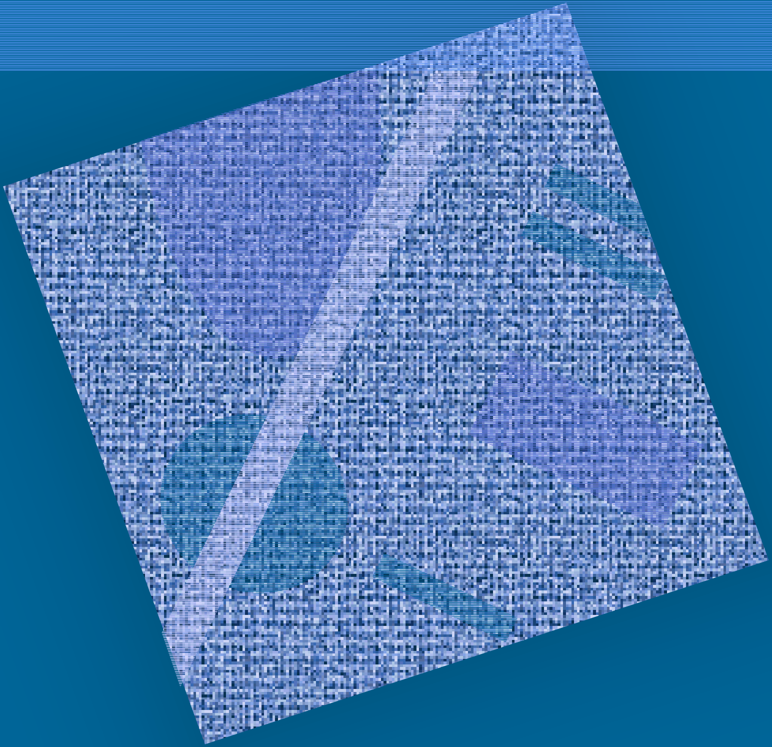
Transmeta Architecture



Major Ideas

General Architecture

Emulated Precise Exceptions

What to do with It

Background

- Transmeta Corporation
  - Paul Allen (Microsoft), George Soros (Soros Funds)
  - David R. Ditzel (Sun) Chief Tech Officer
  - Edmund J. Kelly, Malcolm John Wing, Robert F. Cmelik
  - Linus B. Torvalds, February 1997 → ...
- Patent 5832205
  - applied August 20, 1996
  - granted November 3, 1998
  - many (a few) other patents …
- Crusoe prosessor
  - published January 19, 2000

Basic Idea(s) (5)

faster

cheaper

- Create a new processor which, when coupled with "morph host" emulator, can run Intel/Windows code faster than state-of-the-art Intel processor, *or* with same speed but with less electric power
- New processor can be implemented with significantly fewer gates than competitive processors
- Compete with Intel, friendly with Microsoft
  - sell chip with emulator code to system manufacturers (Dell, IBM, Sun, etc etc)
- x86 binary is new binary standard
- Native OS not so important
  - could be Linux

Major General Ideas

- Emulation can be faster than direct execution
- TLB used to solve new problems
  - track memory accesses for memory mapped I/O
  - track memory accesses for self-modifying code
- Most of executed code generated “on-the fly”
  - not compiled before execution begins
  - extremely optimized dynamic code generation
- Optimized code allows for simpler machine
  - smaller, faster, uses less power?

Major General Ideas (contd)

- Self-modified code (dynamically created code) can be generated so that it is extremely optimized for execution
  - issue dependencies, reorder, reschedule problems solved at code generation (<u>not</u> in HW)
  - processor does not need to solve these
- Optimize for speed only when needed
  - do <u>not optimize</u> for speed when exact state change is required (<u>this is the tricky part</u>!)
- Alias detection to assist keeping globals is registers

Major General Ideas (contd)

- NOT: faster <u>and</u> with less power

Class action suit (27.7.2001) ... stating that ... a revolutionary process that delivered longer battery life in Mobile Internet Computers while delivering high performance ....

