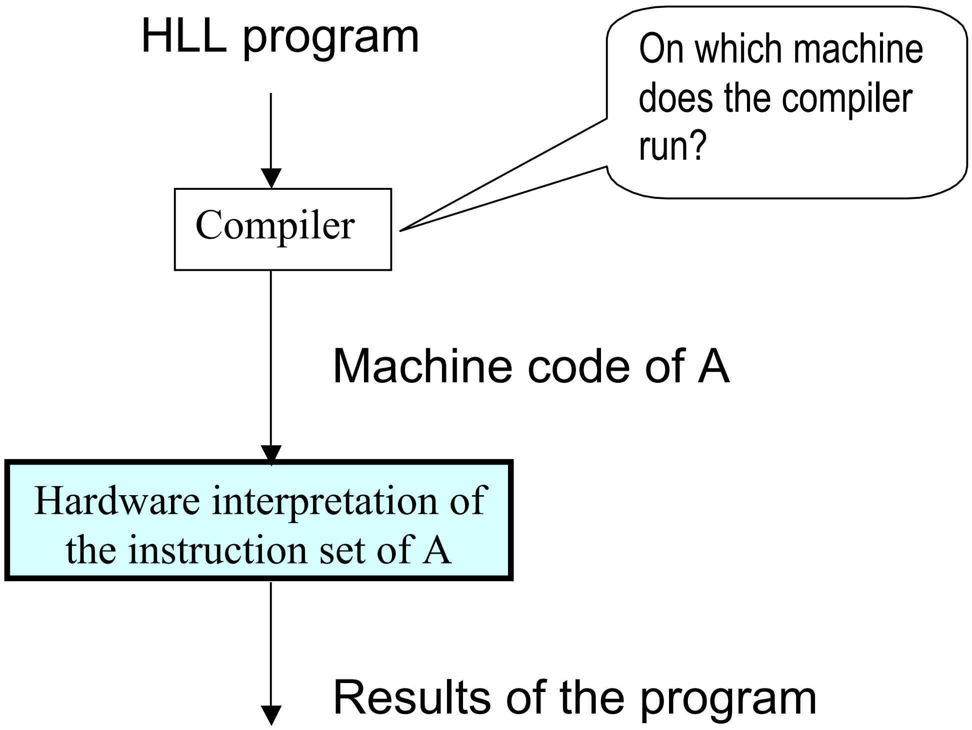
<u>Case Study : Transmeta's Crusoe</u>

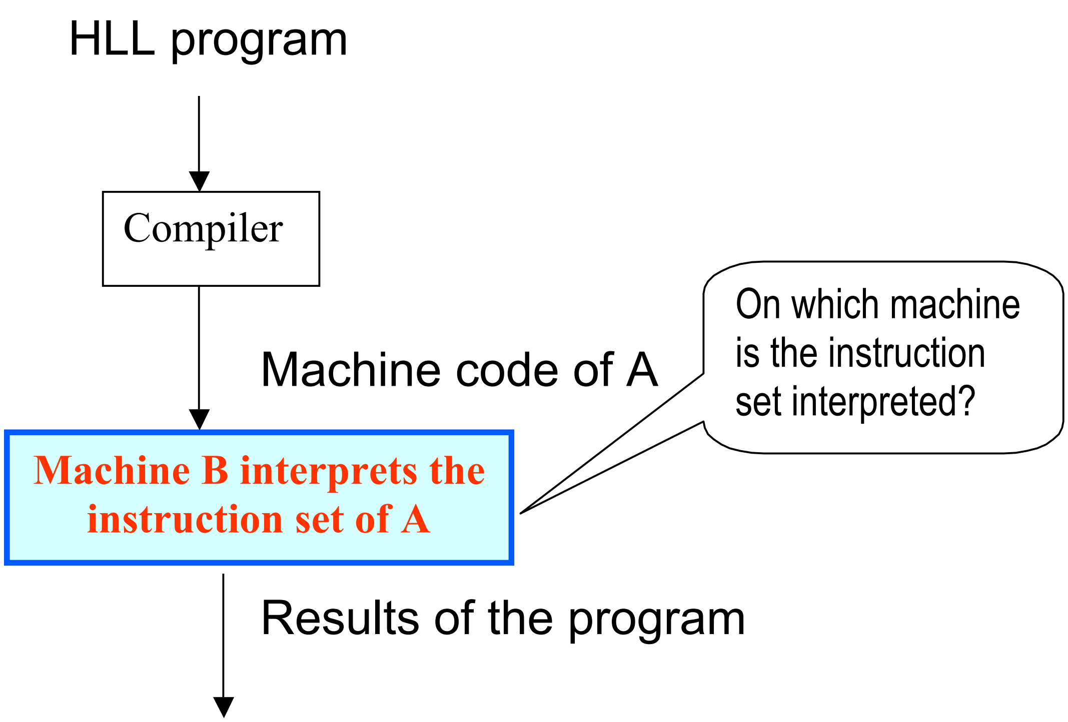
Motivation

David Ditzel (SUN microsystems) observed that Microprocessor complexity is getting worse, and they consume too much power. This led to the birth of Crusoe (nicknamed Transmeta's magic show).

(The magic) Crusoe runs x86 software on a completely different (and simpler) hardware platform, sometimes faster, and consumes 1/3 to 1/30th power

ABC of Emulation





Big question

Can emulation speed-up instruction execution?

