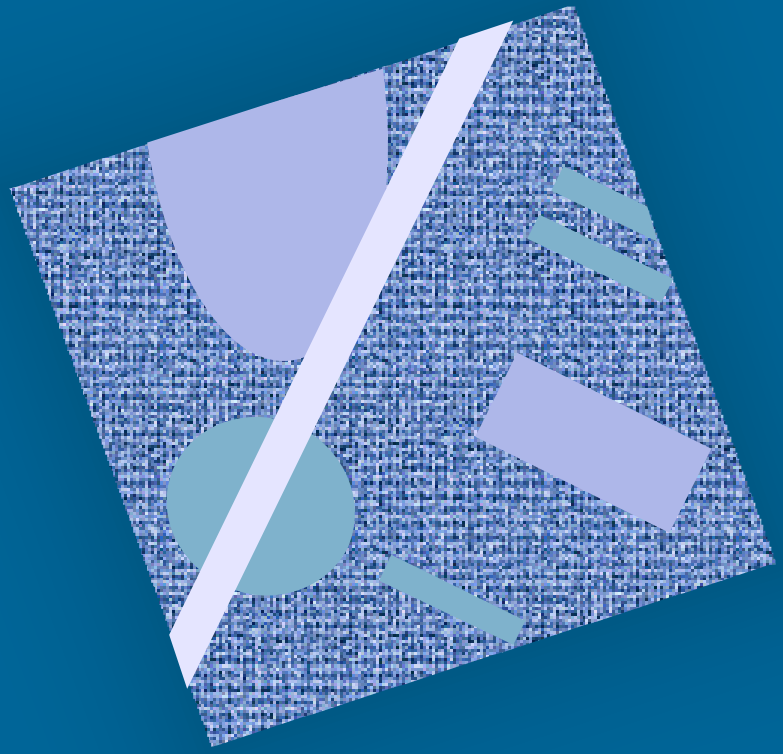
Transmeta Architecture



Major Ideas

General Architecture

Emulated Precise Exceptions

What to do with It

Copyright Teemu Kerola 2000

Background

- Transmeta Corporation
  - Paul Allen (Microsoft), George Soros (Soros Funds)
  - David R. Ditzel (Sun)
  - Edmund J. Kelly, Malcolm John Wing, Robert F. Cmelik (?)
  - Linus B. Torvalds, February 1997 → ...

CEO

- Patent 5958061
  - applied July 24, 1996
  - granted September 28, 1999
  - other patents …
- Crusoe prosessor
  - published January 19, 2000

Copyright Teemu Kerola 2000

Basic Idea

- Create a new processor which, when coupled with “morph host” emulator, can run Intel/Windows code faster than state-of-the-art Intel processor
- 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

Copyright Teemu Kerola 2000

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

Copyright Teemu Kerola 2000

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
  - processor does not need to solve these
- Optimize for speed only when needed
  - do not optimize for speed when exact state change is required
- Alias detection to assist keeping globals is registers

Copyright Teemu Kerola 2000

Major Emulation Ideas

- Target processor state kept in dedicated HW registers
  - working state, committed state
- Memory store buffer keeps uncommitted 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)

Copyright Teemu Kerola 2000

General Architecture

- VLIW implementation
  - 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?

Copyright Teemu Kerola 2000

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

Copyright Teemu Kerola 2000

General Architecture (contd)

- TLB
  - new features to solve new problems
    - earlier 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”)

Copyright Teemu Kerola 2000

General Architecture (contd)

- Target memory store buffer
  - implemented with special register 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 outside, not internally as is usual with speculative instructions

