**Review**

- **Benchmarks**
  - Attempt to predict performance
  - Updated every few years
  - Measure everything from simulation of desktop graphics programs to battery life

- **Megahertz Myth**
  - MHz ≠ performance, it’s just one factor

---

**MIPS is example of RISC**

- **RISC** = Reduced Instruction Set Computer
  - Term coined at Berkeley, ideas pioneered by IBM, Berkeley, Stanford

- **RISC characteristics:***
  - Load-store architecture
  - Fixed-length instructions (typically 32 bits)
  - Three-address architecture

- **RISC examples:** MIPS, SPARC, IBM/Motorola PowerPC, Compaq Alpha, ARM, SH4, HP-PA, ...

---

**MIPS vs. 80386**

<table>
<thead>
<tr>
<th>MIPS</th>
<th>80386</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address:</td>
<td>32-bit</td>
</tr>
<tr>
<td>Page size:</td>
<td>4KB</td>
</tr>
<tr>
<td>Data aligned</td>
<td>Data <strong>unaligned</strong></td>
</tr>
<tr>
<td>Destination reg: Left</td>
<td>Right</td>
</tr>
<tr>
<td>add $rd,$rs1,$rs2</td>
<td>add %rs1,%rs2,%rd</td>
</tr>
<tr>
<td>Regs: $0, $1, ..., $31</td>
<td>%r0, %r1, ..., %r7</td>
</tr>
<tr>
<td>Reg = 0: $0</td>
<td>(n.a.)</td>
</tr>
<tr>
<td>Return address: $31</td>
<td>(n.a.)</td>
</tr>
</tbody>
</table>

---

**MIPS vs. Intel 80x86**

- **MIPS:** “**Three-address architecture**”
  - Arithmetic-logic specify all 3 operands
  - add $s0,$s1,$s2 # s0=s1+s2
  - Benefit: fewer instructions ⇒ performance

- **x86:** “**Two-address architecture**”
  - Only 2 operands, so the destination is also one of the sources
  - add $s1,$s0 # s0=s0+s1
  - Often true in C statements: c += b;
  - Benefit: smaller instructions ⇒ smaller code

---

**MIPS vs. Intel 80x86**

- **MIPS:** “**load-store architecture**”
  - Only Load/Store access memory; rest operations register-register; e.g.,
  - lw $t0, 12($gp)
  - add $s0,$s0,$t0 # s0=s0+Mem[12+gp]
  - Benefit: simpler hardware ⇒ easier to pipeline, higher performance

- **x86:** “**register-memory architecture**”
  - All operations can have an operand in memory; other operand is a register; e.g.,
  - add 12($gp),%s0 # s0=s0+Mem[12+gp]
  - Benefit: fewer instructions ⇒ smaller code
### MIPS vs. Intel 80x86

#### MIPS: “fixed-length instructions”
- All instructions same size, e.g., 4 bytes
- Simple hardware \(\Rightarrow\) performance
- Branches can be multiples of 4 bytes

#### x86: “variable-length instructions”
- Instructions are multiple of bytes: 1 to 17;
  \(\Rightarrow\) small code size (30% smaller?)
- More recent performance benefit: better instruction cache hit rates
- Instructions can include 8- or 32-bit immediates

### Unusual features of 80x86

- 8 32-bit Registers have names;
  16-bit 8086 names with “e” prefix:
  - eax, ecx, edx, ebx, esp, ebp, esi, edi
- 80x86 word is 16 bits, double word is 32 bits
- PC is called eip (instruction pointer)
- leal (load effective address)
  - Calculate address like a load, but load address into register, not data
  - Load 32-bit address:
    - `leal -4000000(%ebp),%esi`
    - \# esi = ebp - 400000

### Instructions: MIPS vs. 80x86

<table>
<thead>
<tr>
<th>MIPS</th>
<th>80x86</th>
</tr>
</thead>
<tbody>
<tr>
<td>addu, addiu</td>
<td>addl</td>
</tr>
<tr>
<td>subu</td>
<td>subl</td>
</tr>
<tr>
<td>and, or, xor</td>
<td>andl, orl, xorl</td>
</tr>
<tr>
<td>sll, srl, sra</td>
<td>sall, shrl, sarl</td>
</tr>
<tr>
<td>lw</td>
<td>movl mem, reg</td>
</tr>
<tr>
<td>sw</td>
<td>movl reg, mem</td>
</tr>
<tr>
<td>mov</td>
<td>movl reg, reg</td>
</tr>
<tr>
<td>li</td>
<td>movl imm, reg</td>
</tr>
<tr>
<td>lui</td>
<td>n.a.</td>
</tr>
</tbody>
</table>

