ECE 571 – Advanced Microprocessor-Based Design Lecture 6

Vince Weaver

https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

16 September 2024

Announcements

- Homeworks
  - HW#1 will be graded soon
  - HW#2 due Friday, posted a bit late (a reading)
- Optional Readings
  - Pipeline Discussion: Computer Organization (RiscV) / Patterson and Hennesey
    Section 4.11 "Real Stuff: The ARM Cortex-A53 and Intel Core i7 Pipelines"
  - Power/Energy: Computer Architecture / Hennesey

and Patterson
Section 1.5 “Trends in Power and Energy in Integrated Circuits”

Pipeline Question from Last Time

- From 2-stage to Pentium 4 31-stage?
  What can you do with that many?
  P6,10:Fetch*2,Decode*3,Rename,ROB Rd,Rdy/Sch,Dispat
  P4,20:NxtIP*2,Fetch*2,Drive,Alloc,Rename*2,Queue,Sch*
  RF*2,Ex,Flgs,BrCk,Drive

How to Take Advantage of Parallelism in Code

- Eventually Physics gets in the way
- You can't make a single CPU core any faster
- What can you do with all the extra transistors?

Finding Parallelism with Software

- Can have a super-scalar multi-pipeline chip, but it's expensive to decode and find dependencies on the fly
- Can you do this in software at compile time instead?
- VLIW as found in itanium
- Compiler pre-finds 3 instructions with no dependencies that can run at once on 3-wide superscalar
- Turns out this is a really hard software problem

Finding Parallelism with Vector Math

- Certain types of math are extremely parallel
- Common example is vector addition, where each sub-addition has no dependencies and can happen in parallel
- Can design SIMD (single-instruction multiple-data) instructions that with one instruction can work on very wide datatypes

SSE example (From Wikipedia)

Doing a 4 element single-prevision vector add would take 4 separate floating point adds:

```
vec_res.x = v1.x + v2.x;
vec_res.y = v1.y + v2.y;
vec_res.z = v1.z + v2.z;
vec_res.w = v1.w + v2.w;
```

With SSE you only need one add instruction:

```
 movaps xmm0, [v1]                ;xmm0 = v1.w :
 addps xmm0, [v2]                 ;xmm0 = v1.w+v2
v1.x+v2.x
 movaps [vec_res], xmm0
```

aps=aligned packed single-precision float

Project idea?

SIMD / Vector Instructions

- x86: MMX/SSE/SSE2/AVX/AVX2
  semi-related FMA
- MMX (mostly deprecated), AMD's 3DNow! (deprecated)
- PowerPC Altivec
- ARM: Neon / A64 SIMD / "Scalabale Vector Extensions" SVE

SSE / x86

- SSE (streaming SIMD): 128-bit registers XMM0 - XMM7, can be used as 4 32-bit floats
- SSE2 : 2*64bit int or float, 4 * 32-bit int or float, 8x16 bit int, 16x8-bit int
- SSE3 : minor update, add dsp and others
- SSSE3 (the s is for supplemental): shuffle, horizontal add
- SSE4 : popcnt, dot product

AVX / x86

- AVX (advanced vector extensions) – now 256 bits, YMM0-YMM15 low bits are the XMM registers. Now twice as many.
  Also adds three operand instructions a=b+c
- AVX2 – 3 operand Fused-Multiply Add, more 256 instructions

