Topics

“Power tends to corrupt; absolute power corrupts absolutely.”

– Lord Acton

British historian, late 19th and early 20th centuries

• Viewpoint of a high-level language
• Viewpoint of the microarchitecture
• Example microarchitectures

Structured Computer Organization

Level 5: Problem-oriented language level
  Translation (compiler)

Level 4: Assembly language level
  Translation (assembler)

Level 3: Operating system machine level
  Partial interpretation (operating system)

Level 2: Instruction set architecture level
  Interpretation (microprogram) or direct execution

Level 1: Microarchitecture level
  Hardware

Level 0: Digital logic level

Java™ as a High-Level Language

• The Java programming language is strongly typed
  – Every variable and every expression has a type that is known at compile time

• Data types limit the following:
  – Values that a variable can hold
  – Values that an expression can produce
  – Operations supported on those values
  • Determines the meaning of those operations

• Strong typing helps detect errors at compile time
Primitive Types

- Predefined by the Java programming language and named by a reserved keyword
- Includes the boolean type
  - Restricted to TRUE and FALSE values
- Includes integral types
  - byte, short, int, and long, whose values are 8-bit, 16-bit, 32-bit, and 64-bit signed two’s-complement integers, respectively
  - char, whose values are 16-bit unsigned integers representing Unicode characters
- Includes floating-point types
  - float and double, which are associated with the 32-bit single-precision and 64-bit double-precision IEEE Floating Point Standard 754

Error Checking by the Compiler

- Verifies each operation has proper inputs and outputs
  - e.g. Boolean expressions used in control flow statements
  - e.g. arithmetic operations are of compatible types
  - e.g. strings (of characters) are properly displayed
- Some operations don’t make sense
  - e.g. (char x char)
  - e.g. if (byte) then ...
  - e.g. int = double + double

Operations on Data Types

- Operators on boolean Values
  - Includes relational operators and logical operators
- Operators on Integral Values
  - Includes numerical comparison, arithmetic operators, increment and decrement, bitwise logical and shift operators
- Operators on Floating-Point Values
  - Includes numerical comparison, arithmetic operators, increment and decrement

Instruction Set Design

- Interface between software and hardware
- Both compilers and hardware must understand ISA
  - Compilers translate high-level language into object code
  - Hardware must directly execute or interpret ISA
ISA: What Must be Specified?

- Instruction format or encoding
  - how is it decoded?
- Location of operands and result
  - where other than memory?
  - how many explicit operands?
  - how are memory operands located?
  - which operands can or cannot be in memory?
- Data type and size
- Operations
  - what are supported
- Successor instruction
  - jumps, conditions, branches
- fetch-decode-execute is implicit!

Typical Operations (little change since 1960)

Data Movement
- Load (from memory)
- Store (to memory)
- Memory-to-memory move
- Register-to-register move
- Input (from I/O device)
- Output (to I/O device)
- Push, pop (to/from stack)

Arithmetic
- Integer (binary + decimal) or FP
  - Add, Subtract, Multiply, Divide

Shift
- Shift left/right, rotate left/right

Logical
- Not, and, or, set, clear

Control (Jump/Branch)
- Unconditional, conditional

Subroutine Linkage
- Call, return

Interrupt
- Trap, return

Synchronization
- Test & set (atomic r-m-w)

String
- Search, translate

Graphics (MMX)
- Parallel subword ops (4 16-bit add)

Data Types

Bit: 0, 1

Bit String: sequence of bits of a particular length
- 4 bits is a nibble
- 8 bits is a byte
- 16 bits is a half-word
- 32 bits is a word
- 64 bits is a double-word

Character:
- ASCII 7 bit code
- UNICODE 16 bit code

Decimal:
- Digits 0-9 encoded as 0000b thru 1001b
- Two decimal digits packed per 8 bit byte

Integers:
- 2's Complement

Floating Point:
- Single Precision
  - Exponent
  - Mantissa
- Double Precision
  - Exponent
  - Mantissa
- Extended Precision
  - Exponent
  - Mantissa
Data Storage in Memory

- Memory is just a collection of bits
- The user defines the context or meaning
  - ISA level instructions could use this data after placed in a register

Big Endian

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0x57494C4C</td>
</tr>
<tr>
<td>104</td>
<td>0x49414D20</td>
</tr>
<tr>
<td>108</td>
<td>0x524F4249</td>
</tr>
<tr>
<td>112</td>
<td>0x4E534F4E</td>
</tr>
</tbody>
</table>

Example: String Layout

- String layout starting at word address 100
  char a[16] = "WILLIAM ROBINSON"
- Each byte contains the ASCII representation
  - see Figure 2-41 on page 110

Big Endian

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0x57494C4C</td>
</tr>
<tr>
<td>104</td>
<td>0x49414D20</td>
</tr>
<tr>
<td>108</td>
<td>0x524F4249</td>
</tr>
<tr>
<td>112</td>
<td>0x4E534F4E</td>
</tr>
</tbody>
</table>

Big Endian

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>W I L</td>
</tr>
<tr>
<td>104</td>
<td>I A M</td>
</tr>
<tr>
<td>108</td>
<td>R O B</td>
</tr>
<tr>
<td>112</td>
<td>N S O</td>
</tr>
</tbody>
</table>

From Program 2

- Sample data set

```
1111 1111 1111 1111 1111 1111 1111 1111
1111 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
1111 1000 1000 1000 1000 1000 1000 1000
1111 0000 0000 0000 0000 0000 0000 0000
```

- What is represented by this?
  - Could view it as 2’s complement integer
  - Could parse it into ASCII characters
  - Could be 32-bit color values

Absolute Power

- The ISA has access to all the hardware features of the microarchitecture
  - 1:1 mapping of assembly to ISA
- Does not restrict operands to data types
  - e.g. a register that holds a 2’s complement integer could also be interpreted as unicode characters
- Compilers are designed to prevent this
  - But we’re not working at that level!
Computer Quotes

"0x2B | ~0x2B." — Hamlet

"There are 10 kinds of people in the world, those that understand binary and those that don't."

"If you give someone a program, you will frustrate them for a day; if you teach them how to program, you will frustrate them for a lifetime."

http://www.geocities.com/SiliconValley/Orchard/4218/humor/quotes.html

Summary

• Data forwarding hardware can eliminate stalls due to true data dependencies

• Scoreboarding is a technique to monitor the dynamic scheduling of instructions

• A typical machine will have in-order issue, out-of-order execution, and in-order completion

Example Microarchitectures

• IA-32 (Pentium II)
  – CISC

• Version 9 SPARC (UltraSPARC II)
  – RISC

• Java Virtual Machine (picoJava II)
  – Stack machine

Microarchitecture for IA-32 ISA
Microarchitecture for IA-32 ISA

Microarchitecture for Version 9 SPARC ISA

Microarchitecture for IA-32 ISA

Microarchitecture for Version 9 SPARC ISA
Microarchitecture for JVM ISA

Instruction Folding

Example Architectures

- Pentium II  
  - CISC
- UltraSPARC II  
  - RISC
- picoJava II  
  - Stack machine
Similarity Among Architectures

Implementation of the execution unit
- Use an opcode, 2 source registers, and a destination register (3-register format)
- Execute a micro-instruction in one cycle
- Deep pipelines and branch prediction
- Split I-cache and D-cache

Difference Among Architectures

How the instructions get to the execution unit
- Pentium II has to break CISC instructions into 3-register format
- picoJava II must fold instructions into 3-register format
- UltraSPARC II already has the 3-register format