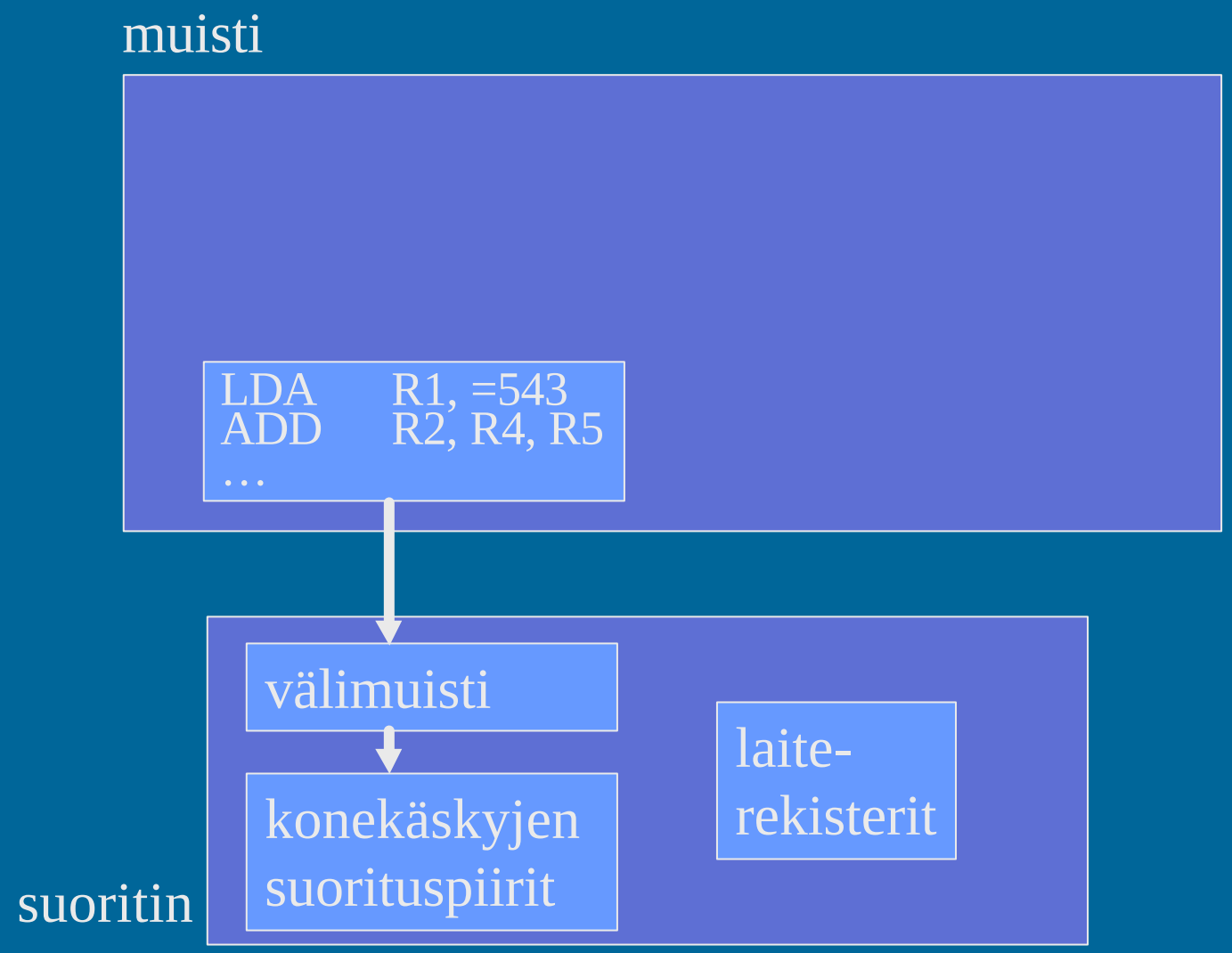
Copyright Teemu Kerola 2000

General Architecture (contd)

