ILP is Dead, Long Live IPC!

(A position paper.)

Augustus K. Uht (aka Gus Uht)
Dept. of Electrical and Computer Engineering

UNIVERSITY OF
Rhode Island

@
URI
μRI
Microarchitecture Research Institute
University of Rhode Island

BARC: February 3, 2006

Outline

1. ILP vs. IPC
2. State of the Art; two nasty trends:
   a) Conventional wisdom: '*…no more exploitable ILP…*'
   or: **'ILP is Dead'**
   b) Power is killing us – CPU power in light-bulb equiv.
3. Industry/Academia Response:
   a) Dual/multi-core chips (multiprocessors)
   b) Does this buy us anything?

4. Solution: **'Long Live IPC!'**

1. ILP and IPC Defined

- ***ILP = Instruction-Level Parallelism:***
  Potential parallelism: what is in the code.
  (Assumption: unlimited resources.)

- ***IPC = Instructions Per Cycle:***
  Realized parallelism:
  what the hardware really gets.
  (Resources limited.)

- ***Overall 'performance' = IPC * clock frequency,***
  in Instructions Per Second.

2. State of the Art

I. Conventional wisdom: ‘ILP is Dead’

  a) Given: ‘Limit studies’ show ILP in 10’s (even **`gcc`**)

  b) In reality, hardware rarely gets IPC > 1, **BECAUSE:**

    i. Conservative research:
    Need to stick to std. CPU model if you want to get published.

    ii. Industry cautious: doesn’t like big changes

II. Power has become excessive, e.g.:
Intel 4.0 GHz Pentium 4:
Heat death of the uni-processor-verse

→ Industry sol’n: *n*-core chips: *n* lower-freq. CPUs, e.g.:

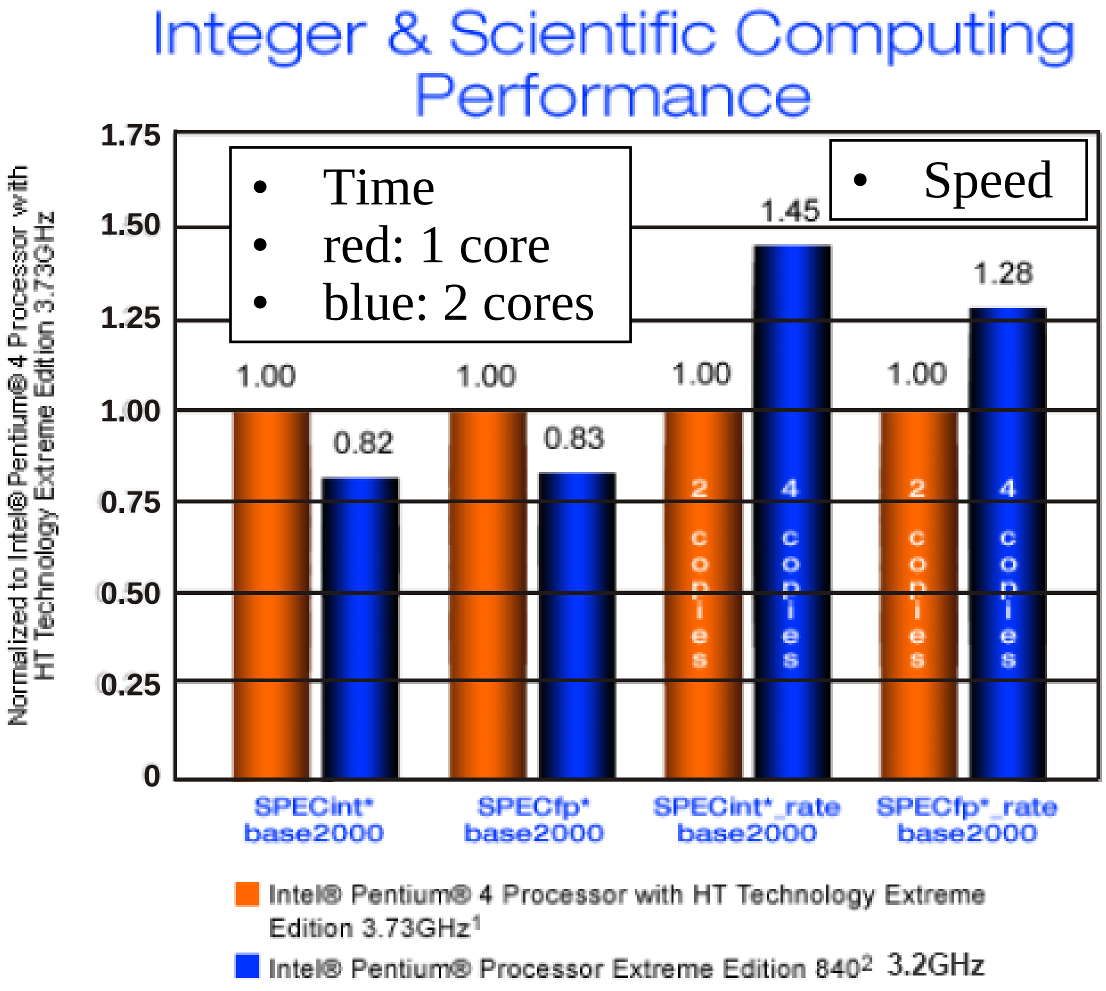
dual perf. >= solo perf., & dual pwr. <= solo pwr.

(Oh, really???)

Multicore Problem 1: (non)-Performance

- Time, 1 to 2 cores:
  - For same power,
  - Twice the cost,
  - Get 18% perf. increase.
  - Likely perf. would actually ***decrease***

- Speed, 1 to 2 cores:
  - < 50% perf. gain
  - Code <u>copies</u> used



- Time
- red: 1 core
- blue: 2 cores

- Speed

© Intel Corp.

Multicore Problem 2:
Can’t Program ‘em

1. Chief of some chip company:
   [*Programmers will **have** to learn how to write parallel programs.*] **!!!**
2. ….and pigs have wings, to wit:
3. Since ~1964:
   a) Tried to build auto-parallelizing compilers. ***NG.***
   b) Tried to make parallel programming easy. ***No cigar.***
4. We don’t know how to write
   good ***sequential*** programs;
   now good ***PARALLEL*** programs?

Solution: Back to the Future

'Long Live IPC!'

1. Recall: community is stuck on old model.
2. Radically new models are promising; examples:
   - TRIPS (IPC ~3, needs compiler support)
   - Levo (IPC ~5, with legacy binaries)

   (Remember: the P6 was radical too, way back when.)
3. Don't use as many transistors as possible.
   → power less of an issue.
4. Multicore processors? Maybe, but used differently.
   1) Use simple cores:
      Cores do not need to be complete or standard processors.
   2) Eliminate basic cross-chip communications.

Summary

'Those who cannot remember the past are condemned to repeat it.'

Parallel programming is a losing proposition for most/all programs, even scientific (time-to-solution).

(But: *APL, anyone?*)

We need IPC!

(We don't need PPC [processors per cycle]…)