### 80386 addressing (ALU instructions too)

- `base reg + offset` (like MIPS)
  - `movl -8000044(%ebp), %eax`
- `base reg + index reg` (2 regs form addr.)
  - `movl (%eax, %ebx),%edi`
    - `# edi = Mem[ebx + eax]`
- `scaled reg + index` (shift one reg by 1,2)
  - `movl(%eax,%edx,4),%ebx`
    - `# ebx = Mem[edx*4 + eax]`
- `scaled reg + index + offset`
  - `movl 12(%eax,%edx,4),%ebx`
    - `# ebx = Mem[edx*4 + eax + 12]`

### Branches in 80x86

- Rather than compare registers, x86 uses special 1-bit registers called “condition codes” that are set as a side-effect of ALU operations
  - S - Sign Bit
  - Z - Zero (result is all 0)
  - C - Carry Out
  - P - Parity: set to 1 if even number of ones in rightmost 8 bits of operation
- Conditional Branch instructions then use condition flags for all comparisons: `<`, `<=`, `>`, `>=`, `==`, `!=`

### Branch: MIPS vs. 80x86

<table>
<thead>
<tr>
<th>MIPS</th>
<th>80x86</th>
</tr>
</thead>
<tbody>
<tr>
<td>beq</td>
<td><code>(cmpl;) je</code> if previous operation set condition code, then cmpl unnecessary</td>
</tr>
<tr>
<td>bne</td>
<td><code>(cmpl;) jne</code></td>
</tr>
<tr>
<td>slt; beq</td>
<td><code>(cmpl;) jlt</code></td>
</tr>
<tr>
<td>slt; bne</td>
<td><code>(cmpl;) jge</code></td>
</tr>
<tr>
<td>jal</td>
<td><code>call</code></td>
</tr>
<tr>
<td>jr %31</td>
<td><code>ret</code></td>
</tr>
</tbody>
</table>
While in C/Assembly: 80x86

C

while (save[i] == k)
    i = i + j;

(i, j, k: %edx, %esi, %ebx)

leal -400(%ebp), %eax

.Loop: cmpl %ebx, (%eax, %edx, 4)
    jne .Exit
    addl %esi, %edx
    j .Loop

.Exit:
Note: cmpl replaces sll, add, lw in loop

Unusual features of 80x86

• Memory Stack is part of instruction set
  - call places return address onto stack, increments esp (Mem[esp] = eip+6; esp += 4)
  - push places value onto stack, increments esp
  - pop gets value from stack, decrements esp
• incl, decl (increment, decrement)
  incl %edx # edx = edx + 1
  - Benefit: smaller instructions ⇒ smaller code

Outline

• Intro to x86
  • Microarchitecture

Intel Internals

• Hardware below instruction set called "microarchitecture"
  • Pentium Pro, Pentium II, Pentium III all based on same microarchitecture (1994)
    - Improved clock rate, increased cache size
  • Pentium 4 has new microarchitecture

Pentium, Pentium Pro, Pentium 4 Pipeline

• Pentium (P5) = 5 stages
  • Pentium Pro, II, III (P6) = 10 stages

Dynamic Scheduling in Pentium Pro, II, III

• PPro doesn’t pipeline 80x86 instructions
  • PPro decode unit translates the Intel instructions into 72-bit "micro-operations" (~MIPS instructions)
  • Takes 1 clock cycle to determine length of 80x86 instructions + 2 more to create the micro-operations
  • Most instructions translate to 1 to 4 micro-operations
  • 10 stage pipeline for micro-operations
Dynamic Scheduling

Consider:
lw $t0 0($t0)         # might miss in mem  
add $s1 $s1 $s1 # will be stalled in  
add $s2 $s1 $s1 # pipe waiting for lw

Solutions:
* Compiler (STATIC) reordering (loops?)
* Hardware (DYNAMIC) reordering

Hardware support for reordering

* Out-of-Order execution (OOO): allow an instruction to execute before prior instructions have executed.
  * Speculation across branches
* When instruction no longer speculative, write results (instruction commit)
  * Fetch/issue in-order, execute OOO, commit in order
  * Watch out for hazards!

Hardware for OOO execution

* Need HW buffer for results of uncommitted instructions: reorder buffer
  * Reorder buffer can be operand source
  * Once operand commits, result is found in register
  * Discard results on mispredicted branches or on exceptions

Dynamic Scheduling in Pentium Pro

Max. instructions issued/clock 3
Max. instr. complete exec./clock 5
Max. instr. committed/clock 3
Instructions in reorder buffer 40
2 integer functional units (FU), 1 floating point FU, 1 branch FU, 1 Load FU, 1 Store FU

Pentium, Pentium Pro, Pentium 4 Pipeline

Pentium (P5) = 5 stages
Pentium Pro, II, III (P6) = 10 stages
Pentium 4 (NetBurst) = 20 stages

Pentium 4

* Still translate from 80x86 to micro-ops
* P4 has better branch predictor, more FUs
* Clock rates:
  * Pentium III 1 GHz v. Pentium IV 1.5 GHz
  * 10 stage pipeline vs. 20 stage pipeline
* Faster memory bus: 400 MHz v. 133 MHz
Pentium 4 features