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

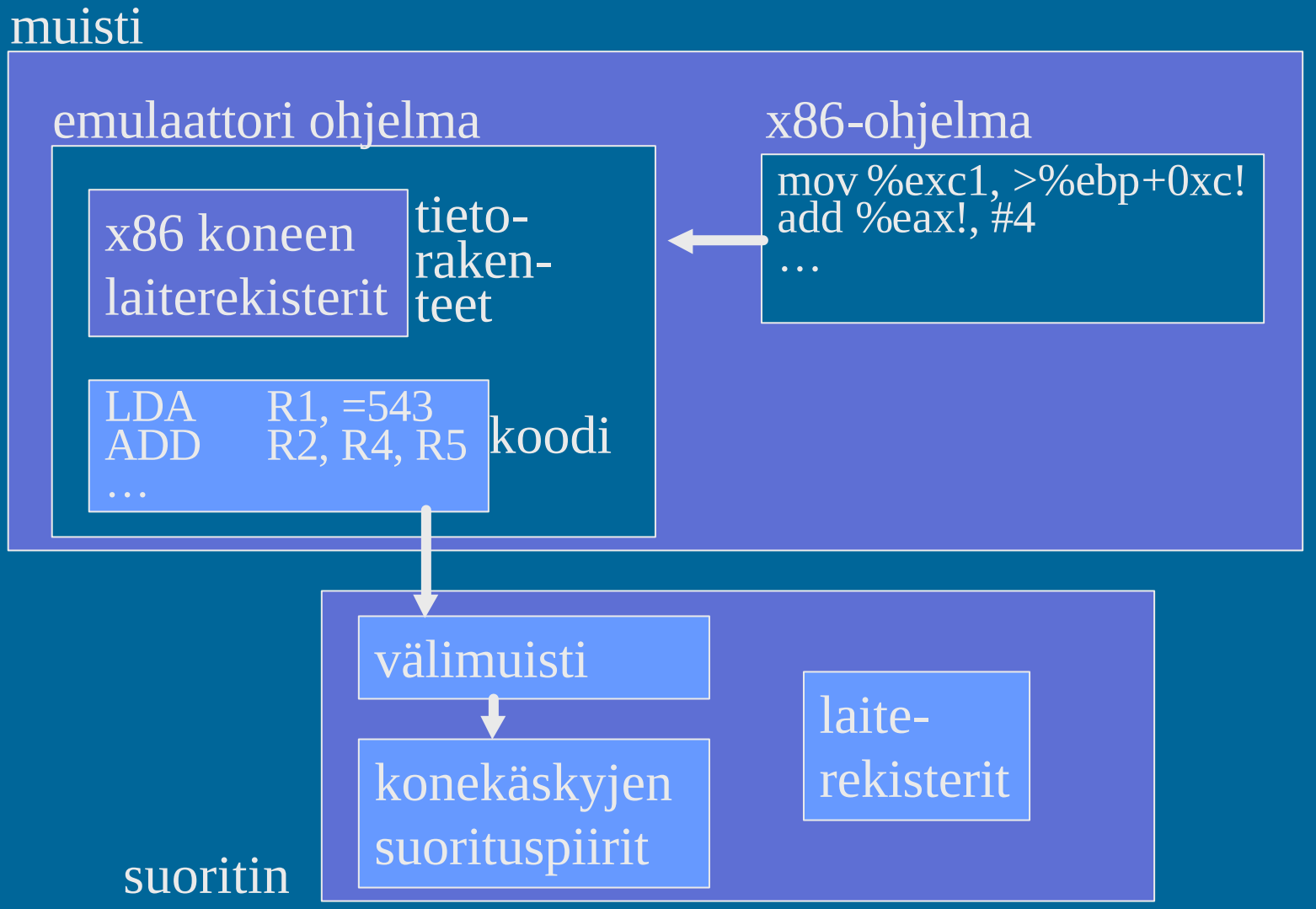
Copyright Teemu Kerola 2000

Tavallisen ohjelman suoritus



Copyright Teemu Kerola 2000

Emulaattoriohjelman suoritus



Copyright Teemu Kerola 2000

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
…
