Computer Basics/Algorithms

INFO/CSE 100, Spring 2006
Fluency in Information Technology

http://www.cs.washington.edu/100
Readings and References

• Reading
  » Fluency with Information Technology
  • Chapters 9, 10
Overview

• During this quarter, we're looking at the actual workings of computer systems

• Organized as “layers of abstraction”
  » application programs
  » higher level languages: Javascript, SQL, …
  » operating system concepts
  » bits, bytes, assembly language
  » transistors, electrons, photons
Layers of Abstraction

- At any level of abstraction, there are
  - elements at that level
  - the building blocks for those elements

- Abstraction
  - isolates a layer from changes in the layer below
  - improves developer productivity by reducing detail needed to accomplish a task
  - helps define a single architecture that can be implemented with more than one organization
Architecture & Organization

• **Architecture (the *logical definition*)**
  » defines elements and interfaces between layers
  » Instruction Set Architecture
    • instructions, registers, addressing

• **Organization (the *physical implementation*)**
  » components and connections
  » how instructions are implemented in hardware
  » many different organizations can implement a single architecture
Computer Architecture

• Specification of how to program a specific computer family
  » what instructions are available?
  » how are the instructions formatted into bits?
  » how many registers and what is their function?
  » how is memory addressed?

• Some examples architectures
  » IBM 360, 370, …
  » PowerPC 601, 603, G5, …
  » Intel x86 286, 386, 486, Pentium, …
  » MIPS R2000, R3000, R4000, R5000, …
Computer Organization

• Processor
  » Data path (ALU) manipulate the bits
  » The control controls the manipulation

• Memory
  » cache memory - smaller, higher speed
  » main memory - larger, slower speed

• Input / Output
  » interface to the rest of the world
A Typical Organization

main memory

processor/memory bus

processor

I/O bus

hard disk
floppy disk
CDROM drive
serial ports
network interface
Anatomy of a Computer

Processor
- ALU
- Control
- Memory

Input
- Mouse
- Keyboard
- Scanner
- Hard Disk
- Floppy Disk

Output
- Monitor
- Printer
- Speakers
Computers…

• Deterministically execute instructions
  » “Deterministically” means that when a computer chooses the next instruction to perform it will make the choice the same way each time
  » Given the program instructions and the current input, you can always predict exactly which instruction will be executed next and what it will do

Computers have no free will and they are not random!
Fetch/Execute Cycle

Computer = instruction execution engine
   » The fetch/execute cycle is the process that executes instructions

Instruction Fetch (IF)
Instruction Decode (ID)
Data Fetch (DF)
Instruction Execution (EX)
Result Return (RR)
Memory ...

Programs and the data they operate on must be in the memory while they are running.

Memory locations

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>o</td>
<td>D</td>
<td>a</td>
<td>w</td>
<td>g</td>
<td>s</td>
<td>!</td>
<td>!</td>
<td>0</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

byte=8 bits

memory addresses

memory contents
Control

• The Fetch/Execute cycle is hardwired into the computer’s control, i.e. it is the actual “engine”
• Depending on the Instruction Set Architecture, the instructions say things like
  » Put in memory location 20 the contents of memory location 10 + contents of memory location 16
  » The instructions executed have the form ADDB 10, 16, 20
    • Add the bytes from memory address 10 and memory address 16 and store the result in memory address 20
The Arithmetic/Logic Unit does the actual computation

Depending on the Instruction Set Architecture, each type of data has its own separate instructions

- **ADD** : add words
- **ADDB** : add bytes
- **ADDBU** : add bytes unsigned
- **ADDH** : add half words
- **ADDHU** : add halves unsigned
- **ADDU** : add words unsigned
- **ADDS** : add short decimal numbers
- **ADDL** : add long decimal numbers

Most computers have only about a 100-150 instructions hard wired
Input/Output

- Input units bring data to memory from outside world; output units send data to outside world from memory
  - Most peripheral devices are “dumb”, meaning that the processor assists in their operation
The PC’s PC

- The program counter (PC) tells where the next instruction comes from
  » In some architectures, instructions are always 4 bytes long, so add 4 to the PC to find the next instruction

Program Counter: 112

<table>
<thead>
<tr>
<th>112</th>
<th>113</th>
<th>114</th>
<th>115</th>
<th>116</th>
<th>117</th>
<th>118</th>
<th>119</th>
<th>120</th>
<th>121</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD 210,216,220</td>
<td>AND 414,418,720</td>
<td>OR</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Clocks Run The Engine

