Advanced Computer Architecture

—

Part I: General Purpose Dynamic Binary Translation,...

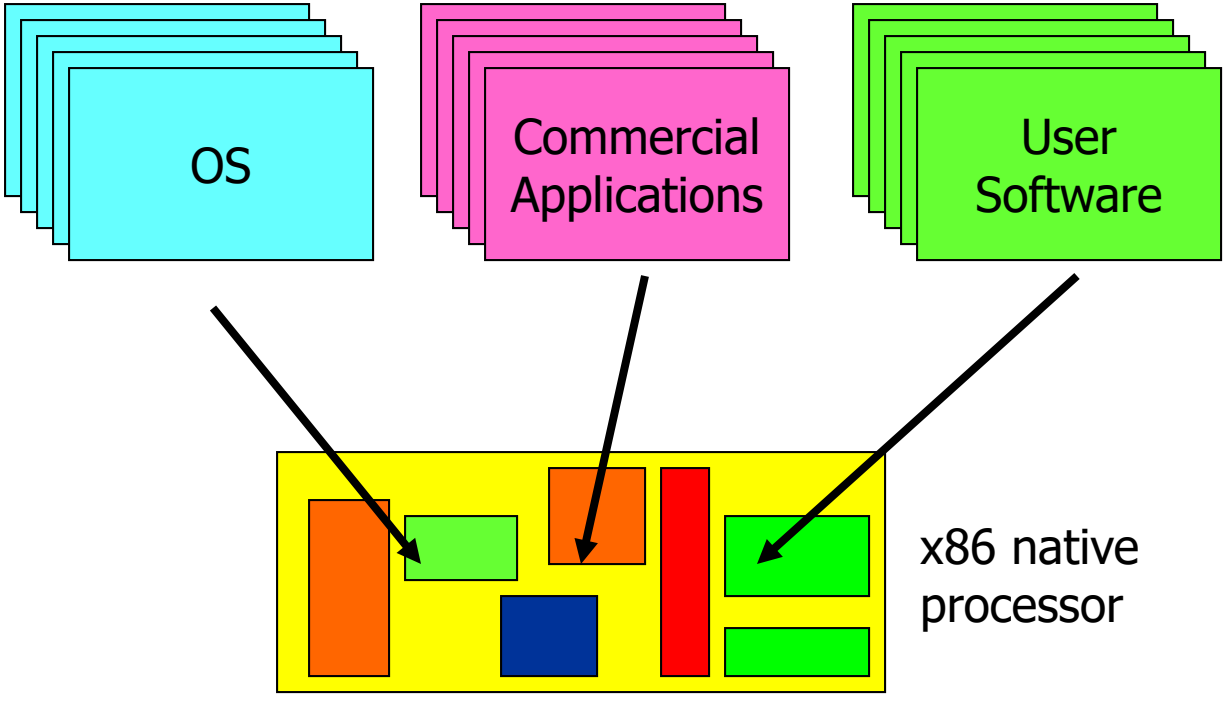
Paolo.Ienne@epfl.ch

EPFL – I&C – LAP

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Binary Compatibility

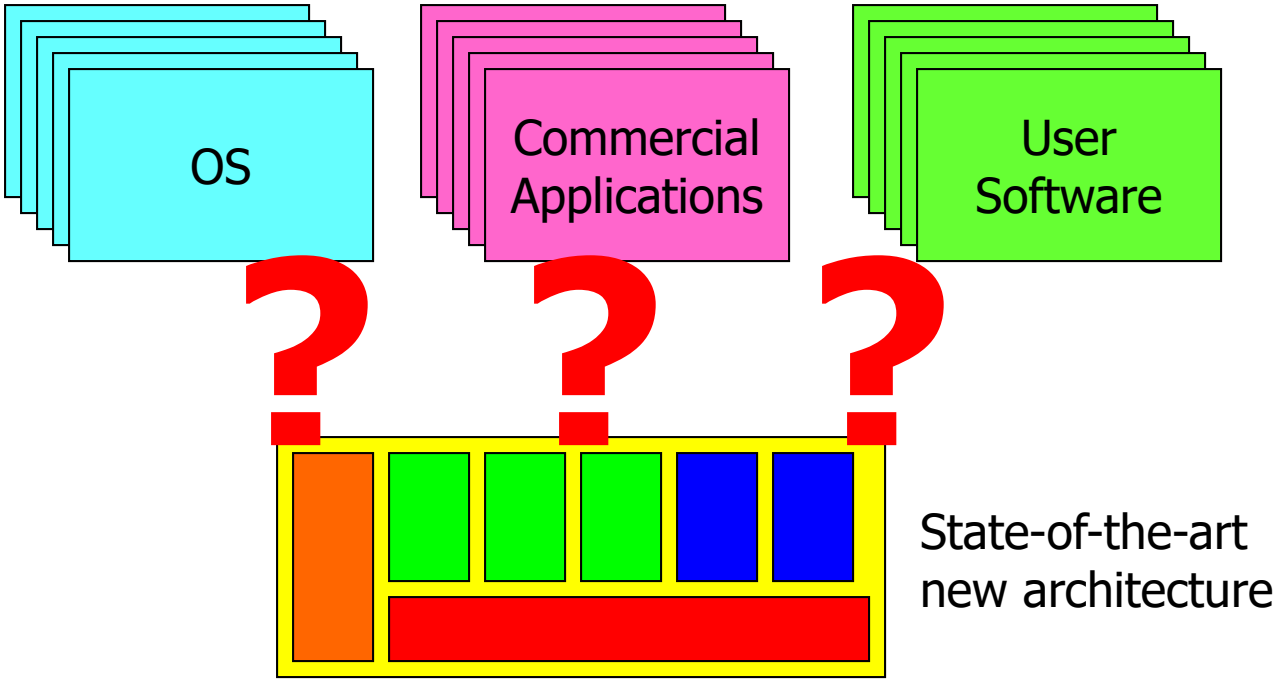
- Single worst obstacle to processor evolution



© Ienne 2003-22

Binary Compatibility

- New architectures cannot be introduced and the scope for enhancement is reduced considerably
- Non IA-32 architectures can ever be established?



© Ienne 2003-22

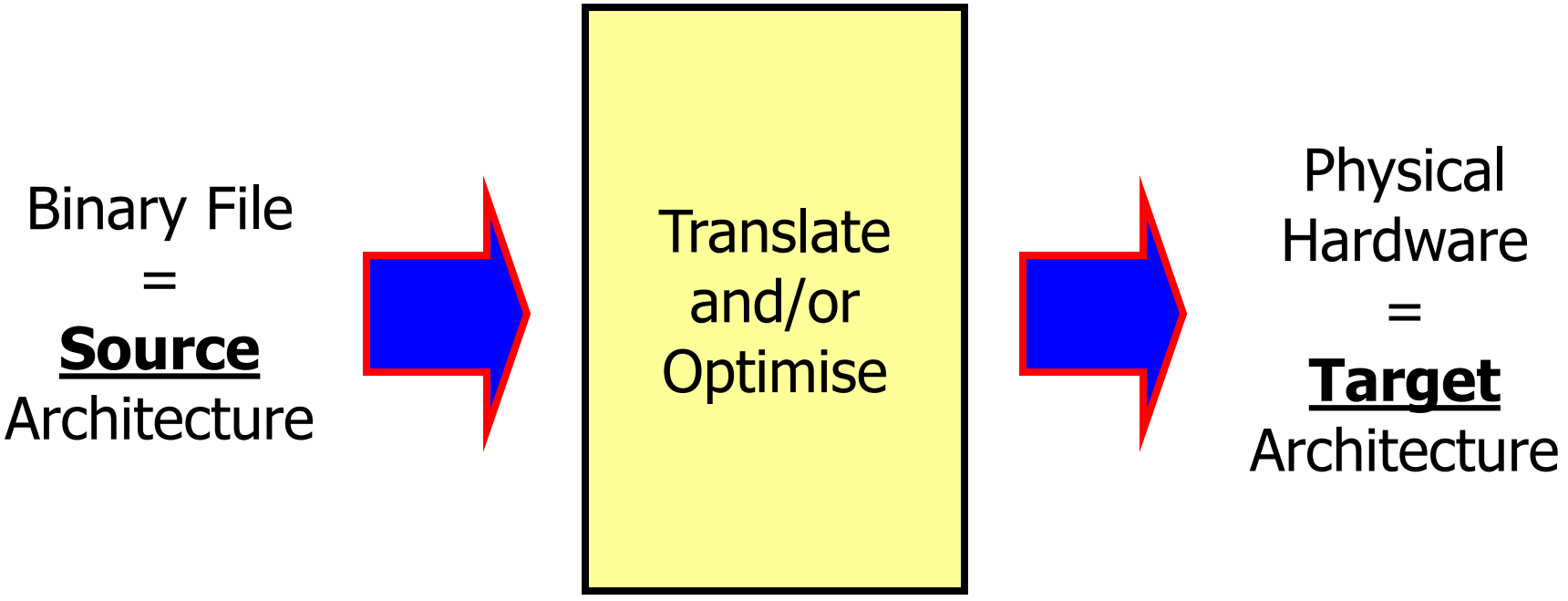
Summary

- Traditional approaches to circumvent (with very moderate success) binary compatibility issues
- Dynamic Binary Translation (DBT)
- Key difficulties, solutions, open problems
- Example applications
- Further work
- Conclusions

