
CS61C - Machine Structures

Lecture 23 - Pentium III, IV and other PC buzzwords

November 22, 2000

David Patterson

<http://www-inst.eecs.berkeley.edu/~cs61c/>

CS61C L23 x86 © UC Regents

1

Review (1/2)

° One way to define clock cycles:

Clock Cycles for program

= **Instructions for a program**
(called "**Instruction Count**")

x **Average Clock cycles Per Instruction**
(abbreviated "**CPI**")

° CPU execution time for program

= **Instruction Count** x **CPI** x **Clock Cycle Time**

CS61C L23 x86 © UC Regents

2

Review (2/2)

° Latency v. Throughput

° Performance doesn't depend on any single factor: need to know Instruction Count, Clocks Per Instruction and Clock Rate to get valid estimations

° User Time: time user needs to wait for program to execute: depends heavily on how OS switches between tasks

° CPU Time: time spent executing a single program: depends solely on design of processor (datapath, pipelining effectiveness, caches, etc.)

CS61C L23 x86 © UC Regents

3

Outline

° Intel 80x86 (Pentium) Instruction Set, History

° Administrivia

° Computers in the News

° Pentium III v. Pentium 4 v. Althon

° Typical PC

° Typical Mac

° Conclusion

CS61C L23 x86 © UC Regents

4

Intel History: ISA evolved since 1978

° 8086: 16-bit, all internal registers 16 bits wide; no general purpose registers; '78

° 8087: + 60 Fl. Pt. instructions, (Prof. Kahan) adds 80-bit-wide stack, but no registers; '80

° 80286: adds elaborate protection model; '82

° 80386: 32-bit; converts 8 16-bit registers into 8 32-bit general purpose registers; new addressing modes; adds paging; '85

° 80486, Pentium, Pentium II: + 4 instructions

° MMX: + 57 instructions for multimedia; '97

° Pentium III: +70 instructions for multimedia; '99

° Pentium 4: +144 instructions for multimedia; '00

CS61C L23 x86 © UC Regents

5

MIPS

vs.

80386

° Address: 32-bit

° 32-bit

° Page size: 4KB

° 4KB

° Data aligned

° Data **unaligned**

° Destination reg: Left

° Right

•add \$rd,\$rs1,\$rs2

•add %rs1,%rs2,%rd

° Regs: \$0, \$1, ..., \$31

° %r0, %r1, ..., **%r7**

° Reg = 0: \$0

° **(n.a.)**

° Return address: \$31

° **(n.a.)**

CS61C L23 x86 © UC Regents

6

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

CS81C L23 x86 © UC Regents

7

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

CS81C L23 x86 © UC Regents

8

MIPS vs. Intel 80x86

- MIPS: “**fixed-length instructions**”
 - All instructions same size, e.g., 4 bytes
 - simple hardware ⇒ performance
 - branches can be multiples of 4 bytes
- x86: “**variable-length instructions**”
 - Instructions are multiple of bytes: 1 to 17;
⇒ small code size (30% smaller?)
 - More Recent Performance Benefit: better instruction cache hit rates
 - Instructions can include 8- or 32-bit immediates

CS81C L23 x86 © UC Regents

9

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, ...

CS81C L23 x86 © UC Regents

10

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 - 4000000

CS81C L23 x86 © UC Regents

11

Instructions: MIPS vs. 80x86

◦ addu, addiu	◦ addl
◦ subu	◦ subl
◦ and, or, xor	◦ andl, orl, xorl
◦ sll, srl, sra	◦ sall, shrl, sarl
◦ lw	◦ movl mem, reg
◦ sw	◦ movl reg, mem
◦ mov	◦ movl reg, reg
◦ li	◦ movl imm, reg
◦ lui	◦ n.a.

