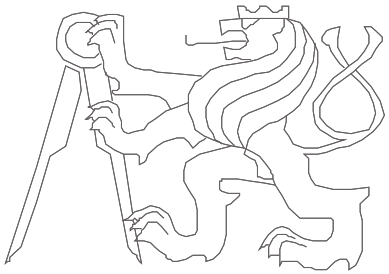
Advanced Computer Architectures

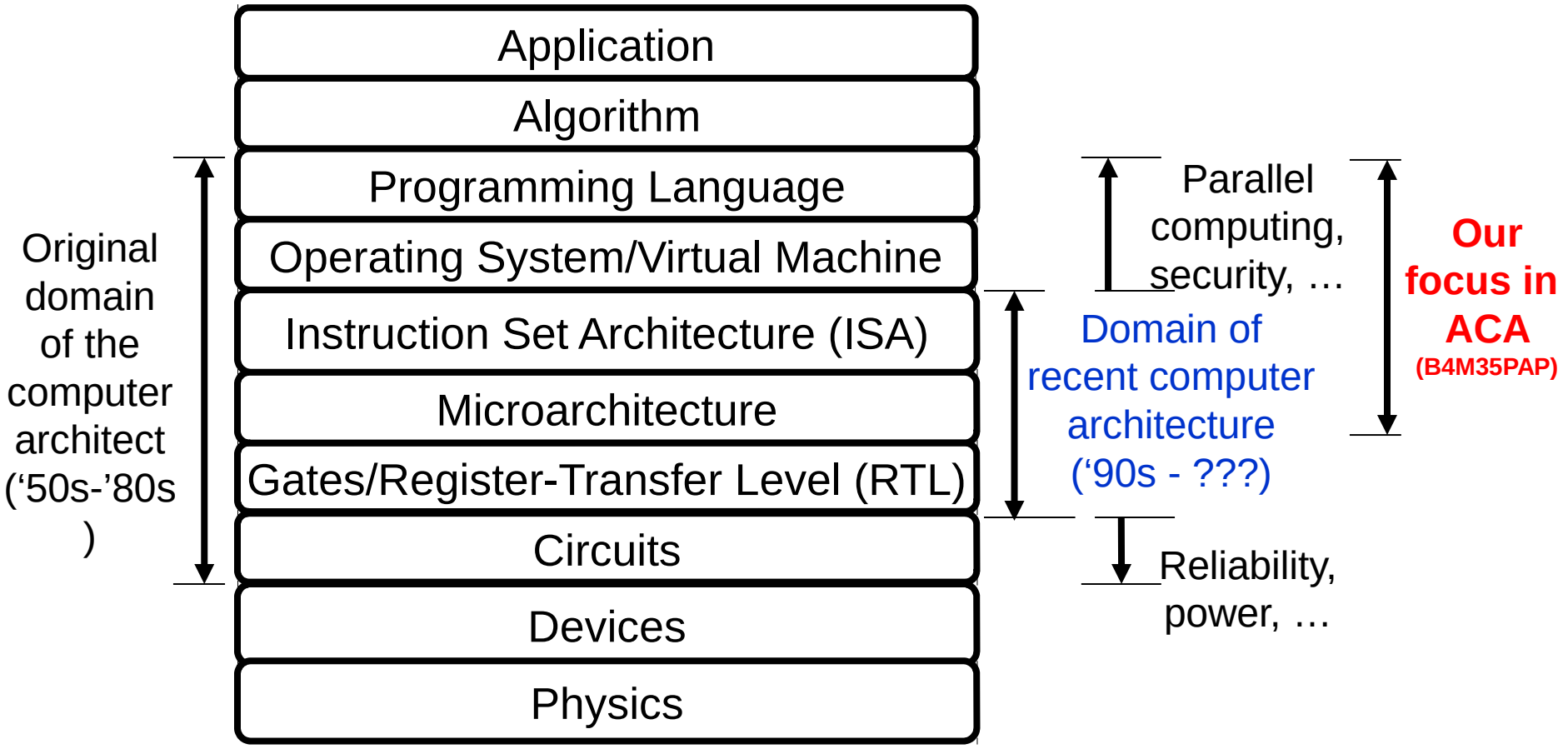
01

Introduction



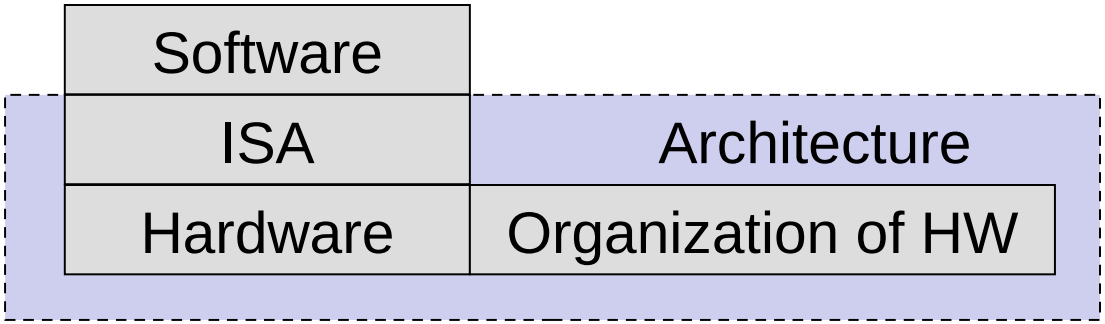
Czech Technical University in Prague, Faculty of Electrical Engineering
Slides authors: Michal Štepanovský, Pavel Píša

Abstraction Layers in Modern Systems



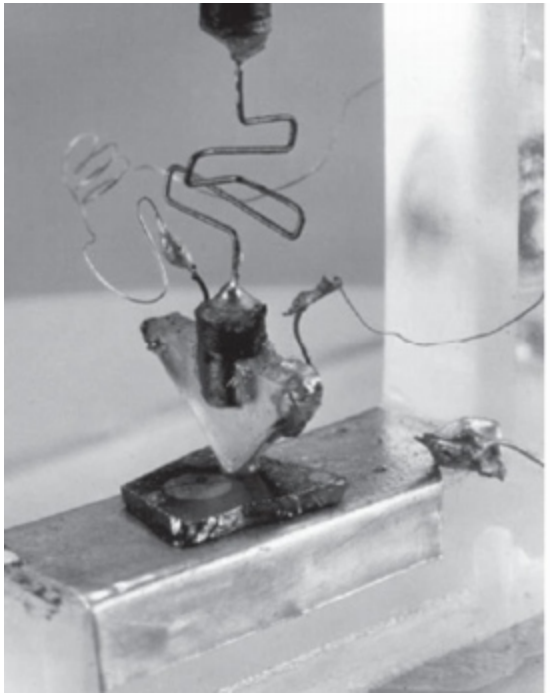
Reference: John Kubiatowicz: EECS 252 Graduate Computer Architecture, Lecture 1. University of California, Berkeley

Instruction Architecture and Their Implementation



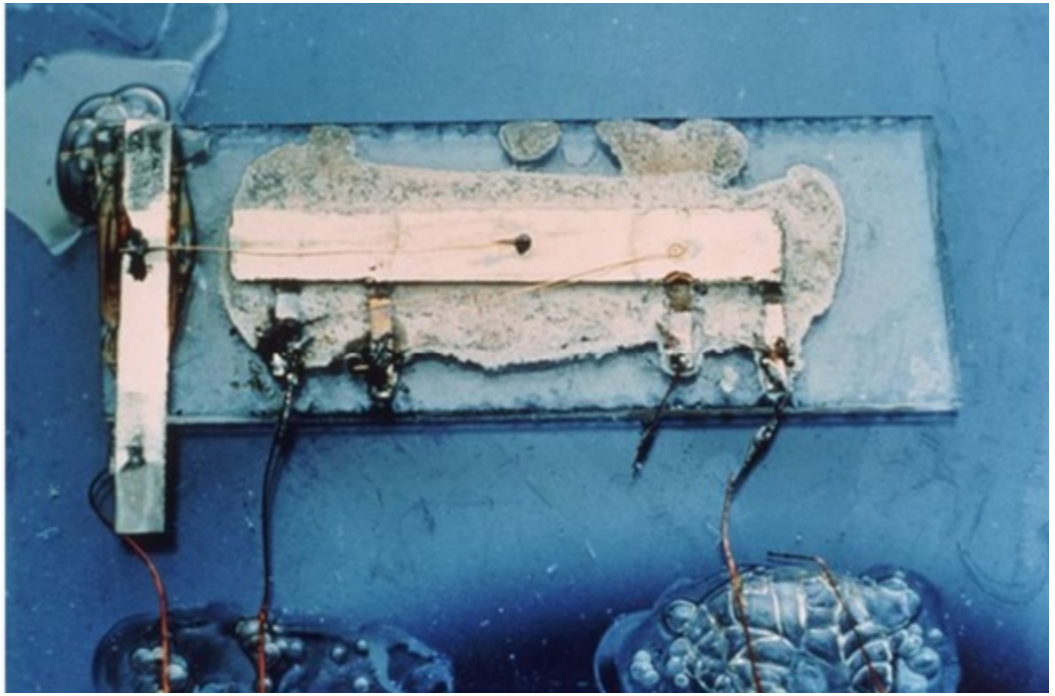
- Two different processors can implement exactly the same ISA, but internal structure (micro-architecture) of these processors can vary significantly (different pipelines, etc.)
- Organization – view from the top (memory subsystem, buses, control unit,…)
- Hardware – view from the bottom (logic circuits, technology, …)

What is it?



First Transistor

(Bell Labs) – Dec. 23, 1947

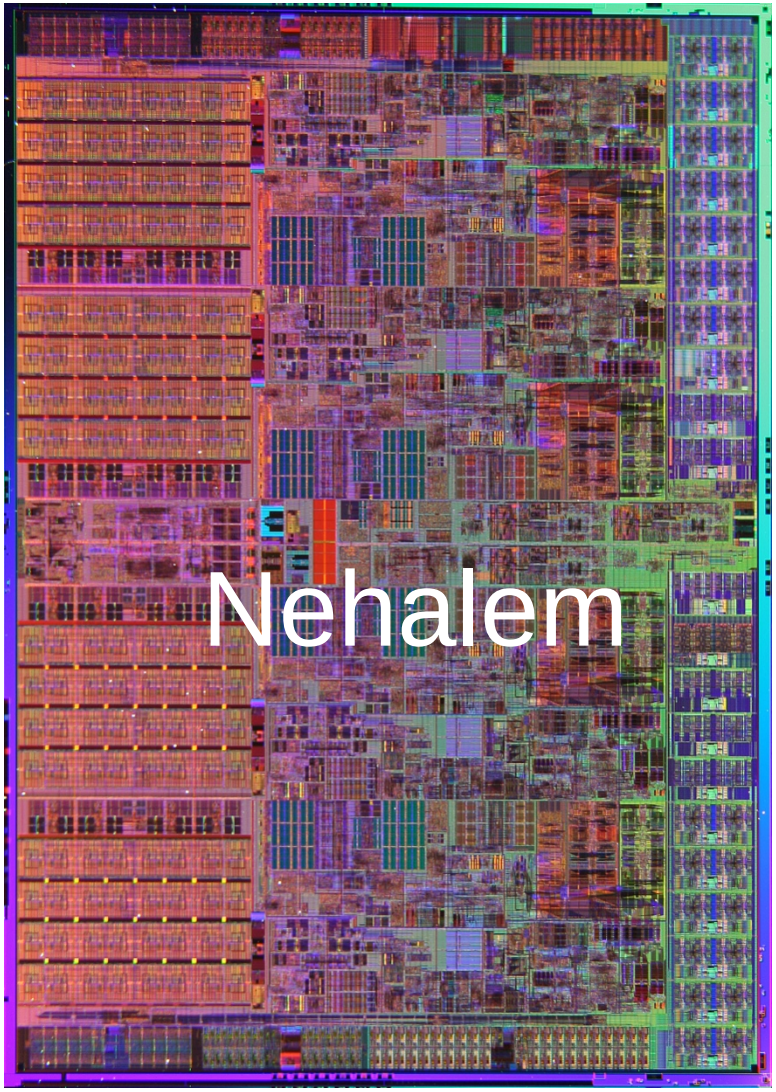


First integrated circuit (IC)

Jack Kilby's original IC (Texas Instruments) - 1958

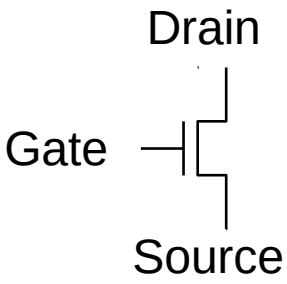
Technology constantly on the move!

