CSE 141-- Introduction to Computer Architecture

Jeff Brown
What is Computer Architecture?

• Hardware Designer
  – thinks about circuits, components, timing, functionality, ease of debugging
  “construction engineer”

• Computer Architect
  – thinks about high-level components, how they fit together, how they work together to deliver performance.
  “building architect”

CSE 141, S2'06

Jeff Brown
Why do I care?

• You may actually do computer architecture someday
• You may actually care about software performance someday
  – The ability of application programs, compilers, operating systems, etc. to deliver performance depends critically on an understanding of the underlying computer organization.
  – That becomes more true every year.
  – Computer architectures become more difficult to understand every year.
Which is faster?

for (i=0; i<N; i=i+1)
  for (j=0; j<N; j=j+1) {
    r = 0;
    for (k=0; k<N; k=k+1)
      r = r + y[i][k] * z[k][j];
    x[i][j] = r;
  }

for (jj=0; jj<N; jj=jj+B)
  for (kk=0; kk<N; kk=kk+B)
    for (i=0; i<N; i=i+1) {
      for (j=jj; j<min(jj+B-1,N); j=j+1)
        r = r + y[i][k] * z[k][j];
      x[i][j] = x[i][j] + r;
    }

CSE 141, S2'06

Jeff Brown
Which is faster?

load R1, addr1 → load R1, addr1
store R1, addr2 → add R0, R2 -> R3
add R0, R2 -> R3 → add R0, R6 -> R7
subtract R4, R3 -> R5 → store R1, addr2
add R0, R6 -> R7 → subtract R4, R3 -> R5
store R7, addr3 → store R7, addr3
Which is faster?

loop1: add ...
    load ...
    add ...
    bne R1, loop1

loop2: add ...
    load ...
    bne R2, loop2

loop1: add ...
    load ...
    add ...
    bne R1, loop1
    nop
    nop

loop2: add ...
    load ...
    bne R2, loop2
Administration

• Instructor: Jeff Brown
• Who are you?
• TAs:
  – Will Chang
  – Steve Checkoway
• Grading
• Integrity
• Course workload
What is Computer Architecture?

Computer Architecture = Machine Organization + Instruction Set Architecture

What the machine looks like

How you talk to the machine
How to Speak Computer

High Level Language Program

Compiler

Assembly Language Program

Assembler

Machine Language Program

Machine Interpretation

Control Signal Spec

temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;

lw $15, 0($2)
lw $16, 4($2)
sw $16, 0($2)
sw $15, 4($2)

10001100011000100000000000000000
10001100111100100000000000001000
10101100111100100000000000000000
10101100011000100000000000001000

ALUOP[0:3] <= InstReg[9:11] & MASK
The Instruction Set Architecture

- that part of the architecture that is visible to the programmer
  - opcodes (available instructions)
  - number and types of registers
  - instruction formats
  - storage access, addressing modes
  - exceptional conditions
The Instruction Set Architecture

° is the agreed-upon interface between all the software that runs on the machine and the hardware that executes it.
The Instruction Execution Cycle

- **Instruction Fetch**: Obtain instruction from program storage
- **Instruction Decode**: Determine required actions and instruction size
- **Operand Fetch**: Locate and obtain operand data
- **Execute**: Compute result value or status
- **Result Store**: Deposit results in storage for later use
- **Next Instruction**: Determine successor instruction
Key ISA decisions

- operations
  - how many?
  - which ones

- operands
  - how many?
  - location
  - types
  - how to specify?

- instruction format
  - size
  - how many formats?

\[
y = x + b
\]

\[
\text{add } r1, r2, r5
\]
Examples of ISAs

• Alpha AXP
• Intel IA-32 ("x86")
• VAX
• MIPS
• SPARC
• IBM 360
• Intel IA-64 ("Itanium", "IPF")
• PowerPC
Once you have decided on an ISA, you must decide how to design the hardware to execute those programs written in the ISA as fast as possible (or as cheaply as possible, or using as little power as possible, …).

This must be done every time a new implementation of the architecture is released, with typically very different technological constraints.
The Challenge of Computer Architecture

• The industry changes faster than any other.
• The ground rules change every year.
  – new problems
  – new opportunities
  – different tradeoffs
• It’s “all” about making programs run faster than the next guy’s machine. Or more efficiently.
The five classic components of computers

- Control
- Datapath
- Memory
- Input
- Output
Course Outline

I. Instruction Set Architecture
II. Computer System Performance and Performance Metrics
III. Computer Arithmetic and Number Systems
IV. CPU Architecture
V. Pipelining
VI. Superscalars
VII. The Memory/Cache Hierarchy
VIII. Parallel Machines
What you can expect to get out of this class

• to become conversant with computer architecture terms and concepts.
• to understand fundamental concepts in computer architecture and how they impact computer and application performance.
• to be able to read and evaluate architectural descriptions of even today’s most complex processors.
• to gain experience designing a working CPU completely from scratch.
• to learn experimental techniques used to evaluate advanced architectural ideas.
Key Points

- High-performance software requires a deep understanding of the underlying machine organization.
- The instruction set architecture defines how software is allowed to use the processor. Multiple computers with vastly different organizations and performance can share an ISA.
- Most every component in a computer system falls into one of five categories.