- Multimedia instructions 128 bits wide vs. 64 bits wide => 144 new instructions
- When used by programs??
- Instruction Cache holds micro-operations vs. 80x86 instructions
  - no decode stages of 80x86 on cache hit
  - called “trace cache” (TC)

Block Diagram of Pentium 4 Microarchitecture

- BTB = Branch Target Buffer (branch predictor)
- I-TLB = Instruction TLB, Trace Cache = Instruction cache
- RF = Register File; AGU = Address Generation Unit

“Double pumped ALU” means ALU clock rate 2X => 2X ALU F.U.s

Pentium, Pentium Pro, Pentium 4 Pipeline

- Pentium (P5) = 5 stages
- Pentium Pro, II, III (P6) = 10 stages
- Pentium 4 (NetBurst) = 20 stages

CS61C: So what’s in it for me? (1st lecture)

Learn some of the big ideas in CS & engineering:

- 5 Classic components of a Computer
- Principle of abstraction, systems built as layers
- Data can be anything (integers, floating point, characters): a program determines what it is
- Stored program concept: instructions just data
- Compilation v. interpretation thru system layers
- Principle of Locality, exploited via a memory hierarchy (cache)
- Greater performance by exploiting parallelism (pipelining)
- Principles/Pitfalls of Performance Measurement

Conventional Wisdom (CW) in Comp Arch

- Old CW: Power free, Transistors expensive
- New CW: Power expensive, Transistors free
- Can put more on chip than can afford to turn on
- Old CW: Chips reliable internally, errors at pins
- New CW: ≤ 65 nm ⇒ high error rates
- Old CW: CPU manufacturers minds closed
- New CW: Power wall + Memory gap = Brick wall
- New idea receptive environment
- Old CW: Uniprocessor performance 2X / 1.5 yrs
- New CW: 2X CPUs per socket / ~ 2 to 3 years
- More simpler processors more power efficient

Massively Parallel Socket

- Processor = new transistor?
- Does it only help power/cost/performance?
- Intel 4004 (1971): 4-bit processor, 2312 transistors, 0.4 MHz, 10 µm PMOS, 11 mm² chip
- RISC II (1983); 32-bit, 5 stage pipeline, 40,760 transistors, 3 MHz, 5 µm NMOS, 60 mm² chip
- 4004 shrinks to ~ 1 mm² at 3 micron
- 125 mm² chip, 65 nm CMOS = 2312 RISC IIs + Icache + Dcache
- RISC II shrinks to ~ 0.02 mm² at 65 nm
- Caches via DRAM or 1 transistor SRAM (www.t-ram.com)?
- Proximity Communication at > 1 TB/s?
- Ivan Sutherland @ Sun spending time in Berkeley!
**20th vs. 21st Century IT Targets**

- **20th Century Measure of Success**
  - Performance (peak vs. delivered)
  - Cost (purchase cost vs. ownership cost, power)

- **21st Century Measure of Success? “SPUR”**
  - Security
  - Privacy
  - Usability
  - Reliability

- Massive parallelism greater chance (this time) if
  - Measure of success is SPUR vs. only cost-perf
  - Uniprocessor performance improvement decelerates

**Other Implications**

- Need to revisit chronic unsolved problem
  - Parallel programming!! (Thanks again Andy)

- Implications for applications:
  - Computing power >>> CDC6600, Cray XMP (choose your favorite) on an economical die inside your watch, cell phone or PDA
    - On your body health monitoring
    - Google + library of congress on your PDA
  - As devices continue to shrink…
    - The need for great HCI critical as ever!

**Administrivia**

- There IS discussion today
  - No lab tomorrow
  - Review session tomorrow instead of lecture
  - Make sure to talk to your TAs and get your labs taken care of.

- If you did well in CS3 or 61{A,B,C} (A- or above) and want to be on staff?
  - Usual path: Lab assistant ⇒ Reader ⇒ TA
  - Fill in form outside 367 Soda before first week of semester…
  - We strongly encourage anyone who gets an A- or above in the class to follow this path…

**Taking advantage of Cal Opportunities**

- “The Godfather answers all of life’s questions”
  - Heard in “You’ve got Mail”

- Why are we the #2 Univ in the WORLD?
  - Research, research, research!

- Whether you want to go to grad school or industry, you need someone to vouch for you! (as is the case with the Mob)

- Techniques
  - Find out what you like, do lots of web research (read published papers), hit OH of Prof, show enthusiasm & initiative (and get to know grad students!)

  http://research.berkeley.edu/

  So says the 2004 ranking from the “Times Higher Education Supplement”

**The Future for Future Cal Alumni**

- What’s The Future?

  - New Millennium
    - Internet, Wireless, Nanotechnology, …
    - Rapid Changes in Technology
    - World’s Best Education
    - Never Give Up!

  “The best way to predict the future is to invent it” – Alan Kay

  The Future is up to you!