- Num of transistors not limiting factor
  - Currently ~ 1 billion transistors/chip
  - Problems:
    - Too much Power, Heat, Latency
    - Not enough Parallelism
- 3-dimensional chip technology?
  - Sandwiches of silicon
  - “Through-Vias” for communication
- On-chip optical connections?
  - Power savings for large packets
- The Intel® Core™ i7 microprocessor (“Nehalem”)
  - 4 cores/chip
  - **45 nm**, Hafnium hi-k dielectric
  - 731M Transistors

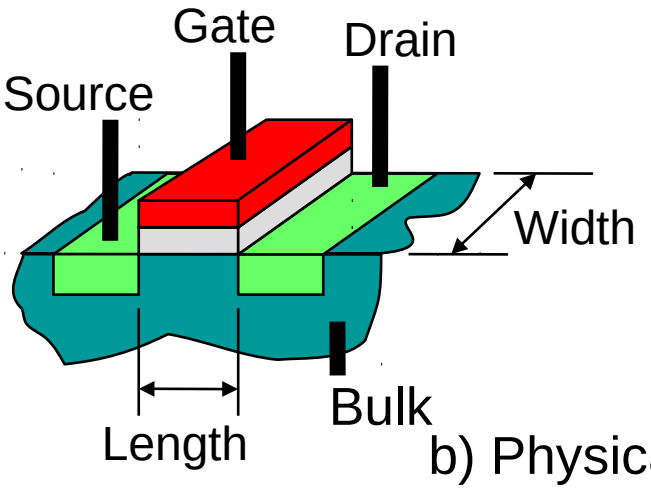


Reference: John Kubiatowicz: EECS 252 Graduate Computer Architecture, Lecture 1. University of California, Berkeley

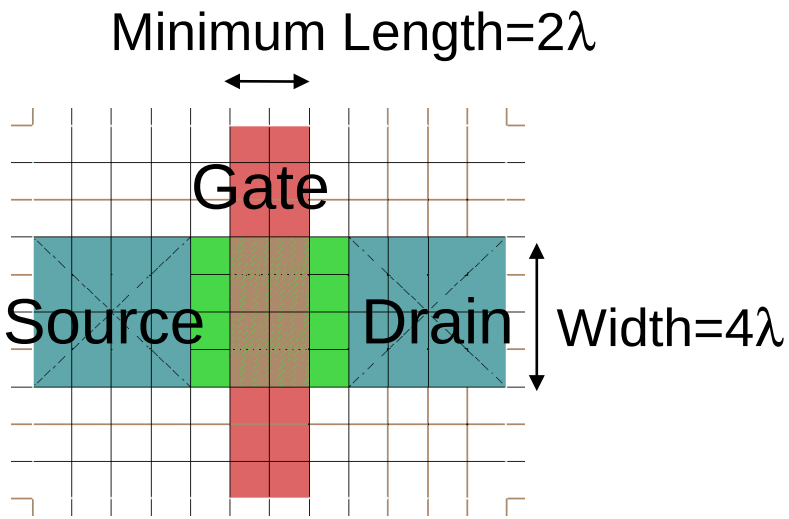
Transistors



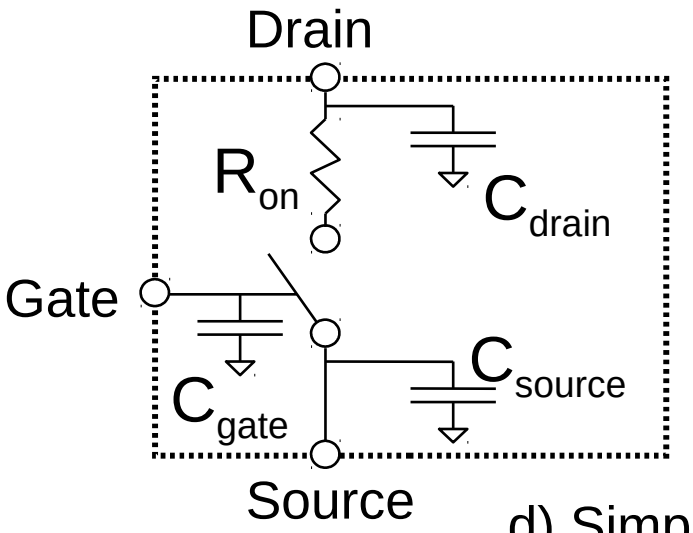
a) Circuit Symbol



b) Physical Realization



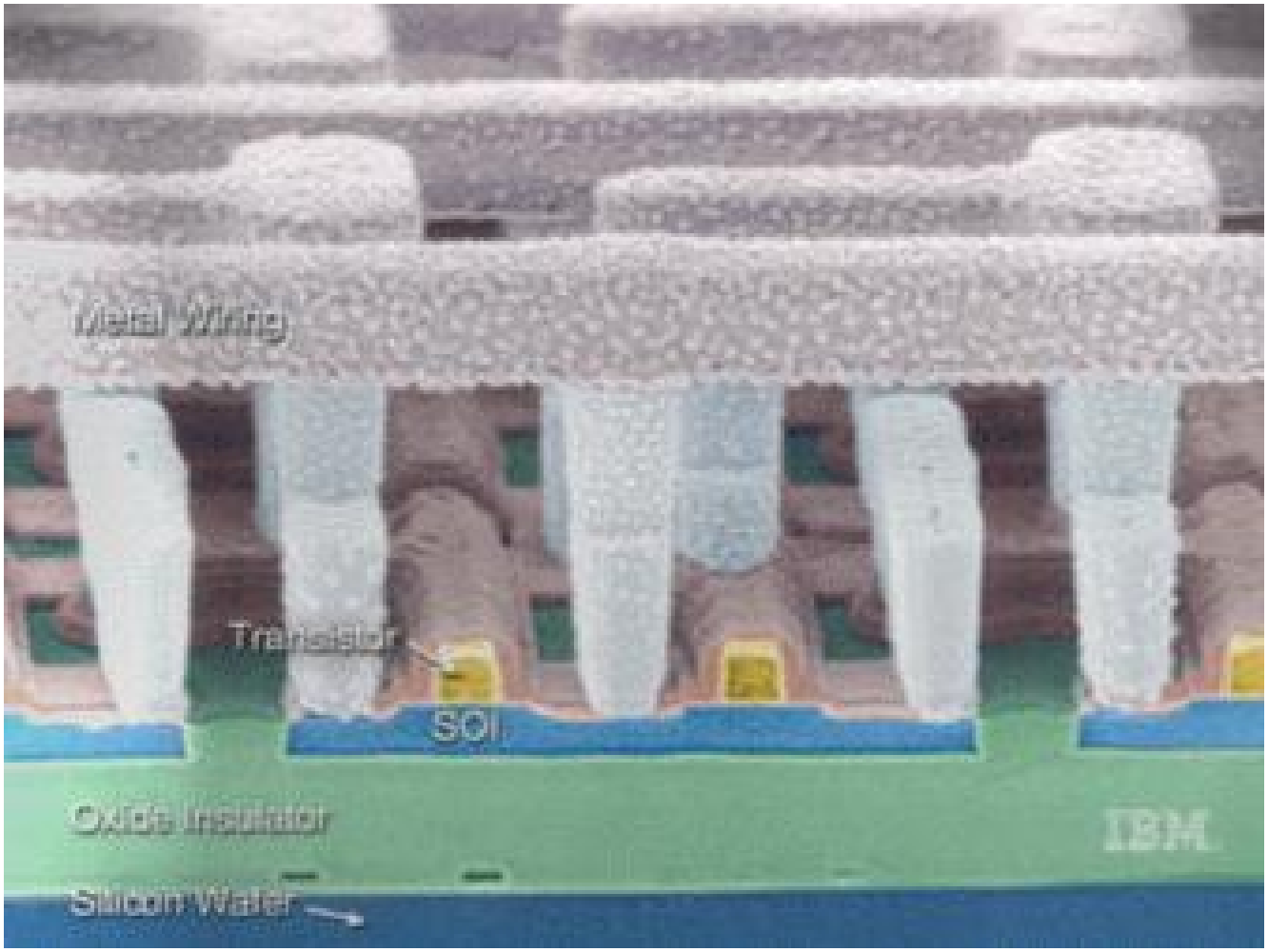
c) Layout View



d) Simple RC Model

K.Asanovic: VLSI for Architects - Advanced VLSI Computer Architecture, http://groups.csail.mit.edu/cag/6.893-f2000/lectures/l02-vlsi-for-archs.pdf, *Copyright 2000, Krste Asanovic.*

