Performance:More Than Just Speed

We are not programming in 1969 (or even 1984) anymore

by
Jon "maddog" Hall
Executive Director
Linux International
and
Board Chair
Linux Professional Institute

Who Am I?

- *Half* Electrical Engineer, *Half* Business, *Half* Computer Software
- In the computer industry since *1969*
  - Mainframes 5 years
  - Unix since 1980
  - Linux since 1994
- Companies (mostly large): Aetna Life and Casualty, Bell Labs, Digital Equipment Corporation (DEC, DECUS), SGI, IBM, Linaro, WIT
- Organizations: USENIX, Linux International, Linux Professional Institute
- ***Programmer***, Systems Administrator, Systems Engineer, Product Manager, Technical Marketing Manager, ***University Educator***, Author, Businessperson, Consultant
- Taught OS design and compiler design
- *Extremely* large systems to *extremely* small ones
- Pragmatic
- Vendor *and* an "open source" customer

Who Needs Performance?

- “CPUs are fast enough”
  - I have been hearing that for over 50 years...
- “JAVA is the only language people need”
- “Nobody codes in assembler language any more”
- “Virtual machines make architecture knowledge obsolete”

Performance

- "Real" problems
  - Petabytes of data, thousands of processors
- Real-time
  - REAL real-time
    - Lower those rods!
  - Linus and "soft real time"
- Cell Phone Apps
  - Saving battery life
- Saving the environment!
  - Only 9000 servers!
- "New" (or at least newly affordable) advancements
  - Field Programmable Gate Arrays (FPGAs)
  - Digital Signal Processors (DSPs)

To Write Really *Great* Code...

...you need to understand machine architecture and that includes machine/assembly language

Examples From My Past

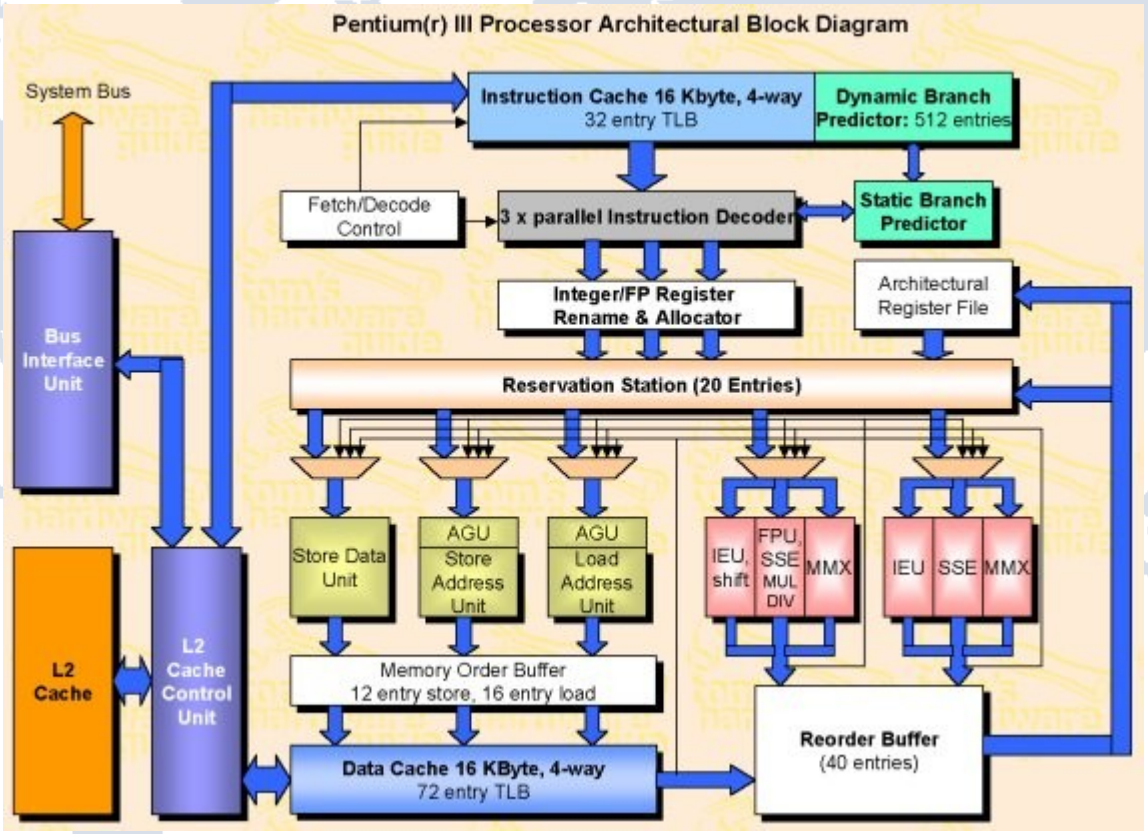
- Compiler errors (?!?)
- Cache
  - Digital Unix – ½ the size, 7% faster
  - 40 times (David Mossberger-Tang)
  - 220 times (not 200%) – PDP-11/70 + RSTS-E
- Tapes (OMG!) - start/stop and streaming tapes