© Ienne 2003-22

Source and Target Architecture

- Translation



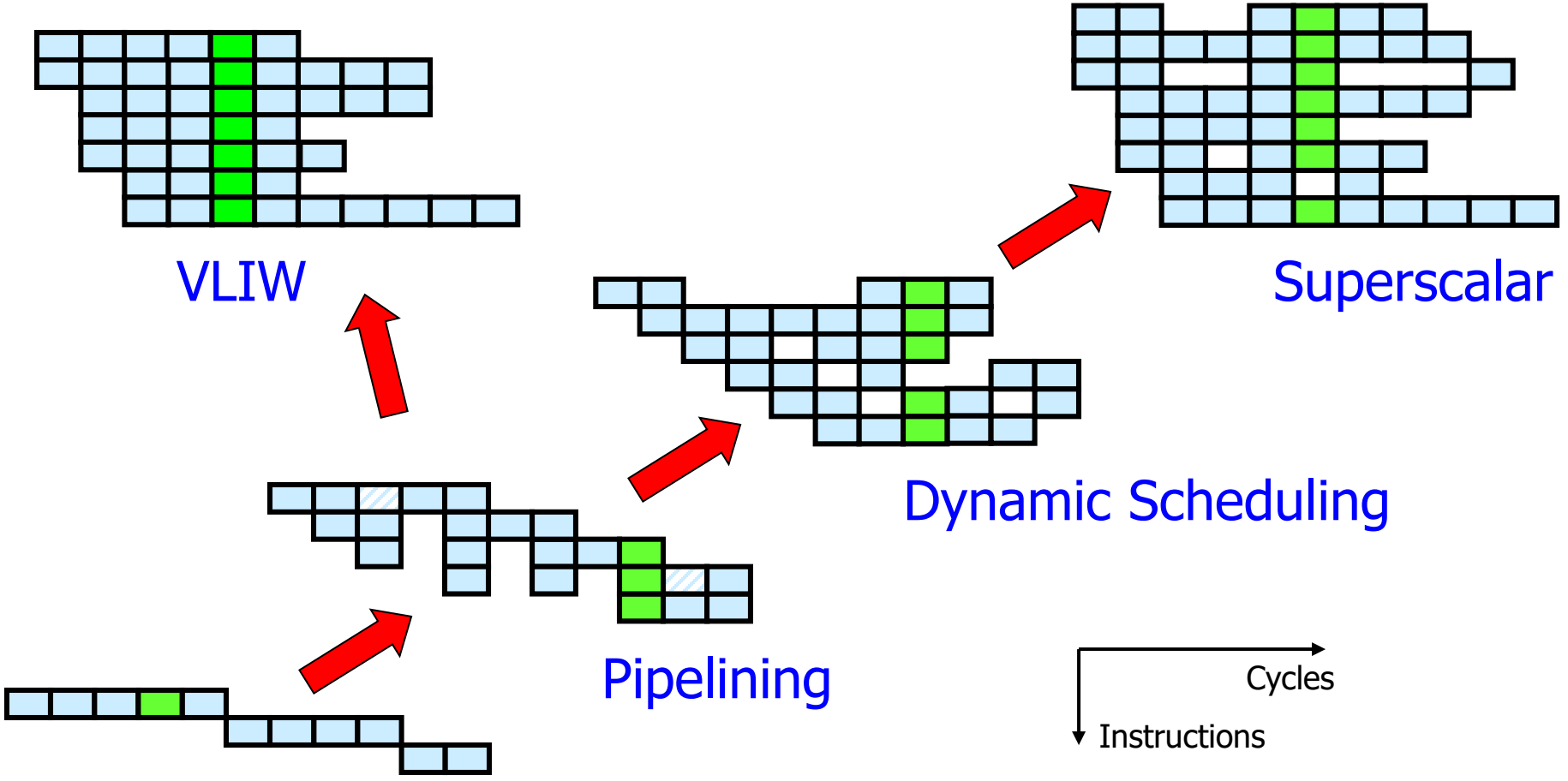
© Ienne 2003-22

Activities in DBT

- Transmeta's Crusoe
  - Commercial processor shipped in 2000
  - Source: IA-32 architecture (Pentium III)
  - Targets very low power, low cost markets
- IBM's Daisy
  - Research project started in 1996
  - Source: PowerPC architecture
  - 16-issue VLIW
- HP Labs' Dynamo
  - Source = Target: PA-RISC to PA-RISC translator (!)
- And many more…

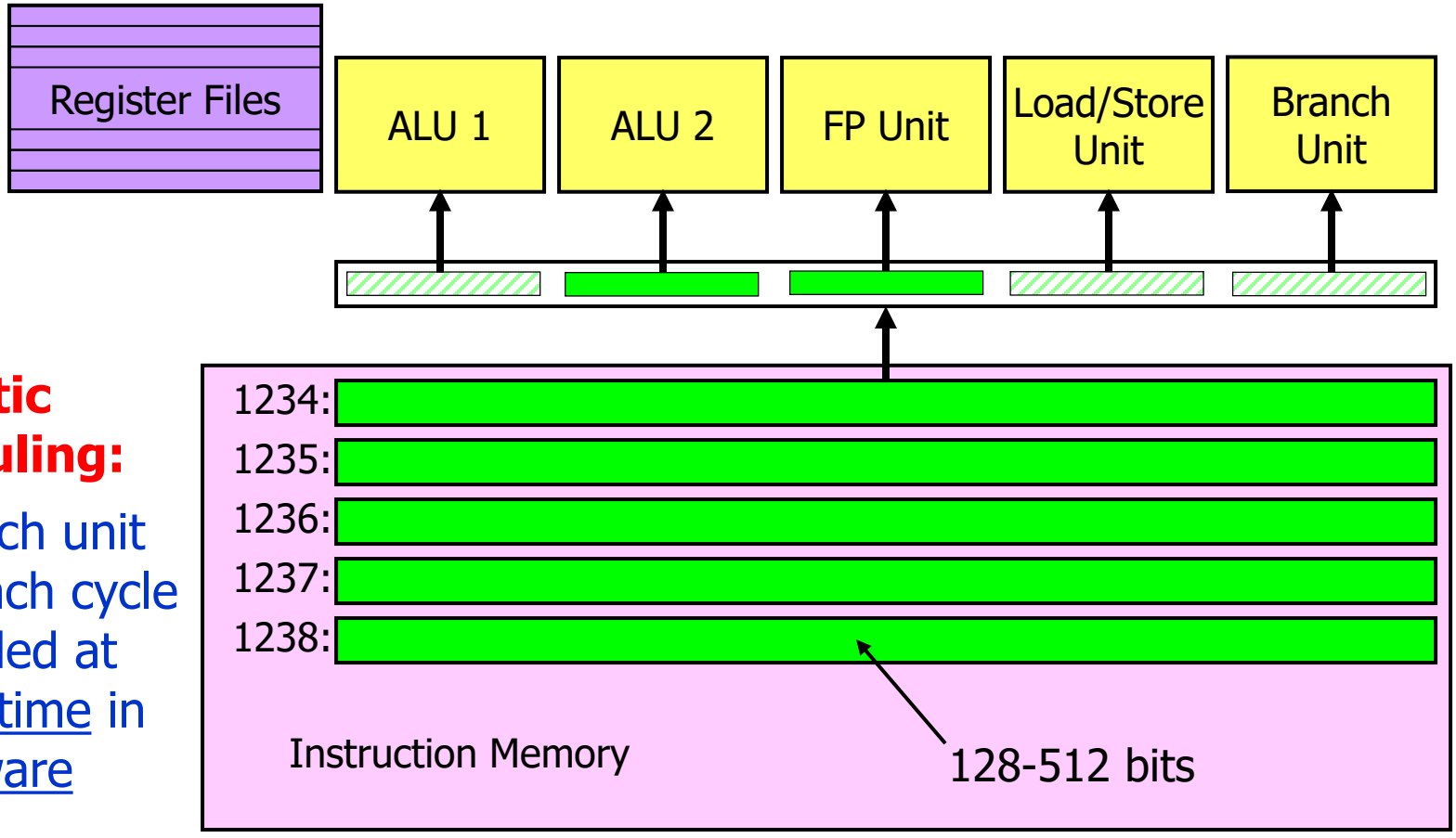
© Ienne 2003-22

Two Ways to Aggressive ILP



© Ienne 2003-22

(Statically Scheduled) Very Long Instruction Word Processor

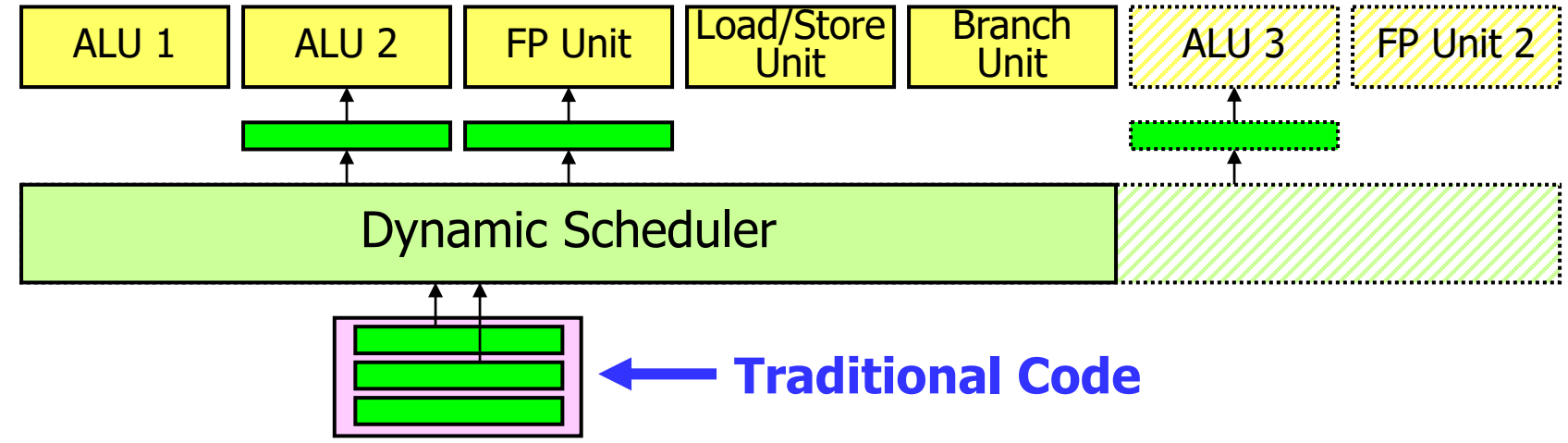


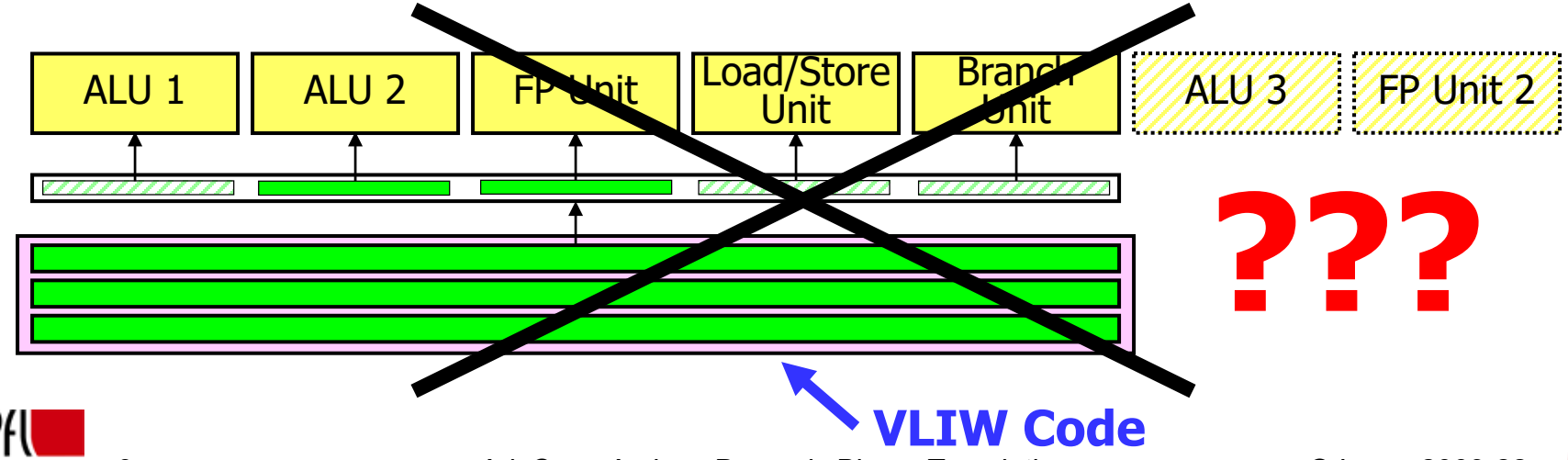
Static Scheduling:

