TI II: Computer Architecture
Introduction

Single Processor Systems
Historical Background
Classification / Taxonomy
Architectural Overview
Examples
The Layered Computer Model
SINGLE PROCESSOR SYSTEMS
1.3

Computer System – the old days …
Computer System – very old days
Computer System – not much to see
Classical architecture of microcomputers
Classical idea of a modular computer system

- SCSI controller
- Sound card
- Modem
- Edge connector
- Card cage
A complete classical single processor system
A complete classical single processor system

- I/O connectors (USB, serial, parallel)
- PCI slots
- CPU socket
- AGP slot
- Power supply
- Memory sockets
- EIDE connector (disk, CD, DVD)
PC Architecture 2014

www.heise.de
Simple single processor system

Do we still have this today? Think of, e.g., mobile phone with main CPU, radio modem, graphics accelerator, GPS, WLAN, ... later more about this!
THE VON NEUMANN ARCHITECTURE
The von Neumann architecture

The von Neumann architecture forms the basis of many hardware architectures presented in this course.

The architecture comprises the following main components
- Central processing unit
  - Control unit
  - ALU / Operating unit
- Memory
- Input/Output units
- Interconnection
The von Neumann architecture

Central control of the computer

A computer consists of several functional units (central processing unit, memory, input/output unit, connection)

The computer is not tailored to a single problem but a general purpose machine. In order to solve a problem a program is stored in the memory (“program controlled universal computer”) – yes, today this sounds so simple…

IMPORTANT
- Instructions (the program) and data (input and output values) are stored in the same memory.
- The memory consists of memory cells with a fixed length, each cell can be addressed individually.
The von Neumann architecture

**Processor, central unit** (CPU: "central processing unit")
- Controls the flow and execution of all instructions

**Control unit**
- Interprets the CPU instructions
- Generates all control commands for other components

**Arithmetic Logical Unit (ALU)**
- Executes all instructions (I/O and memory instructions with the help of these units)

**Input/Output system**
- Interface to the outside world
- Input and output of program and data

**Memory**
- Storage of data and program as sequence of bits

**Interconnection**
The von Neumann Architecture

PRINCIPLE OF OPERATION OF A COMPUTER
Principle of Operation of a Computer

At any time the CPU executes only a **single instruction**. This instruction can only manipulate a **single operand**.
- Traditionally, this is called **SISD** (Single Instruction Single Data).

**Code and data** are stored in the **same memory** without any distinction. There are no memory protection mechanisms – programs can destroy each other, programs can access arbitrary data.

**Memory is unstructured** and is addressed linearly. Interpretation of memory content depends on the context of the current program only.

**Two phase principle** of instruction processing:
- During the interpretation phase the content of a memory cell is fetched based on a **program counter**. This content is then interpreted as an instruction (note: this is a pure interpretation!).
- During the execution phase the content of a memory cell is fetched based on the address contained in the instruction. This content is then interpreted as data.

The instruction flow follows a strict sequential order.
Principle of Operation of a Computer Instruction Execution

```java
public class Interpreter {

    static int PC; // Program counter holds the address of the next instruction
    static int AC; // Register for doing arithmetic, accumulator
    static int instruction; // Current instruction
    static int instructionType; // Type of the current instruction, i.e. what to do
    static int dataLocation; // Address of the data for the instruction
    static int data; // Holds the operand
    static boolean runBit = true; // Bit used to halt the computer

    public static void interpreter(int[] memory, int startingAddress) {
        PC = startingAddress; // Initialize the program

        while (runBit) {
            instruction = memory[PC]; // Fetch next instruction
            PC = PC + 1; // Increment PC
            instructionType = getInstructionType(instruction); // Determine instruction
            dataLocation = findData(instruction, instructionType); // Locate data
            if (dataLocation >= 0) // No operand if -1
                data = memory[dataLocation]; // Fetch data
            execute(instructionType, data); // Execute instruction
        }
    }
}
```
Principle of Operation of a Computer Instruction Execution