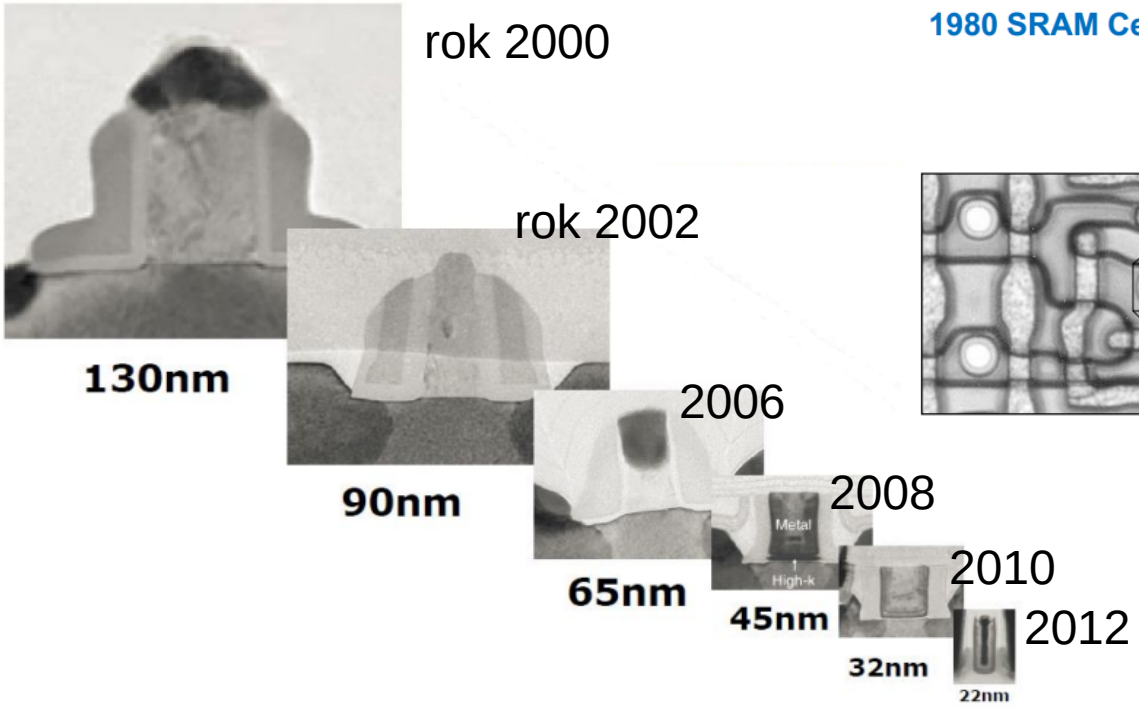
Transistors - Today

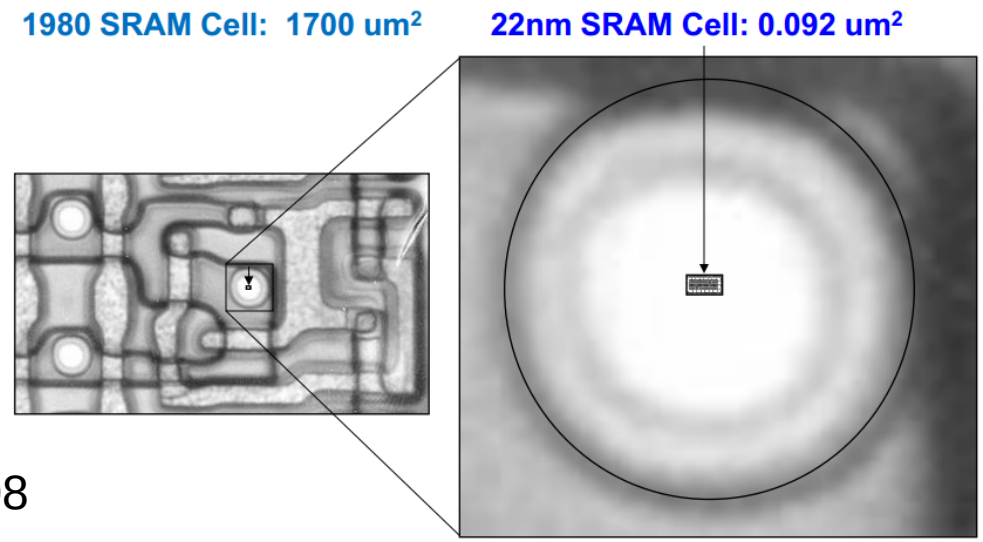


IBM SOI Technology

K.Asanovic: VLSI for Architects - Advanced VLSI Computer Architecture, http://groups.csail.mit.edu/cag/6.893-f2000/lectures/l02-vlsi-for-archs.pdf, *Copyright 2000, Krste Asanovic.*

Gordon Moore





Intel 8008 v roce 1971: 10 mikro

- The speed of the electrical signals is limited by the speed of the light (and dependents on the material properties of the medium) -> at 3.6GHz – period is 0.278 ns => 8.3 cm (the propagation delay of interconnection is higher then gate switching time)
- Ultimate minimum feature size may be determined by atomic spacing – in Si the lattice constant is ~0.5 nm.

Dmitri Nikonov: Beyond CMOS computing - 1. CMOS Scaling. h
ttp://nanohub.org/resources/18348/download/NikonovBeyondCMOS_1_scaling.pdf
S. Demlow: A Brief History of the field of VLSI, ECE 410, January 9, 2012

Syllabus and Subject Goals – Lectures

1. **Lecture – Introduction into modern computer architectures**

   Control flow computers and Data flow computers (Data driven, Demand driven). Classification of computer architectures by Flynn's taxonomy. Parallel processing – multi-core, multiprocessor and multiple computers based systems, the concept of parallel processing. Amdahl's and Gustafson's law. Performance metrics.

2. **Lecture – From the scalar to the superscalar processors (Basic organization of superscalar processor)**

   Superscalar processors with static, dynamic, and hybrid scheduling of instructions execution.

3. **Lecture - Superscalar techniques I – Data flow inside the processor (register data flow)**

   Registers renaming (Tomasul algorithm) and data speculation. Precise exception support.

4. **Lecture - Superscalar techniques II - Instruction flow, speculative execution (Control Speculation)**

   Prediction, predictors and instructions prefetching. Static and dynamic predictions; Smith's predictor, two-level predictors with local and global history, bi-mode, adaptive branch prediction technique, and more. Branch misprediction recovery

5. **Lecture - Superscalar techniques III - Memory data flow, VLIW and EPIC processors**

   Data flow from / to memory. Load bypassing and Load forwarding. Speculative load. Some other ways to reduce memory latency. VLIW and EPIC processors. Use of data parallelism, SIMD and vector instruction in ISA. Loop unrolling and Software pipelining - Execution on WLIV and superscalar processor.

6. **Lecture – Memory subsystem**

   Non-Blocking cache, Victim cache, Virtual memory and cache

Syllabus and Subject Goals – Lectures

7. Lecture – Multiprocessor systems and memory coherence problem.

Multiprocessor computers architectures. Distributed and shared memory systems (DMS, SMS). Symmetric multiprocessor computer architectures. Methods to ensure coherence in SMP.

8. Lecture – Multiprocessor systems and memory consistency problems.

Rules for performing memory operations, ensuring sequential consistency, memory consistency models.

9. Lecture – Parallel Systems Programming I.

Parallel systems programming concepts, using Message Passing Interface (MPI) and Open Multi-Processing (OpenMP) to create parallel programs.

10. Lecture – Programming of parallel systems II.

Synchronization. Code optimization. Cache maintenance, consequences of coherence protocols.

11. Lecture – Interconnection networks

Static and dynamic interconnection networks

12. Lecture – Use of graphic accelerators (GPU) and General-purpose computing on graphics processing units (GPGPU)

13. Lecture – Time and space parallelization in practice.

Sample of selected partitions on processor Intel Nehalem, AMD Optreon, IBM Power4, ARM, AArch64, RISC-V

14. Lecture – Perspectives and limitations of future development and history

Syllabus and Subject Goals – Seminars

Part I – Hardware description language

Seminar project: Design of simple CPU in Verilog/VHDL

Part II – Programming of parallel systems

Seminar project: Development of program for multiprocessor parallel system (cluster of multicore CPUs)

Part III – Semester projects processing, presentation and defense

Seminar project: Design of a simple pipelined CPU, based on work from Part I

Assessment Award Requirements and Exam

- Participation in seminars - compulsory
  (2 absences without confirmation from a doctor tolerated)
- Submitting of all seminar projects
- Exam:
  - Passing the written part of the exam (at least 60%)
  - Oral examination

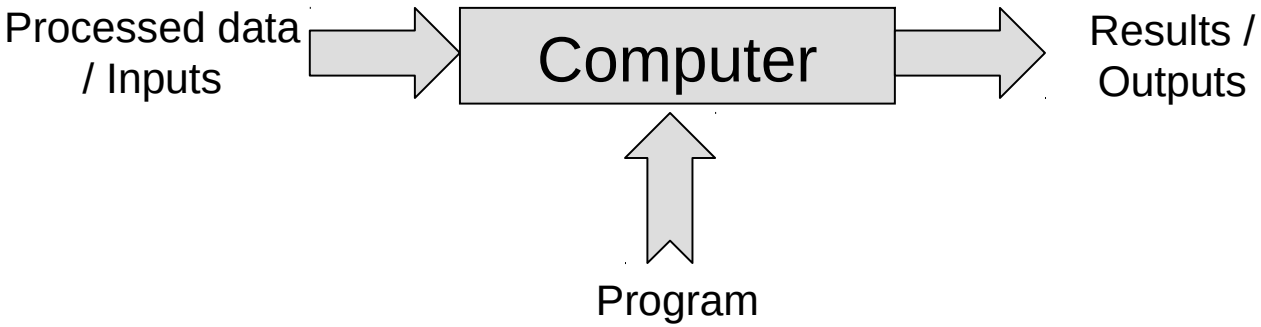
Subject web pages: **https://cw.fel.cvut.cz/wiki/courses/b4m35pap**

Literature:

- Hennesy, J. L., Patterson, D. A.: Computer Architecture : A Quantitative Approach, Third Edition, San Francisco, Morgan Kaufmann Publishers, Inc., 2002
- Shen, J.P., Lipasti, M.H.: Modern Processor Design : Fundamentals of Superscalar Processors, First Edition, New York, McGraw-Hill Inc., 2004
- Grama A., Gupta, A. et al.:Introduction to Parallel Computing, Second Edition, Addison Wesley, 2003

General View of Computer and Its Operation

- We understand to the term *computer* as a general purpose device that can be programmed to process the input data **automatically**



- The program can be defined as a specification of set of operations over the operands (data being processed) that are required to perform the required task (obtaining the results).

- F = (A*B + C*D) / (E+2),
  A,B,C,D,E – inputs; F – outputs;
  …expressed in programming language to compute result → program

General View of Computer and Its Operation

Considering: F = (A*B + C*D) / (E+2)

Control Flow approach:

- Mult Reg1, A, B
- Mult Reg2, C, D
- Add Reg1, Reg1, Reg2
- Add Reg2, E, 2
- Div F, Reg1, Reg2
- Write F

The computer activity is determined by the sequence of instructions…

General View of Computer and Its Operation

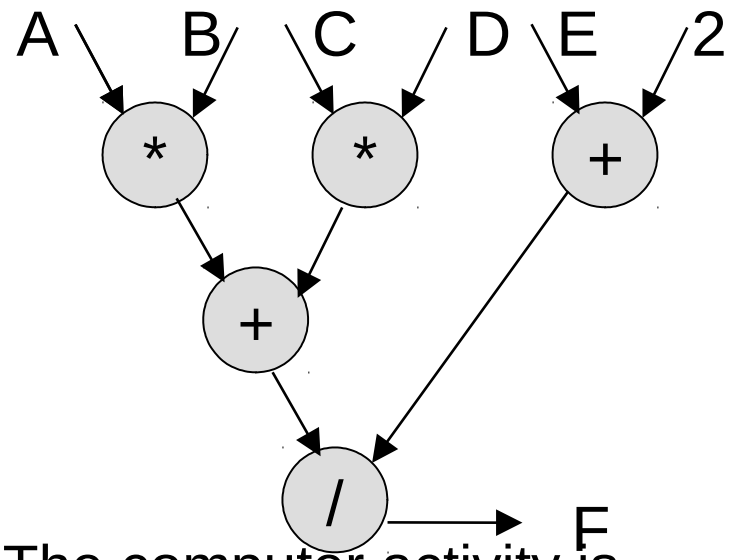
Considering: F = (A*B + C*D) / (E+2)

Control Flow approach:

- Mult Reg1, A, B
- Mult Reg2, C, D
- Add Reg1, Reg1, Reg2
- Add Reg2, E, 2
- Div F, Reg1, Reg2
- Write F

The computer activity is determined by the sequence of instructions…

Data Flow approach:

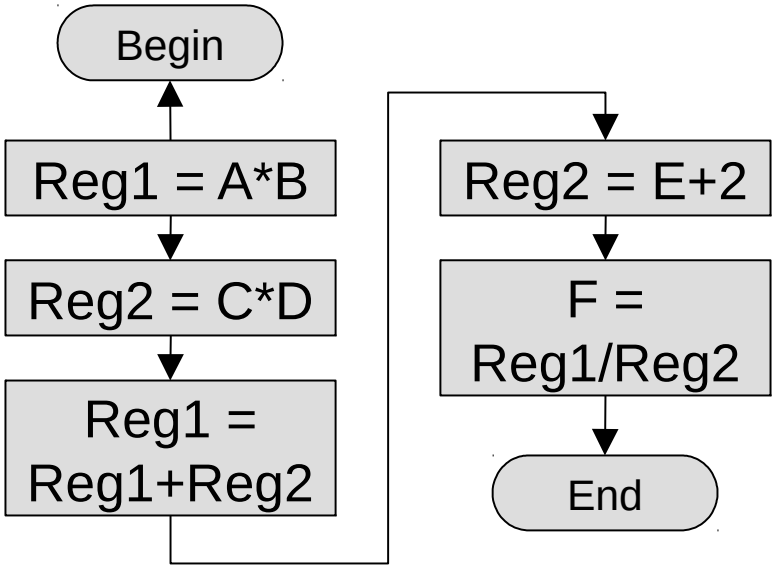


The computer activity is determined by demands (demand driven computer) or by the presence of operands (data driven)…