http://www.milberg.com/transmeta/

Major Emulation Ideas

- Target processor (I.e., Intel processor) state kept in dedicated HW registers
  - working state ("speculated" state?), committed state
- Memory store buffer keeps uncommitted ("speculated") emulated memory state
- Specific instructions support emulation
  - commit, rollback (exact exceptions)
  - prot (aliases)
- TLB (and VM) designed to support emulation
  - A/N-bit (mem-mapped I/O), T-bit (self-mod. code)

General Architecture

- VLIW implementation
  - VLIW = Very Long Instruction Word
  - 4 simultaneous RISC instructions
    - one each of float, int, load/store, branch
  - no circuitry for issue dependencies, reorder, optimize, reschedule
    - compiler takes care of these
  - what about data, control and structural dependencies?
    - part of issue dependencies?
    - data & structural dependencies under compiler control?

General Architecture (contd)

- Large register set
  - native regs: 64 INT, 32 FP
    - extra regs for renaming
  - target architecture regs: complete CPU state
    - INT, FP, control
    - working regs for normal emulation
    - committed regs for saving emulated processor state

Reax, Recx, Rseq, Reip

General Architecture (contd)

- TLB
  - new features to solve new problems
    - before used to solve also memory protection problems in addition to plain VM address mapping
  - A/N-bit for memory-mapped I/O detection
    - trap to emulator, which creates precise code
    - memory-mapped I/O requires precise emulated processor state changes
  - T-bit for self-modifying code detection
    - trap to emulator, which recreates emulating code in instruction cache ("translation buffer")

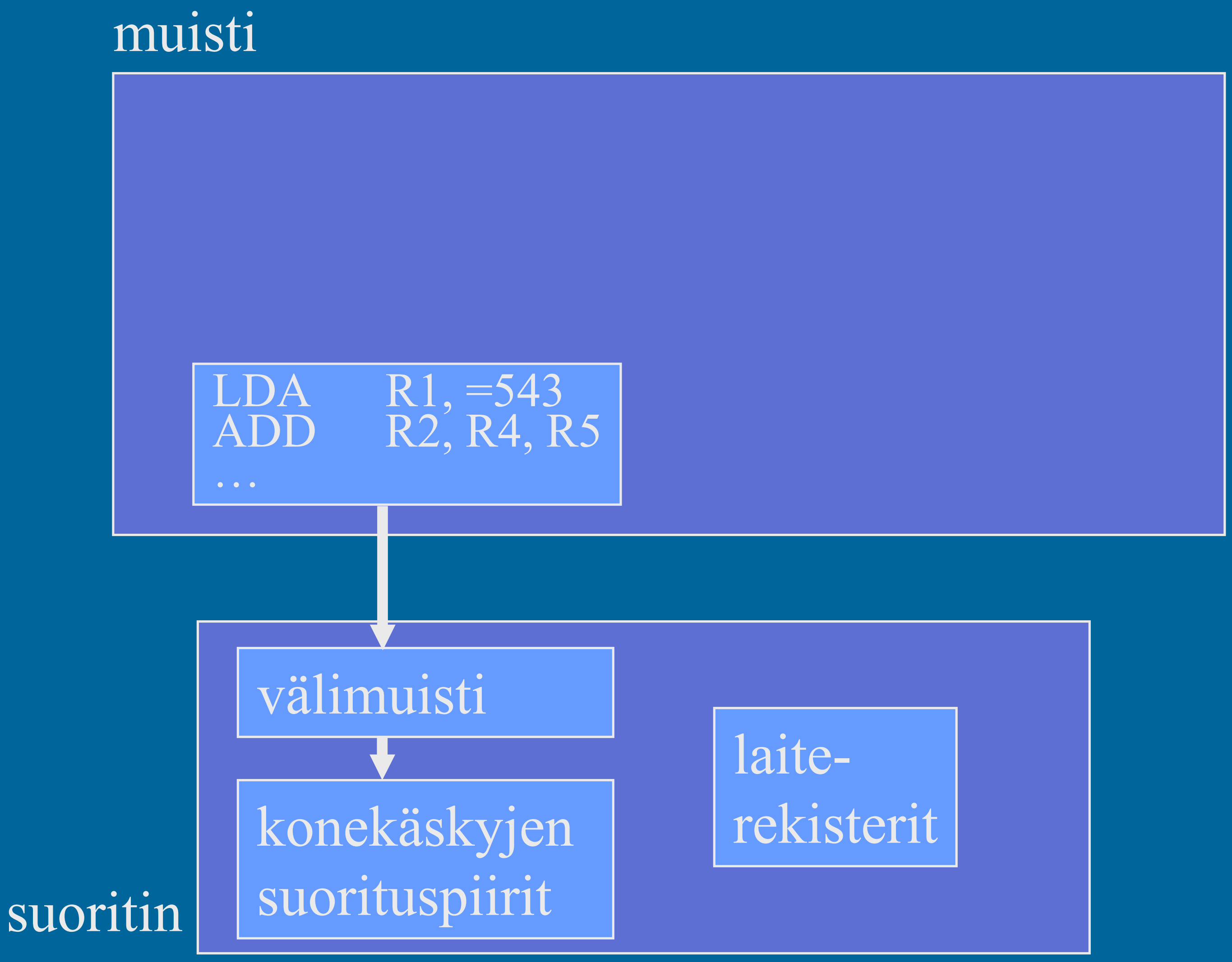
General Architecture (contd)

