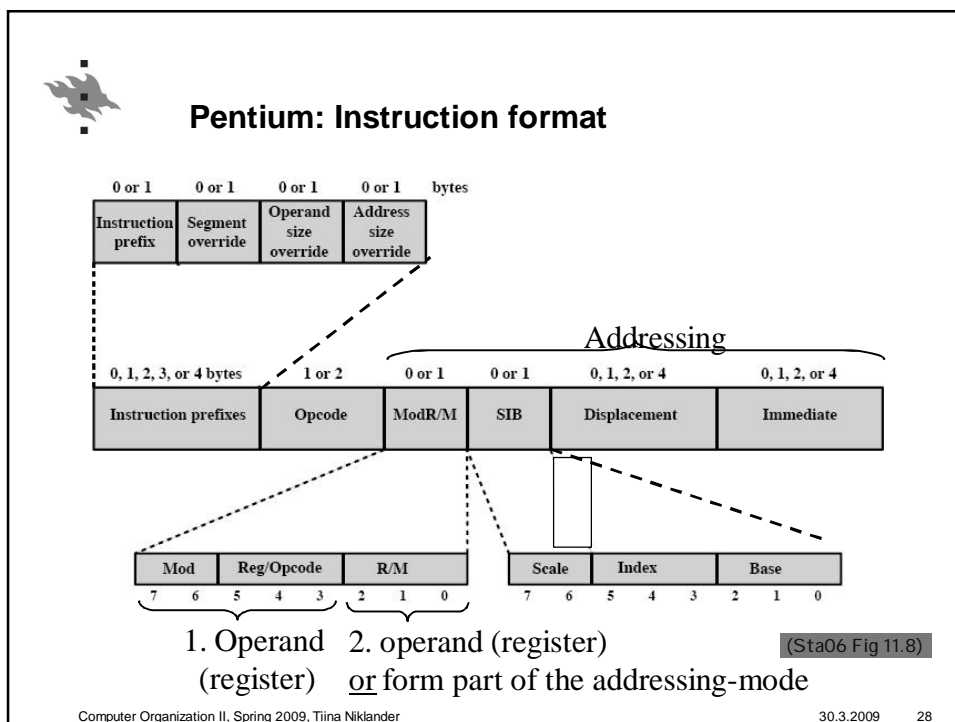
Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 25




Pentium: Instruction format

- CISC
 - Complex Instruction Set Computer
 - Lot of alternative fields op-code only one always!
 - Part may be present or absent in the bit sequence
 - Prefix 0-4 bytes
 - Interpretation of the rest of the bit sequence depends on the content of the preceding fields
- Plenty of alternative addressing modes (*osoitustapa*)
 - At most one operand can be in the memory
 - 24 different
- Backward compatibility
 - OLD 16-bit 8086-programns must still work
 - How to handle old instructions: emulate, simulate?

Computer Organization II, Spring 2009, Tiina Niklander
30.3.2009 27






See Sta06 Fig 11.8

Pentium: Instruction format

- Instruction prefix (optional)
 - LOCK –exclusive use of shared memory in multiprocessor env.
 - REP – repeat operation to all characters of a string
- Segment override (optional)
 - Use the segment register explicitly specified in the instruction
 - Else use the default segment register (implicit assumption)
- Operand size override (optional)
 - Switch between 16 or 32 bit operand, override default size
- Address size override (optional)
 - Switch between 16 or 32 bit addressing. Override the default, which could be either

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 29




See Sta06 Fig 11.8

Pentium: Instruction format

- Opcode
 - Each instruction has its own bit sequence (incl. opcode)
 - Bits specify the size of the operand (8/16/32b)
- ModR/m(optional)
 - Indicate, whether operand is in a register or in memory
 - What addressing mode (osoitusmuoto) to be used
 - Sometimes enhance the opcode information (with 3 bits)
- SIB = Scale/Index/Base (optional)
 - Some addressing modes need extra information
 - Scale: scale factor for indexing (element size)
 - Index: index register (number)
 - Base: base register (number)

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 30




See Sta06 Fig 11.8

Pentium: Instruction format

- Displacement (optional)
 - Certain addressing modes need this
 - 0, 1, 2 or 4 bytes (0, 8, 16 or 32 bits)
- Immediate (optional)
 - Certain addressing modes need this, value for operand
 - 0, 1, 2 or 4 bytes

