CS-200
Computer Architecture

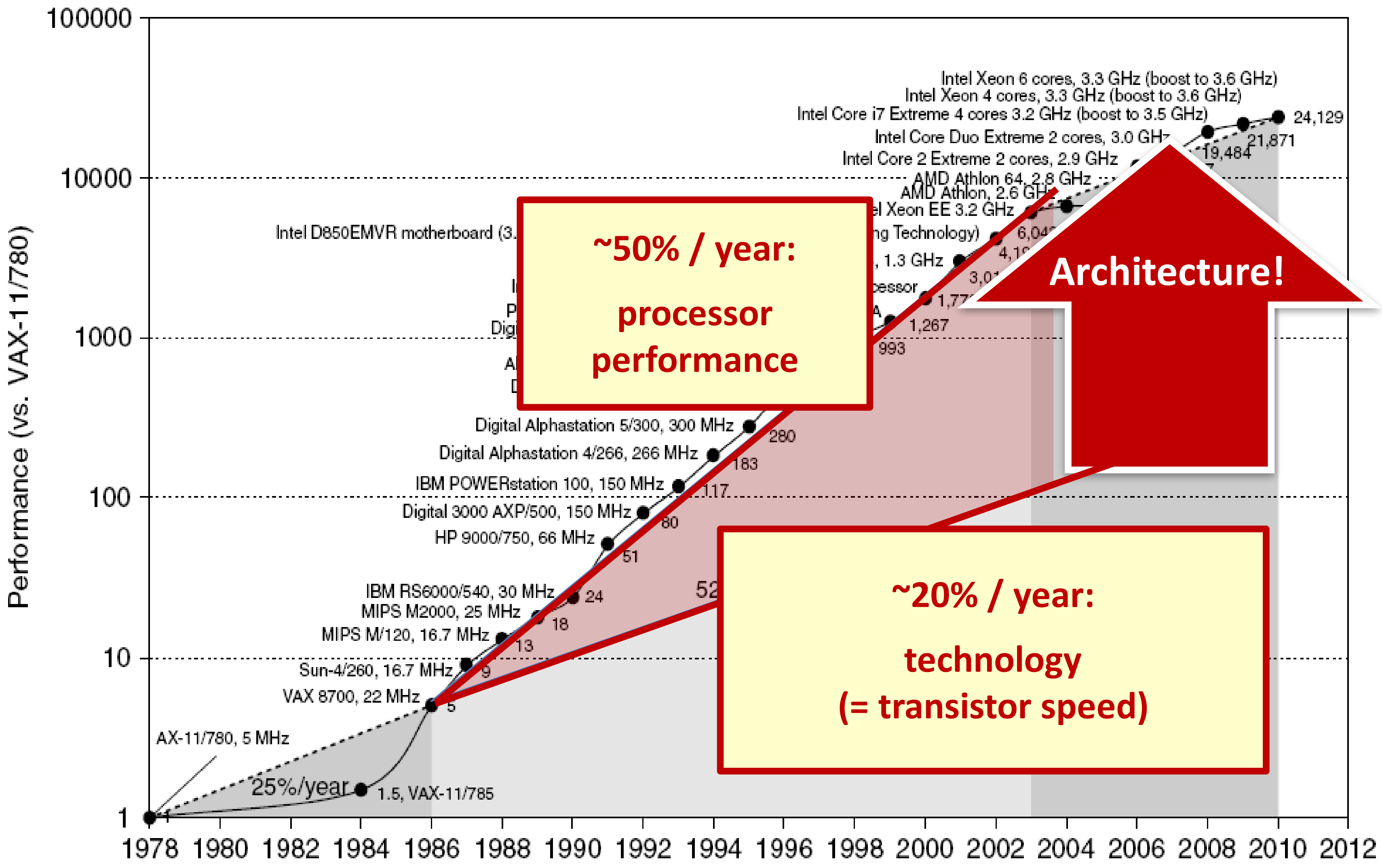
—

Part 4a. Instruction Level Parallelism
Performance

Paolo Ienne

<paolo.ienne@epfl.ch>

Remember?



Source: Hennessy & Patterson, © MK 2011

So Far about Performance…

- Different parts of a system do not benefit equally from manufacturing technology advances
  - Memories are “slower and slower” → **Caches**
- We have done nothing to speed-up the processor itself

What is “Performance”?