• The rate that a computer “spins around” the Fetch/Execute cycle is controlled by its clock
  » Current clocks run 2-3 GHz
  » The computer tries to do at least one instruction per cycle, depending on the instruction and the availability of memory contents
  » Modern processors often try to do more than one instruction per cycle

Clock rate is not a good indicator of speed anymore, because several things are happening every clock cycle
Programming

• Converting the complicated tasks we want the computer to do into simple instructions
  » Computers can be programmed to convert the complex into the simple
  » Computers only understand binary digits
• Developers created assembly language as a convenient form for instructions
  » For example; ADDD 2000, 8000, 4000
  » Translation from assembly to binary is called assembling
• High-level languages are used to complete more complex tasks easily
  » Translation from a programming language to assembly is called compilation
Algorithm

• Algorithm
  » a precise, systematic method to produce a desired result

• For example, the placeholder technique for deleting a short string except where it occurs in longer strings is an algorithm with an easy specification:

  \[
  \text{longStringWithShortStringInIt} \leftarrow \text{placeholder} \\
  \text{ShortString} \leftarrow e \\
  \text{placeholder} \leftarrow \text{longStringWithShortStringInIt}
  \]
Properties of an Algorithm

• For an algorithm to be well specified it must have …
  » Inputs specified
    • The range of possible inputs is well defined
  » Outputs specified
    • The desired output is well defined
  » Definiteness
    • The steps to take are definite and understandable
  » Effectiveness
    • The steps must be possible to accomplish
  » Finiteness
    • A processor that follows the algorithm will eventually finish
Communicating…

• People can fill in missing steps, but can get swamped by lots of details and clutter
• Computers cannot fill in missing steps, but can manage lots and lots of detail without error
• What helps when communicating with computers?
  » Be organized and consistent in all the details
  » Invent abstractions to help specify the basic ideas accurately and consistently
  » Analyze your algorithm and its implementation, because you won’t get to interact later
Example: Directions to the Bookstore

- The student operates at a higher level of abstraction with a richer vocabulary of shorthands
- An algorithm is a plan for how to accomplish a task
  » A program is an implementation of an algorithm
- Good algorithms (at any level of abstraction) require precision
Algorithm Analysis: What is it?

• What is an algorithm?
  » A sequence of steps that accomplishes a task

• Many different algorithms may correctly solve a given task
  » can it be implemented with available equipment?
  » will it complete within this lifetime?
  » will it require gigabytes of memory?
Algorithm Analysis: Why do it?

• Understand the mathematical fundamentals needed to analyze algorithms
• Learn how to compare the efficiency of different algorithms in terms of running time and memory usage
• Study a number of standard algorithms for data manipulation and learn to use them for solving new problems
Programs vs Algorithms

• A program is an algorithm specialized to a particular situation
  » an Algorithm
    longStringWithShortStringInIt ← placeholder
    ShortString ← e
    placeholder ← longStringWithShortStringInIt
  » a Program that implements the Algorithm
    ↓↓ ← #  // replace double <newlines> with <#>
    ↓ ← e   // delete all single < newlines>
    # ← ↓↓  // restore all double <newlines>
Programming as Communication

• When we write a program, we are communicating with
  » the computer
  » other people

• The computer reads our program as the set of instructions that it should perform
  » It just needs to know how, not why

• Other people read our programs to understand how and why
  » Programs that don't work (bugs)
  » Program evolution - new features
  » Performance improvement
An algorithm to alphabetize CDs

define variable named *Artist*

use *Artist* to refer to the name of the group that made a CD

for all slots in the rack starting at one end

call the current slot *alpha*

for all the remaining slots in the rack

call the next slot *beta*

Exchange?

If *Artist* of the CD in the *beta* slot is earlier in the alphabet than the *Artist* of the CD in the *alpha* slot, interchange the CDs

next *beta*

next *alpha*

done
Another sorting demo…

• With plastic bottles!
Summary

• We can figure out many algorithms on our own, abstracting from specific cases
• We can learn from others who have studied particular algorithms in depth
• We abstract parts of an algorithm or program to understand them
  » Thinking of how the program works and reasoning about its properties allows us to know why an algorithm works … and then we can get the computer to do it for us