What each unit does in each cycle is decided at compile time in software

© Ienne 2003-22

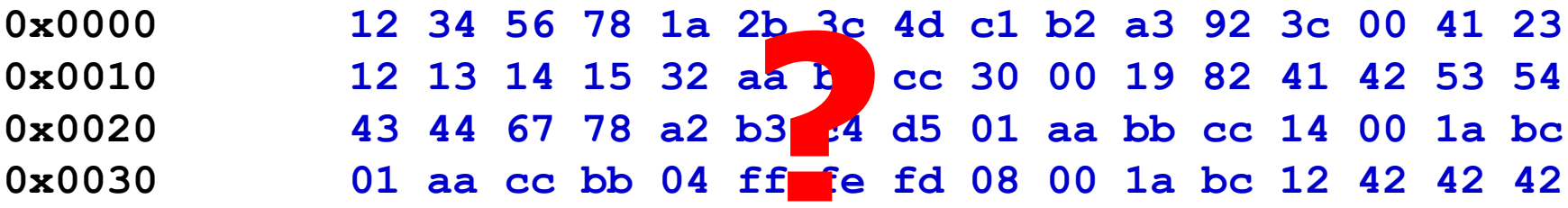
VLIW Binary Is Incompatible with More Aggressive Implementations





© Ienne 2003-22

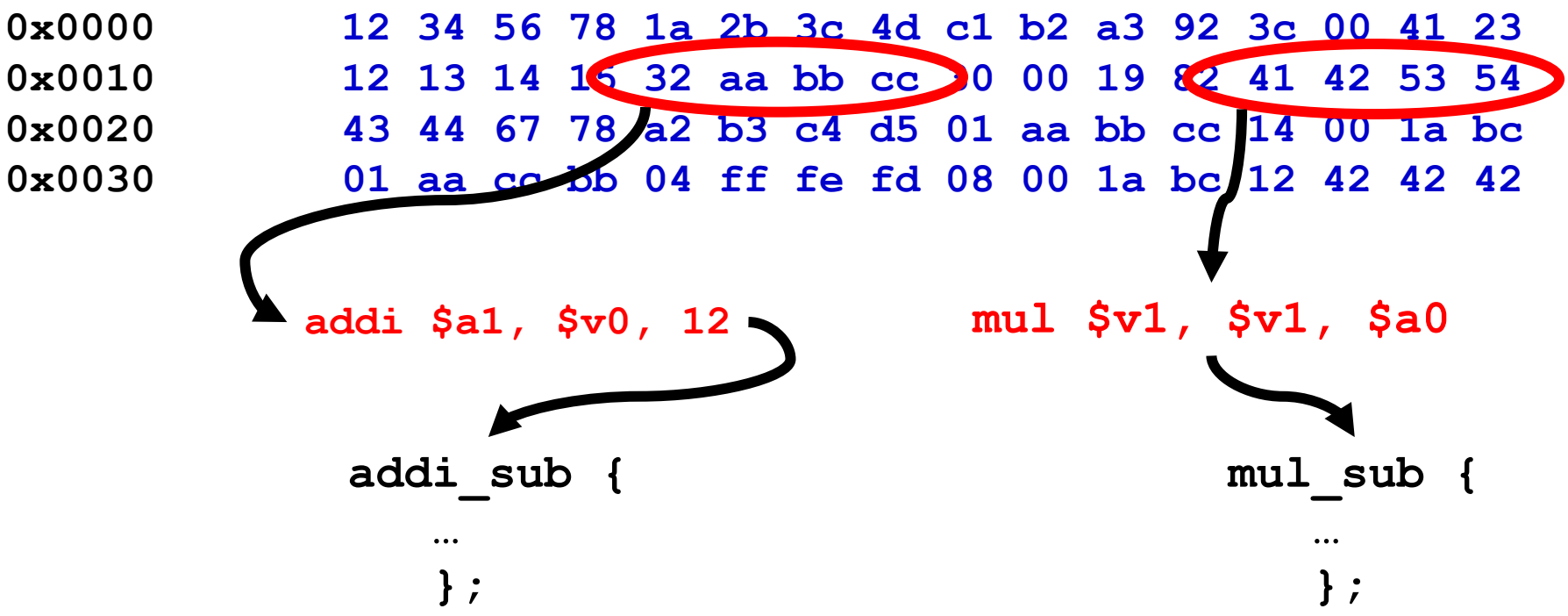
Emulation or Static Translation



- **Emulation**: Instruction by instruction simulation of the source architecture
- **Static Translation**: Conversion from machine code of the source architecture to the machine code of the target architecture

© Ienne 2003-22

Emulation



Performance cannot be but poor…

© Ienne 2003-22

Static Translation

```
0x0000    12 34 56 78 1a 2b 3c 4d c1 b2 a3 92 3c 00 41 23
0x0010    12 13 14 15 32 aa bb cc 00 00 19 82 41 42 53 54
0x0020    43 44 67 78 a2 b3 c4 d5 01 aa bb cc 14 00 1a bc
0x0030    01 aa cc bb 04 ff fe fd 08 00 1a bc 12 42 42 42
```

`add $a1, $a1, $v1`	`ADD BX, CX`
`addi $a1, $v0, 12`	`ADD BX, 12`
`mul $v1, $v1, $a0`	`MUL EX, DX`
`jal label`	`CALL label`

```
0x0000    30 00 19 82 41 04 ff fe fd 08 78 a2 b3 53 54 c1
0x0010    12 13 14 15 32 aa bb cc 12 34 56 78 1a b2 a3 92
0x0020    2b 3c 00 41 23 3c 4d c4 d5 01 aa bb cc 00 1a bc
0x0030    01 00 1a bc 12 42 42 42 42 43 44 67 aa 14 cc bb
```


© Ienne 2003-22

Difficulties of Static Translation

- **Code identification**: all code must be discovered statically and separated from embedded data
- **Self-modifying code**: what to do with it? Additional hardware to allow support of source architecture?
- **Precise Exceptions**: no 1-to-1 relation between target instructions and source ones
- **OS**: Support of shared libraries and system calls

Never a 100% solution!

© Ienne 2003-22

Dynamic Binary Translation

- The basic idea of a hybrid approach:

Merge emulation and translation to get the best of both worlds

(and finally get much more…)

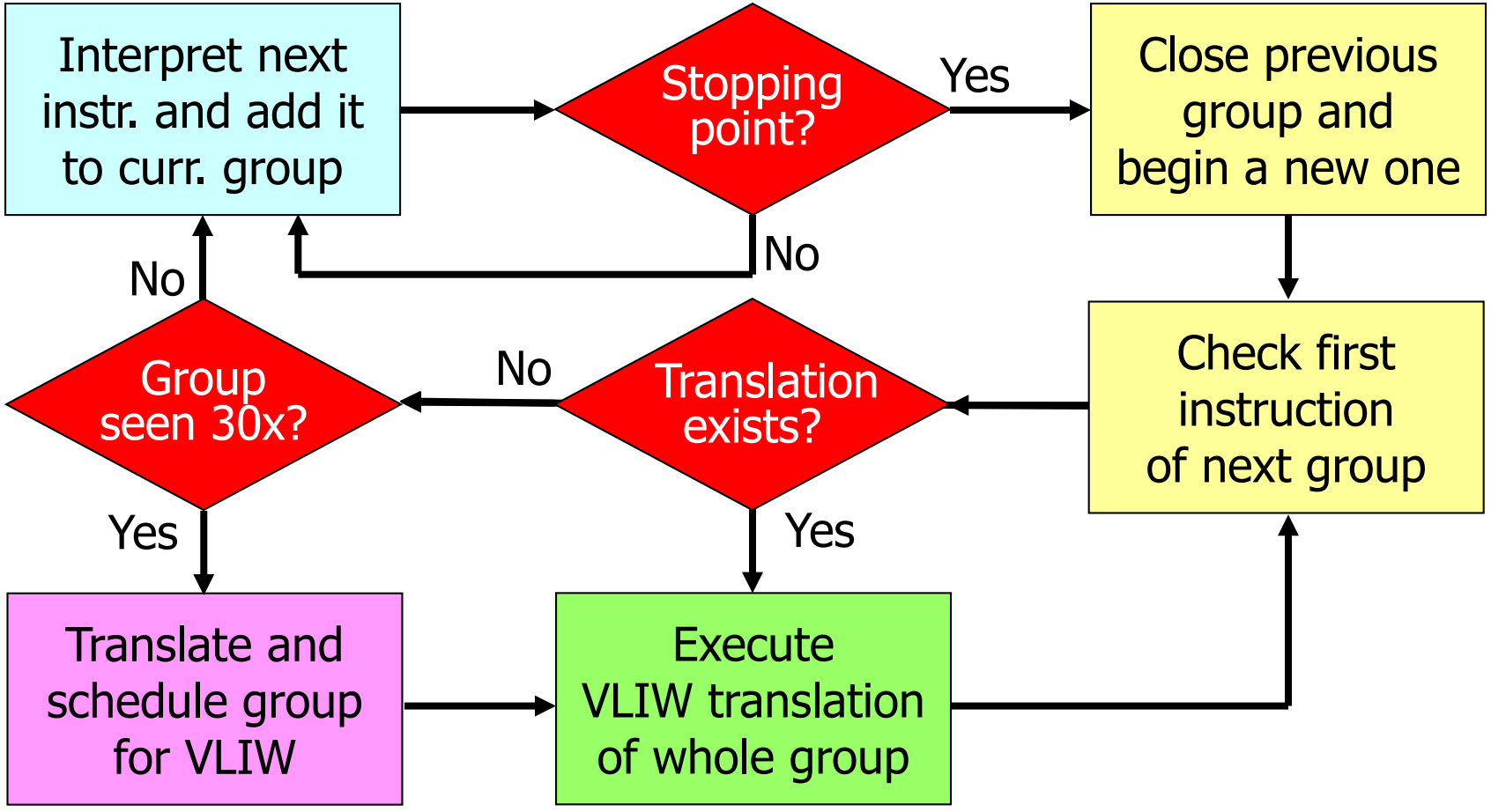
© Ienne 2003-22

Dynamic Binary Translation

- How to put the idea in practice?
  - Start by emulate everything
    - Explore code
    - Profile (control flow, data, etc.)
  - Translate and optimise code reused frequently
    - Optimise when it is worth spending the effort
  - Use translation when available
    - Run efficiently important code