General View of Computer and Its Operation

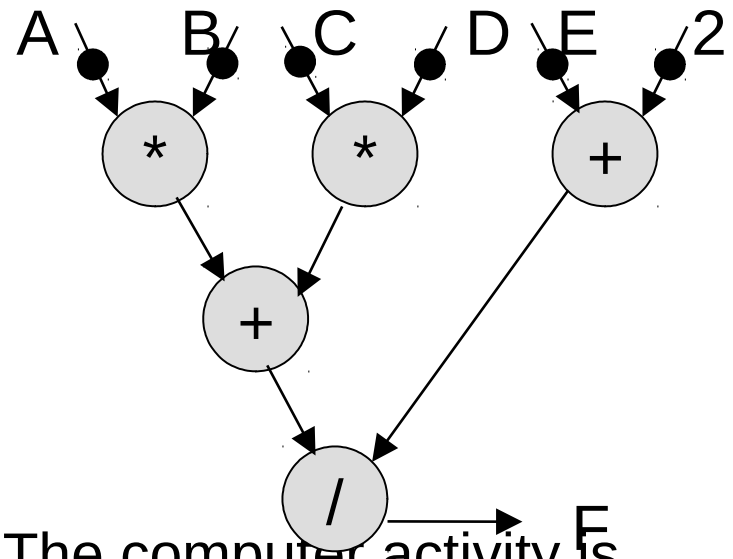
Considering: F = (A*B + C*D) / (E+2)

Control Flow approach:



The computer activity is determined by the sequence of instructions…

Data Flow approach:



The computer activity is determined by demands (demand driven computer) or by the presence of operands (data driven)…

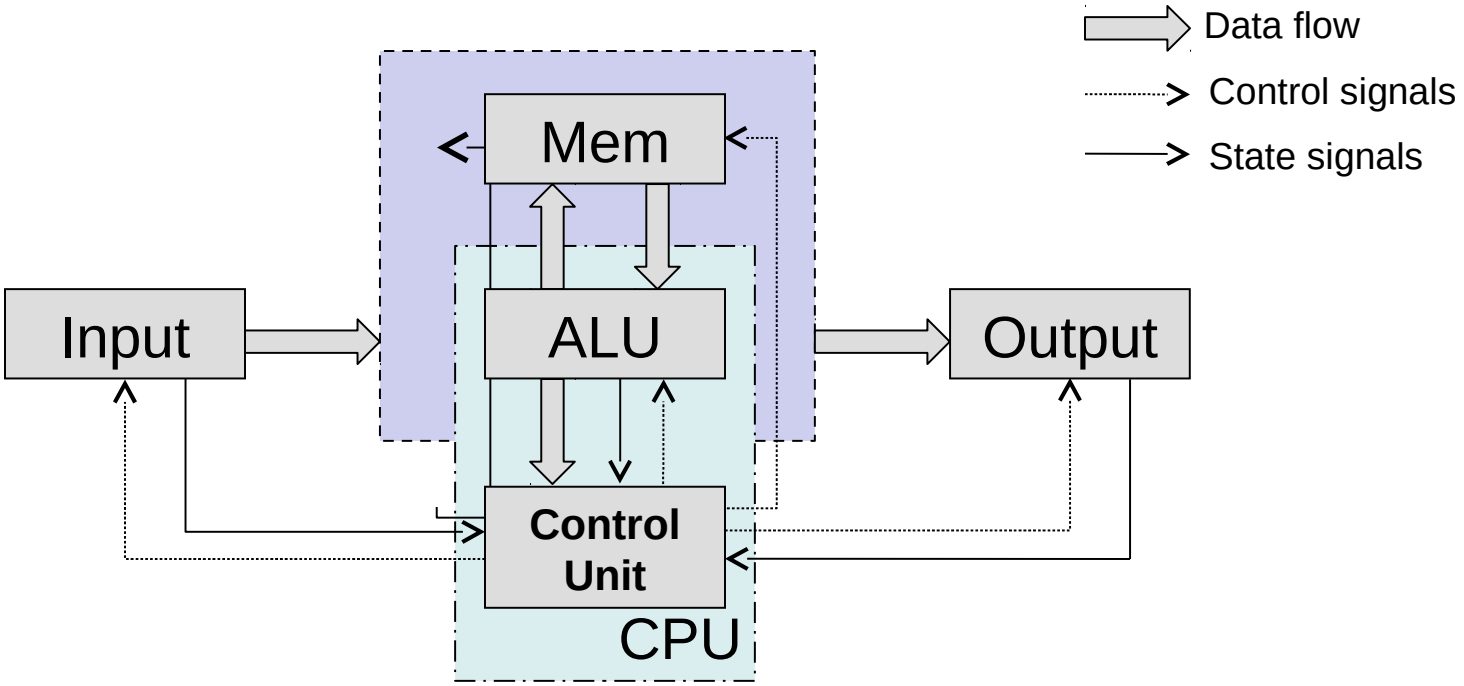
Control Flow vs. Data Flow

- The instructions are executed sequentially as they are written, respectively as determined by the program counter (PC' = PC+1) .
- Special control flow instructions (jumps) are used to alleviate this rule
  → Imperative programming
- Optional out-of-order execution of instructions (enabling better HW utilization) have to produce the same results as the program in their original form

- Instruction in DF computer is represented as the {operation; source(s); destination} template
- Instruction is executed everytime when all its source(s) operands are available/computed
- DF computer do not have a program counter
- The order of instructions execution is based on data dependencies and resources availability → independence of the instruction from the location in the program

Control Flow

- Four subsystems (von Neumann): Memory (Mem), Arithmetic logic unit (ALU), Control Unit, Input/Output subsystem (I/O)
- Control Unit – controls all subsystems – instructions cycle (instruction fetch, decode, execute etc.)



Data Flow

Basic elements:

- Nodes (add, if, switch, merge, not,…), Edges (data, control), Tokens (data, control)

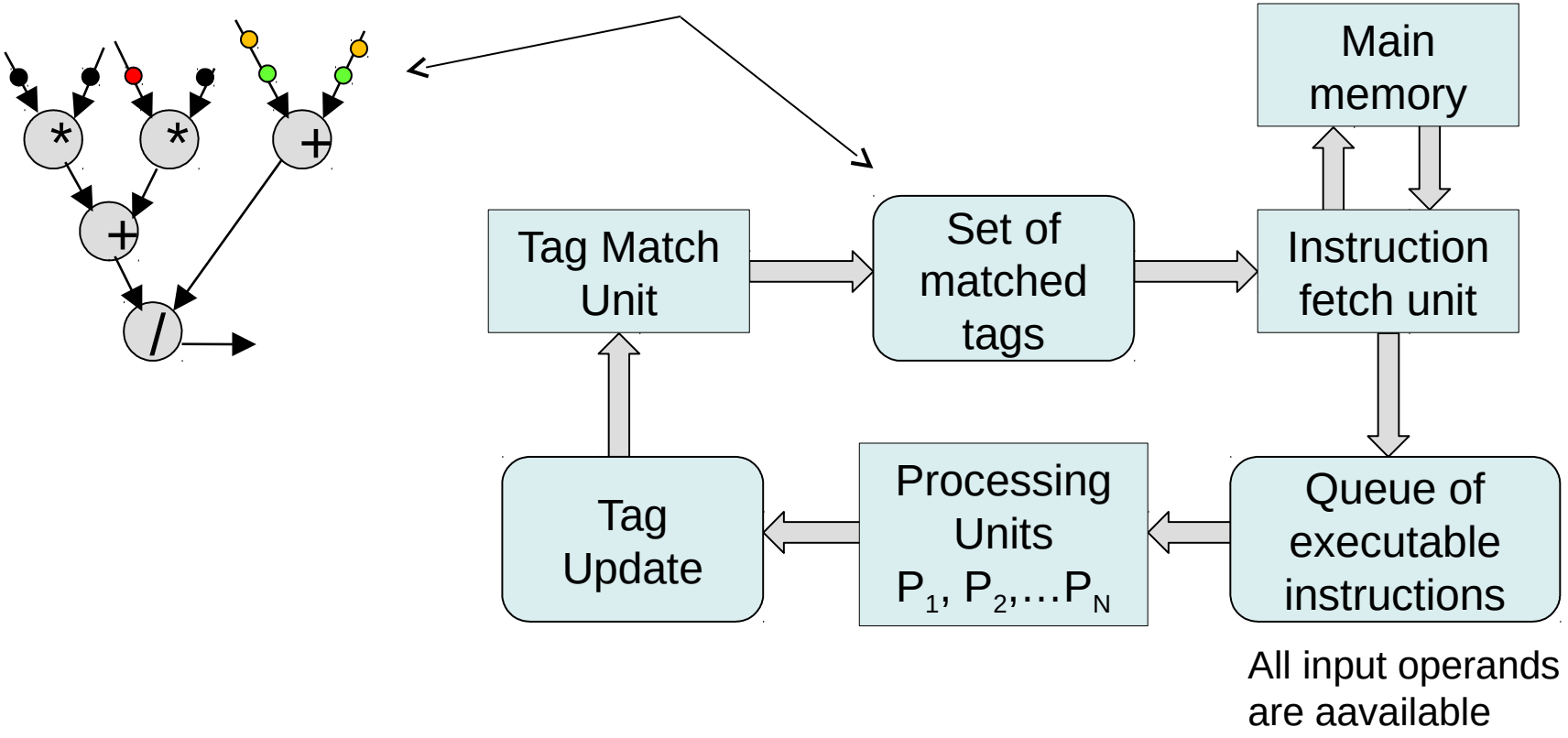
Architecture:

- Static (only one token per edge allowed – ensured by graph construction, so there is no reason to map certain node to the more computational nodes (processing element or unit – PU; program graph is statically allocated to physical nodes PUs)
- Dynamic (more tokens, ordering identified by Tag)
- Hybrid (more tokens; two modes of execution: 1. FIFO (incoming tokens are placed into queue), 2. ordering is identified by Tag)

Execution:

- Static (execution is fired only when no token is on output, and all operation inputs are ready, tokens are present)
- Dynamic

Dynamic architecture of DF computer



Data Flow – Mechanisms to support DF

- Evaluation of conditions for nodes to be executable (all input operands available).
- Ready nodes (instruction) scheduling to available processing units/processors/PU
- Token flow between nodes (from output node towards input node)
- Dynamic nodes (or subgraphs) support – due to cycles in the graph (Certain node in the graph may need several instances)
- Distinction between tokens of different instances of the same node
- Interaction with computer (I/O)
- Detection of EndOfProgram

Data Flow Architectures – Conclusion

- Development and progress in 70s and 80-ies
- Nowadays DF used in specialized devices (notably FPGA, SoC)

  e.g., in a reconfigurable SoC, for example for MPEG4 video decoding[1] (9 hierarchical FSM, 31 states, 44 transitions, 89 SDF nodes),

  or in coprocessor for image processing[2] (bar code recognition)

  and more...

Sources:

[1] Sunghyun Lee , Sungjoo Yoo , Kiyoung Choi: Reconfigurable SoC Design with Hierarchical FSM and Synchronous Dataflow Model (2002).

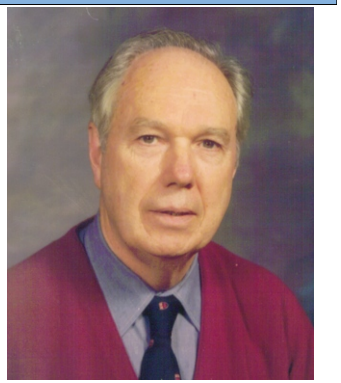
[2] Richard Thavot, Romuald Mosqueron et al. : Dataflow design of a co-processor architecture for image processing. DASIP 2008, Bruxelles, Belgium, 24-26 November 2008.