- **Processor frequency?**
  - Is it better an Intel Core i7-7700K at **4.2 GHz** or an AMD Ryzen 5 5600X at **3.7 GHz**? And how much better the best one is?
- **Memory speed? Cache efficiency**?
  - Is it better to have **8 MiB of 4-way set-associative** cache or **16 MiB of direct mapped** cache?
  - Is it better to have **three levels of overall smaller** caches or **two levels of overall bigger** caches?

We need a metric!

Elapsed Time, CPU Time,…

None of the above matters in itself

- What matters is how long it takes to perform a job a user needs!

OS command to inquire about execution time

My job

```
[110]icvm0100> time latex mypaper >& /dev/null
0.79u 0.17s 0:01.20 80.0%
[111]icvm0100>
```

Elapsed Time: 1.20s after I started it, my job was completed

System CPU Time: The processor has spent 0.17s executing instructions of the *operating system* on behalf of my program

User CPU Time: The processor has spent 0.79s executing instructions of my program (`latex`)

Elapsed Time, CPU Time,…

```
[110]icvm0100> time latex mypaper >& /dev/null
0.79u 0.17s 0:01.20 80.0%
[111]icvm0100>
```

80% of the Elapsed Time (= 0.96s/1.20s) has been spent on my job:
system I/O, other jobs, other users,…

User CPU Time + System CPU Time ≠ Elapsed Time:
The processor has spent 0.96s executing for me
but the result took 1.20s to become ready

We are interested in

Elapsed Time on an Unloaded System

Often simply "Execution Time" for brevity…

Relative Performance

- **Speedup**
  - How faster system X is compared to system Y

$$Speedup = \frac{Performance_X}{Performance_Y} = \frac{Execution\ Time_Y}{Execution\ Time_X}$$

- **Common Performance Indices**
  - Speedups of systems compared to a single standard system
  - SPEC CPU, Geekbench, Cinebench, and LinPack HPL, EEMBC ("Embassy") CoreMark

The classic CPU benchmark

Dedicated to embedded processors

Relate Performance to Hardware Implementation

- In hardware, our measure of time is the **clock period** or **cycle**
- We are often interested to relate the execution time to this "hardware quantum"
- **Cycles per Instruction (CPI)**
  - Average number of cycles per instruction executed

$$CPI = \left(\frac{Execution\ Time}{Clock\ Period} \right) \Big/ \left(Total\ Instruction\ Count \right)$$

- **Instructions per Cycle (IPC) ← 1 / CPI**
  - Average instructions executed per cycle
  - Normally below unity, unless the processor executes several instructions in parallel

Improving Performance?

- Performance being 1/Execution Time, rewriting the definition of CPI and IPC:

Implement the processor in a fast technology ☺

$$Performance = \frac{1}{Execution\ Time} =$$

$$= \frac{f_{clock}}{\boxed{Instruction\ Count} \cdot \boxed{CPI}} = \frac{\boxed{f_{clock}} \cdot \boxed{IPC}}{Instruction\ Count}$$

Execute several instructions in parallel ☺

Use fewer complex instructions (CISC)?
But then we need more cycles per instruction or a slower clock… ☹

Make instructions simple so that we can have less CPI or a faster clock (RISC)?
But then we need more instructions for the same job… ☹

Many Other Considerations Influence the Performance

A few random examples:

- *Instruction Count* depends also from the **compiler**
  - It is more important to have an instruction-set which a compiler can use very effectively (best instructions for the required job) rather than a “reduced” or a “complex” instruction-set—otherwise the instruction count will be larger than needed…
- *CPI* depends on the **cache performance**
  - If the overall code size grows, the cache will be less effective (more misses)…
- Very fast clock cycles will require more often instructions from the memory
  - Performance of the cache becomes more critical…

What to Improve to Increase Performance?

- **Amdahl's Law** (law of diminishing returns)
  - **The performance enhancement possible with a given improvement is limited by the amount the improved feature is used**
- Typical software situation:
  - If a program spends 20% of the time in subroutine X, the maximum reduction in execution time one can get from optimising X is 20%, that is a speedup of 1/(1 - 0.2) = 1.25x
- In a processor:
  - If the instruction Y is used 0.1% of the time, is it worth to make it faster? It is probably better to look for the instruction which is used 20% of the time…

Look for where most of the time goes!

