Scientific Computing: Not Only for the NERDS

Zhongming Qu March 13, 2014

Contents

1 Brief Introduction

Computers usherred us into this so-called information era. Computers are not new. Broadly speaking, any 'thing' that assists people in calculation can be count as computer, or a simpler version, calculator. Computers have really changed the way we do things. We use them everywhere. And in most situations them are fairly easy to use.

The purpose of this lecture is, however, to 1) reveal the dirty internals of computers as much as possible, 2) help build an adequate level of appreciation of the complexity of modern computer systems, 3) demystify some common incorrect beliefs regarding computers, and 4) inspire ideas. This lecture aims at a very general audience. No prior knowledge is required to understand anything. Absolute newbies, computer lovers, amateur and semi-expert programmers, and maybe even computer experts can have some refreshment on their existing perspectives, if any, regarding computers.

2 What's in a CPU?

2.1 1+1=2 Revisited

2.2 Operator and Operands

[graph]

2.3 Compoud Arithmetic Expressions

[1+3*5]

2.4 Tree Structure of Compound Arithmetic Expressions

[graph:1+3*5]

2.5 Parsing/Flattening Trees of Arithmetic Expressions

[lecture:1+3*5]

2.6 Machine Representation of Numbers

[int, long int, unsigned int]
[float, double, long double]
[32bit, 64bit]
[IEEE standard]

2.7 Instructions and Registers

[instructions: add, mul, div]
[registers: graph from internet]
[registers may be wider than the data]
[registers may have special uses]
[instruction sets]

2.8 Vendors and Instruction Sets

[MMX,SEE,SEE2,SEE3,SEE4,SEE4.1,SEE4.2,AVX,AVX2]
[3DNow!]
[MIPS,MPIS64]
[PPC]

2.9 Machine Language

=instruction
[machine dependent]

2.10 Assembly Language and Assemblers

[one-one map between instruction and human readable words]
[machne dependent, but also sometimes maybe software/OS/convention dependent]

2.11 Ex ASM: 1+3*5

[cpu/cpdexp.exe]

2.12 Registers and Caches

[Registers' schematic plot from wiki]
[Cache size exmaples]
[RAM, disk, speed difference]

3 High Level Languages

3.1 A Taste of C

[hello world example]

3.2 Compiling and Compilers

[Compilation of each source file generates one object file]
[Optimizing compiling is artificial intelligence]
[Compiling usually involves at least three phases: parsing, compiling and assembling]
[Show the compiler theory book]
[Design of languages]
[Philosophy of C]

3.3 Linking and Linker

[Linking of one or more object file(s) generates one executable, which is indeed machine instructions]

3.4 Libraries, Shared Objects

[static library vs. dynamic library]
[Windows dll, Linux so, MaxOSX dylib]
[good libraries, bad libraries]

3.5 Fortran

[ASM helloworld exmaple]
[After compilation, object files are same with C]
[Philosophy of Fortran]

3.6 Ex ASM: Fortran & C

[linking-cfortran/addtwo.exe]

3.7 C++ vs. C: Huge Difference

[name mangling: theory and practice]
[mixed linkage: extern "C"]

3.8 Ex ASM: C & C++

[linking-ccpp/addtwo.exe]

3.9 Java

[java virtual machine (JVM)]
[java runtime environment (JRE)]
[java compiler javac generates java virtual machine instructions]

3.10 Ex Java

[java/helloworld.exe]

3.11 Python, Matlab, Mathematica, R

[Interpretative vs. Compiling]
[Interpreter vs. Compiler]
[Rule of Thumb of Interpretative Tools]

4 Compiler Optimazation

4.1 Two Versions of a Summation

[vec/addmany1.exe]
[vec/addmany2.exe]

4.2 Vectorization

[single instruciton multi-data (SIMD)]
[compiler flags, code directives, and dirty tricks]

4.3 Ex ASM: Vectorization

[vec/addmany1.exe]
[vec/addmany2.exe]

4.4 Two Versions of Scaling

[vec/scale1.exe]
[vec/scale2.exe]

4.5 Many Variables

[vec/manyvars.exe] (to exhaust registers)

4.6 Loop Unrolling

[vec/unroll.exe]

5 Memory Optimization

5.1 Accessing Memory: Cacheline

[unitstride.exe]
[stride.exe]
[data locality]

5.2 Accessing Meory; TLB

[tlbmiss.exe]
[tlbhit.exe]

5.3 Data Storage

[big O notation]
[operation cost: init, insert, append, at, pop, delete, destroy]
[storage cost]

5.4 Vector: Sequential Storage and Random Access

[hard to add/delete, easy to access]

5.5 Linked List: Random Storage and Sequential Access

[easy to add/delete, hard to access]

5.6 Hash Tables

[near constant access, insert, delete time cost]

5.7 Trees, Packed Trees

[general trees: sequential access]
[packed trees: random access]

5.8 Stack, Queue, Vector

6 Parallelism: OverView

6.1 Instruction and Thread

6.2 Thread and Process

6.3 Thread and Core

[different from thread and process]

Appendices