Example: interpreter(memory, 256);

memory[256] = 80  memory[261] = 7
memory[257] = 0  memory[262] = 20
memory[258] = 5  memory[263] = 2
memory[259] = 80  memory[264] = 0
memory[260] = 1  memory[265] = 1

Byte sequence: 80 0 5 80 1 7 20 2 0 1
Bit sequence: 01010000 00000000 00000101
             01010000 00000001 00000111
             00010100 00000010 00000000 00000001

Assembler representation (MMIX Style)
- LDB $0,5
- LDB $1,7
- ADD $2,$0,$1

High-level programming language representation
\[ Z = X + Y \]
Pros and Cons of the von Neumann architecture

Advantages
- Principle of minimal hardware requirements
- Principle of minimal memory requirements

Disadvantages
- The main interconnection (memory ↔ CPU)
  is the central bottleneck: the “von Neumann bottleneck”
- Programs have to take into account the sequential data flow across the von Neumann bottleneck
  → Influences on higher programming languages (“intellectual bottleneck”)
- Low structuring of data (a long sequence of bits…)
- The instruction determines the *operand type*
- No memory protection
Look around:
Not everything is von Neumann!

Basic “von Neumann-architecture”: data and program are stored in the same memory

Typical for the PC architecture…
- well, depends on your viewpoint…
Harvard Architecture

Classical definition of the Harvard architecture
- Separation of data and program memory

Most processors with microarchitecture
- von Neumann from the outside
- Harvard from the inside
- Reason
  - Different time scales and locality when caching data and instructions
High-level comparison

von Neumann
- general purpose memory
- CPU

Harvard
- program memory
- CPU
- data memory
MICRO COMPUTERS
**Special purpose micro processors**

Micro processors for special applications exist next to universal microprocessors (standard-micro-processss):

- Micro controller
- Signal processors
- Coprocessors

![Image of micro processor](image-url)
Definition of a Micro computer system

Micro processor system:
- Digital system, using a micro processor as central control and/or arithmetic unit

Micro computer:
- includes a micro processor that communicates with memory, controllers and interfaces for external devices using the system bus.

Special cases of microcomputers:
- Single-chip microcomputer
  - All components of the microcomputer are located on one chip.
- Single-circuit microcomputer (dt. Ein-Platinen-Mikrocomputer)
  - All components of the microcomputer are on one circuit board.

Microcomputer system:
- Microcomputer with connected external devices
- Can be small – think of the Internet of Things!
Basic concept of a micro computer

The IBM PC is a modified von Neumann architecture and was introduced by IBM fall 1981. The interconnection structure was realized by a **bus**.

- The bus connects the CPU with the main memory, several controllers and the input/output system.
Interconnection

The bus comprises the
- data bus
- address bus
- control bus

The bus is controlled by a **bus controller** and uses a buffer for data and addresses to communicate with the CPU.

The CPU fetches each instruction sequentially from the main memory.

The controllers are accessed with an I/O mapped model
- The internal registers of a controller are accessed with special I/O instructions using port addresses.
Components of a Computer

Hardware: all mechanical and electronic components
Software: all programs running on the computer
Firmware: micro-programs stored in ROM, somewhere in-between SW and HW

Research: Where to place a function – software or hardware?
Hardware and Software are logically equivalent!
History of Computers

See for example:

PERFORMANCE OF PROCESSORS
Anwendung vom Mooreschen Gesetz auf CPU-Chips

„Alle 18 Monate verdoppelt sich die verfügbare Rechenleistung“
120 Years of Moore’s Law