© Ienne 2003-22

Dynamic Binary Translation Typical Execution Flow



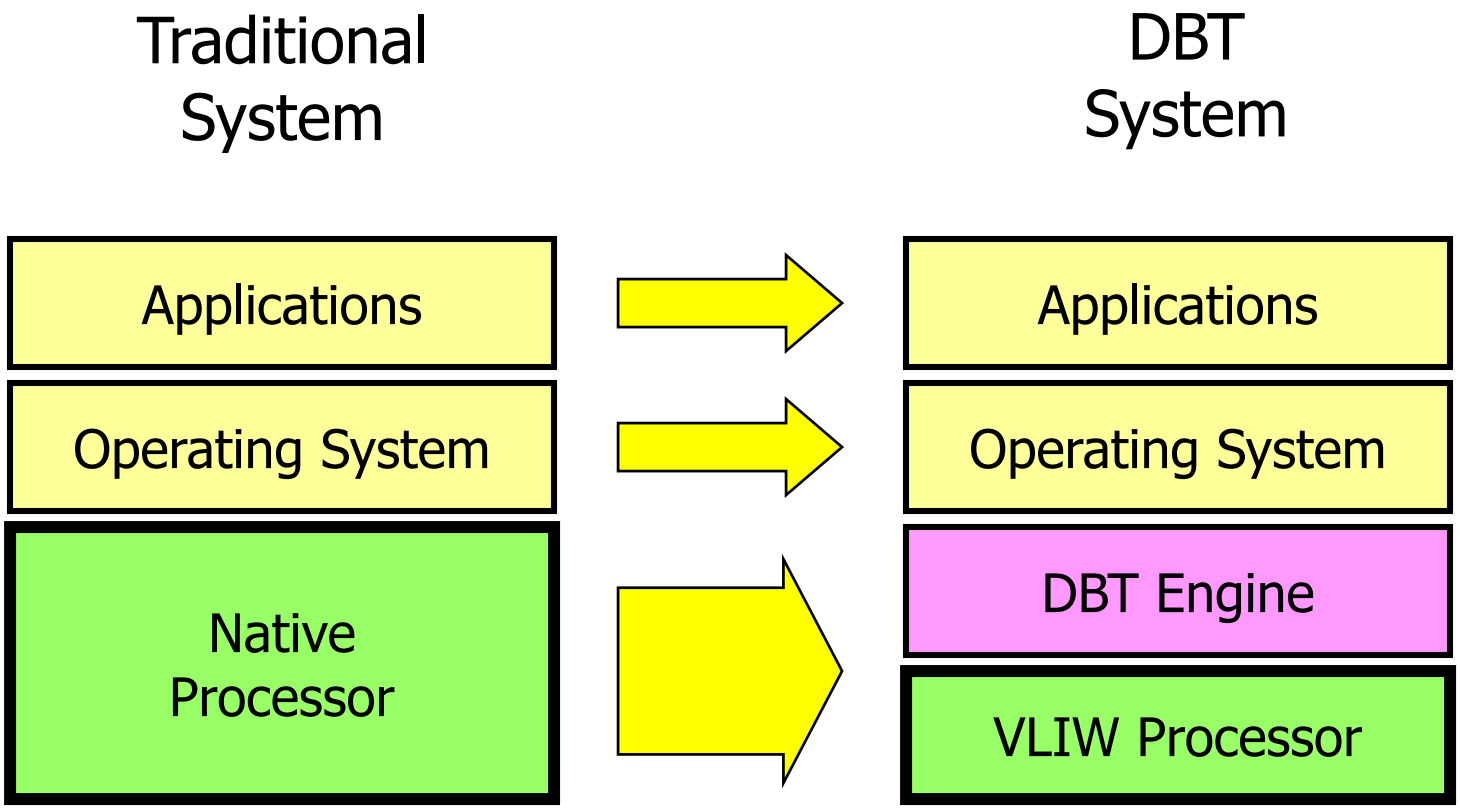
© Ienne 2003-22

Typical Optimisations to Translated Code

- ILP scheduling (data and control speculation)
- Loop unrolling
- Alias analysis
- Load-store telescoping
- Copy propagation
- Combining
- Unification
- Limited dead-code elimination

© Ienne 2003-22

DBT Hardware/Software Interface



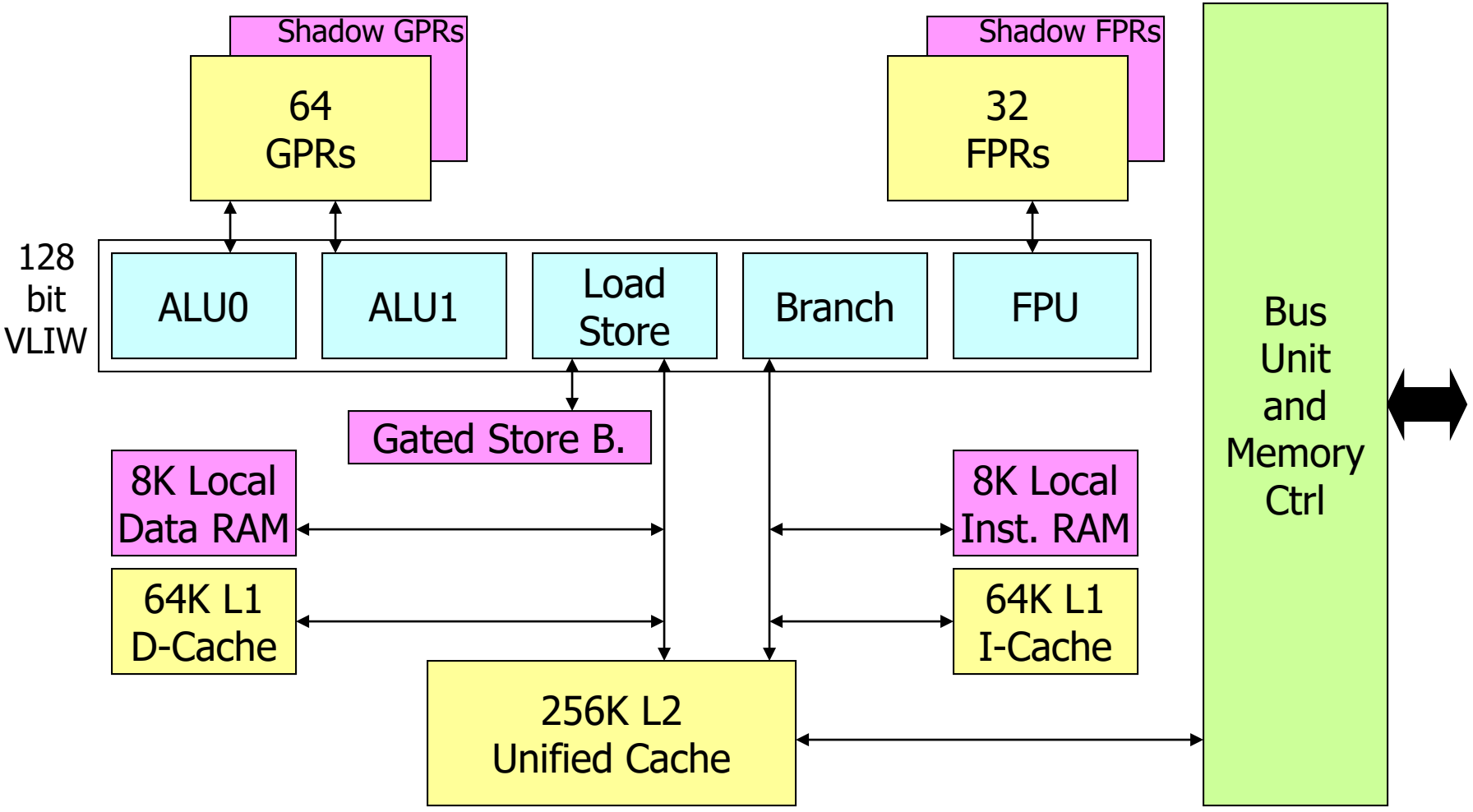
© Ienne 2003-22

DBT Engine

- **Additional layer of software** which also takes over the hardware scheduler of superscalar processors
- Many names: Virtual Machine Monitor (Daisy), Code Morphing Software (Crusoe), etc.

