VLIW architecture and Binary translation

Michael Wenbing Zhang
Dinggao Pan

What is VLIW

Very Long Instruction Word

- One long instruction consists of many tightly coupled independent operations from various source

Explore Instruction Level Parallelism

Hardware simpler, more complex compiler

- Pipelining, Superscalar, Out-of-Order Execution requires extra hardwares

Compiler Method: Trace scheduling

Key Target: Increase Instruction Level Parallelism(ILP)

Statically predict frequently executed trace

- Trace as a stream of instructions

Preprocessing prevents illegal code motion

Compact long stream of instructions as if they are a basic block, compensate for any loss due to the compact afterwards

- May result in huge increase in code size and does not work well if the behavior of the code varies with different inputs.

Another Trace scheduling is done with the new code generated by the first

Contemporary VLIW machine - Intel Itanium

IA64 architecture (64-bit machine)

Explicit Parallel Instruction Computing (EPIC) by HP

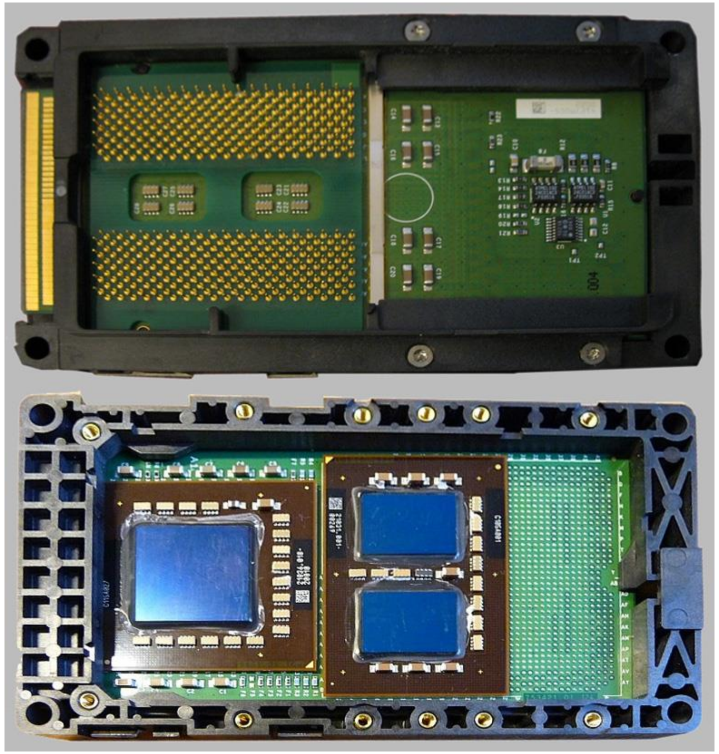
128 bits per bundle

3 instruction per bundle

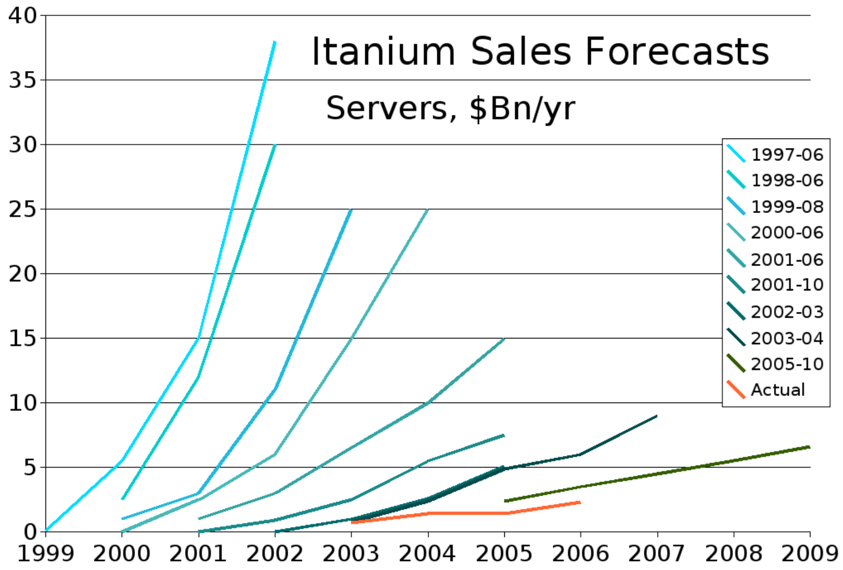
Extra bits to have CPU addressing more register and indicate which instructions to execute in parallel

Predicted by Intel and HP to soon replace x86 architecture machines

Actual performance not superior than CISC and RISC machines



Itanium(Merced) Package



Binary Translation

Method to port software program across system

Emulate one instruction set by another through translation of binary code

Two types of Binary Translation

- Static binary translation
  - Convert all binaries of a executable into target architecture binary code without running the code first
  - May fail to translate some branches whose value is know only at run-time
- Dynamic binary translation
  - Convert a short sequence and cache the result to have the CPU executing