Source: Ray Kurzweil, DFJ
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1961</td>
<td>UNIVAC I</td>
<td>1,000</td>
<td>125,000</td>
<td>2,000</td>
<td>48</td>
<td>$1,000,000</td>
<td>1</td>
<td>$6,107,600</td>
<td>1</td>
</tr>
<tr>
<td>1964</td>
<td>IBM S/360 model 50</td>
<td>60</td>
<td>10,000</td>
<td>500,000</td>
<td>64</td>
<td>$1,000,000</td>
<td>263</td>
<td>$4,792,300</td>
<td>318</td>
</tr>
<tr>
<td>1965</td>
<td>PDF-8</td>
<td>8</td>
<td>500</td>
<td>330,000</td>
<td>4</td>
<td>$16,000</td>
<td>10,855</td>
<td>$75,390</td>
<td>13,135</td>
</tr>
<tr>
<td>1976</td>
<td>Cray-1</td>
<td>58</td>
<td>60,000</td>
<td>166,000,000</td>
<td>32,000</td>
<td>$4,000,000</td>
<td>21,842</td>
<td>$10,756,800</td>
<td>51,604</td>
</tr>
<tr>
<td>1981</td>
<td>IBM PC</td>
<td>1</td>
<td>150</td>
<td>240,000</td>
<td>256</td>
<td>$3,000</td>
<td>42,105</td>
<td>$5,461</td>
<td>154,673</td>
</tr>
<tr>
<td>1991</td>
<td>HP 9000/ model 750</td>
<td>2</td>
<td>500</td>
<td>50,000,000</td>
<td>15,384</td>
<td>$7,400</td>
<td>3,556,188</td>
<td>$9,401</td>
<td>16,122,356</td>
</tr>
<tr>
<td>1996</td>
<td>Intel PPro PC (200 MHz)</td>
<td>2</td>
<td>500</td>
<td>400,000,000</td>
<td>15,384</td>
<td>$4,400</td>
<td>47,846,890</td>
<td>$4,945</td>
<td>239,078,908</td>
</tr>
<tr>
<td>2003</td>
<td>Intel Pentium 4 PC (3.0 GHz)</td>
<td>2</td>
<td>500</td>
<td>6,000,000,000</td>
<td>262,144</td>
<td>$1,600</td>
<td>1,875,000,000</td>
<td>$1,600</td>
<td>11,452,000,000</td>
</tr>
</tbody>
</table>

Source: Patterson, Hennessy, Computer Organization and Design
Example Processor: Intel Core i9

Up to 18 cores
Up to $2000
Up to 1.3 Tflop/s dual precision
Up to 165 W TDP
But also with 1kW …

Limits?
Useful?
How to go even faster?
Current Mainboards
Yet another Example: High Performance for 500€
Physical Layout
High Graphics Performance

- 4 compute processors, 2 command processors, DX12 firmware, new work distribution
- 4 shader engines: 4.688 G primitives/second (2.6x)
- 4 shader arrays:
  - 6 TFLOPs: 40 compute units x 128 FLOPs x 1.172 GHz (4.3x)
  - 187.5 G bilinear texels/second (4.3x)
- 8 RB color/depth engines: 37.5 G pixels/second (2.6x)
- 2 MB L2 cache with bypass and index buffer access (4x)
- 1 MB parameter cache (4x)
- Conservative occlusion query
- Delta color and depth compression
- Compressed texture access
- OOO rasterization
- Xbox One S and 360 compatibility
Many Cores, a lot of Caching

- 8 x86 cores, 2.3 GHz (1.3x)
- 32 KB L1I, 32 KB L1D caches per core
- 4 MB shared L2 cache (2 MB per 4 core cluster)
- Lower main memory latency (up to 20%)
- 12 channels and 192 banks of main memory (3x and 6x)
- 2048 entry L2I TLB and L2D TLB for 4 KB pages (4x)
- 32 entry L1I TLB for 4 KB pages, 8 entry L1I TLB for 2 MB pages
- 40 entry L1D TLB for 4 KB pages, 8 entry L1D TLB for 2 MB pages,
  256 entry L2D TLB for 2MB pages
- Page Descriptor Cache of nested translations (up to 4.3% performance)
Many Interfaces

Totally confused? At the end of the course you will understand the basic principles even behind these state-of-the-art systems!
WHAT IS COMPUTER ARCHITECTURE?
What is Computer Architecture?

Different opinions exist
- Hardware structure, components, interfaces
- Basic operation principle, applications
- Only external view
- Internal and external view
- …

Computer architecture is NOT (only) standard PC architecture!
- The vast majority of computers are embedded systems, specialized solutions
- One size does NOT fit it all

 ➤ Should be: Computer Architectures
Amdahl, Blaauw and Brooks 1967

“Computer architecture is defined as the attributes and behavior of a computer as seen by a machine-language programmer. This definition includes the instruction set, instruction formats, operation codes, addressing modes, and all registers and memory locations that may be directly manipulated by a machine language programmer.