Benchmarks

- Performance indices such as SPEC CPU, Geekbench, Cinebench, and LinPack HPL, EEMBC (“Embassy”) CoreMark need a **precise definition of the user job(s) to run**
- Serious **benchmark suites** are collections of large and representative user programs spanning all areas of typical use, often agreed between manufacturers
- They do not only define the programs (in C, C++, FORTRAN, Java), but also how to compile them, what data to run them on, etc.

SPEC CPU2006 Integer

Benchmark Description	CPU2000 Integer	Lng	RT	CPU2006 Integer	Lng	RT
GNU C compiler	176.gcc	C	1,100	403.gcc	C	8,050
Manipulates strings & prime numbers in Perl language	253.perlbmk	C	1,800	400.perlbench	C	9,766
Minimum cost network flow solver (combinatorial optimization)	181.mcf	C	1,800	429.mcf	C	9,120
Data compression utility	256.bzip2	C	1,500	401.bzip2	C	9,644
Data compression utility	164.gzip	C	1,400			
Video compression & decompression				464.h264ref	C	22,235
Artificial intelligence, plays game of Chess	186.crafty	C	1,000	458.sjeng	C	12,141
Artificial intelligence, plays game of Go				445.gobmk	C	10,489
Artificial intelligence used in games for finding 2D paths across terrains				473.astar	C++	7,017
Natural language processing	197.parser	C	1,800			
XML processing				483.xalancbmk	C++	6,869
FPGA circuit placement and routing	175.vpr	C	1,400			
EDA place and route simulator	300.twolf	C	3,000			
Search gene sequence				456.hmmer	C	9,333
Ray tracing	252.eon	C++	1,300			
Computational group theory	254.gap	C	1,100			
Database program	255.vortex	C	1,900			
Library for simulating a quantum computer				462.libquantum	C	20,704
Discrete event simulation				471.omnetpp	C++	6,270
	hours	5.3	19,100	hours	36.6	131,638

Source: Microprocessor Report, © Cahners 2007

- **Reference Time (RT)** measured on a Sun Ultra 5 + 300MHz UltraSPARC III + 256KB L2 cache → **100** SPEC2000
- Note the complexity of the benchmark: 36.6 hrs and 88.3 hrs of runtime (on a relatively old machine) for Int and FP respectively

SPEC CPU2006 Floating Point

Benchmark Description	CPU2000 Floating Pnt	CPU2000 Lng	CPU2000 RTime	CPU2006 Floating Pnt	CPU2006 Lng	CPU2006 RTime
Weather prediction, shallow water model	171.swim	F77	3,100			
Velocity & distribution of pollutants based on temperature, wind	301.apsi	F77	2,600			
Weather modeling (30km area over 2 days)				481.wrf	C/F	11,215
Physics, particle accelerator model	200.sixtrack	F77	1,100			
Parabolic/elliptic partial differential equations	173.applu	F77	2,100			
Multi-grid solver in 3D potential field	172.mgrid	F77	1,800			
General relativity, solves Einstein evolution equations				436.cactusADM	C/F	11,927
Computational electromagnetics (solves Maxwell equations in 3D)				459.GemsFDTD	F	10,583
Quantum chromodynamics	168.wupwise	F77	1,600			
Quantum chromodynamics, gauge field generation with dynamical quarks				433.milc	C	9,180
Fluid dynamics, analysis of oscillatory instability	178.galgel	F90	2,900			
Fluid dynamics, computes 3D transonic transient laminar viscous flow				410.bwaves	F	13,592
Computational fluid dynamics for simulation of astrophysical phenomena				434.zeusmp	F	9,096
Fluid dynamics, large eddy simulations with linear-eddy model in 3D				437.leslie3d	F	9,358
Fluid dynamics, simulates incompressible fluids in 3D				470.lbm	C	13,718
Molecular dynamics (simulations based on newtonian equations of motion)				435.gromacs	C/F	7,132
Biomolecular dynamics, simulates large system with 92,224 atoms				444.namd	C++	8,018
Computational chemistry	188.ammp	C	2,200			
Quantum chemistry package (object-oriented design)				465.tonto	F	9,822
Quantum chemistry, wide range of self-consistent field calculations				416.gamess	F	19,575
Computer vision, face recognition	187.facerec	F90	1,900			
Speech recognition system				482.sphinx3	C	19,528
3D graphics library	177.mesa	C	1,400			
Neural network simulation (adaptive resonance theory)	179.art	C	2,600			
Earthquake modeling (finite element simulation)	183.equake	C	1,300			
Crash modeling (finite element simulation)	191.fma3d	F90	2,100			
Number theory (testing for primes)	189.lucas	F90	2,000			
Structural mechanics (finite elements for linear & nonlinear 3D structures)				454.calculix	C/F	8,250
Finite element analysis (program library)				447.dealII	C++	11,486
Linear programming optimization (railroad planning, airlift models)				450.soplex	C++	8,338
Image ray tracing (400x400 anti-aliased image with abstract objects)				453.povray	C++	5,346
	hours	8.0	28,700	hours	52	186,164
	hours	13.3	47,810	hours	88.3	317,802