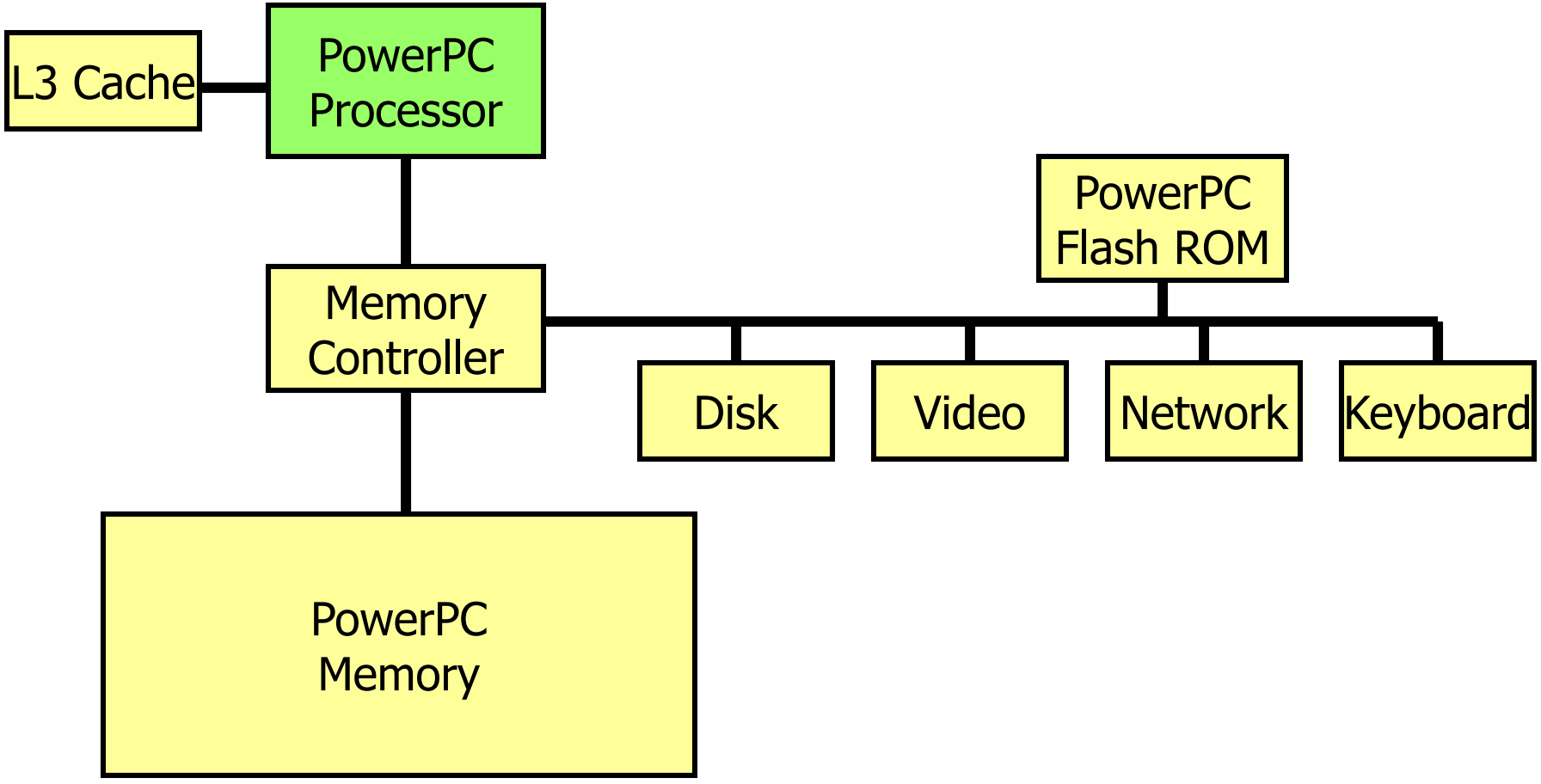
© Ienne 2003-22

Example of DBT Processor Transmeta Crusoe TM5400



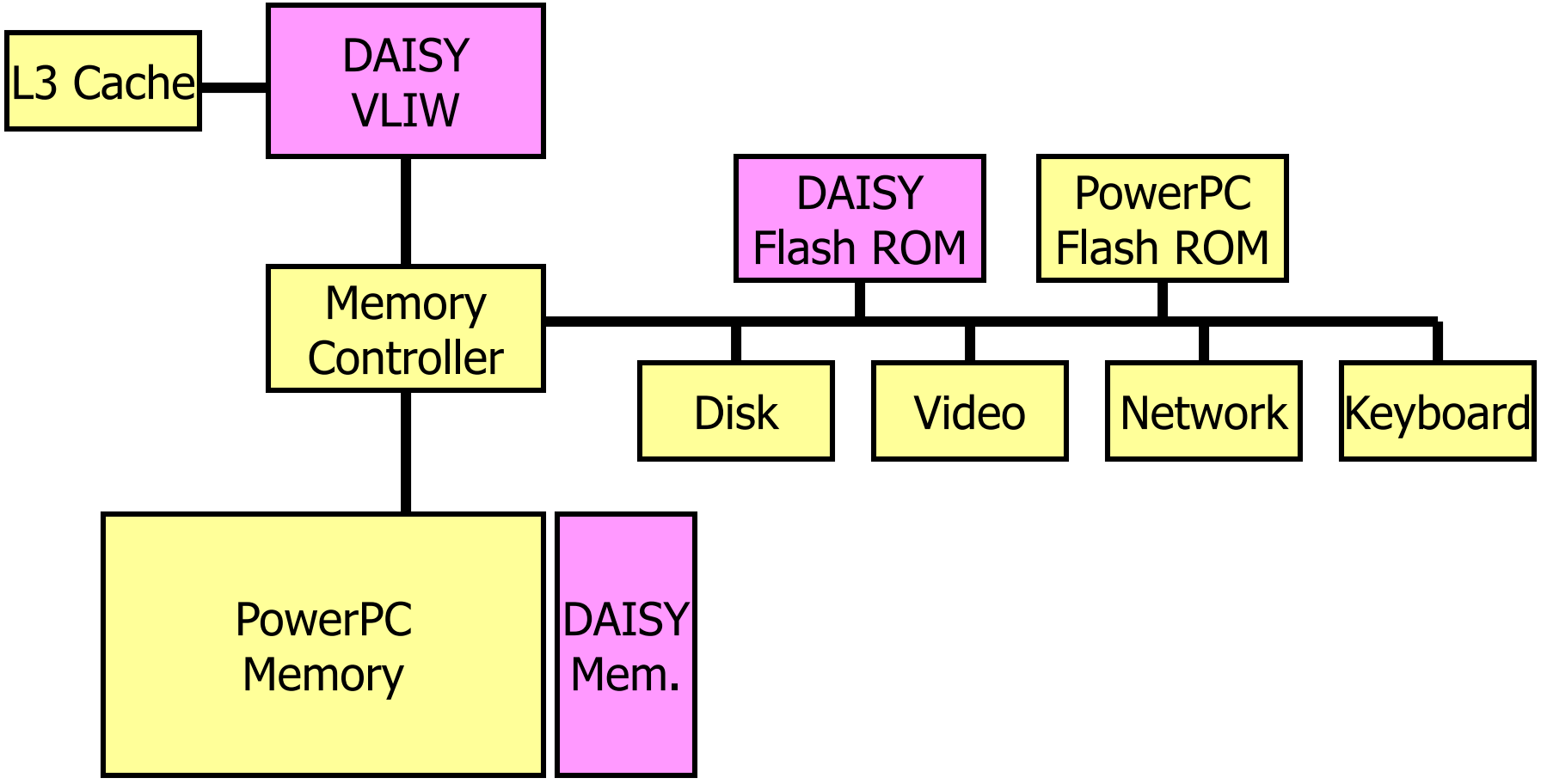
© Ienne 2003-22

Traditional System



© Ienne 2003-22

DBT System (Daisy)



© Ienne 2003-22

Translation Cache

- Keep translated pages in memory for later **reuse**
- Not a real cache: can be a translation buffer in main memory
- **Essential** to leverage the high cost of translation and of good optimisations
- Trade-off: Cost of memory vs. higher reuse
- Research topic: Best allocation policies?
- Invalidate translated pages when modification in the corresponding source page

© Ienne 2003-22

Difficult Problems for DBT

- Self-modifying code
- Precise exceptions
- Address translations, aliasing
- Self-referential code
- Management of translation cache
- Real-time code
- Boot code

© Ienne 2003-22

Example of Problems: DBT and Exceptions

- Asynchronous exceptions
  - **Can be delayed**, no big deal
  - Wait until end of group
  - Translate exception handler
  - Invoke translated exception handler

© Ienne 2003-22

Example of Problems: DBT and Exceptions

- Synchronous exceptions (e.g., Crusoe)
  - During **emulation**, no issue
  - If **synchronous** exception during the execution of a **translated** and optimised group of VLIW instructions, unclear instruction and state w.r.t. source architecture (speculative, out-of-order, etc.)
  - **Revert status** to beginning of current translated group
  - **Re-emulate** source architecture to find the exact point of the exception and to leave the processor in the architecturally correct state
  - **Invoke** translated exception handler

© Ienne 2003-22

Example of Problems: DBT and Exceptions

- Not so simple…
- Reverting status needs some architectural support (Crusoe)
  - Set of **shadow registers** which get the value of the main registers at the end of a group
  - Gated **store buffer** which holds pending stores for commit at the end of a group
- Side advantages in optimisation potential

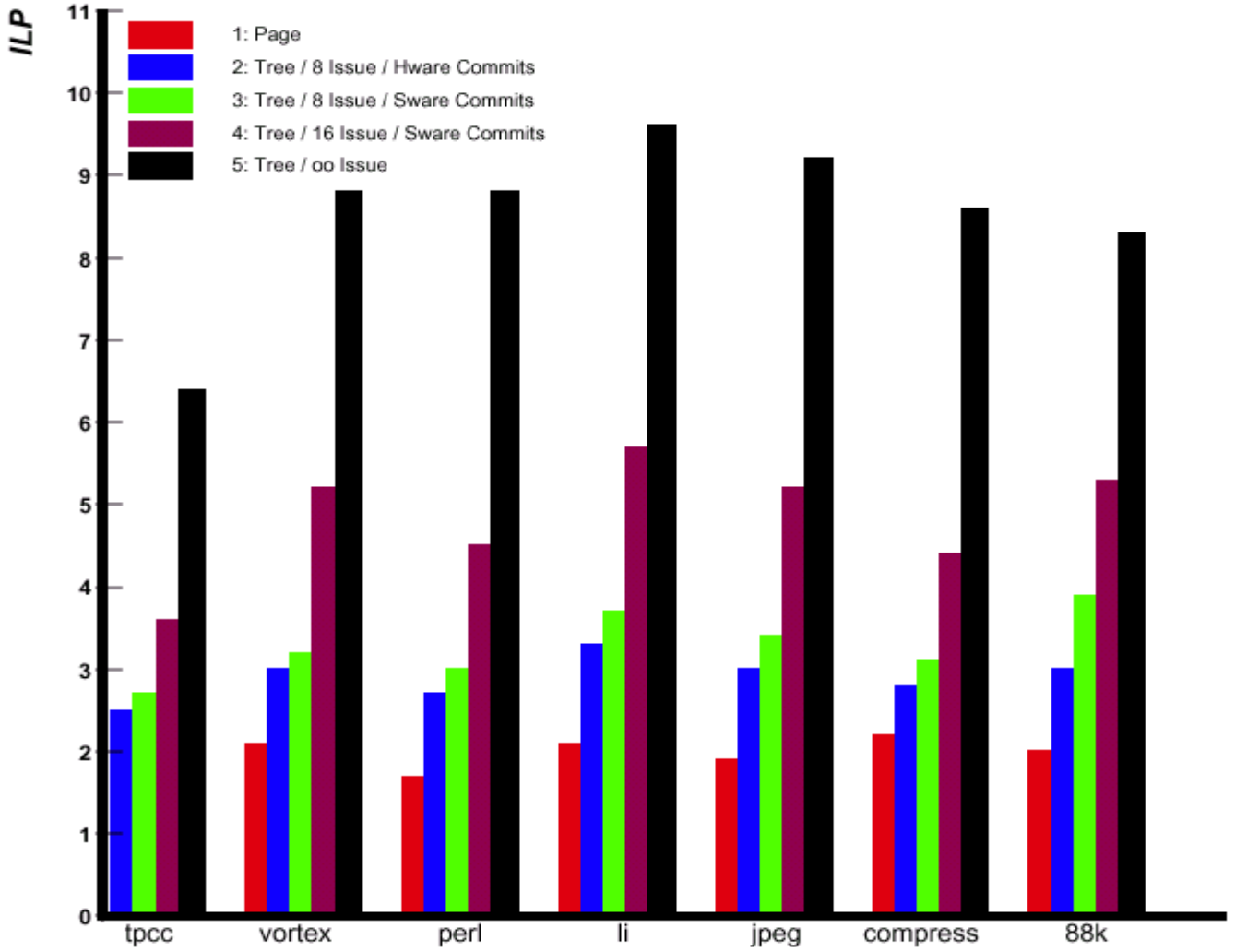
© Ienne 2003-22

Does it Really Work? Some Performance Figures

- Daisy achieves ILP up to 3-4 instructions per cycle
- Transmeta T5400 at 667 MHz is about the same of Pentium III at 500 MHz (Doug Laird, Transmeta VP)
- Dynamo improves execution time up to 22%