Add new powerful instruction

David Ditzel tried with SUN Sparc

Superscalar processing

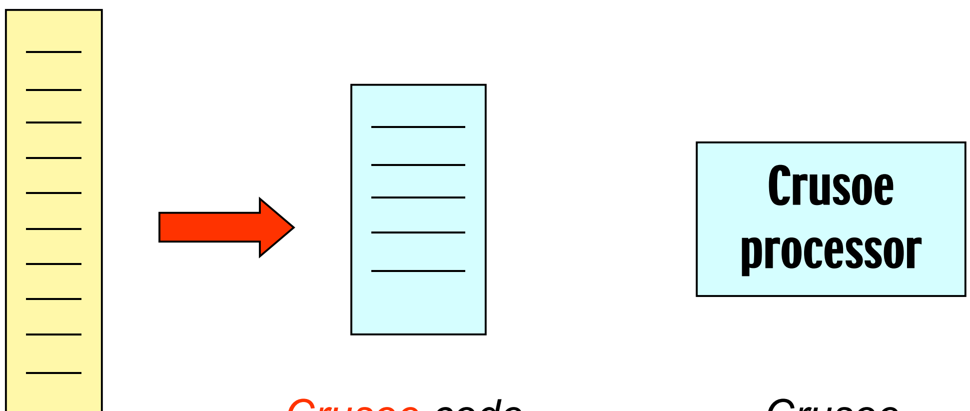
Control unit hardware become complex.

VLIW approach

Convert x86 instructions into VLIW formal for another machine B. The hardware of machine B will execute these in parallel.

Transmeta's Crusoe

runs x86 codes on a non-x86 platform, and sometimes, more efficiently! Uses emulation.



x86 binary

Crusoe code to interpret the x86 binary

Crusoe executes this code

Additional codes may be needed to "patch" the architectural differences between the source and the target architectures, making (traditional) emulation somewhat slower.

But the parallelism in the VLIW approach compensated for the slowness.

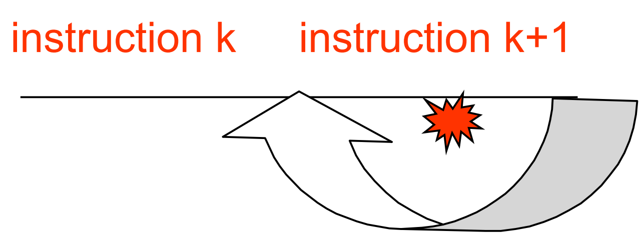
Typical problems with emulation

Condition codes set differently in the two machines.

X86 may have a condition code register that Crusoe does not have. Crusoe had to use a few additional registers to mimic the condition code register.

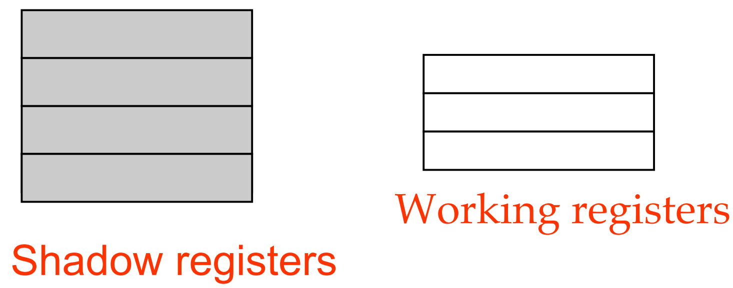
Exception handling can be a big headache.

To restart a faulting instruction, the program has to roll back to the previous state prior to that instruction.



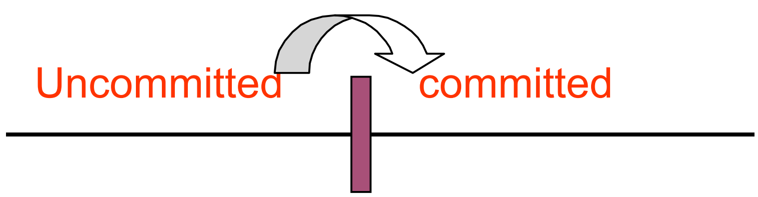
Instructions are re-shuffles in the VLIW version

Introduced instructions for **commit** and rollback.



Commit instruction duplicates the shadow registers into the working registers. The shadows are preserved for a possible rollback.

Rollback instruction copies the shadow registers into the working registers. The shadow registers can be re-used when an “all clear” is received.



Concept of gated store

Secret of success

Use of VLIW in emulation.

Each x86 instruction = 1-4 crusoe instructions, called atoms.

These are bundled into molecules (each molecule may contain up to four atoms).

Memory = load/store	Compute = integer/FP/multimedia	ALU = typical 3-register op	immediate

Memory	Compute	ALU	branch

The bundling is done on-the-fly by the crusoe software.

Crusoe can schedule 1-4 instructions per clock cycle.

The optimizer further optimizes frequently used codes.

Inside the Crusoe Processor

A simple RISC architecture

2 integer units, 1 FP unit, 1 load-store unit, 1 BPU

64 integer and 32 FP registers

48 integer shadow registers and 16 FP shadow registers.

Crusoe pipeline

Six stages for integers:

Fetch	Fetch	Decode	Read	Execute	Writeback

Ten stages for floating point

Code Morphing

The buzzword for emulation that helped market the idea to venture capitalists.

Lessons learned

Using code morphing, you can run legacy software without using legacy hardware, and sometimes with better efficiency.

Power management

- Crusoe is a simple RISC processor. Compared to x86, the transistor count is only 50%, so power consumption is naturally much lower. Many hardware functions were off-loaded to software. Note the use of virtual devices which simulates the registers within the interfaces
- 10,000 separate clocks for different combinations of functional units. Only run the clocks for the active functional units.
- **LongRun** technology reduced the supply voltage and clock frequency to get the job dome in time using minimum necessary power (like dimmer control).

DVD playback

Mobile Pentium 3 consumes 1.13 watts

Crusoe consumes 0.42 watts