Today

- (Some) College Students learning
  - "Microsoft Office and Oracle" instead of "Office Systems and Databases"
  - JAVA and "IT/TI" instead of Assembly and Operating Systems
  - Virtual machines instead of "real iron"
- High school students
  - Games and HTML

How Most High School Students See Computers

How Computers Really Look



Ways of Holding Numbers

- EBCDIC/ASCII Codes
  - One “Digit” per byte
- Packed Decimal
  - Two “Digits” per byte
  - Just a “squidge less”
- Binary
- Floating point (mantissa and exponent)

Which One Is Used for Indexing?

Please do not say “EBCDIC/ASCII”

Real Life Effects of “Dumb Down”

“Incoming freshmen know less than they knew 20 years ago” - Raspberry Pi Foundation

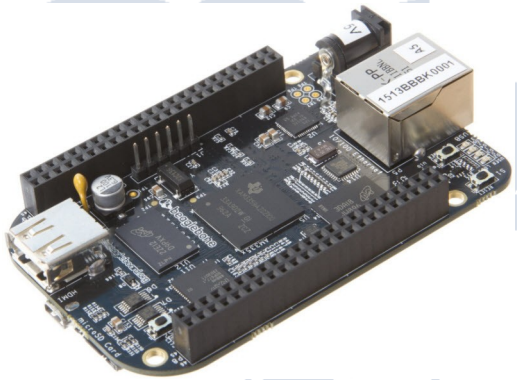
The Raspberry Pi was created to help fix this problem.

Raspberry Pi – 35 USD

- Single Core ARM – 700Mhz
- ½ Gbyte RAM
- 3D GPU
- One HDMI port
- USB 2.0
- 10/100 Ethernet
- 802.11 b/g Wireless
- Bluetooth 4.0
- GPIO Pins
- 3W

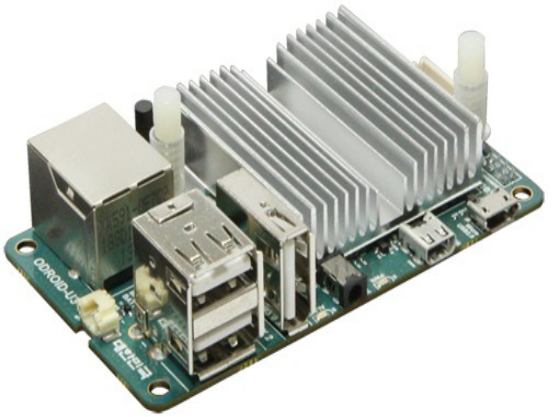
- Four Core 64-bit ARM8 – 1.5GHz
- Two GB RAM (4 and 8 GB available)
- 3D GPU
- Two micro-HDMI ports (4Kp60)
- USB – two 2.0, two 3.0
- Gbit Ethernet
- 802.11 b/ac Wireless (2.4 and 5.0 Ghz)
- Bluetooth 5.0 BLE
- GPIO Pins
- 13.5 W