© Ienne 2003-22

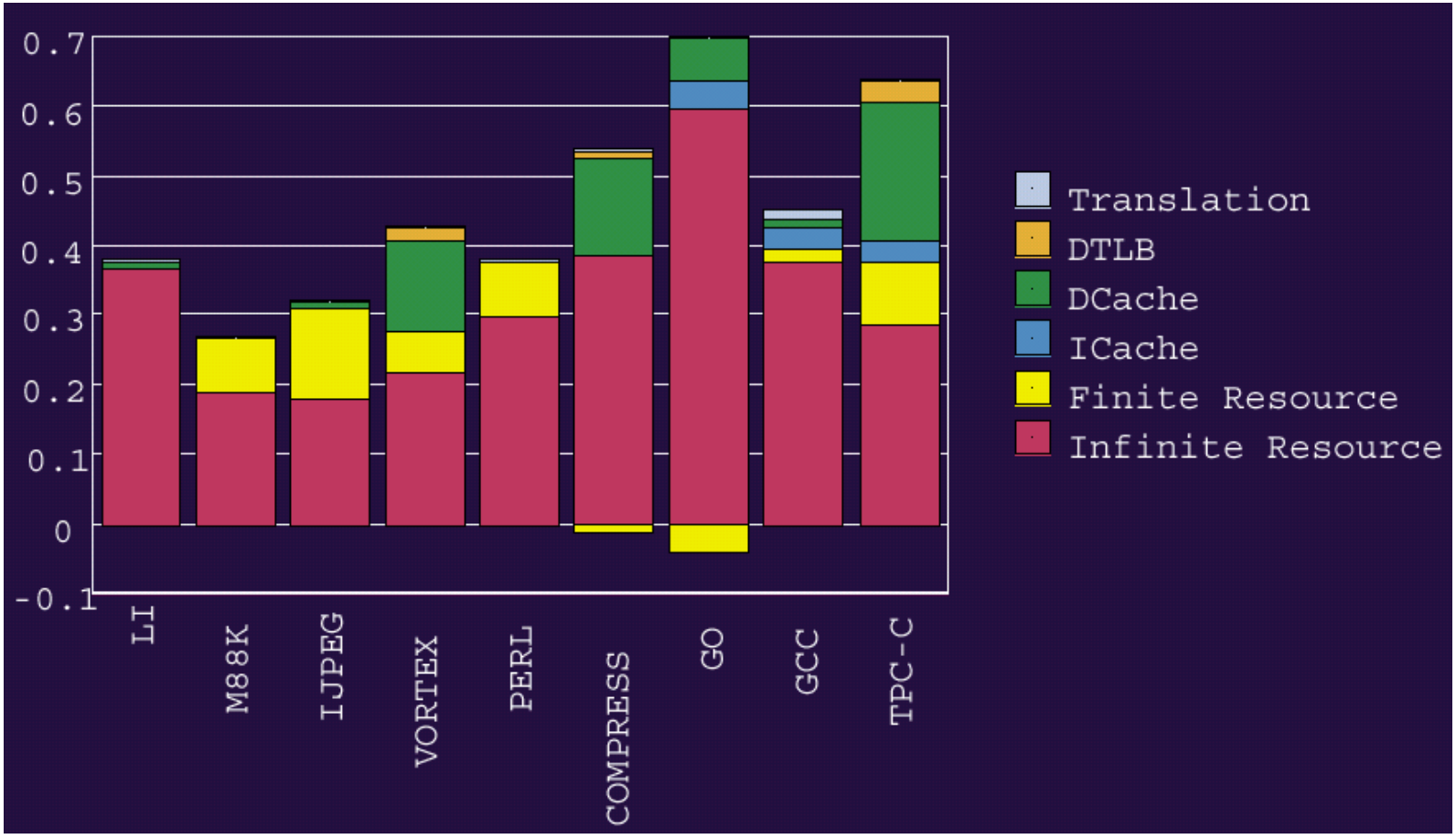
Daisy ILP with Infinite Cache



Source: Ebcioglu et al., © IEEE 2000

© Ienne 2003-22

Daisy Break-out of CPI Performance



Source: Ebcioglu et al., © IEEE 2000

© Ienne 2003-22

Additional Optimisations in DBT

- **Block Reordering**: Make target image execution as sequential as possible
- **Memory Colouring**: Improve mapping of translated code to fit target memory hierarchy
- **Code Specialization**: Clone procedures based on constant parameter values

© Ienne 2003-22

Benefits of DBT

- Compatibility
  - With native implementations
  - Across different VLIWs sizes and generations
- Reliability and possibilities to upgrade
  - Software patches for bugs in translator
  - Software patches for optimiser enhancements
  - Translator can be used to hide hardware bugs

© Ienne 2003-22

Benefits of DBT

- Low hardware cost
  - SW scheduler: smaller chip with higher yield
  - Fast in-order implementations possible
- Higher instruction-level parallelism
  - Dynamic groups can be made arbitrarily large
- Low-power consumption
  - Memory consumes less than logic: schedule once and then fetch from memory (?)

© Ienne 2003-22

Issues of DBT

- Reduced resources for the user
  - Cycles: lost performance for translation
  - Memory
- Slow at start (emulation) and real-time difficulties
- Debugging difficulties
  - Target machine code far removed from source code
  - Non-deterministic behaviour of real-systems

© Ienne 2003-22

Open Problems of DBT

- Can a DBT VLIW machine be ever any better than a well-conceived superscalar?
- Better light-weight optimisations possible?
- Real-time problems solvable?
- Which translation cache management policy is best?
- Target architecture ever exposed to users?

© Ienne 2003-22

DBT for Multiple Source Architectures?

- Efficient translation requires some hardware support for the source architecture in the target hardware architecture
  - Opcodes
  - Condition code registers
  - Floating-point formats
  - Timer registers
  - Segment registers
  - Address translation and MMU
  - **Other awkward details** (8-bit reg. access, alignment,…)
- Crusoe's VLIW has some features to address IA-32's legacy oddities

© Ienne 2003-22

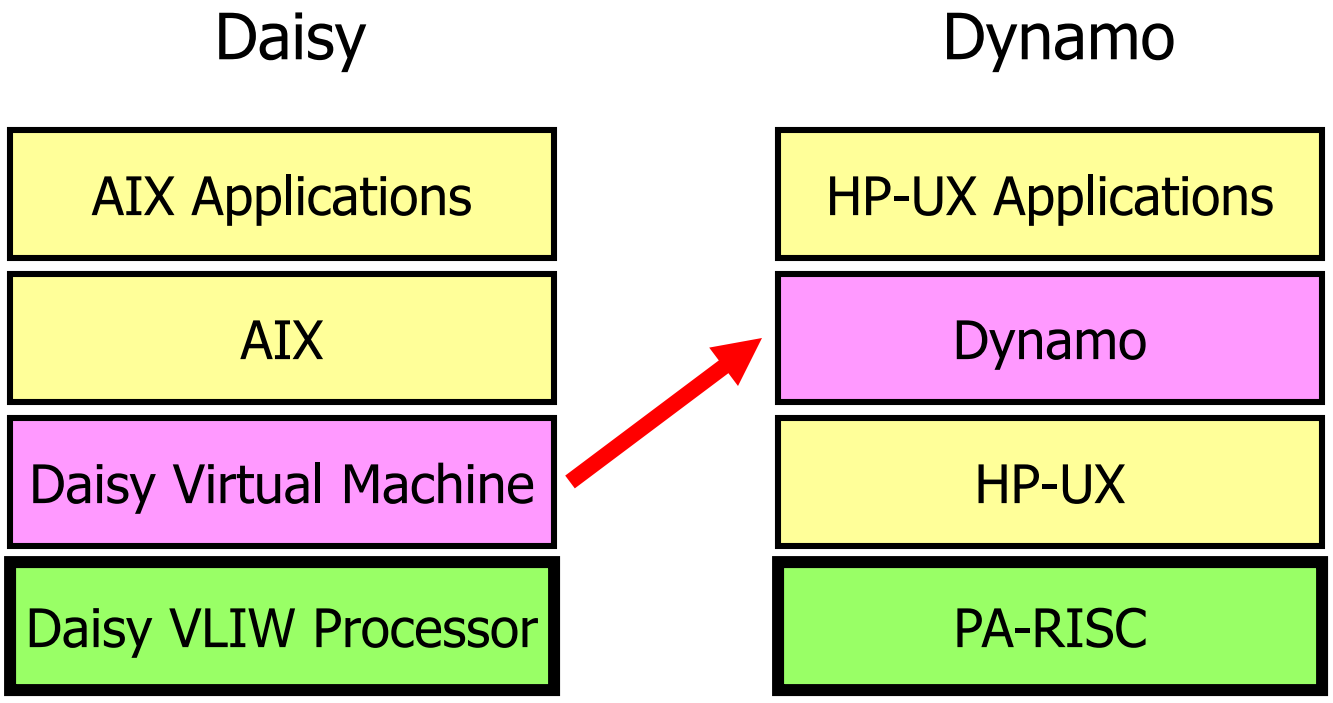
More Aggressive Application: Dynamo's Dynamic Optimisation