Implementation is defined as the actual hardware structure, logic design, and data path organization of a particular embodiment of the architecture.”
Another view: processor architecture

The processor architecture (Instruction Set Architecture) comprises the description of attributes and functions of a system as seen from a machine language programmer’s point of view.

The specification of the processor architecture comprises:
- instruction set
- instruction formats
- addressing modes
- interrupt handling
- logical address space
- Register/memory model (as far as a programmer can access it)

The processor architecture does not describe details of the implementation or hardware – all internal operations and components are explicitly excluded.
From the architects of the DEC Alpha microprocessor 1992

"Thus, the architecture is a document that describes the behavior of all possible implementations; an implementation is typically a single computer chip.

The architecture and software written to the architecture are intended to last several decades, while individual implementations will have much shorter lifetimes.

The architecture must therefore carefully describe the behavior that a machine-language programmer sees, but must not describe the means by which a particular implementation achieves that behavior."
Processor micro architecture

An implementation (micro architecture) describes the hardware structure, all data paths, the internal logic etc. of a certain realization of the processor architecture, thus a real microprocessor.

The micro architecture defines:
- Number and stages of pipelines
- Usage of super scalar technology
- Number of internal functional units (ALUs)
- Organization of cache memory

The definition of a processor architecture (ISA, instruction set architecture) enables the use of programs independent of a certain internal implementation of a microprocessor.

All microprocessors following the same processor architecture specification are called “binary compatible” (i.e., the same binaries run on them).
CLASSIFICATION OF COMPUTERS
Classification of Computers: Levels of and techniques for parallelism

How to classify those many different systems, different architectures?
- By hardware? Changes too fast…
- By software? Can run on many systems…
- By size? Price? ???
- Here: by parallelism supported

A parallel program can be seen as a partially ordered set of instructions. The order is given by the dependencies among the instructions. Independent instructions can be executed in parallel.

Levels of parallelism: within a program/a set of programs

Techniques for parallelism: implemented in hardware
Classification of Computers: 5 levels of parallelism

Program level

Process level (or task level)
- Tasks (heavy weighted processes, coarse-grained tasks)

Block level
- Threads, Light-Weighted Processes

Instruction level
- Basic instructions (of the chosen language, of the instruction set)

Sub-operation level
- Basic instructions of a language/an instruction set can be divided into sub-operations/micro operations by a compiler/the processor (e.g., Pentium 4 uses micro instructions internally)
Level of parallelism and granularity

The granularity depends on the relation of computation effort over communication or synchronization effort.

Typically, program, process, and thread level are considered as large grained parallelism.

Thus, instruction and sub-operation level may support fine grained parallelism.

Sometimes, the block or thread level is considered as medium grained.
# Techniques of parallelism vs. levels of parallelism

## Techniques

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Program level</th>
<th>Process level</th>
<th>Block level</th>
<th>Instruction level</th>
<th>Sub-operation level</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Loosely coupled computers</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hyper- and Meta-computer</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Workstation-Cluster</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Coupled multi processor systems</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message coupled</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory coupled (SMP)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory coupled (DSM)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large grained data flow principle</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Processor internal structures</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction pipelining</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Super scalar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>VLIW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Overlapping I/O and CPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Fine grained data flow principle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td><strong>SIMD and microarchitectures</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vector and field computers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Micro architectures</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
PERFORMANCE ASSESSMENT OF COMPUTER SYSTEMS
Performance assessment of computer systems

Needed for:
- Selection of a new computer system
- Tuning of an existing system
- Design of new computer systems

Methods for performance assessment
- Evaluation of hardware parameters
- Run-time measurements of existing programs
- Monitoring during operation of real computer systems
- Model theoretic approaches
Methods for performance assessment

Performance parameters
- Hypothetical maximum performance
- MIPS or MOPS (Millions of Instructions/Operations per Second)
- MFLOPS (Millions of Floating Point Operations per Second, today Tera FLOPS, TFLOPS ~10^{12})
- MLIPS (Millions of Logical Inferences per Second)