Source: Microprocessor Report, © Cahners 2007

SPEC CPU2017 Speed and Rate

SPECspeed	SPECrate	Language	Application
600.perlbench_s	500.perlbench_r	C	Perl interpreter
602.gcc_s	502.gcc_r	C	Gnu C compiler
605.mcf_s	505.mcf_r	C	Route planning
620.omnetpp_s	520.omnetpp_r	C++	Discrete event simulation—computer N/W
623.xalancbmk_s	523.xalancbmk_r	C++	XML-to-HTML conversion via XSLT
625.x264_s	525.x264_r	C	Video compression
631.deepsjeng_s	531.deepsjeng_r	C++	AI: alpha-beta tree search (Chess)
641.leela_s	541.leela_r	C++	AI: Monte Carlo tree search (Go)
648.exchange2_s	548.exchange2_r	Fortran	AI: recursive solution generator (Sudoku)
657.xz_s*	557.xz_r	C	General data compression
603.bwaves_s	503.bwaves_r	Fortran	Explosion modeling
607.cactuBSSN_s	507.cactuBSSN_r	C++, C, Fortran	Physics: relativity
Not applicable†	508.namd_r	C++	Molecular dynamics
Not applicable†	510.parest_r	C++	Biomedical imaging: optical tomography
Not applicable†	511.povray_r	C++, C	Ray tracing
619.lbm_s	519.lbm_r	C	Fluid dynamics
621.wrf_s	521.wrf_r	Fortran, C	Weather forecasting
Not applicable†	526.blender_r	C++, C	3D rendering and animation
627.cam4_s	527.cam4_r	Fortran, C	Atmosphere modeling
628.pop2_s	Not applicable‡	Fortran, C	Wide-scale ocean modeling (climate level)
638.imagick_s	538.imagick_r	C	Image manipulation
644.nab_s	544.nab_r	C	Molecular dynamics
649.fotonik3d_s	549.fotonik3d_r	Fortran	Computational electromagnetics
654.roms_s	554.roms_r	Fortran	Regional ocean modeling

Source: Microprocessor Report, © The Linley Group 2017

SPECspeed

Measures the completion time of a task (latency test)

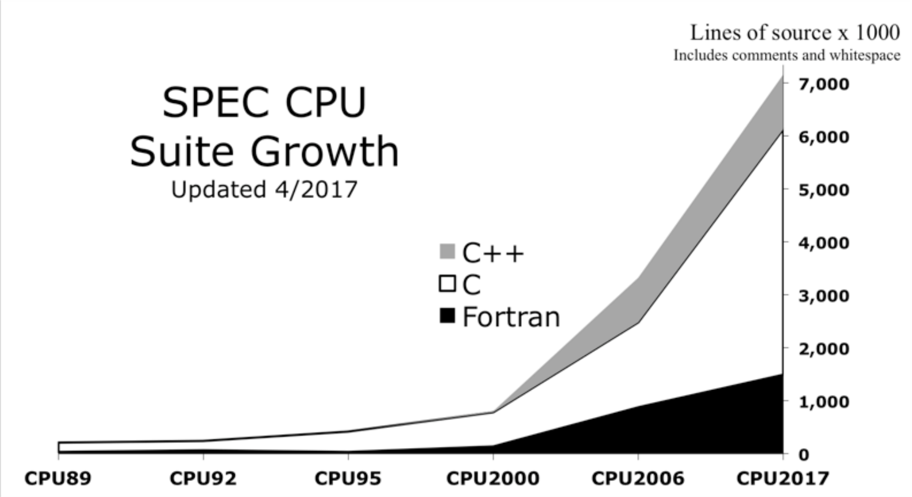
SPECrate

Measures the number of tasks performed in the unit of time (throughput test)