[3] John A. Sharp: Data Flow Computing: Theory and Practice

[4] http://comjnl.oxfordjournals.org/cgi/reprint/33/3/230

Classification of computer architectures by Flynn's taxonomy

proposed by Michael J. Flynn in 1966

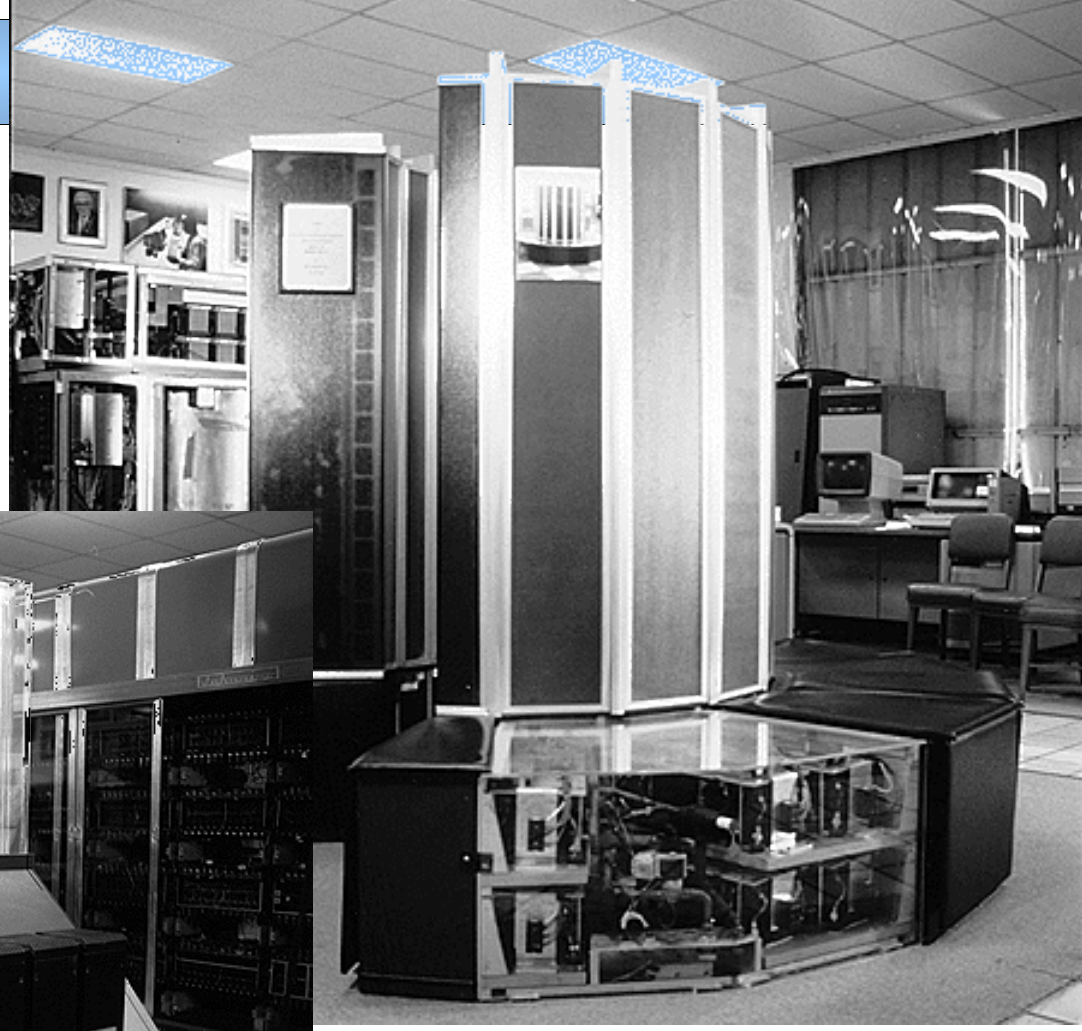


http://arith.stanford.edu/~flynn/

- **SISD** (Single Instruction stream Single Data stream)
  classic von Neumann architecture without parallelism
- **SIMD** (Single Instruction stream Multiple Data stream)
  many Processing Units (PU/PE) ; In history: CM-5, ILLIAC IV; vector and matrix computers; Today, various ISA extensions for (short 4 to 512 element) vector operations (MMX, SSE,..), and also in GPU $\rightarrow$ local support
- **MISD** (Multiple Instruction stream Single Data stream)
  pipelining? Questionable if MISD? ; Yes for fault-tolerant systems – same data flow processed multiple times and results are compared
- **MIMD** (Multiple Instruction stream Multiple Data stream)
  multithreading, multi-core CPUs, multi-processor systems and multiple interconnected computers (Cluster, Grid, Cloud)
  - SPMD (Single Program stream Multiple Data stream)
  - MPMD (Multiple Program stream Multiple Data stream)

Cray II

World fastest computer from 1985 until 1989.
1,9 Gflops peak vector performance



Cray I (1976)

Zdroj:http://ed-thelen.org/comp-hist/Shustek/ShustekTour-03.html

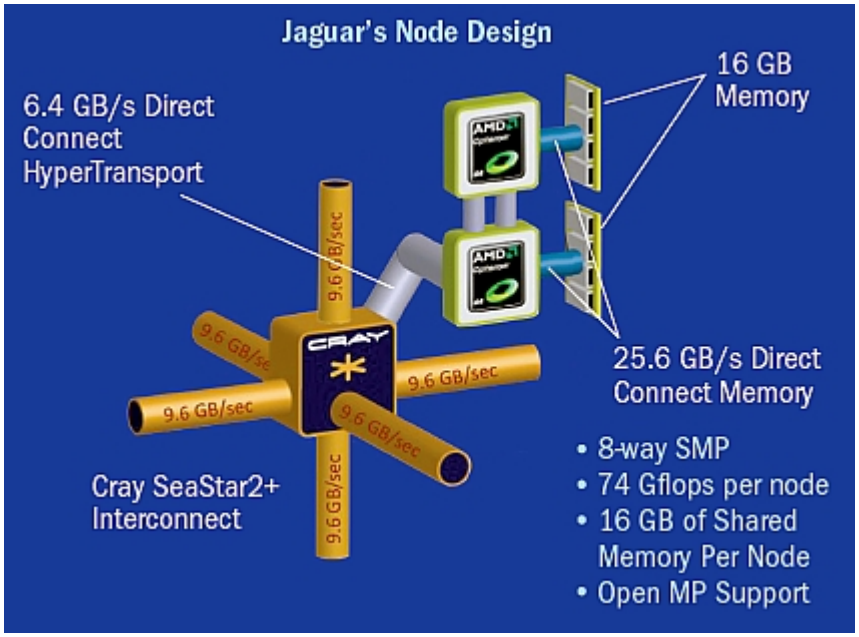
Another Supercomputer Designed by Cray

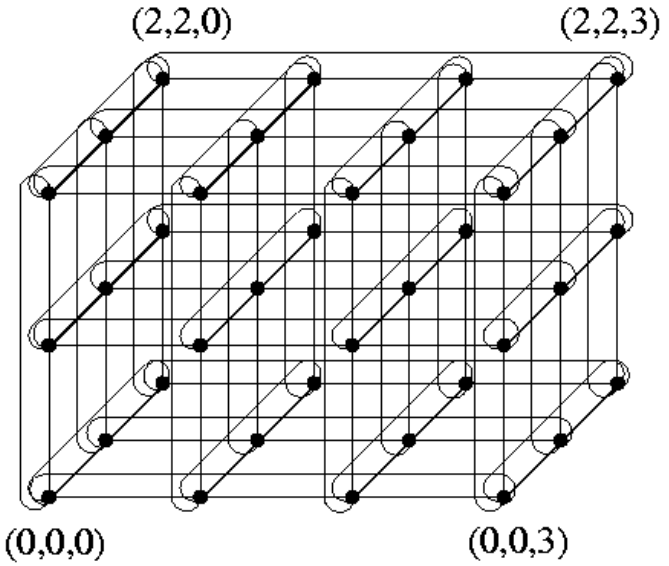
Another Supercomputer Designed by Cray

- Cray XT5-HE (Top system in September 2010)
- MPP (Massivly Patallel Processor)
- Performance 1 759 000 Gflops (Rmax), Rpeak = 2,3 PFlops
- 224 162 compuatational nodes (37 376 processors)
- dual hex-core AMD Opteron 2435 (Istanbul) processors operating at 2.6GHz (10.4 GFlops)
- 300TB memory
- Disk capacity 10.7 PB
- Linux

Information sources:
http://www.top500.org/system/details/10184
http://www.nccs.gov/computing-resources/jaguar/

Cray XT5-HE





- Each node – two 6-core processors
- Each node – 25.6 GB/s throughput to the local memory, 6.4GB/s to the network
- Nodes interconnected in 3D torus
- 12 OpenMP or MPI tasks in node, MPI between nodes
- Libraries for message passing (sending/receiving): MPI 2.0, SHMEM

http://www.scidacreview.org/0901/html/hardware.html

Top 500 most powerful supercomputers – in 2018

Architecture

- Cluster (87.4%), MPP (12.6%)

Operating system

- Linux (100%)

Number of processing units (VE)

- 2049-4096 (22%), 4k - 8k (58%), 8k – 128k (18%), more than 128 000 VE (1%)

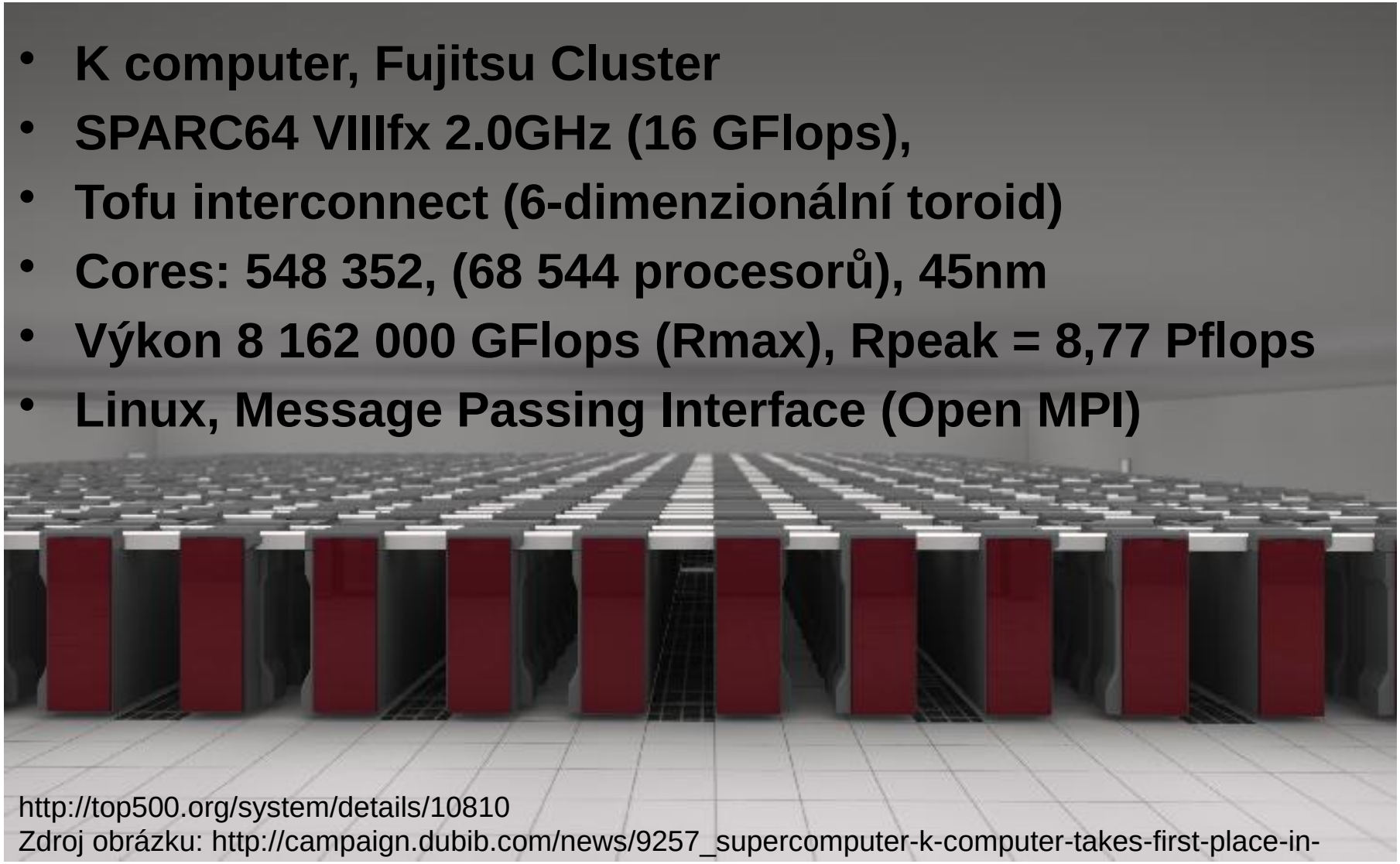
Accelerator

- Nvidia Tesla P100 48.2, Other Tesla 27.8, Intel Xeon Phi 3%

Vendors from Top 10

IBM (3), Cray (4), China NRCPC/NUDT (2), Fujistu (2)