```


Copyright Teemu Kerola 2000

Transmetan emulaattoriohjelma

(x86 koneen emuloidut laiterekisterit laitteistossa)

Dynaamisesti generoidut (optimoitut) käskysarjat x86-koneen konekäskyjoukkoja varten

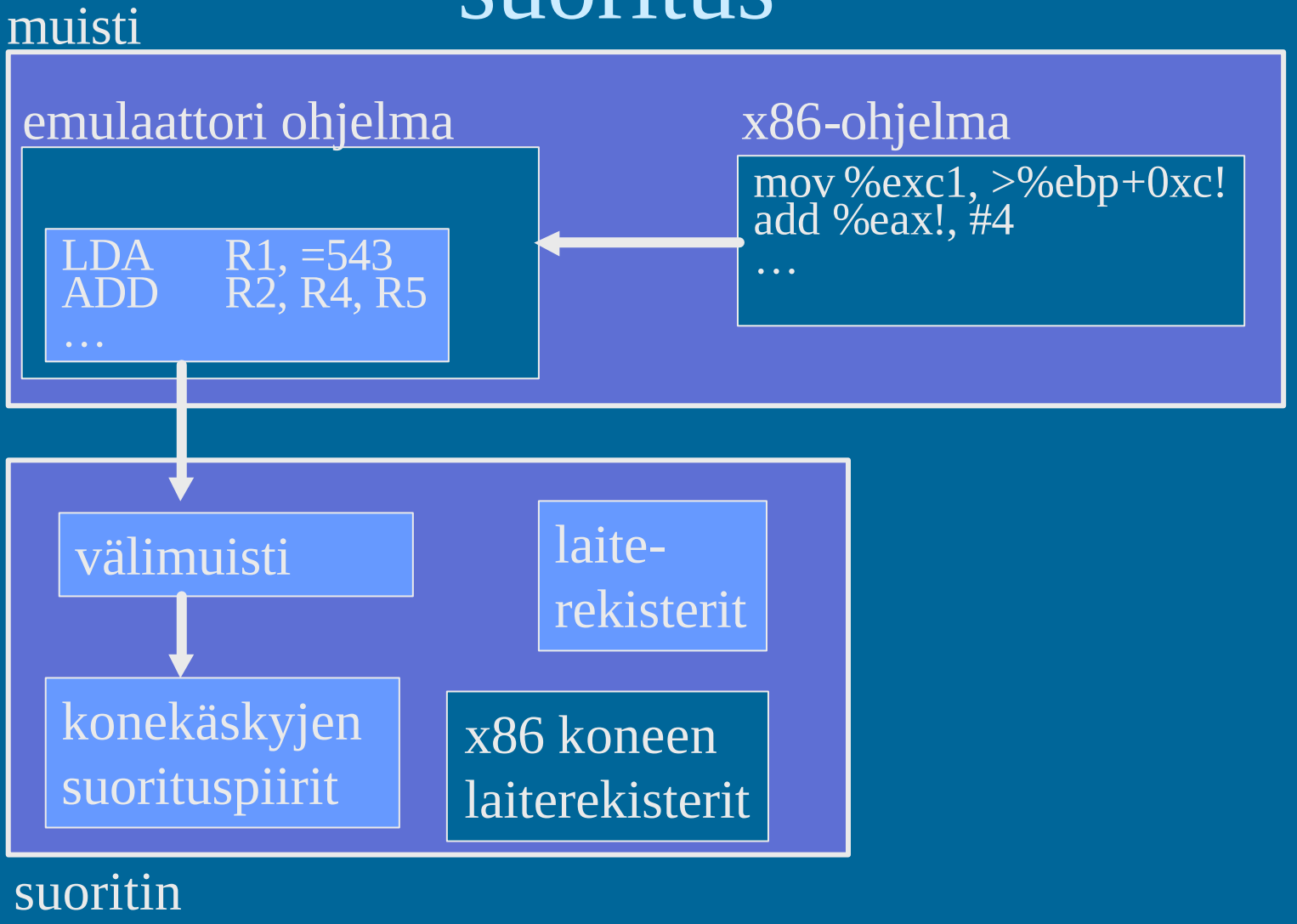
```
LDA     R1, =543
ADD     R2, R4, R5
…
```