Mixes
- Theoretical mix of operations

Benchmark programs
- Number crunching, office applications, …

Monitors / measurement during operation
- Hardware monitor
- Software monitor

Model theoretic approaches
- Analytical methods
- Software simulation
Mixes

Calculation of an assumed average run-time based on the durations of $n$ operations according to the formula:

$$T = \sum_{i=1}^{n} p_i t_i$$

$T$  average duration

$n$  number of distinct operations

$t_i$  duration of operation $i$

$p_i$  relative weight of operation $i$ (i.e., relative number of appearances)

The following must hold:

$$\sum_{i=1}^{n} p_i = 1 \quad \text{and} \quad p_i \geq 0$$
Benchmark programs

Sieve of Erathostenes, Ackermann function
Whetstone (1970, typ. Fortran, lots of floating point arithmetic)
Dhrystone (begin of 80s, 53% assignments, 32% control statements, 15% function/procedure calls)
Savage-Benchmark (mathematical standard functions)
Basic Linear Algebra Subprograms (BLAS), core of the LINPACK/LAPACK (Linear Algebra Package) software package
Lawrence Livermore Loops (for vectorizing compilers)

SPEC-Benchmark Suite
**SPEC benchmarks**

SPEC Standard Performance Evaluation Corporation
- since 1989, many vendors, general purpose applications, focuses on calculation speed and throughput
  ([www.spec.org](http://www.spec.org))

Many benchmark suites exist:
- SPEC CPU2006, CPU2017 (Integer, Floating point)
- SPEC MPI2007, OMP2012, ACCEL (High Performance Computing)
- SPEC VIRT_SC 2013 (virtualized servers)
- SPEC Cloud_IaaS 2016
- ...

“[…]. SPEC is once again seeking to encourage those outside of SPEC to assist us in locating applications that
could be used in the next CPU-intensive benchmark suite, currently planned to be SPEC CPU2004.”
# Integer Component of SPEC CPU2017

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>500.perlbench_r</td>
<td>600.perlbench_s</td>
<td>C</td>
<td>362</td>
<td>Perl interpreter</td>
</tr>
<tr>
<td>502.gcc_r</td>
<td>602.gcc_s</td>
<td>C</td>
<td>1,304</td>
<td>GNU C compiler</td>
</tr>
<tr>
<td>505.mcf_r</td>
<td>605.mcf_s</td>
<td>C</td>
<td>3</td>
<td>Route planning</td>
</tr>
<tr>
<td>520.omnetpp_r</td>
<td>620.omnetpp_s</td>
<td>C++</td>
<td>134</td>
<td>Discrete Event simulation - computer network</td>
</tr>
<tr>
<td>523.xalancbmk_r</td>
<td>623.xalancbmk_s</td>
<td>C++</td>
<td>520</td>
<td>XML to HTML conversion via XSLT</td>
</tr>
<tr>
<td>525.x264_r</td>
<td>625.x264_s</td>
<td>C</td>
<td>96</td>
<td>Video compression</td>
</tr>
<tr>
<td>531.deepsjeng_r</td>
<td>631.deepsjeng_s</td>
<td>C++</td>
<td>10</td>
<td>Artificial Intelligence: alpha-beta tree search (Chess)</td>
</tr>
<tr>
<td>541.leela_r</td>
<td>641.leela_s</td>
<td>C++</td>
<td>21</td>
<td>Artificial Intelligence: Monte Carlo tree search (Go)</td>
</tr>
<tr>
<td>548.exchange2_r</td>
<td>648.exchange2_s</td>
<td>Fortran</td>
<td>1</td>
<td>Artificial Intelligence: recursive solution generator (Sudoku)</td>
</tr>
<tr>
<td>557.xz_r</td>
<td>657.xz_s</td>
<td>C</td>
<td>33</td>
<td>General data compression</td>
</tr>
</tbody>
</table>
## Floating Point Component of SPEC CPU2017