The most powerful in September 2011, 2014 in the 4. place

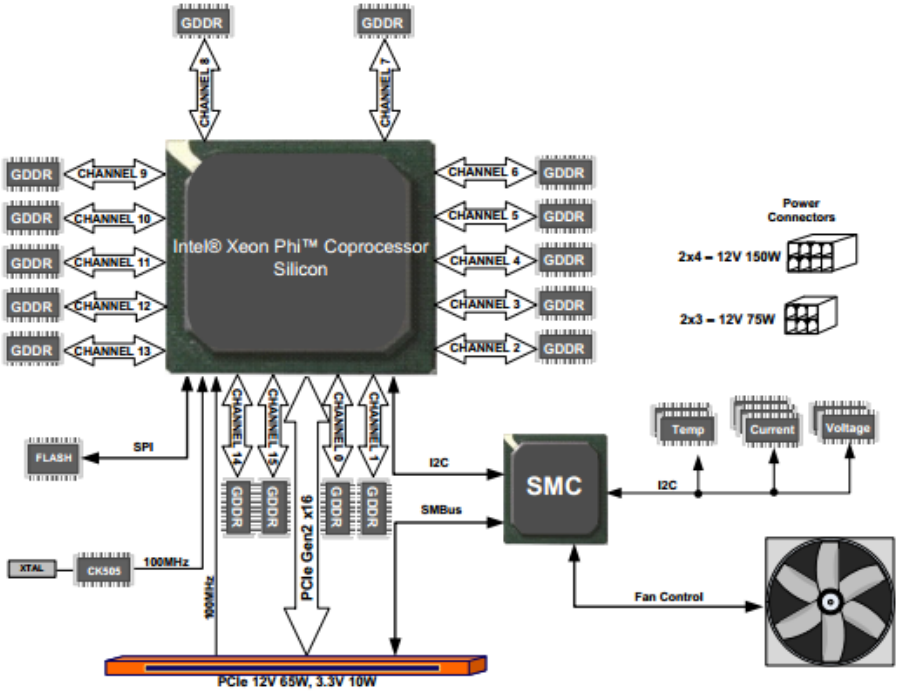


- **K computer, Fujitsu Cluster**
- **SPARC64 VIIIfx 2.0GHz (16 GFlops),**
- **Tofu interconnect (6-dimenzionální toroid)**
- **Cores: 548 352, (68 544 procesorů), 45nm**
- **Výkon 8 162 000 GFlops (Rmax), Rpeak = 8,77 Pflops**
- **Linux, Message Passing Interface (Open MPI)**

http://top500.org/system/details/10810
Zdroj obrázku: http://campaign.dubib.com/news/9257_supercomputer-k-computer-takes-first-place-in-world

The most powerful - September 2014, September 2015

- Tianhe-2 (MilkyWay-2)
- Intel Xeon E5-2692 2.2GHz
  (12 cores, 24 threads,
  12 x 32 KB L1 i caches,
  12 x 32 KB L1 d caches,
  12 x 256 KB L2 caches,
  1 x 30 MB L3 cache)
- Intel Xeon Phi 31S1P (57/228)
- Each node: two Xeon processors and three Xeon Phi coprocessors
- Cores: 3 120 000
- **Perf. 33 862 700 GFlops (Rmax), Rpeak = 54.9 Pflops**
- **New trend: Gflops/Watt**

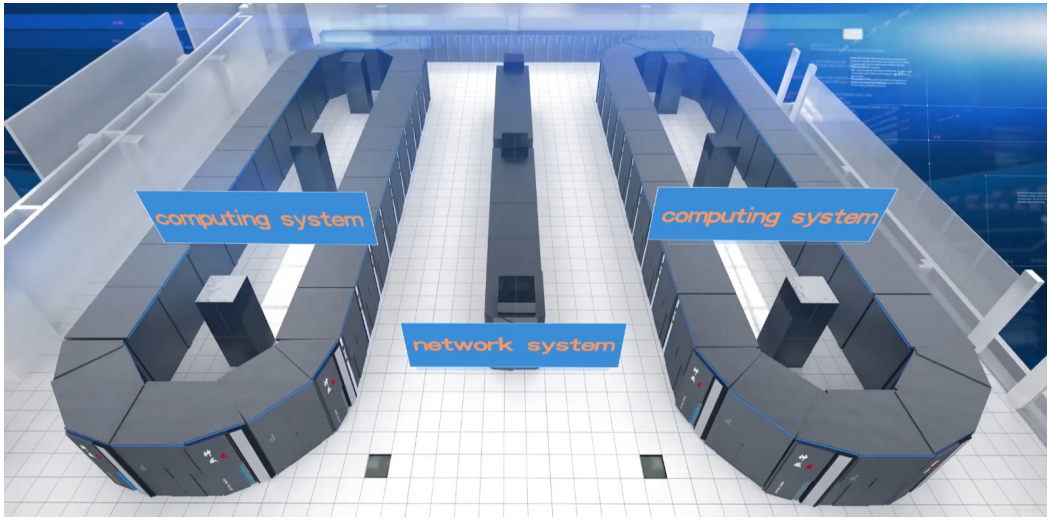


http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/xeon-phi-coprocessor-datasheet.pdf

The most powerful - 2016 to June 2018

Sunway TaihuLight

- 93 PFLOPS (LINPACK benchmark), peak 125 PFLOPS
- Interconnection 14 GB/s, Bisection 70 GB/s
- Memory 1.31 PB, Storage 20 PB
- 40,960 SW26010 (Chinese) – total 10,649,600 cores
- SW26010 256 processing cores + 4 management
- 64 KB of scratchpad memory for data (and 16 KB for instructions)
- Sunway RaiseOS 2.0.5
  (Linux based)
- OpenACC
  (for open accelerators)
  programming standard
- Power Consumption
  15 MW (LINPACK)



Summit supercomputer – IBM AC922

June 2018, US Oak Ridge National Laboratory (ORNL), 200 PetaFLOPS, 4600 “nodes”, 2× IBM Power9 CPU + 6× Nvidia Volta GV100

Source: http://www.tomshardware.com/

96 lanes of PCIe 4.0, 400Gb/s

NVLink 2.0, 100GB/s CPU-to-GPU, GPU-to-GPU

2TB DDR4-2666 per node

1.6 TB NV RAM per node

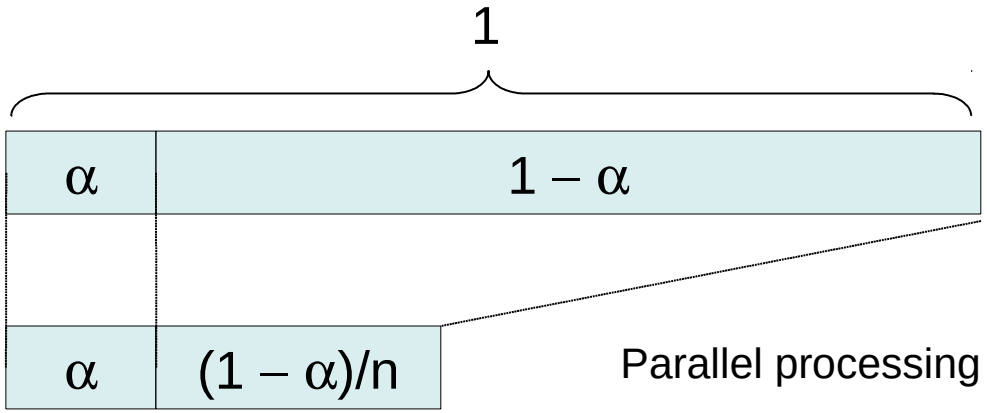
250 PB storage

POWER9-SO, Global Foundries 14nm FinFET, 8×10^9 tran., 17-layer, 24 cores, 96 threads (SMT4)

120MB L3 eDRAM (2 CPU 10MB), 256GB/s

Amdahl's LAW

- Every program consists of some serial (sequential) portion. Let's call it an α.

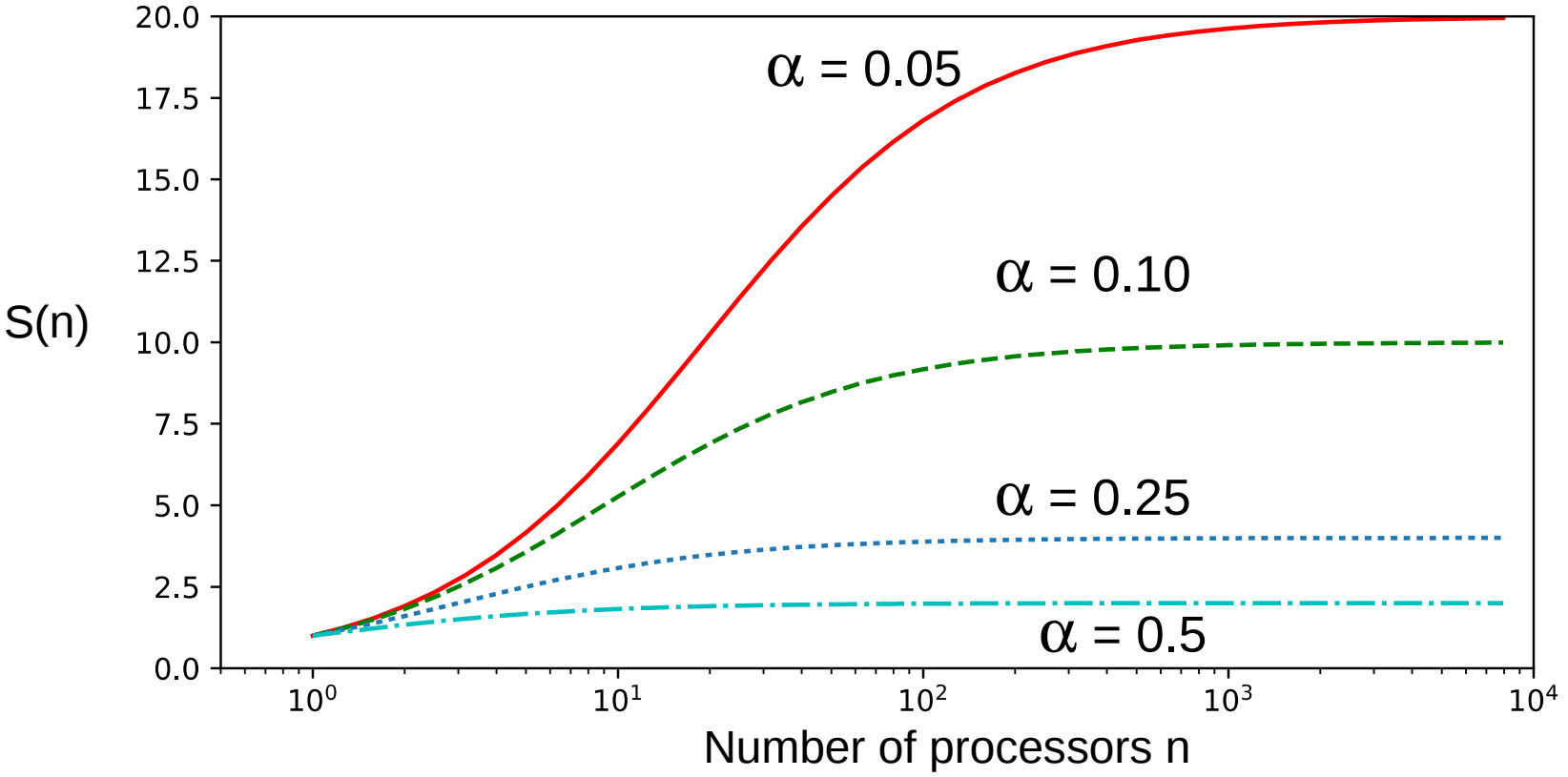


$$S(n) = \frac{T(1)}{T(n)} = \frac{\alpha + 1 - \alpha}{\alpha + (1-\alpha)/n} = \frac{n}{(n-1)\alpha + 1}$$

