Announcements

New lecture room PH A18A
No lecture on Sept. 12th
Reader 1 is complete
Recommended habit
  • Check web page for schedule
  • Read in advance
  • Form reading groups to discuss papers

You must have all received email from the class list
How to Average

Arithmetic mean
- sum of all execution times/number
- valid only if all programs run equally often

Weighted arithmetic mean
- “weight” factors to give equal importance to all programs

Other Averages

E.g., 30 mph for first 10 miles, 90 mph for next 10 miles
What is the average speed?

Average speed = (30 + 90)/2 = 60 mph!

Average speed = total distance / total time
= 20 / (10/30 + 10/90)
= 45 mph
**Harmonic Mean**

Harmonic mean of rates = $1/\{\Sigma 1/rate(i)/n\}$

Use harmonic mean if forced to start and end with rates

---

**Dealing with Ratios**

<table>
<thead>
<tr>
<th>Program 1</th>
<th>Machine A</th>
<th>Machine B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 2</td>
<td>1000</td>
<td>100</td>
</tr>
</tbody>
</table>

If we take ratios with respect to Machine A (Machine B)

<table>
<thead>
<tr>
<th>Program 1</th>
<th>Machine A</th>
<th>Machine B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 2</td>
<td>1 (0.1)</td>
<td>10 (1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Program 2</th>
<th>Machine A</th>
<th>Machine B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 2</td>
<td>1 (10)</td>
<td>0.1 (1)</td>
</tr>
</tbody>
</table>
Dealing with Ratios (Cont.)

Averages different depending on which is the reference

Geometric mean of ratios = \[ \sqrt[n]{\prod_{1}^{n} \text{ratio}(i)} \]

Use geometric mean if forced to use ratios

Independent of reference machine

In the example, mean for machine A and B is 1

- normalized with respect to either machine

But.....

Geometric mean of ratios is not proportional to total time

Arithmetic mean in example says machine B is 9.1 times faster

Geometric mean says they are equal

If we took total execution time, A an B are equal only if

- program 1 is run 100 times more than program 2

Geometric mean will mispredict for three or more machines

- Use AM for times, HM for rates, GM for ratios
Burks, Goldstine, and von Neumann

Classic paper:
- many observations are still true
- most historians credit Eckert and Mauchly for the ideas

“It is evident that the machine must be capable of storing in some manner not only the data but also the instructions which govern the actual machine.”

“Conceptually we have discussed above two different forms of memory: storage of numbers and storage of orders. The memory organ can be used to store both numbers and orders.”

Arithmetic:
- binary arithmetic
- two’s complement
- iterative carry
- iterative multiply (carry-save adders)
- rounding vs jamming
- non-restoring division
- no floating-point. Why?
Burks, Goldstine, and von Neumann

Control:
- 40-bit data
- 20-bit instructions
  - 8-bit opcodes
  - 12-bit addresses
- basic instructions
  - conditional and unconditional branches
  - data transfer
  - ALU and shift
  - store into orders. Why?

Instruction Sets

“Instruction set architecture (ISA) is the structure of a computer that a machine language programmer (or a compiler) must understand to write a correct (timing independent) program for that machine”

IBM introducing 360 in 1964

Instruction set aspects
- operands
- memory issues
- operations (mostly control)
- assumes you know what simple pipelines look like (EE 365)
Operand Storage

Why in the processor?
- faster access
- shorter address

Accumulator
- less hardware
- high memory traffic
- likely bottleneck

Stack
- code density
- bottleneck while pipelining (why?)
- e.g., JAVA VM

Operand Storage

Registers (8 to 256 words)
- flexible
- register must be named
- code density
- name space
Caches vs. Registers

Registers
✓ faster (no addressing modes, no tags)
✓ deterministic access time
✓ easier to add ports
✓ short identifier
✗ must save/restore on procedure calls
✗ can not take address of register
✗ fixed size (can not fit FP, strings, structures/records)
✗ compilers must manage (an advantage?)

Registers vs. Caches

Can we add registers?
✓ Hold operands longer (reducing memory traffic & runtime)
✗ longer register specifiers (except with register windows)
✗ slower registers
✗ more state slows context switches
**Operands for ALU Instructions**

ALU instructions combine operands

Number of explicit operands

- two, \( r_i = r_j \) op \( r_j \)
- three, \( r_k = r_i \) op \( r_j \)

Operands in registers or memory

- any combo, orthogonal, e.g., VAX
- at least one register, not orthogonal, e.g., IBM 360/370
- all registers, orthogonal but loads/stores, e.g., Cray, RISCs