Computer Organization II, Spring 2009, Tiina Niklander

30.3.2009 31




Computer Organization II

PowerPC

Computer Organization II, Spring 2009, Tiina Niklander


30.3.2009 32



PowerPC: Instruction set (käskykanta)

- RISC
 - Reduced Instruction Set Computer
- Fixed instruction length (32b), few formats and modes
 - Instructions usually have 3 operands
- Small number of different instructions
 - Easier hardware implementation, faster execution
 - Longer programs?
- Only 2 memory reference formats
 - Load/Store-architecture
- 32 general registers
- Fixed data size (32/64)
- No string operations in the instruction set
 - Libraries (in the programs)

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 33



PowerPC: Addressing modes

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	Operand = (GPR)
Immediate	Operand = I
Floating-Point Computation	
Register	Operand = (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)

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 34

PowerPC: Instruction format (Sta06 Fig 11.9)

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

← 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 sg/dbl	Dest Register	Src Register	Src Register	Src Register	Fadd, etc.	R
------------	---------------	--------------	--------------	--------------	------------	---

(e) Floating-point arithmetic instructions

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 35

PowerPC: Instruction format

- Simple branch (jump) instructions
 - CR: which bit of the condition register (CR) is tested
 - L (Link): subroutine call (return address to link register!)
 - A (Absolute): addressing mode absolute (destination address fully here) or relative to program counter (PC)

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

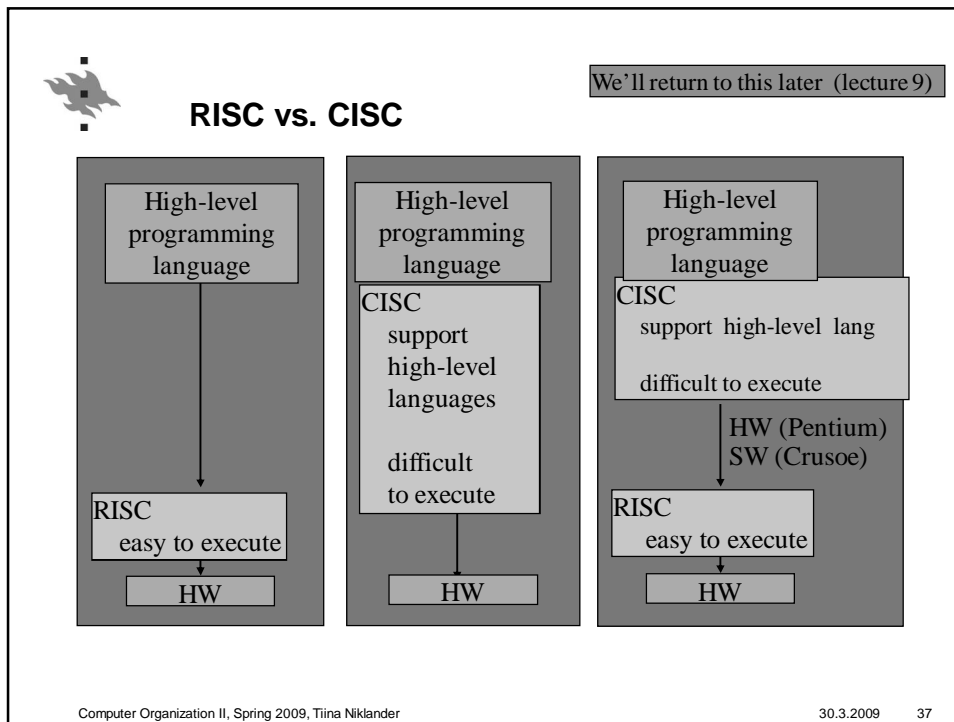
(a) Branch instructions

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

(b) Condition register logical instructions

(Sta06 Fig 11.9)

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 36



Review Questions / Kertauskysymyksiä

- Fields of the instruction?
- How does CPU know if the integer is 16 b or 32 b?
- Meaning of Big-Endian?
- Benefits of fixed instruction size vs variable size instruction format?

- 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 käskyformaattista verrattuna vaihtelevanpituiseen formattiin?

Computer Organization II, Spring 2009, Tiina Niklander 30.3.2009 38