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>503.bwaves_r</td>
<td>603.bwaves_s</td>
<td>Fortran</td>
<td>1</td>
<td>Explosion modeling</td>
</tr>
<tr>
<td>507.cactusBSSN_r</td>
<td>607.cactusBSSN_s</td>
<td>C++, C, Fortran</td>
<td>257</td>
<td>Physics: relativity</td>
</tr>
<tr>
<td>508.namd_r</td>
<td></td>
<td>C++</td>
<td>8</td>
<td>Molecular dynamics</td>
</tr>
<tr>
<td>510.parest_r</td>
<td></td>
<td>C++</td>
<td>427</td>
<td>Biomedical imaging: optical tomography with finite elements</td>
</tr>
<tr>
<td>511.povray_r</td>
<td></td>
<td>C++, C</td>
<td>170</td>
<td>Ray tracing</td>
</tr>
<tr>
<td>519.lbm_r</td>
<td>619.lbm_s</td>
<td>C</td>
<td>1</td>
<td>Fluid dynamics</td>
</tr>
<tr>
<td>521.wrf_r</td>
<td>621.wrf_s</td>
<td>Fortran, C</td>
<td>991</td>
<td>Weather forecasting</td>
</tr>
<tr>
<td>526.blender_r</td>
<td></td>
<td>C++, C</td>
<td>1,577</td>
<td>3D rendering and animation</td>
</tr>
<tr>
<td>527.cam4_r</td>
<td>627.cam4_s</td>
<td>Fortran, C</td>
<td>467</td>
<td>Atmosphere modeling</td>
</tr>
<tr>
<td>538.imagick_r</td>
<td>638.imagick_s</td>
<td>Fortran, C</td>
<td>338</td>
<td>Wide-scale ocean modeling (climate level)</td>
</tr>
<tr>
<td>544.nab_r</td>
<td>644.nab_s</td>
<td>C</td>
<td>24</td>
<td>Molecular dynamics</td>
</tr>
<tr>
<td>549.fotonik3d_r</td>
<td>649.fotonik3d_s</td>
<td>Fortran</td>
<td>14</td>
<td>Computational Electromagnetics</td>
</tr>
<tr>
<td>554.roms_r</td>
<td>654.roms_s</td>
<td>Fortran</td>
<td>210</td>
<td>Regional ocean modeling</td>
</tr>
</tbody>
</table>

[1] For multi-language benchmarks, the first one listed determines library and link options ([details](https://www.spec.org/cpu2017/results/cpu2017.html))

[2] KLOC = line count (including comments/whitespace) for source files used in a build / 1000

See [https://www.spec.org/cpu2017/results/cpu2017.html](https://www.spec.org/cpu2017/results/cpu2017.html) for results
<table>
<thead>
<tr>
<th>RANK</th>
<th>SITE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>National Super Computer Center in Guangzhou, China</td>
</tr>
<tr>
<td>2</td>
<td>DOE/SC/Oak Ridge National Laboratory, United States</td>
</tr>
<tr>
<td>3</td>
<td>DOE/NNSA/LLNL, United States</td>
</tr>
<tr>
<td>4</td>
<td>RIKEN Advanced Institute for Computational Science (AICS), Japan</td>
</tr>
<tr>
<td>5</td>
<td>DOE/SC/Argonne National Laboratory, United States</td>
</tr>
<tr>
<td>6</td>
<td>Swiss National Supercomputing Centre (CSCS), Switzerland</td>
</tr>
<tr>
<td>7</td>
<td>Texas Advanced Computing Center/U of Texas, United States</td>
</tr>
<tr>
<td>8</td>
<td>Forschungszentrum Juelich (FZJ), Germany</td>
</tr>
<tr>
<td>9</td>
<td>DOE/NNSA/LLNL, United States</td>
</tr>
<tr>
<td>10</td>
<td>Government, United States</td>
</tr>
</tbody>
</table>

**November 2014**

<table>
<thead>
<tr>
<th>RANK</th>
<th>SITE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>National Super Computer Center in Guangzhou, China</td>
</tr>
<tr>
<td>2</td>
<td>DOE/SC/Oak Ridge National Laboratory, United States</td>
</tr>
<tr>
<td>3</td>
<td>DOE/NNSA/LLNL, United States</td>
</tr>
</tbody>
</table>