CS81C L23 x86 © UC Regents

12

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]`

CS81C L23 x86 © UC Regents

13

Branch 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: `<`, `<=`, `>`, `>=`, `==`, `!=`

CS81C L23 x86 © UC Regents

14

Branch: MIPS vs. 80x86

◦ <code>beq</code>	◦ <code>(cmpl;) je</code> if previous operation set condition code, then <code>cmpl</code> unnecessary
◦ <code>bne</code>	◦ <code>(cmpl;) jne</code>
◦ <code>slt; beq</code>	◦ <code>(cmpl;) jlt</code>
◦ <code>slt; bne</code>	◦ <code>(cmpl;) jge</code>
◦ <code>jal</code>	◦ <code>call</code>
◦ <code>jr \$31</code>	◦ <code>ret</code>

CS81C L23 x86 © UC Regents

15

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)
X     jne .Exit
8     addl %esi,%edx
6     j .Loop
.Exit:
```

Note: `cmpl` replaces `sll`, `add`, `lw` in loop

CS81C L23 x86 © UC Regents

16

Administrivia: Rest of 61C

•Rest of 61C slower pace

- no more homeworks, projects, labs
W 11/24 X86, PC buzzwords and 61C; RAID Lab

W 11/29 Review: Pipelines; Feedback "lab"
F 12/1 Review: Caches/TLB/VM; Section 7.5

M 12/4 Deadline to correct your grade record

W 12/6 Review: Interrupts (A.7); Feedback lab
F 12/8 61C Summary / Your Cal heritage / HKN Course Evaluation

Sun 12/10 Final Review, 2PM (155 Dwinelle)
Tues 12/12 Final (5PM 1 Pimintel)

CS81C L23 x86 © UC Regents

17

Computers in the News

◦ Need More CPU Speed? Henry Norr, November 20, 2000, S.F. Chronicle

"Stand by to duck and cover -- you're about to be barraged by a new wave of clock-speed and performance claims from the leading makers of PC processors.

Today's release of the Pentium 4, running at up to 1.5 GHz, will put Intel back in the lead in the gigahertz (formerly megahertz) derby over rival Advanced Micro Devices and its Athlon chip. With standard benchmarks and real-life applications, the question is cloudier -- basically, it all depends on what test you use -- but Intel will no doubt be spending millions to promote its chip's advantages."

CS81C L23 x86 © UC Regents

18

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

CS81C L23 x86 © UC Regents

19

Unusual features of 80x86

- cl is the old count register, & can be used to repeat an instruction; it is 8 rightmost bits of ecx
- Used by shift to get a variable shift; uses cl to indicate variable shift
 - movl (%esi),%ecx # ecx = M[esi]
 - sall %cl,%eax,%ebx # ebx << ecx
- Positive constants start with \$; regs with %
 - cmpl \$999999,%edx
- 16-bits called **word**; 32-bits **double word** or **long word** (**halfword** and **word** in MIPS)

CS81C L23 x86 © UC Regents

20

Unusual features of 80x86: Floating Pt.

- Floating point uses a separate stack; load, push operands, perform operation, pop result
 - fldl (%esp)
 - # fpstack = M[esp],
 - # convert integer to FP
 - flds -8000048(%ebp)
 - # push M[ebp-8000048]
 - fsubp %st,%st(1)
 - # subtract top 2 elements
 - fstps -8000048(%ebp)
 - # M[ebp-8000048] = difference

CS81C L23 x86 © UC Regents

21

MIPS vs. Intel 80x86 Operations

- MIPS, HP-PA: “**fixed-length operations**”
 - All operations on same data size: 4 bytes; whole register changes
 - Goal: simple hardware and high performance
- x86: “**variable-length operations**”
 - Operations are multiple of bytes: 1, 2, 4
 - Only part of register changes if op < 4 bytes
 - Condition codes are set based on width of operation for Carry, Sign, Zero

CS81C L23 x86 © UC Regents

22

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

CS81C L23 x86 © UC Regents

23

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