- Target memory store buffer
  - implemented with special register(s) to support emulation
  - keeps track on which target processor memory stores are committed and which are not
  - uncommitted memory stores can be discarded at rollback
    - modify HW registers implementing it
    - commit & rollback controlled from <u>outside</u> of the processor, not internally as is usual with speculative instructions

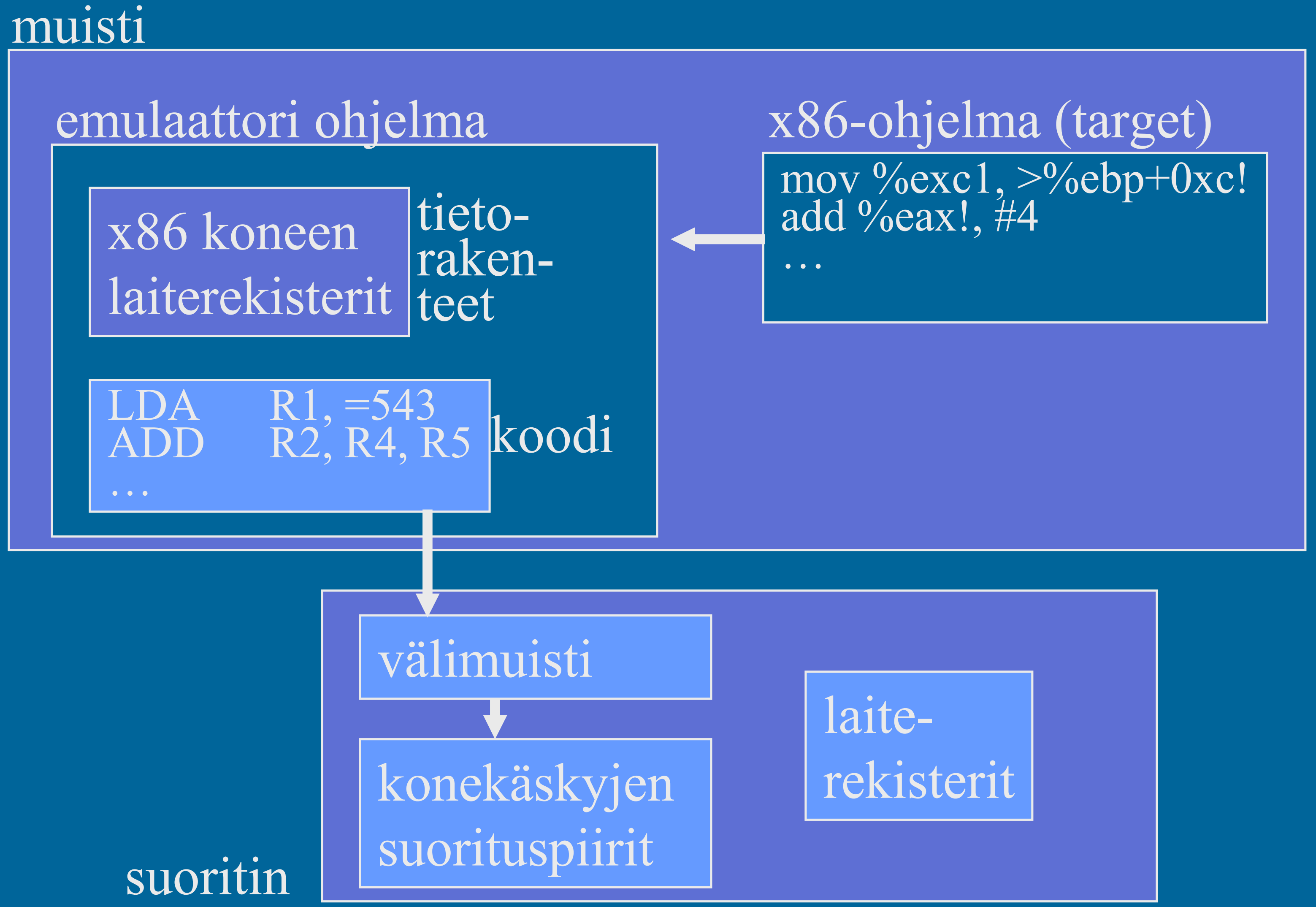
General Architecture (contd)