SPEC CPU2000 Example
Complete and Complex Programs

Benchmark	Language	KLOC	Resident size (Mbytes)	Virtual size (Mbytes)	Description
SPECint2000					
164.gzip	C	7.6	181	200	Compression
175.vpr	C	13.6	50	55.2	FPGA circuit placement and routing
176.gcc	C	193.0	155	158	C programming language compiler
181.mcf	C	1.9	190	192	Combinatorial optimization
186.crafty	C	20.7	2.1	4.2	Game playing: Chess
197.parser	C	10.3	37	62.5	Word processing
252.eon	C++	34.2	0.7	3.3	Computer visualization
253.perlbmk	C	79.2	146	159	Perl programming language
254.gap	C	62.5	193	196	Group theory, interpreter
255.vortex	C	54.3	72	81	Object-oriented database
256.bzip2	C	3.9	185	200	Compression
300.twolf	C	19.2	1.9	4.1	Place and route simulator
SPECfp2000					
168.wupwise	F77	1.8	176	177	Physics: Quantum chromodynamics
171.swim	F77	0.4	191	192	Shallow water modeling
172.mgrid	F77	0.5	56	56.7	Multigrid solver: 3D potential field
173.applu	F77	7.9	181	191	Partial differential equations
177.mesa	C	81.8	9.5	24.7	3D graphics library
178.galgel	F90	14.1	63	155	Computational fluid dynamics
179.art	C	1.2	3.7	5.9	Image recognition/neural networks
183.equake	C	1.2	49	51.1	Seismic wave propagation simulation
187.facerec	F90	2.4	16	18.5	Image processing: Face recognition
188.ammp	C	12.9	26	30	Computational chemistry
189.lucas	F90	2.8	142	143	Number theory/primality testing
191.fma3d	F90	59.8	103	105	Finite-element crash simulation
200.sixtrack	F77	47.1	26	59.8	Nuclear physics accelerator design
301.apsi	F77	6.4	191	192	Meteorology: Pollutant distribution

Source: Computer, © IEEE 2000



Source: https://www.spec.org/cpu2017/Docs/overview.html#Q19, © SPEC 2017

SPEC CPU2006 required 1 GB of physical memory, CPU2017 requires 16 GB

SPECspeed 2017
on Intel Xeon E5-2650L 1.8 GHz

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Perl interpreter	perlbench	2684	0.42	0.556	627	1774	2.83
GNU C compiler	gcc	2322	0.67	0.556	863	3976	4.61
Route planning	mcf	1786	1.22	0.556	1215	4721	3.89
Discrete Event simulation - computer network	omnetpp	1107	0.82	0.556	507	1630	3.21
XML to HTML conversion via XSLT	xalancbmk	1314	0.75	0.556	549	1417	2.58
Video compression	x264	4488	0.32	0.556	813	1763	2.17
Artificial Intelligence: alpha-beta tree search (Chess)	deepsjeng	2216	0.57	0.556	698	1432	2.05
Artificial Intelligence: Monte Carlo tree search (Go)	leela	2236	0.79	0.556	987	1703	1.73
Artificial Intelligence: recursive solution generator (Sudoku)	exchange2	6683	0.46	0.556	1718	2939	1.71
General data compression	xz	8533	1.32	0.556	6290	6182	0.98
Geometric mean	–	–	–	–	–	–	2.36

Summary

- Performance measurement is all but easy: many **heterogeneous parameters** come into the picture
- What really matters most for a user is the **elapsed time on an unloaded system**
- **CPI** and **IPC** help relate hardware features of the processor to performance, but there are **many pitfalls**, hidden dependencies, "second-order" effects…
- **Benchmarks** are the only practical way to assess performance—and serious unbiased benchmarks are difficult to design

References

- Patterson & Hennessy, COD – RISC-V Edition
  - **Section 1.6**, **1.9**, and **1.11**