CS81C L23 x86 © UC Regents

24

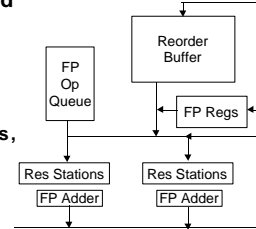
Hardware support

- **Out-of-Order execution:** allow a instructions to execute before branch is resolved (“HW undo”)
- When instruction no longer speculative, write results (**instruction commit**)
- Fetch in-order, execute out-of-order, commit in order

Hardware for out of order 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

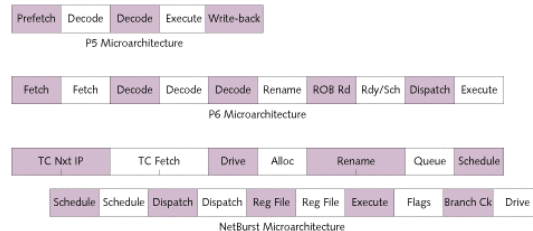
Pentium 4

- 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
- Caches
 - Pentium III: L1I 16KB, L1D 16KB, L2 256 KB
 - Pentium 4: L1I 8 KB, L1D 8 KB, L2 256 KB
 - Block size: PIII 32B v. P4 128B

Pentium 4 features

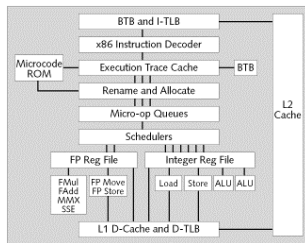
- Multimedia instructions 128 bits wide vs. 64 bits wide => 144 new instructions
 - When used by programs??
- Instruction Cache holds micro-operations vs. 80x86 instructions
 - no decode stages of 80x86 on cache hit
 - called “trace cache” (TC)
- Using RAMBUS DRAM
 - Bandwidth faster, latency same as SDRAM
 - Cost 3X vs. SDRAM

Pentium, Pentium Pro, Pentium 4 Pipeline



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

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

CS81C L23 x86 © UC Regents

31

Pentium III v. Pentium 4 in benchmarks

- PC World magazine, Nov. 20, 2000
 - WorldBench 2000 benchmark (business)
 - P4 score @ 1.5 GHz: 164 (bigger is better)
 - PIII score @ 1.0 GHz: 167
 - AMD Althon @ 1.2 GHz: 180
 - (Media apps do better on P4 v. PIII)
- S.F. Chronicle 11/20/00: "... the challenge for AMD now will be to argue that frequency is not the most important thing - precisely the position Intel has argued while its Pentium III lagged behind the Athlon in clock speed."

CS81C L23 x86 © UC Regents

32


Why?

- Instruction count is the same for x86
- Clock rates: P4 > Althon > PIII
- How can this be?

CS81C L23 x86 © UC Regents

33


Mac Internals

- CompUSA, \$1800, G4 Cube 
- Processor: PowerPC G4
- Processor Speed: 450 MHz
- Bus Speed: 100 MHz
- Cache Size: 1024 KB
- Memory Technology: SDRAM
- Installed Memory: 64 MB
- Maximum Memory: 1.5 GB
- Hard Drive Capacity: 20 GB
- Drive Controllers: IDE (ATA Ultra 66)
- DVD-ROM Read Speed: ? X
- Network Support: Ethernet (10/100 Mbps)

CS81C L23 x86 © UC Regents

35


PC Internals

- CompUSA, \$1400, HP 8766C 
- Processor: Intel Pentium III
- Processor Speed: 900 MHz
- Bus Speed: 100 MHz
- Cache Size: 256 KB
- Memory Technology: SDRAM
- Installed Memory: 128 MB
- Maximum Memory: 768 MB
- Hard Drive Capacity: 40 GB
- Drive Controllers: IDE (ATA)
- CD-ROM Read Speed: 24 X
- CD-ROM Rewrite Speed: 4 X
- DVD-ROM Read Speed: 12 X
- Network Support: Ethernet (10/100 Mbps)

CS81C L23 x86 © UC Regents

36

PC Internals

- Dell, \$2000, Dim. 8100 
- Processor: Intel Pentium 4
- Processor Speed: 1400 MHz
- Bus Speed: 400 MHz
- Cache Size: 256 KB
- Memory Technology: RDRDRAM
- Installed Memory: 128 MB
- Maximum Memory: 1024 MB
- Hard Drive Capacity: 40 GB
- Drive Controllers: IDE (ATA)
- DVD-ROM Read Speed: 12 X
- Network Support: (optional)

CS81C L23 x86 © UC Regents

37

"And in Conclusion.." 1/1

- **Once you've learned one RISC instruction set, easy to pick up the rest**
 - **ARM, Compaq/DEC Alpha, Hitachi SuperH, HP PA, IBM/Motorola PowerPC, Sun SPARC, ...**
- **Intel 80x86 is a horse of another color**
- **RISC emphasis: performance, HW simplicity**
- **80x86 emphasis: code size**
- **Pentium 4 goes to longer clock rate to increase clock frequency; what about Execution time? Clock rates is higher but so is CPI**