**June 2015**

<table>
<thead>
<tr>
<th>RANK</th>
<th>SITE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>National Supercomputer Center in Wuxi, China</td>
</tr>
<tr>
<td>2</td>
<td>DOE/SC/Oak Ridge National Laboratory, United States</td>
</tr>
<tr>
<td>3</td>
<td>DOE/NNSA/LLNL, United States</td>
</tr>
<tr>
<td>4</td>
<td>RIKEN Advanced Institute for Computational Science (AICS), Japan</td>
</tr>
<tr>
<td>5</td>
<td>DOE/SC/Argonne National Laboratory, United States</td>
</tr>
<tr>
<td>6</td>
<td>Swiss National Supercomputing Centre (CSCS), Switzerland</td>
</tr>
<tr>
<td>7</td>
<td>King Abdullah University of Science and Technology, Saudi Arabia</td>
</tr>
<tr>
<td>8</td>
<td>Texas Advanced Computing Center/U of Texas, United States</td>
</tr>
<tr>
<td>9</td>
<td>Forschungszentrum Juelich (FZJ), Germany</td>
</tr>
<tr>
<td>10</td>
<td>DOE/NNSA/LLNL, United States</td>
</tr>
</tbody>
</table>

**August 2016**

<table>
<thead>
<tr>
<th>RANK</th>
<th>SITE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunway TaihuLight, Sunway MPP, Sunway SW26010-260C, 1.45GHz, Sunway, NRCC, National Supercomputing Center in Wuxi, China</td>
</tr>
<tr>
<td>2</td>
<td>Tianhe-2 (MILkyWay-2), TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.00GHz, TH Express-2, Intel Xeon Phi 3151P, NUDT, National Supercomputer Center in Guangzhou, China</td>
</tr>
<tr>
<td>3</td>
<td>Piz Daint, Cray XC50, Xeon E5-2609v3 12C 2.6GHz, Aries interconnect, NVIDIA Tesla P100, Cray Inc, Swiss National Supercomputing Center (CSCS), Switzerland</td>
</tr>
<tr>
<td>4</td>
<td>Titan, Cray XK7, Opteron 6274 16C 2.00GHz, Cray Gemini interconnect, NVIDIA K20c, Cray Inc, DOE/SC/Oak Ridge National Laboratory, United States</td>
</tr>
<tr>
<td>5</td>
<td>Sequoia, BlueGene/Q, Power BGQ 16C 1.60 GHz, Custom, IBM, DOE/NNSA/LLNL, United States</td>
</tr>
<tr>
<td>6</td>
<td>Gari, Cray XC40, Intel Xeon Phi 7250 68C 1.40GHz, Aries interconnect, Cray Inc, DOE/SC/LLNL/NERSC, United States</td>
</tr>
<tr>
<td>7</td>
<td>Oakforest-PACS - PRIMERGY CX4000 M1, Intel Xeon Phi 7250 68C 1.40GHz, Intel Omni-Path, Fujitsu, Joint Center for Advanced High Performance Computing, Japan</td>
</tr>
<tr>
<td>8</td>
<td>K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect, Fujitsu, RIKEN Advanced Institute for Computational Science (AICS), Japan</td>
</tr>
<tr>
<td>9</td>
<td>Mira, BlueGene/Q, Power BQC 16C 1.60GHz, Custom, IBM, 786.332, 8.568, 10.066, 3.946</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,632</td>
</tr>
<tr>
<td>12,600</td>
</tr>
<tr>
<td>13,554</td>
</tr>
<tr>
<td>13,554</td>
</tr>
<tr>
<td>15,371</td>
</tr>
<tr>
<td>17,808</td>
</tr>
<tr>
<td>2,272</td>
</tr>
<tr>
<td>3,739</td>
</tr>
<tr>
<td>2,719</td>
</tr>
<tr>
<td>2,719</td>
</tr>
</tbody>
</table>