- RISC instruction set
  - explicitly parallel code (VLIW)
  - *commit* instruction supports emulation
    - commits emulated processor and memory state
    - use when coherent <u>target processor</u> (Intel) state!
  - *rollback* instruction (?) supports emulation
    - some or all of it can be in emulator code
    - recover latest committed emulated target register state
    - delete uncommitted writes from store buffer
    - retranslate emulation code for precise state changes
      - *commit* now after every emulated instruction?
  - *prot* instruction for alias detection

Tavallisen ohjelman suoritus



Tavallisen emulaattoriohjelman suoritus



Tavallinen emulaattoriohjelma

x86-emulaattori (ohjelma)

x86 koneen emuloidut laiterekisterit tietorakenteena

Proseduraalinen pääohjelma, jossa "ikuisessa silmukassa" haetaan x86-konekäskyjä muistista ja emuloidaan niitä yksi kerrallaan sopivalla aliohjelmalla

Staattinen aliohjelma jokaista x86 koneen konekäskyä varten

```
LDA    R1, =543
ADD    R2, R4, R5
…
```

Transmetan emulaattoriohjelma

(x86 koneen emuloidut laiterekisterit laitteistossa)

Dynaamisesti generoidut (optimoitut?) käskysarjat emuloivat x86-koneen konekäskyjä (sekalaisessa?) järjestyksessä