- Static optimisation in compiler backend is limited
  - Often profile-based optimisations not used
  - Lots of runtime info not available
  - Static optimisations are typically implementation independent
- "Translate" with the same source and target architecture: **dynamic optimisation!**

© Ienne 2003-22

More Aggressive Application: Dynamo's Dynamic Optimisation

- Slightly different emphasis: centred on user code, no optimisation of the OS



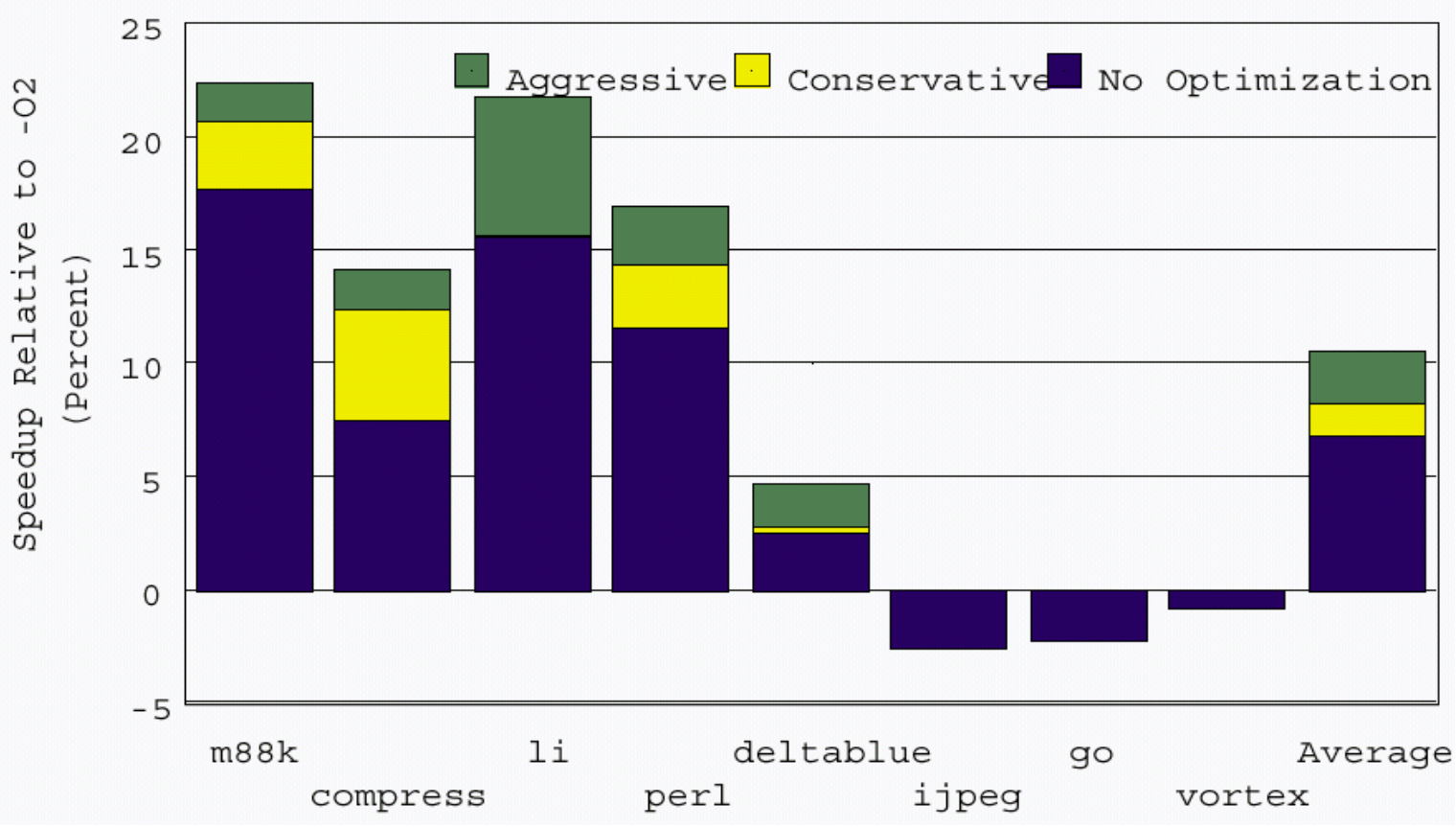
© Ienne 2003-22

More Aggressive Application: Dynamo's Dynamic Optimisation

- Optimisations:
  - Identify long instruction groups (**traces**)
  - Extend traces over
    - Indirect branches
    - Function calls and returns
    - Virtual function calls
  - Optimise traces: classic ILP optimisations, remove unconditional branches,…
- Dynamo can bail out…

© Ienne 2003-22

More Aggressive Application: Dynamo Speedup over –O2

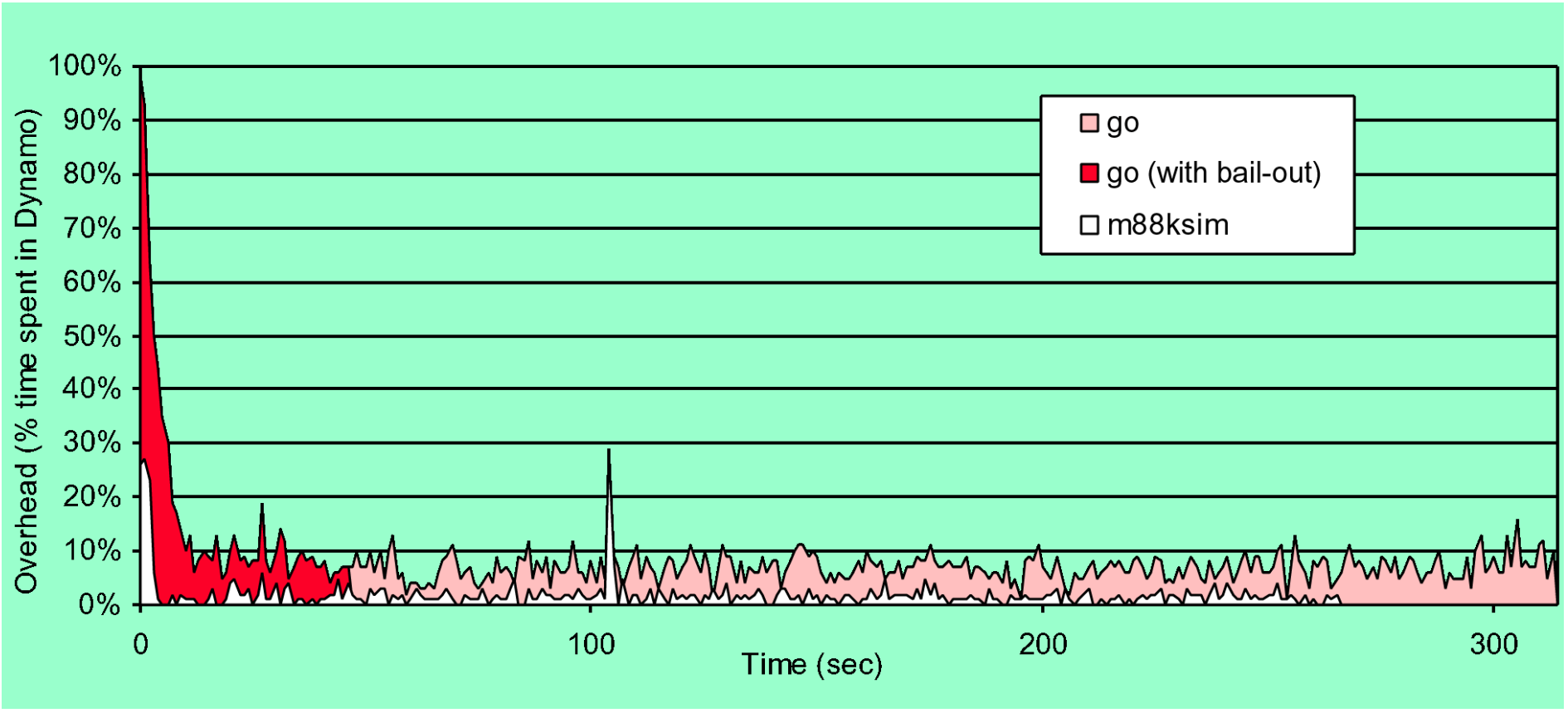


Source: Bala et al., © ACM 2000

- No optimisation ➔ only dynamic inlining with trace selection
- `go` and `vortex` ➔ bail-out

© Ienne 2003-22

More Aggressive Application: Dynamo Bail-Out

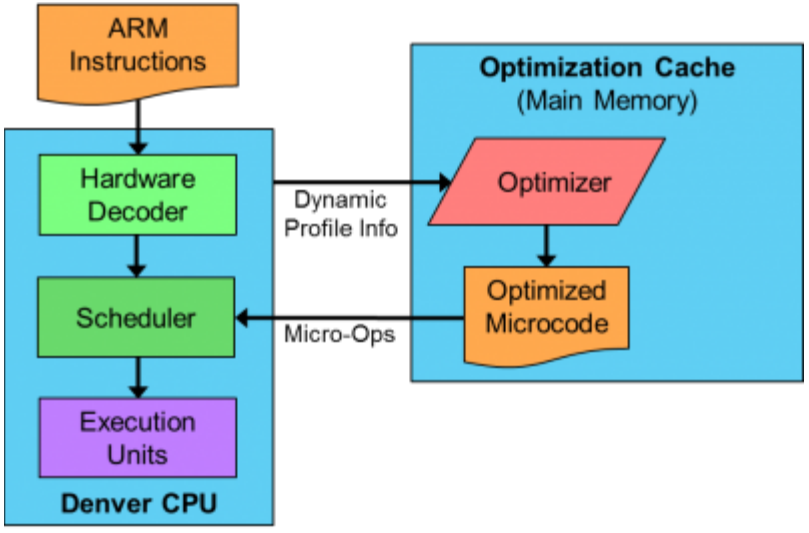


Source: Bala et al., © ACM 2000

© Ienne 2003-22

Is the Idea Dead with Transmeta? Not at All…

- Nvidia announced in 2014 its first CPU design, code-named Denver
- Native ARMv8 → 2 instructions per cycle
  - Avoid the troubles of emulation
  - Uses trivial translation ("decoding") into a proprietary µ-op set
- Can perform DBT to the proprietary µ-op set → 7 µ-op per cycle
  - Fully exploits the µ-op set only with proper recompilation / sw optimization