$$\lim_{n \to \infty} S(n) = \lim_{n \to \infty} \frac{n}{(n-1)\alpha + 1} = \frac{1}{\alpha}$$

Amdahl's LAW

- Speed-up by execution on 1 to 8000 processing units (PU) and α choices 0.05, 0.10, 0.25 a 0.5.



Amdahl's LAW

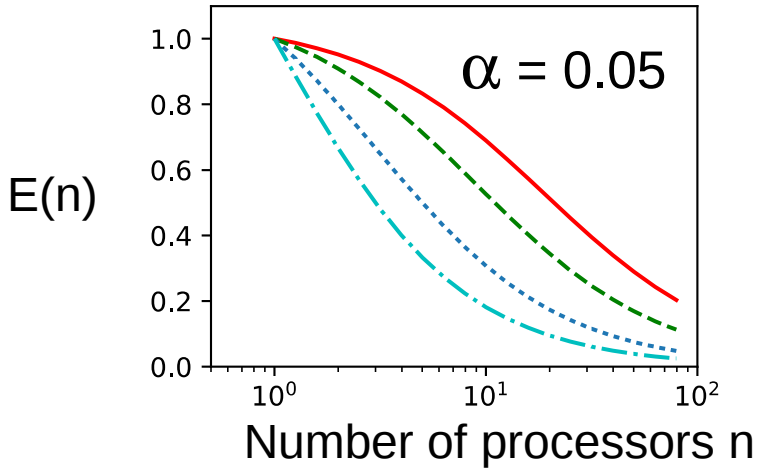
- Speed-up:

$$S(n) = \frac{T(1)}{T(n)}$$

Efficiency:

$$E(n) = \frac{S(n)}{n} = \frac{T(1)}{nT(n)}$$

where $T(1)$ is the computing time when only one Processing Unit (PU) is used, $T(n)$ when n PUs are used.



Is time enhanced/reduced by factor two when we use 2 PUs?

Amdahl's LAW

- Speed-up: Efficiency:

$$S(n) = \frac{T(1)}{T(n)} \qquad E(n) = \frac{S(n)}{n} = \frac{T(1)}{nT(n)}$$

where $T(1)$ is the computing time when only one Processing Unit (PU) is used, $T(n)$ when n PUs are used.

Amdahl's law:

- We have a serial computer (only one PU). The execution of our program takes 10 hours, but only 1 hour is serial portion, the rest of the program can be parallelized.

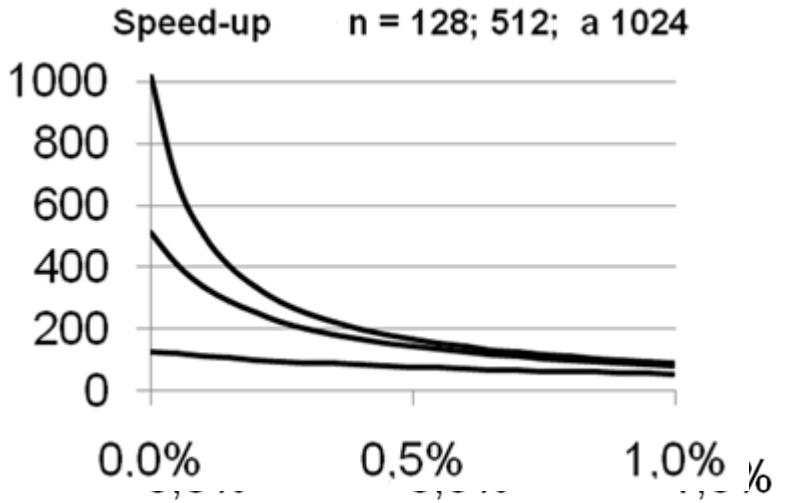
  When increasing number of PUs, we will never under 1 hour! We can express speed-up for n $\rightarrow \infty$:

$$\lim_{n \to \infty} S(n) = \lim_{n \to \infty} \frac{T(1)}{T(n)} = \frac{10}{1} = 10$$

Amdahl's LAW

- Example Let's consider $\alpha = 0{,}05$ (5%).

For
$n = 5$: $S(5) = 4.17$
$n = 10$: $S(10) = 6.9$
$n = 100$: $S(100) = 16.9$
$n \to \infty$: $S(\infty) = 20$

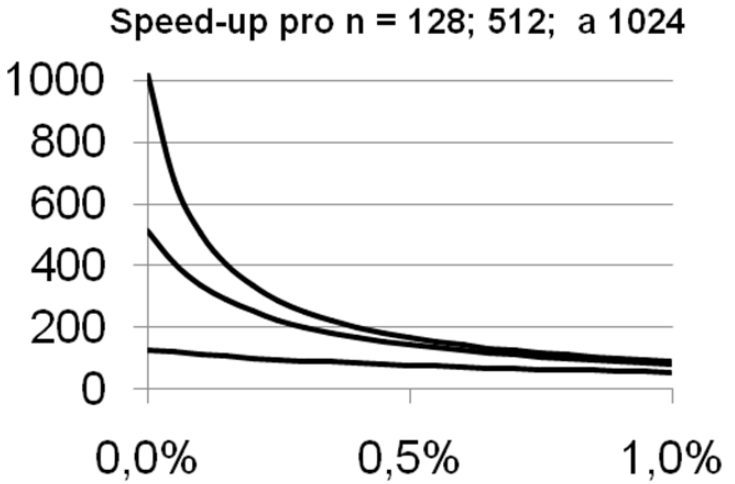


How many PU would you like to buy?

Amdahl's law

- Example. Let's consider $\alpha = 0{,}05$ (5%).

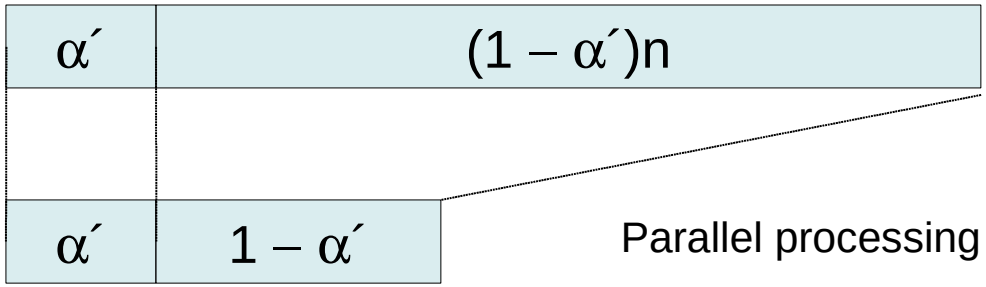
For $n = 5$: $S(5) = 4.17$
$n = 10$: $S(10) = 6.9$
$n = 100$: $S(100) = 16.9$
$n \to \infty$: $S(\infty) = 20$



- Cray XT5-HE has n = 224 162,
  K computer: n = 548 352.
  77% of most powerful supercomputers has number of PUs over 4k.
  Is such a large number of PUs/cores effectively usable? / What type of problems are these systems designed for?

Gustafson's Law

- Example: We have a parallel computer. The execution of our program takes 10 hours, but only 1 hour is serial portion, the rest of the program is parallelized. If the same task will be solved on serial computer, it will takes T(1) = 1 + n*9 hours. Therefore S(n) = (1 + n*9) / 10; and then S(n $\rightarrow \infty$) = ∞.



$$S(n) = \frac{T(1)}{T(n)} = \frac{\alpha' + (1-\alpha')n}{\alpha' + 1 - \alpha'} = n - (n-1)\alpha'$$

$$\alpha' \rightarrow 0 \quad \Rightarrow \quad S(n) \rightarrow n$$

Gustafson's Law

- Example. Let's consider $\alpha' = 0{,}05$ (5%).

  For $n = 5$: $S(5) = 4.8$
  $n = 10$: $S(10) = 9.55$
  $n = 100$: $S(100) = 95.05$
  $n \to \infty$: $S(\infty) = \infty$

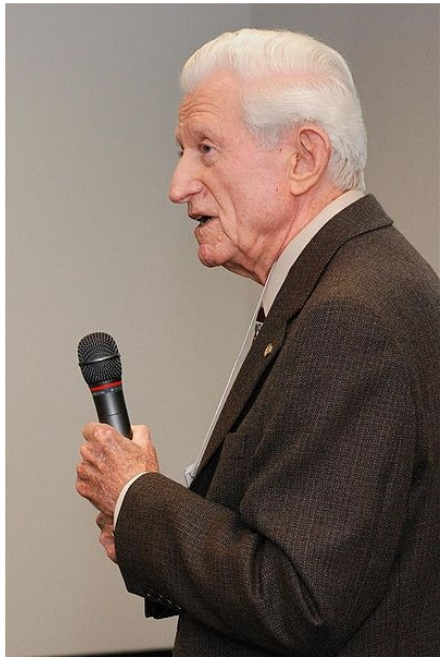
Summary:

- Amdahl – Pessimistic: A sequential portion of the program requires same time on a parallel computer as it takes on a serial one.
- Gustafson – Optimistic: A sequential portion of the program requires same time on a parallel computer as it takes on a serial one .

Where is a difference? α is not same as α'; Gustafson's law assumes the changing magnitude of the problem with the changing computing environment without increase of time required for serial program portion.

Amdahl vs. Gustafson

- born November 16, 1922
- Amdahl, G.M., Validity of single-processor approach to achieving large-scale computing capability, *Proceedings of AFIPS Conference,* Reston, VA. 1967. pp. 483-485.



- born January 19, 1955
- Gustafson, J.L., Reevaluating Amdahl's Law, CACM, 31(5), 1988. pp. 532-533.

[1] http://en.wikipedia.org/wiki/Gene_Amdahl
[2] http://en.wikipedia.org/wiki/John_Gustafson_%28scientist%29

Amdahl's and Gustafson's law

- **Example**
  We can replace the existing processor (PU) with a new, 10 times more powerful one. The current processor is loaded at 50%, the rest of the time (50%) waits for I/O operations. What total gain would the CPU replace achieve?

- **Example**
  Let's assume we've achieved an 80x acceleration on a 100-processor system. Which part (percentage) of the program is sequential? How many percent of the time does the sequential part of the program work on this system? Are these percentages the same?