Load	Add	ftSub	
	Sub	ftMul	brEqu
Store	Add		Jump

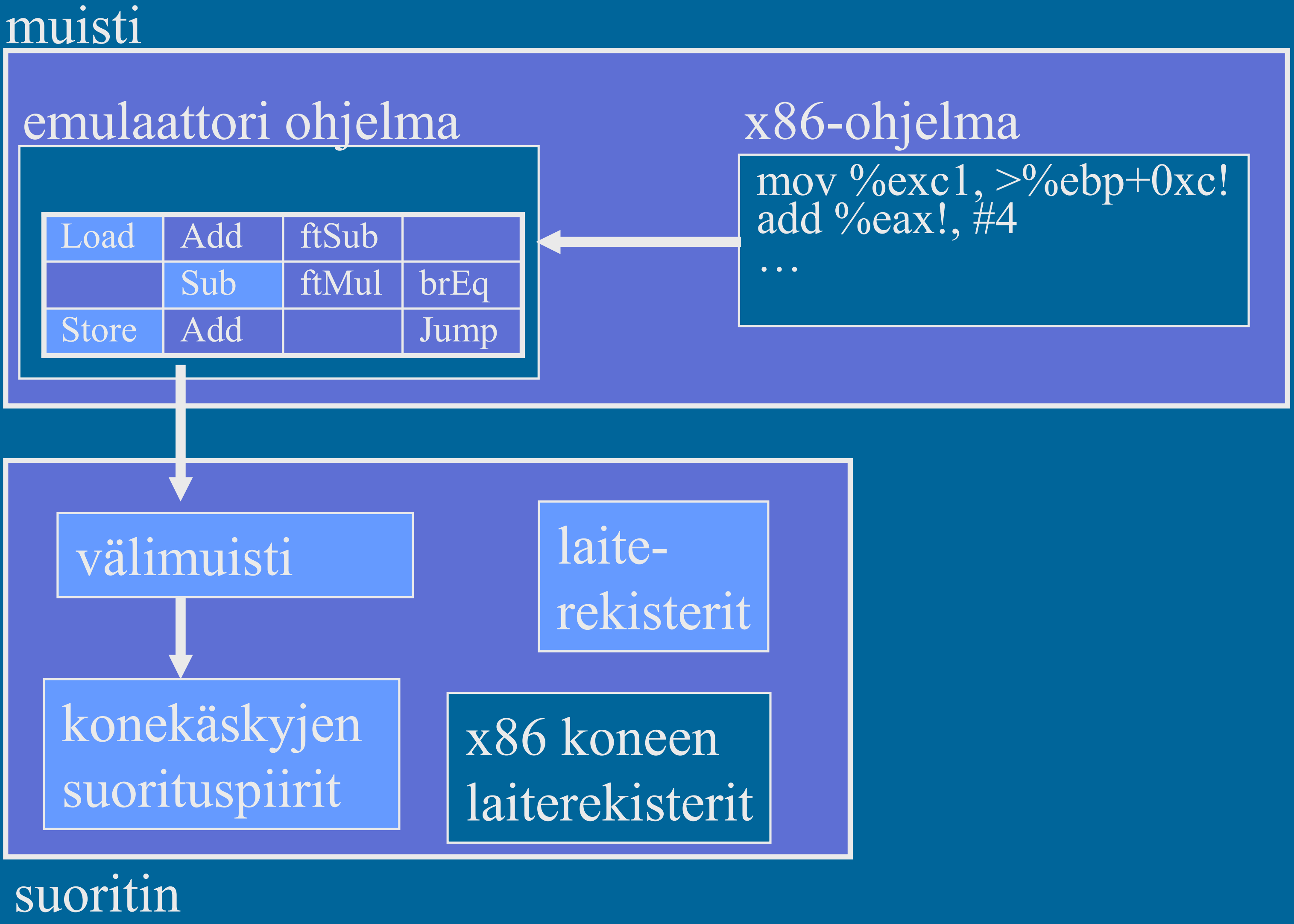
Tapahtumaperustainen pääohjelma, joka valvoo emulointia ja generoi suoritettavia konekäskyjä välimuistiin:

jos emuloitavaa käskysarjaa ei vielä ole generoitu omalle konekielelle, niin generoi se (käännöspuskuriin) ja anna sille suoritusvuoro