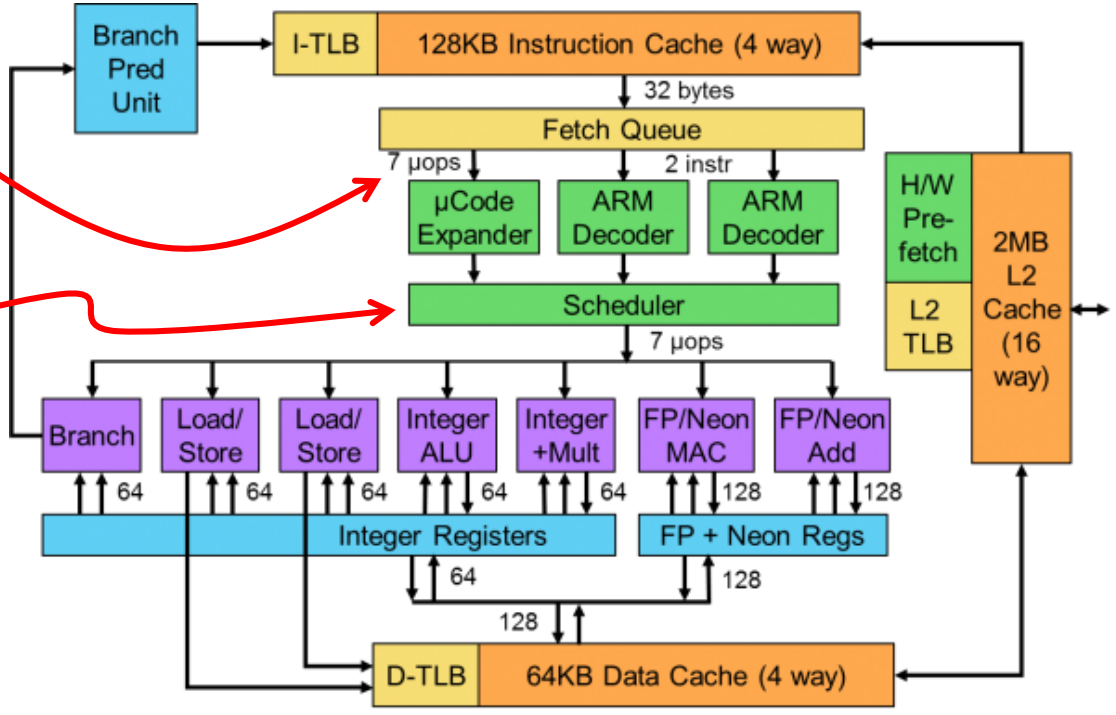
Source: Gwennap, © The Linley Group 2014

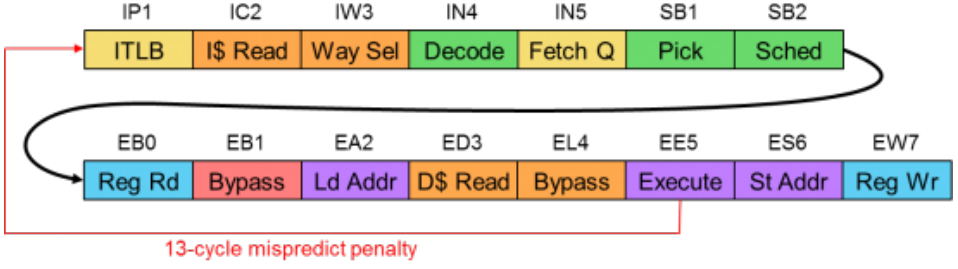
© Ienne 2003-22

Denver Microarchitecture: A Simple In-Order Superscalar

µ-ops are fetched in bundles: similar to a VLIW in that parallel operations are selected at compile time

No fancy OOO scheduler! Simply check data dependences and stall



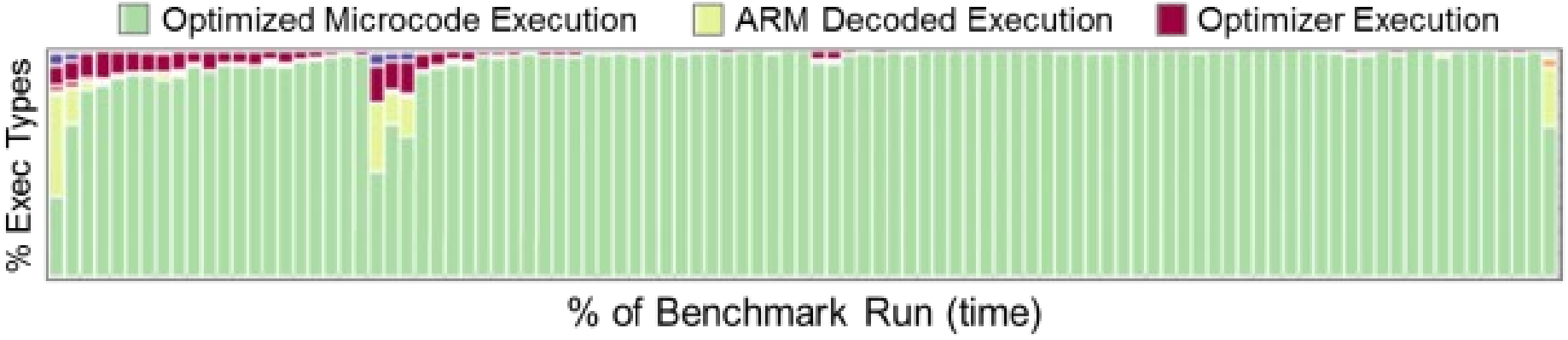


Skewed pipeline can bundle a Load/ALU/Store dependent sequence

Source: Gwennap, © The Linley Group 2014

© Ienne 2003-22

A Sample Execution Trace



The optimizer is invoked for new parts of code never visited before or for translations that have been evicted, as natural

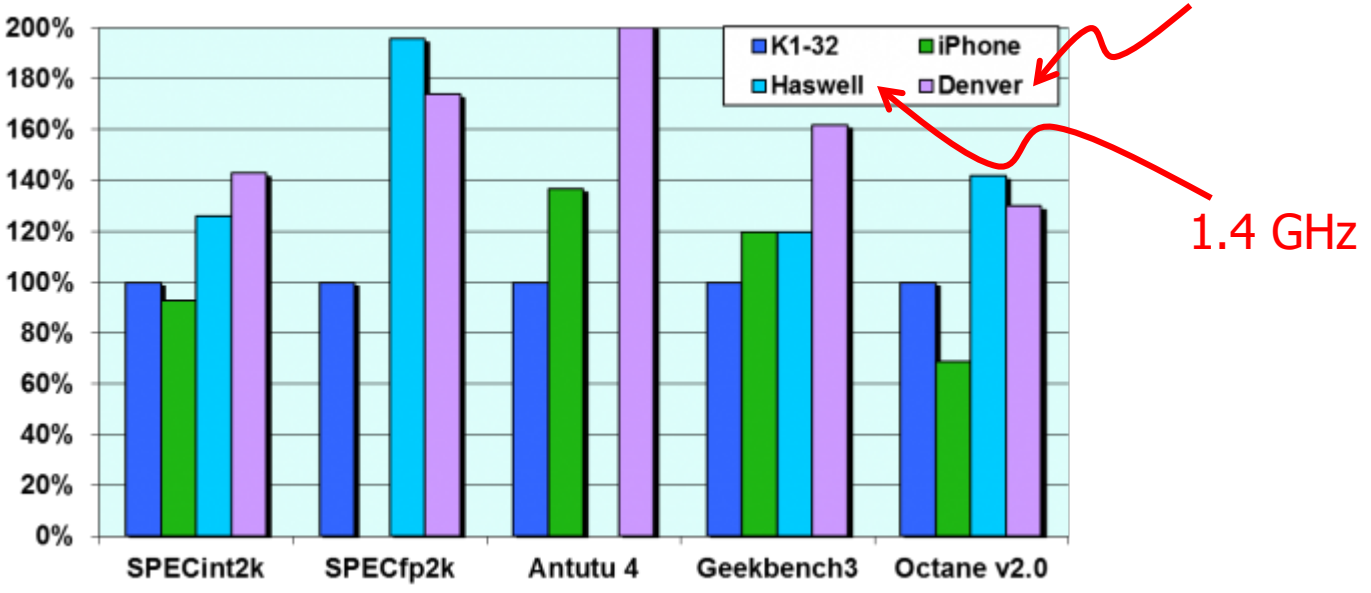
Yet, the optimizer can also be invoked if the branch behaviour changes (e.g., a branch predictor is modified)

Source: Gwennap, © The Linley Group 2014

© Ienne 2003-22

The Net Result: The Fastest ARM

	Snapdragon 805	Apple A7	Tegra K1-32	Tegra K1-64	Snapdragon 810
CPU ISA	ARMv7	ARMv8	ARMv7	ARMv8	ARMv8
CPU Type	4x Krait 400	2x Cyclone	4x A15	2x Denver	4x A57 + 4x A53
CPU Speed	2.5GHz	1.4GHz	2.3GHz	2.5GHz	2.2GHz‡
CPU Perf*‡	3.5 ST / 11.9	5.0 ST / 9.6	3.9 ST / 13.3	6.3 ST / 11.9	4.4 ST / 17.9
GPU Type	Adreno 420	SGX5 MP4	Kepler-192	Kepler-192	Adreno 430
GPU Perf†§	19fps	13fps	27fps	27fps	25fps‡
Video Decode	4K	1080p	4K	4K	4K
ISP	1.0GP/s	Not disclosed	1.2GP/s	1.2GP/s	1.2GP/s
IC Process	28nm HPM	28nm HPM	28nm HPM	28nm HPM	20nm HKMG
First Devices	2Q14	3Q13	2Q14	4Q14 (est)	1H15 (est)



Source: Gwennap, © The Linley Group 2014

© Ienne 2003-22

References

- Tutorial by Erik Altman and Kemal Ebcioğlu (IBM T. J. Watson Research Center), given at MICRO-33 in December 2000
  http://www.microarch.org/micro33/tutorial/tutorial.html
- Daisy
  - http://www.research.ibm.com/daisy
  - E. R. Altman et al., Advances and Future Challenges in Binary Translation and Optimization, IEEE Proceedings, 89(11):1710-22, November 2001
- Dynamo
  - http://www.hpl.hp.com/cambridge/projects/Dynamo
  - V. Bala et al., Dynamo: A Transparent Dynamic Optimization System, PLDI 2000
- IEEE Transactions on Computers, June 2001 – Special issue on Dynamic Optimisation
- IEEE Computer, March 2000 – Special issue on Binary Translation
- Gwennap, Nvidia's First CPU Is a Winner, MPR 18th August 2014

© Ienne 2003-22