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)

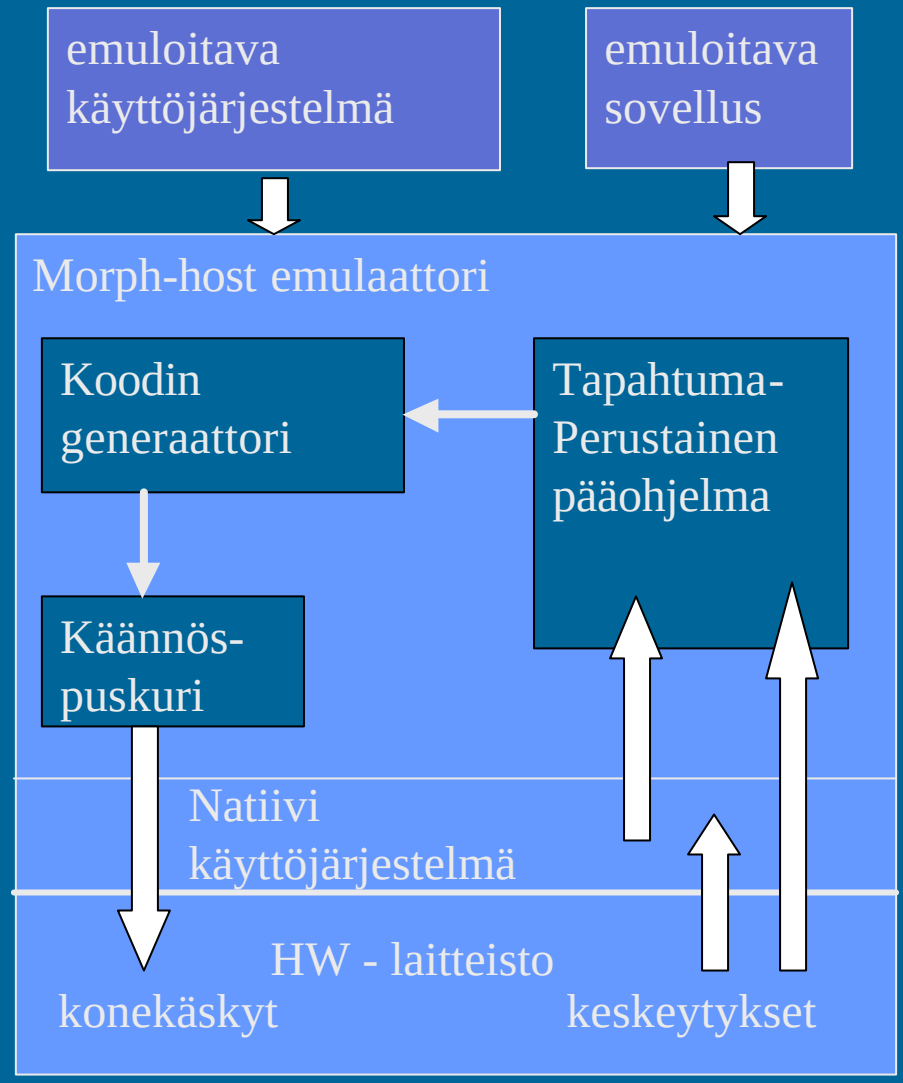
Copyright Teemu Kerola 2000

Transmetan emulaattoriohjelman suoritus



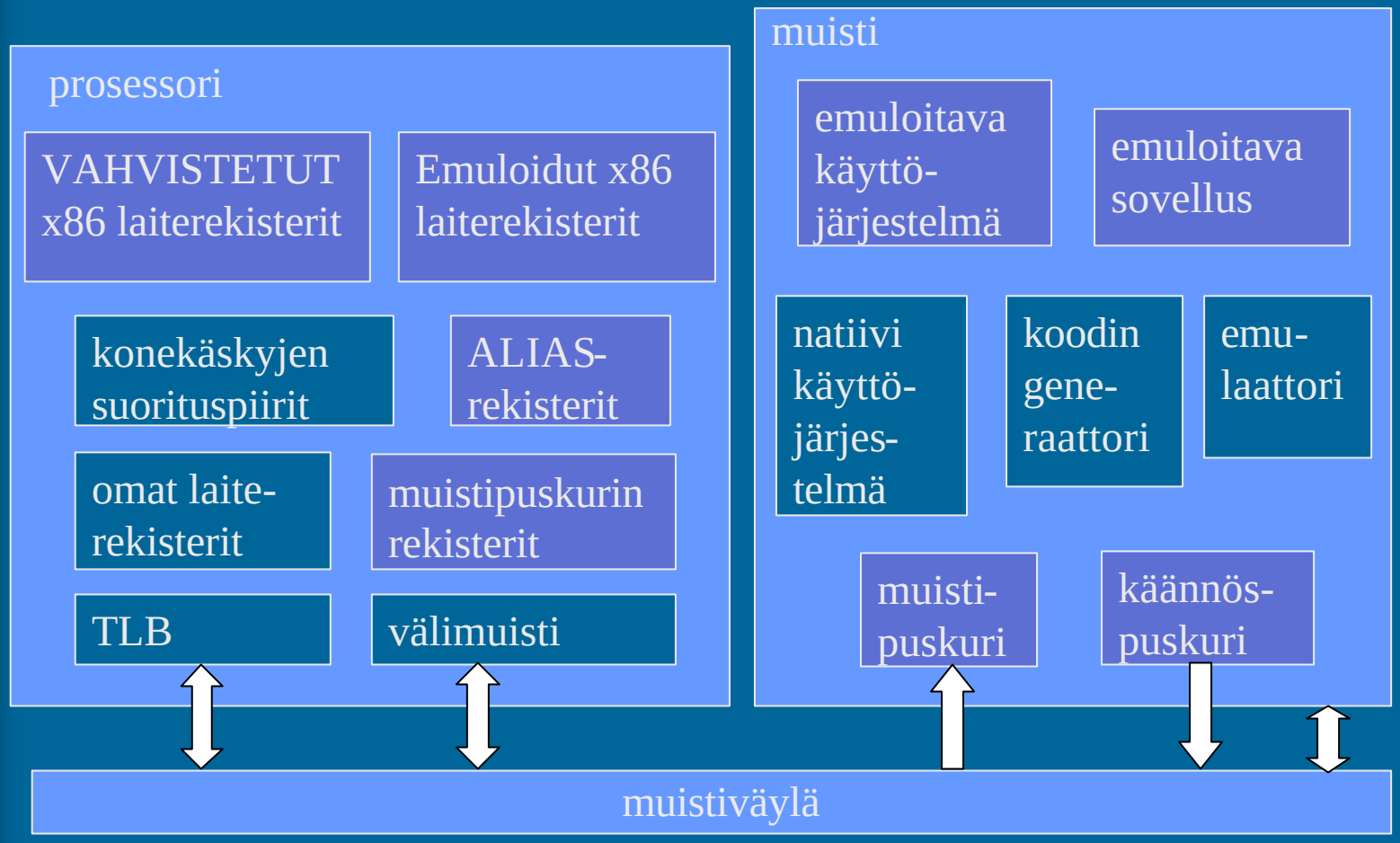
Copyright Teemu Kerola 2000

Transmetan prosessorin looginen rakenne



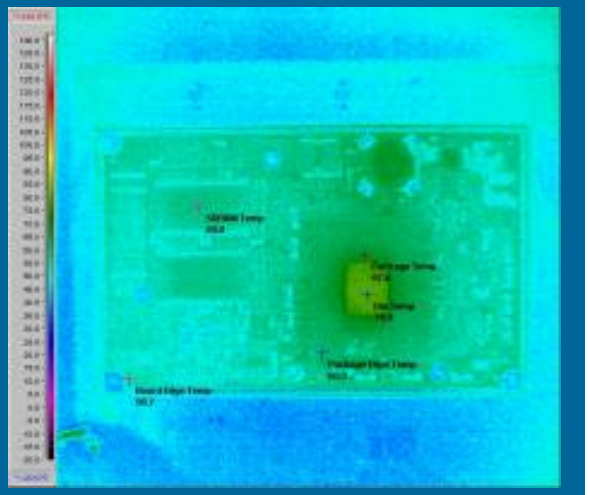