Computer performance

Aspects of performance

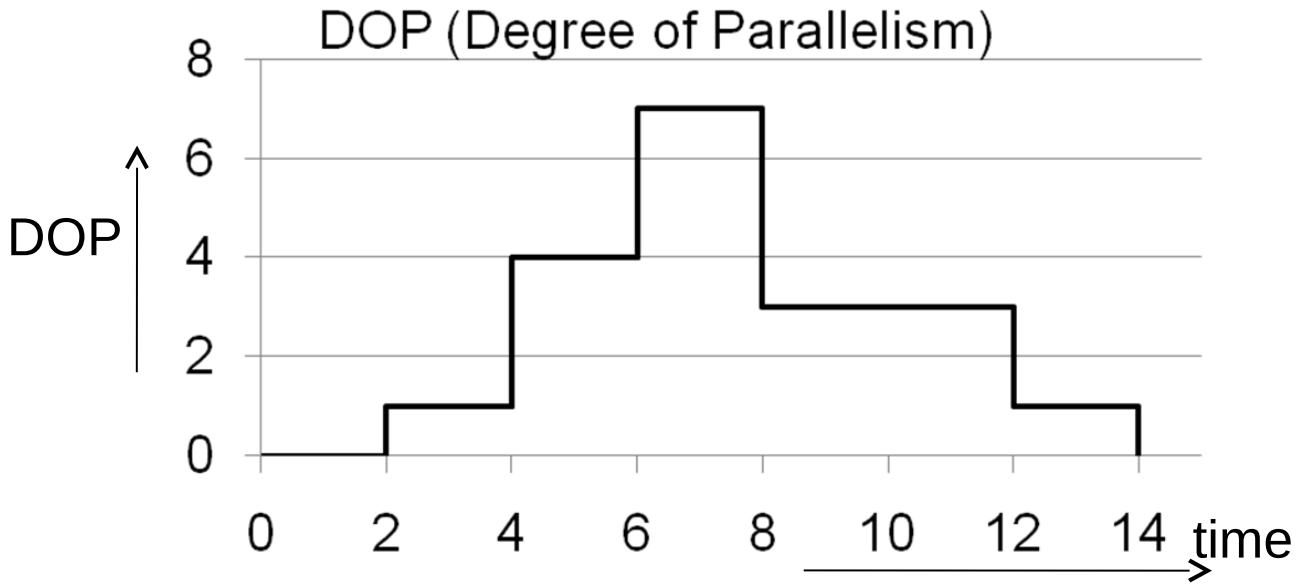
- Processing speed
- Response time
- Throughput
- Availability
- Latency
- Scalability
- Power consumption

In parallel processing (parallel computers):

- degree of parallelism

Metrics of performance

- Degree of parallelism (parallelism profile)
  is the number of PU used to execute for each time period



$$DOP_{STR} = \frac{\sum_i DOP_i . \Delta t_i}{\sum_i \Delta t_i} = \frac{1}{t_n - t_0} \sum_i DOP_i . \Delta t_i = \frac{1}{14-2}\left(1\times 2 + 4\times 2 + 7\times 2 + \ldots\right)$$

Metrics of performance

- Compare of computer performances – which one is the best?

	computer A	computer B	computer C
program 1	12 min	1 hour	1 hour
program 2	20 min	40 min	10 min
program 3	5 hours	1 hour	2 hours

Metrics of performance

The total amount of time required to execute a particular program is:

$$T = IC.CPI.T_{CLK}$$

IC – Instruction Count (to be executed from program to compute results)
CPI – Cycles per Instruction
T_{CLK} – time period of cycle (reciprocal of the clock frequency)

For a given ISA (Instruction Set Architecture), the processor design is the optimization of CPI and T_{CLK}.

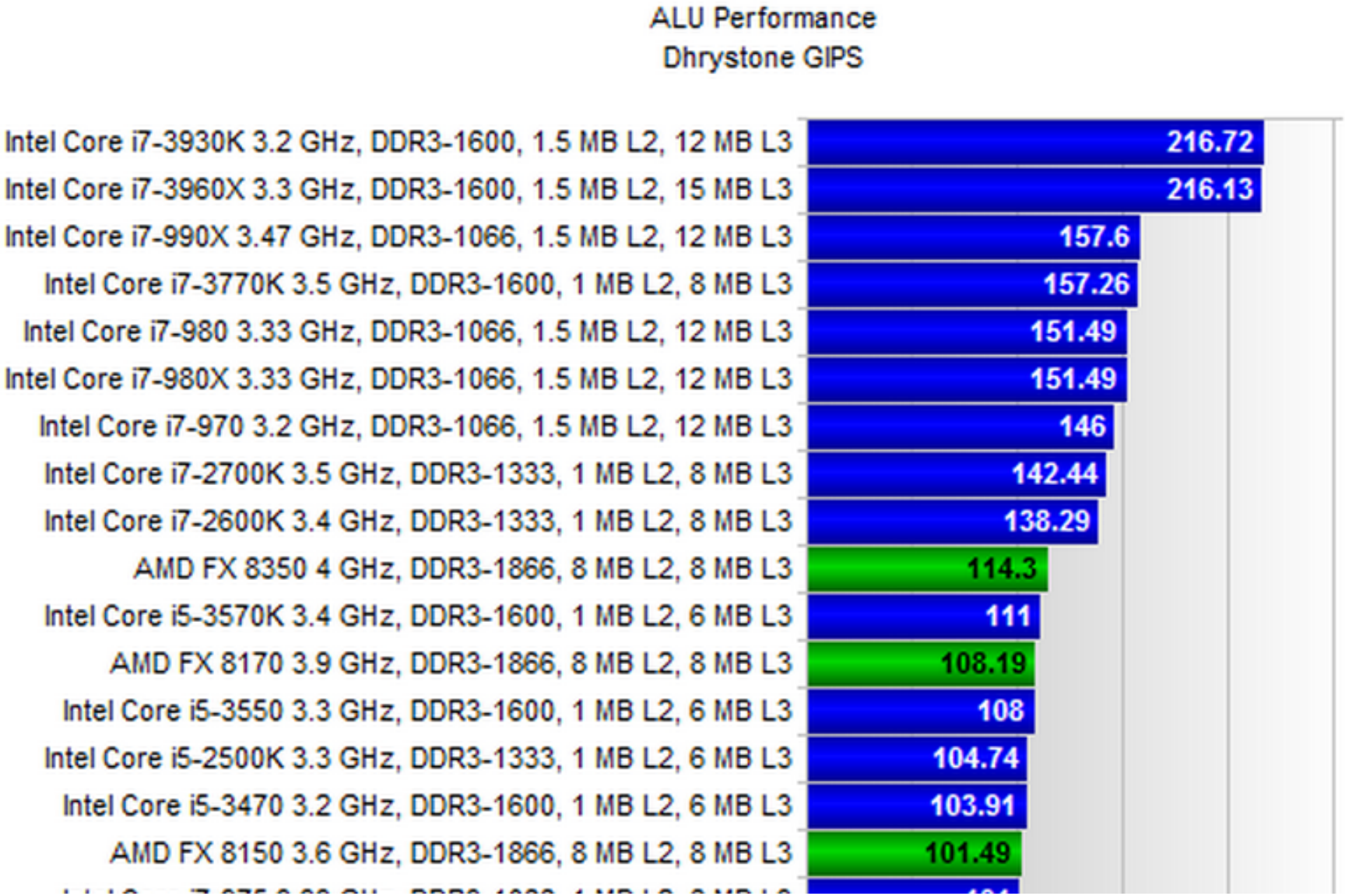
MIPS (Million Instructions per second) where $IPS = IC / T = IPC_{str}.f_{CLK}$, MOPS (Operations), MFLOPS (Floating point Operations) - all is dependent on the program and data to be processed and depends on instruction set (ISA) used on given computer… (Dhrystone MIPS are different, normalized to predefined scale for given task)

Program throughput::

$$W = 1 / T = IPC.f_{CLK} / IC$$

IPC – Instructions per Clock

Metrics of performance



Metrics of performance

Příklad

2 GHz processor executes the program with the following mix of instructions. What is the mean of CPI, the speed of program execution (MIPS), the program throughput (Wp) and the execution time (T)?

Instruction type	Number of instructions	Number of clock cycles
Celočíselní aritmetika	45000	1
Přenos údajů	32000	2
Pohyblivá řádová čárka	15000	3
Přenos řízení(skoky)	8000	2

Used symbols

- T_{CLK} – Clock period
- f – Clock frequency:
- IC – Program size (instruction count)
- C – Total number of cycles needed to execute program

Metrics of performance

Let's $\{R_i\}$ are execution speeds of different programs

i = 1, 2, ..., m measured in MIPS (MFLOPS), or IPS (FLOPS)

- The arithmetic mean performance: $R_a = \sum_{i=1}^{m} \frac{R_i}{m} = \frac{1}{m} \sum_{i=1}^{m} R_i$

  R_a is equally weighted *(1/m)* in all programs and is proportional to the sum of the IPC, but not the sum of execution times (inversely proportional). Arithmetic mean (average) not generally usable:

$$R_a = \frac{1}{2}(R_1 + R_2) = \frac{1}{2}\left(\frac{IC_1}{T_1} + \frac{IC_2}{T_2}\right) = \frac{1}{2}\left(\frac{IC_1}{IC_1 . CPI_1 T_{CLK}} + \frac{IC_2}{IC_2 . CPI_2 T_{CLK}}\right) =$$

$$= \frac{1}{T_{CLK}}\left(\frac{IPC_1 + IPC_2}{2}\right) = \frac{1}{T_{CLK}}\left(\frac{IC_1}{2C_1} + \frac{IC_2}{2C_2}\right) \quad but \quad IPC_{1,2} = \frac{IC_1 + IC_2}{C_1 + C_2}$$

  Only iff $C_1 = C_2$ (total number of cycles of both programs is equal) tnen R_a is usable

- In praxis: The arithmetic mean of execution speed of two (or more) different programs is not related to overall execution speed! Not usable!

Metrics of performance

- The geometric mean:

$$R_g = \prod_{i=1}^{m} R_i^{\frac{1}{m}}$$

It does not summarize real performance. It has no inverse relation to overall execution time of all programs. Usable only for comparison with normalized results to reference compute.

- The harmonic mean:

$$R_h = \frac{m}{\sum_{i=1}^{m} \frac{1}{R_i}}$$

$$R_h = \frac{2}{\frac{1}{R_1}+\frac{1}{R_2}} = \ldots = \frac{1}{T_{CLK}}\left(\frac{2}{CPI_1+CPI_2}\right) = \frac{1}{T_{CLK}}\frac{2IC_1IC_2}{C_1IC_2+C_2IC_1}$$

Only iff $IC_1 = IC_2$ (both programs are of the same size)then R_h is usable

- There exist even weighted versions of these performance metrics.

Metrics of performance conclusion

- So, how we can do a comparison between computers (processors)?
- The “processor speed” highly depends on used benchmark!
- Actually, it is not the “processor speed”, but the speed of execution of that benchmark

Todat trends and next lecture topics

The goal is to maximize the program throughput

$$W = 1 / T = IPC \cdot f_{CLK} / IC$$

Technology

- The speed of the electrical signals is limited by the speed of the light (and dependents on the material properties of the medium) -> at 3.6GHz – period is 0.278 ns => 8.3 cm
- Today: 10 nm CMOS technology

  (The atomic radius of Si is 0.111 nm)

Architecture (Hardware + Organization of HW + ISA)

- Hardware supported parallelism at lowest level (bit-level parallelism),
- Instruction parallelism (pipelining, superscalar organization, speculative execution, out-of-order execution,…)
- Thread level parallelism,
- Data parallelism (see loop-level parallelism),
- Task-level parallelism (functional and control parallelism)