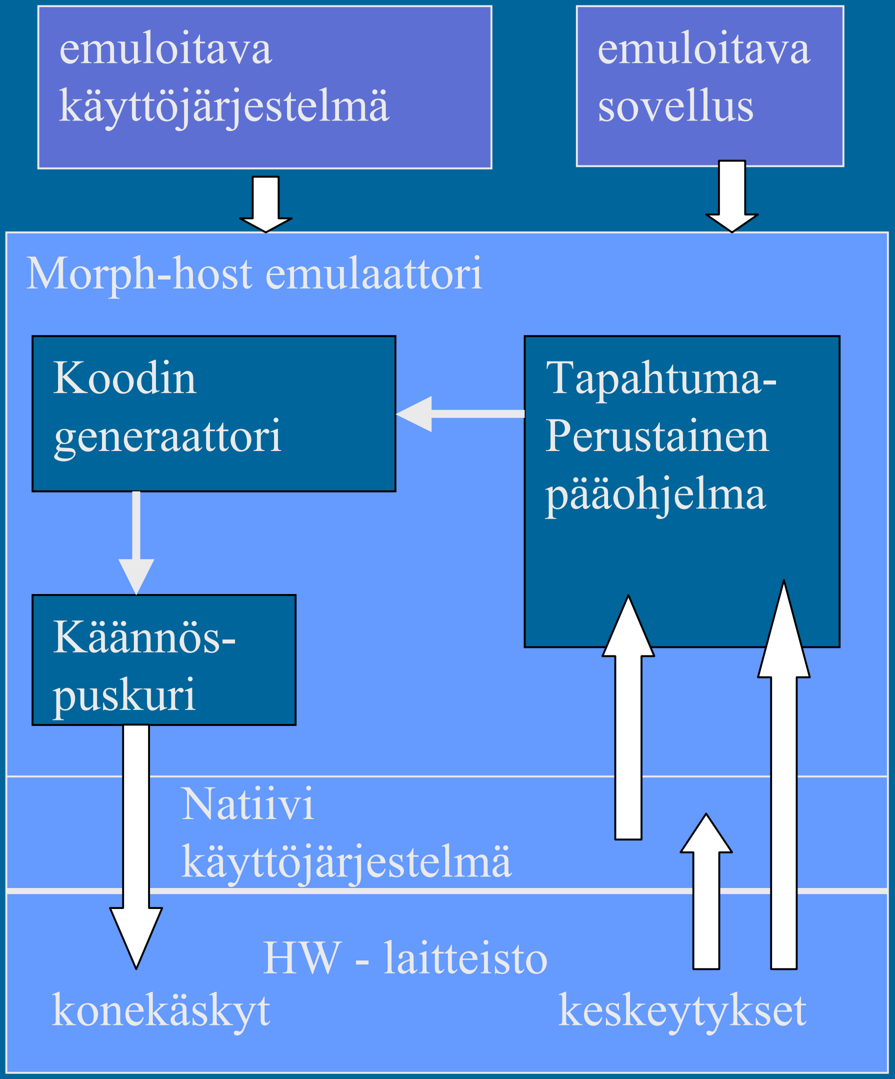
jos emuloitu epätarkka keskeytys, niin peruuta talletettuun tilaan, generoi hidas mutta täsmällinen emulointikoodi ja jatka.

jos emuloitu tarkka keskeytys, niin käsittele se ja jatka nopealla suorituksella (optimoidut käskysarjat)