Copyright Teemu Kerola 2000

Transmetan prosessorin fyysinen rakenne



Copyright Teemu Kerola 2000

What Will Transmeta do with Crusoe?



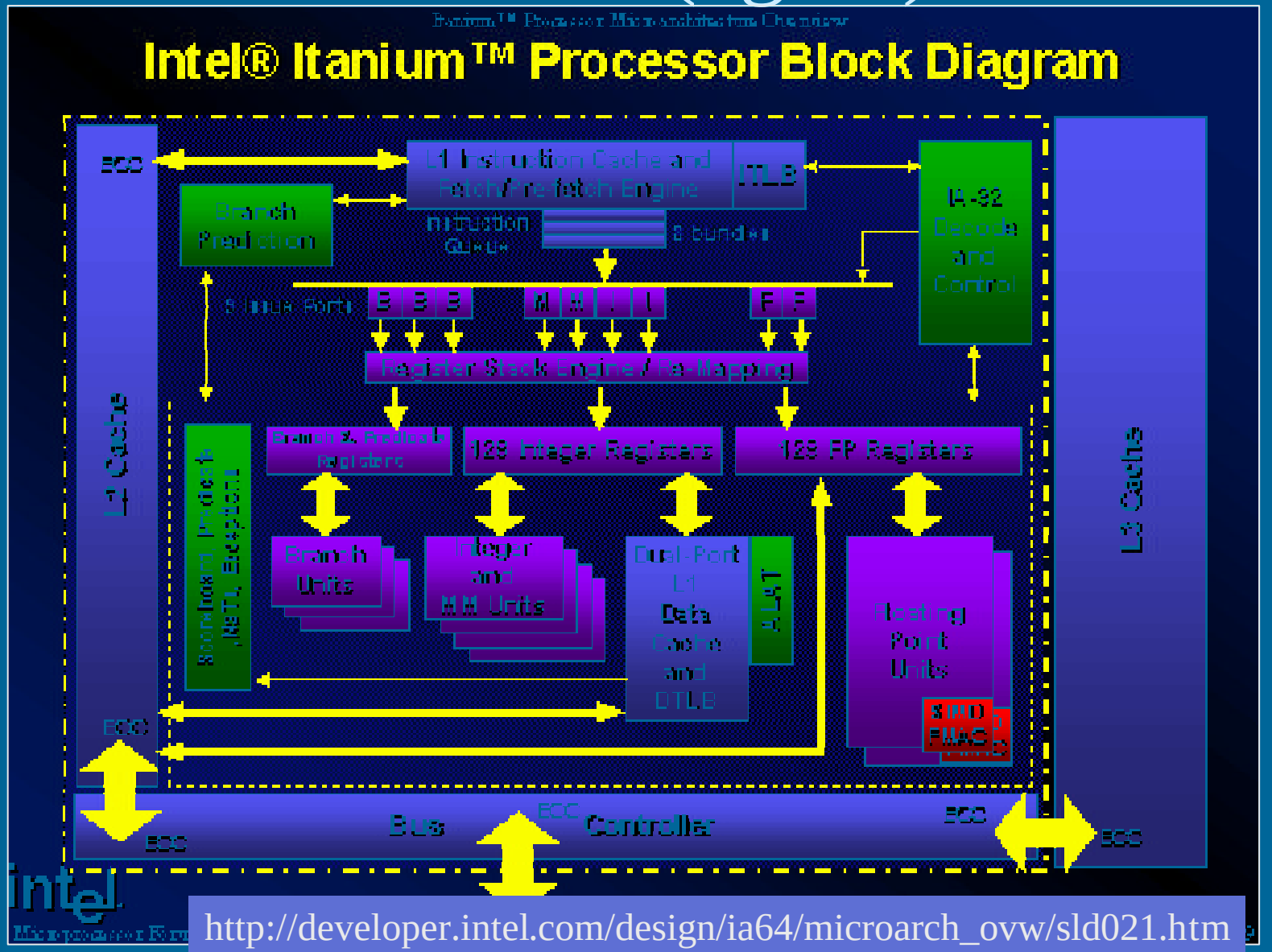
- Optimize for speed or size?
  - Small size ⇒ cheaper, less power
- License it to Intel?
- Have someone else manufacture it and compete with Intel?
  - IBM
  - Motorola, AMD?
- Make products based on Crusoe?
  - Internet window?
  - Visual phone?

TM3120, TM5400

low power

Copyright Teemu Kerola 2000

-- The End (again) --



http://developer.intel.com/design/ia64/microarch_ovw/sld021.htm

Copyright Teemu Kerola 2000