**June 2017**

<table>
<thead>
<tr>
<th>RANK</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunway TaihuLight, Sunway MPP, Sunway SW26010-260C, 1.45GHz, Sunway, NRCC, National Supercomputing Center in Wuxi, China</td>
</tr>
<tr>
<td>2</td>
<td>Tianhe-2 (MILkyWay-2), TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.00GHz, TH Express-2, Intel Xeon Phi 3151P, NUDT, National Supercomputer Center in Guangzhou, China</td>
</tr>
<tr>
<td>3</td>
<td>Piz Daint, Cray XC50, Xeon E5-2609v3 12C 2.6GHz, Aries interconnect, NVIDIA Tesla P100, Cray Inc, Swiss National Supercomputing Center (CSCS), Switzerland</td>
</tr>
<tr>
<td>4</td>
<td>Titan, Cray XK7, Opteron 6274 16C 2.00GHz, Cray Gemini interconnect, NVIDIA K20c, Cray Inc, DOE/SC/Oak Ridge National Laboratory, United States</td>
</tr>
<tr>
<td>5</td>
<td>Sequoia, BlueGene/Q, Power BGQ 16C 1.60 GHz, Custom, IBM, DOE/NNSA/LLNL, United States</td>
</tr>
<tr>
<td>6</td>
<td>Gari, Cray XC40, Intel Xeon Phi 7250 68C 1.40GHz, Aries interconnect, Cray Inc, DOE/SC/LLNL/NERSC, United States</td>
</tr>
<tr>
<td>7</td>
<td>Oakforest-PACS - PRIMERGY CX4000 M1, Intel Xeon Phi 7250 68C 1.40GHz, Intel Omni-Path, Fujitsu, Joint Center for Advanced High Performance Computing, Japan</td>
</tr>
<tr>
<td>8</td>
<td>K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect, Fujitsu, RIKEN Advanced Institute for Computational Science (AICS), Japan</td>
</tr>
<tr>
<td>9</td>
<td>Mira, BlueGene/Q, Power BQC 16C 1.60GHz, Custom, IBM, 786.332, 8.568, 10.066, 3.946</td>
</tr>
</tbody>
</table>
Model theoretic approaches

Analytical methods
- deterministic queuing models (parameter: fixed values)
- stochastic queuing models (parameter: average values)
- operational queuing models (parameter: measured values)

Software simulations
- deterministic simulation
- stochastic simulation
- simulation based on traces
THE LAYERED COMPUTER MODEL
A Six-level Computer

Level 5: Problem-oriented language level
- Translation (Compiler)

Level 4: Assembly language level
- Translation (Assembler)

Level 3: Operating system machine level
- Partial interpretation (operating system)

Level 2: ISA (Instruction Set Architecture) level
- Interpretation (microprogram) or direct execution

Level 1: Microarchitecture level
- Hardware

Level 0: Digital logic level

A Six-level Computer – Examples

- **Level 0**: Digital logic level
  - Hardware

- **Level 1**: Microarchitecture level
  - Interpretation (microprogram) or direct execution

- **Level 2**: ISA (Instruction Set Architecture) level
  - Partial interpretation (operating system)

- **Level 3**: Operating system machine level
  - Translation (Assembler)

- **Level 4**: Assembly language level
  - Translation (Assembler)

- **Level 5**: Problem-oriented language level
  - Translation (Compiler)

---

**Languages and Compilers**

- **Java, C#, C++, C, Haskell, Cobol, Swift...**
  - javac, gcc, .NET

- **Java Byte Code, MSIL/CIL**
  - JVM, CLR; JIT/Interpreter

- **Unix, Windows, iOS**
  - JVM, CLR; JIT/Interpreter

- **x64, x86, PPC, ARM, ...**
  - microprogram/none

- **Netburst, ISSE, ASX, <none>, ...**
  - hardware

- **Core i9-7980XE, ARM9, Ryzen, ...**

---

**Microarchitectures**

- Core i9-7980XE, ARM9, Ryzen, ...
Summary

Classical computer architecture
- von Neumann
- Harvard

Universal & special purpose computers

μComputer

Milestones of computer development

A definition of computer architecture

Classification of computers

Performance assessment

Layered computer model