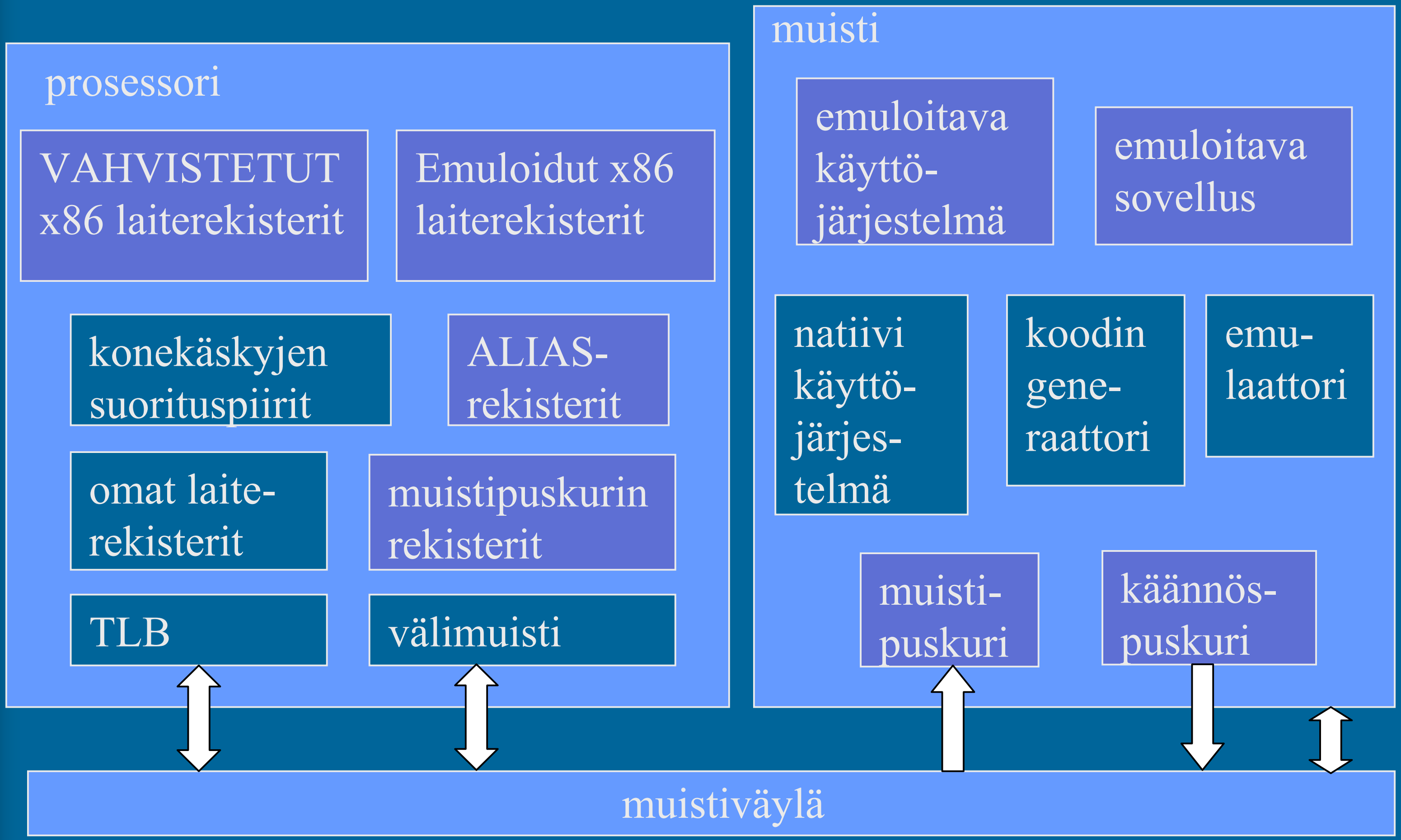
Transmetan emulaattoriohjelman suoritus



Transmetan prosessorin looginen rakenne

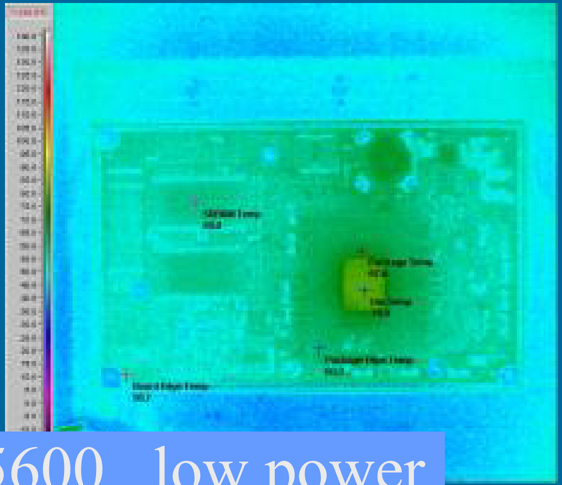


Transmetan prosessorin fyysinen rakenne



prosessori
VAHVISTETUT x86 laiterekisterit
Emuloidut x86 laiterekisterit
konekäskyjen suorituspiirit
ALIAS-rekisterit
omat laite-rekisterit
muistipuskurin rekisterit
TLB
välimuisti
muisti
emuloitava käyttö-järjestelmä
emuloitava sovellus
natiivi käyttö-järjes-telmä
koodin gene-raattori
emu-laattori
muisti-puskuri
käännös-puskuri
muistiväylä

What Will Transmeta do with Crusoe?



- Optimize for speed or size?
  - Small size ⇒ cheaper, less power
- Someone else will manufacture chips and compete with Intel?
  - IBM

TM3200, TM5400, …, TM5600 low power
TM5800 high speed

- Someone else will use the chips to make products to compete with Intel-based products
  - NEC, Fujitsu, Sony, …
- Midori Linux
  - Open Source project for delivering system software on small devices
    - Gateway, Hitachi, FIC

-- The End (again) --

”Aqua 3400 Portable Wireless Internet Access Device, Transmeta 400MHz, 8.4" TFT touch-screen”

”NEC Versa DayLite combines the power-saving 600 Mhz Crusoe TM5600 processor with dual battery systems that NEC claims will extend battery life to up to 7.5 hours on a single charge”