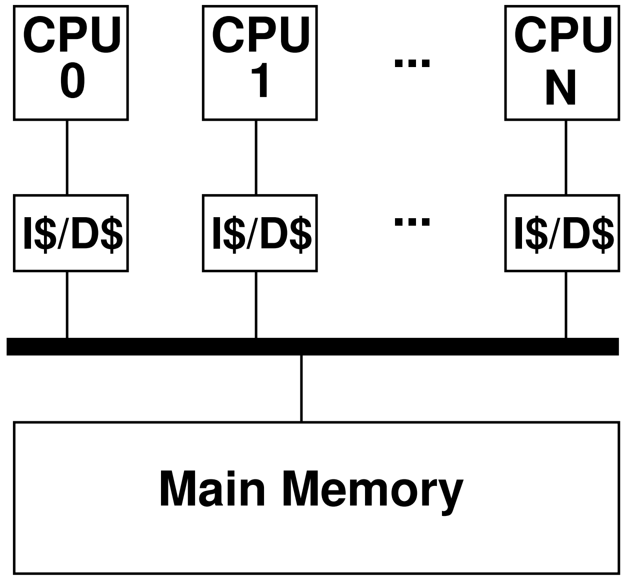
AVX-512 / x86

- Originally on (discontinued) Xeon Phis (knights landing)
- now 512 bits, ZMM0-ZMM31
- Problems, power hungry, chip have to scale down frequency so perf tradeoff complex
- Due to P/E cores on hybrid chips Intel disabled on Alder Lake and Raptor Lake

Finding Parallelism with More Cores

- Multi-core / Chip-Multi Processing (CMP)
- Moore's law gives you lots of transistors.
- Why not just put more cores on the die?
- Exploits multi-programmed parallelism rather than instruction-level parallelism

CMP Diagram



Multi-core Programming

- One way is multi-programming, have one core for each task
  That often only works up to about 4-8 cores though
- Other option is multi-threaded programming, writing programs to split work among different cores
- This is hard. Really hard.
- See ECE574 cluster Computing

Multi-core Programming Challenges

- Distributed vs Shared Memory
- Shared Memory most common
- Memory consistency models, how to instructions interleave `https://arangodb.com/2021/02/cpp-memory`
- Software can have race conditions, need locking, hard to get right
- Hardware has to ensure cache coherency (protocols)

Cache Coherency

- How do you handle data being worked on by multiple processors, each with own cache of main memory?
- Cache coherency protocols.
- Many and varied. MESI is a common one
- Directory vs Snoopy

MESI

- Modified, Exclusive, Shared, Invalid

Hybrid/Heterogeneous Core Systems

- Do all cores need same features?
- ARM big.LITTLE systems, a few powerful cores for compute heavy tasks, many more power-efficient cores for regular workloads. Can help battery life
- Recent Intel has P (performance) and E (efficiency) cores doing similar
- Makes CPU job scheduling much harder
- Also issues with performance counters
- TODO: diagram

Finding Parallelism with better Pipeline Utilization

- On super-scalar systems pipelines might not always be full, especially if waiting on memory
- How can we utilize those wasted pipeline bubbles

Hardware Multi-threading

- SMT (simultaneous multithreading), Intel Hyperthreading
- Hyrbid of multi-core and multi-pipeline
- Why not duplicate fetch/decode logic, and have two programs execute at once on same set of pipelines.
- If one is idle/stalled, run instructions from other thread
- Looks to OS as if you have two cores, but really just one with two instruction dispatch stages
- Extra logic to make sure that pipelines used fairly, the results get committed to the right register file, etc.