Many Little Computers: 45 USD – 199 USD

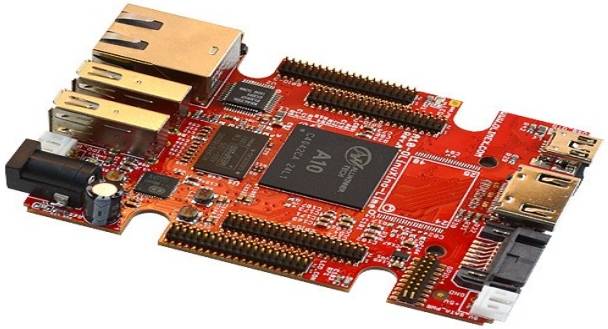


BeagleBoneBlack

Hackberry 10



ODROID-U3



OlimoX - LIME

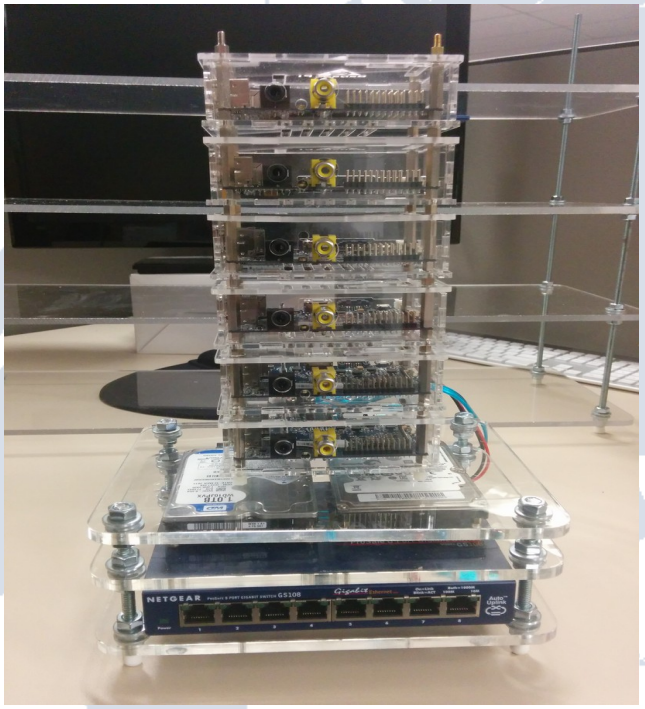
Pandaboard

Galileo

Why Do I Show You All This?

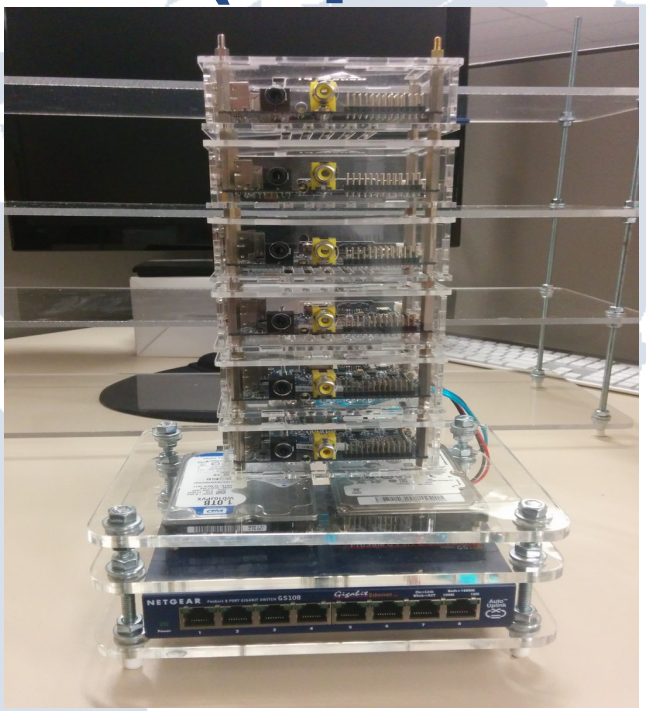
Because Of THIS!

- 12 ARMv7 Cores at 1 GHz each
- 6 GBytes of RAM
- 6 HDMI ports
- 6 SATA ports (currently driving two disks)
- IR on board
- 2 TB SATA disk
- 8 Port Gbit ETHERNET
- 70 Watts
- Fits in standard briefcase



Because Of THIS (Updated)!

- 24 ARMv8 Cores at 1.6 GHz each in RPi 4
- 48 GBytes of RAM (8GB each)
- 12 HDMI ports
- 12 USB-3 ports (currently two driving two SATA disks)
- 2 TB SATA disk (RAID)
- 8 Port Gbit ETHERNET
- 100 Watts
- Fits in standard briefcase



Why Is This Interesting?

- Can be used to teach
  - HPC computing
  - HA computing
  - heterogeneous computing (programming and systems administration)
- Very portable, can be assembled in minutes
- Very modular
- Prototype cost: 800 USD
  - Currently using “Raspberry Pi 4”
- Production cost: < 600-1000 USD
  - Could also use (6) new “Labrador/Model D”

GNU/Linux: Programming For The *Future*

- "Beowulf" supercomputers (1994)
  - Non Uniform Memory Architecture (NUMA)
  - Please *do not* build one out of RPi Zeros
    - Not a big one, anyway…..
- GPUs – not just for graphics anymore
- Field Programmable Gate Arrays (FPGA)
  - More efficient than GPU
  - More flexible than an ASIC
- Quantum computing – nooooooooooo!

Can we make our code *better*?

Summary

- Learn ***SOME*** assembly/machine language (any assembly language)
- Choose your algorithms and data carefully
- Use the right language for the right job
- Examine the assembly/machine language that is generated
- Speedups of “only” 2-3x are “ok”….you don't need 1000x (but that is really cool)

Resources

- "The Definitive Guide to GCC: 2nd Edition" by William von Hagen (Apress, 2006)
- "The Art of Debugging with GDB, DDD, and Eclipse" by Matloff and Salzmann (No Starch Press, 2008)
- "Valgrind 3.3: Advanced Debugging and Profiling for GNU/Linux applications" by Seward, Nethercote et. al. (Network Theory Ltd., 2008)

More Resources

- "ARM Assembly Language – an Introduction" by J.R. Gibson (Lulu, 2007)

Questions, Comments, Ideas?