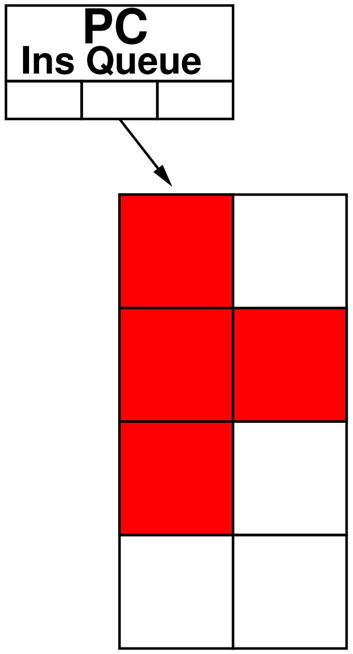
SMT Variations

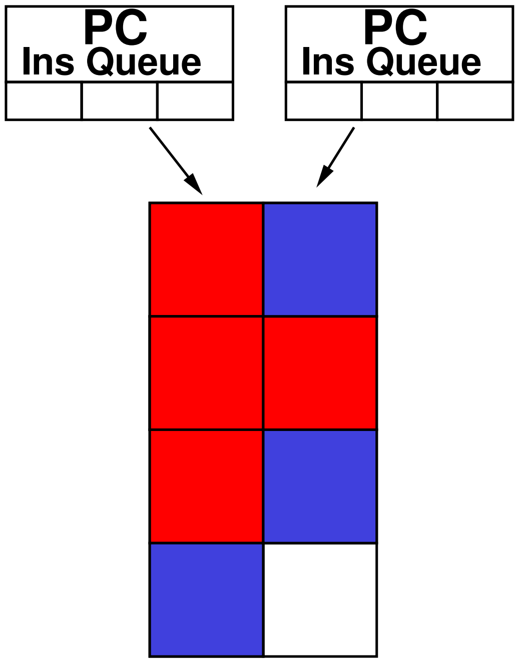
- Fine-grained – rotate threads every cycle
- Coarse-grained – rotate threads only if long latency event happens (cache miss)
- Simultaneous – issue from any combination of threads, to maximize use of pipeline (have to be superscalar)
- Aside on Sun Niagara

SMT Downsides

- Can actually slow down code (especially if both threads trying to use same functional units, also if both using memory heavily as cache is often shared)
- Security? Information Leakage?
- How wide (Sun Niagara aside)

SMT Diagram





Real-World Pipelining Examples (from P&H)

- ARM Cortex-A53 (found in Pi3)
  - Eight-stage pipeline
  - Dynamic multi-issue, two instructions
  - Static in-order pipeline
  - First 3 stages fetch two insns at a time, filling a 13-entry instruction queue (branch predictors)
  - Pipelines: one for load, one for store, two for ALU, one multiply, one divide, one FP/SIMD (mul/div/sqrt)

one FP/SIMD for other

- ◦ What's the peak possible IPC?
- ◦ Patterson and Hennesey report SPEC CPU 2006 INT results. Best case is hmmer (search for gene sequence) with IPC 1.03 (CPI 0.97). Worst is mcf (public transit vehicle scheduling) IPC 0.12 (CPI 8.56). Mostly memory constrained.
- ◦ In-order so depends a lot on compiler to get good performance.
- ◦ 100mW (1 core at 1GHz)

- Intel Core i7 920 (Nehalem, 2008)

- Decodes CISC instructions to micro-ops
- Can issue up to 6 micro-ops per cycle
- 14 pipeline stages
- dynamic out-of-order with speculation
- register renaming, useful with speculation, as no need to store snapshot to undo speculation, just mark the speculated register results as invalid
- Instruction fetch, fetches 16 bytes. If wrong, 15 cycle penalty
- Predecode instruction buffer – transform 16 bytes (x86 insns 1-15 bytes) into x86 insns

- 18-instruction instruction queue.
- Micro-op decode – three decoders handle decode of instructions that map to 1 uop. One other handles microcode engine that produces longer sequences, up to 4uops a cycle.
- Can also do micro-op fusion (fuse two different insns into one uops, such as cmp/branch)
- Micro-ops go ins a 28-entry uop buffer

  Loop Stream Detector – if code is in tight loop (less than 28 insns) it can execute from this buffer and not need to fetch.

- Instruction issue. Reservation station. Up to six uops can be issued
- Finished instructions go back to reservation station and retirement unit, wait to update register state when determined it is no longer speculative.
- Once instruction hits the head of the reorder buffer, instruction commits and is removed from re-order buffer
- Even though 6 uops can issue, only 4 can be finished a turn? What's the peak IPC? (4)
- Again, SPECCPU. Best is libquantum IPC=2.2 (CPI

0.44). Worst, again, mcf IPC=0.37 (CPI=2.67)

- ◦ Where do the wasted cycles go? Stalls? But also mis-speculation where work is done and then thrown out.
- ◦ 130 Watts (2.66GHz)