# COMPUTER ARCHITECTURE

## 7 Measuring CPU performance

# 7 Measuring CPU performance - objectives:

- Understanding basic concepts:
  - Relative frequency of instructions
  - Instruction execution time
  - CPI, MIPS
  - CPUtime of program
- Understanding performance measurements:
  - CPU time
  - Synthetic benchmark programs
  - SPEC

# 7 Measuring CPU performance

## Measuring CPU performance

- Computer performance is traditionally measured primarily by the performance of the CPU.
- Memory and I / O device are usually ignored.
- Memory and I / O is supposed to be powerful enough not to cause the CPU to wait.
- In such a case, the performance of the computer can be equated with the CPU performance.

## The frequency of various types of instructions

By measuring (counting), we can determine how often (percentage of the total time of the program) a program executes each group of instructions.

- **Static frequency** - the number of various types of instructions in the program.
- **Dynamic frequency** - the number of various types of instructions during the program execution.
  - Case: the program has 20 instructions, 6 of which are in a loop which is executed 10 times; there are 20 instructions in total, however, the number of instructions executed in the program is 74.
  - dynamic frequency of instructions of the type *i* is denoted by $p_i$ and expressed as a percentage of the total program execution time.

## Frequency of instructions

- An example of dynamic frequencies of certain types of instructions for a given machine and two different programs P1 and P2:

| i | Instruction types | $p_i$ % for program P1 | $p_i$ % for program P2 |
|---|---|---|---|
| 1 | Load / Store | 40% | 25% |
| 2 | ALU | 45% | 35% |
| 3 | Control instructions | 10% | 10% |
| 4 | I/O instructions | 5% | 8% |
| 5 | FP instructions | 0 | 22% |
|  | Total | 100% | 100% |

## Execution time of various types of instructions

- The duration of the instruction execution is normally expressed in clock periods. Therefore, the information applies to all (same type) processors with different clock frequencies.

- The duration of each type of instruction is expressed with parameter **CPI (Cycles Per Instruction**) and is equal to the number of clock periods needed to execute the instruction

- For all instructions the CPI is an integer number (instructions in the CPU are always executed in integer number of clock periods)

- If we multiply the CPI with the clock period $t_{CPU}$ we get the instruction execution time in seconds.

## Execution time of certain types of instructions

- Case: An example for instruction execution time in seconds, at various clock frequencies, if the instructions is executed in 5 clock periods $\Rightarrow$ *CPI* = 5 :

| CPI [clock periods] | $f_{CPU}$ [Hz] | $t_{CPU}$ [S] | Instruction execution time [s] |
|---|---|---|---|
| 5 | 100 MHz | 10 ns | 50 ns |
| 5 | 800 MHz | 1.25 ns | 6.25 ns |
| 5 | 2 GHz | 0.5 ns | 2.5 ns |
| 5 | 3 GHz | 0.33 ns | 1.65 ns |

1 MHz = $10^6$ Hz 1 ns = $10^{-9}$ s 1 GHz = $10^9$ Hz

1 Hz = 1/s = one period per second

## Measuring CPU performance

For some program we can from following data:

- dynamic frequency of types of instructions *$p_i$*
- and the number of clock periods *$CPI_i$* needed to execute certain type of instructions

calculate the average number of clock periods for execution of one instruction.

- **Average number** of clock periods *CPI,* which are necessary for the execution of one instruction in a specific program with *n* different types of instructions is expressed as follows:

$$CPI = \sum_{i=1}^{n} CPI_i \cdot p_i$$

- Case: Calculation of *CPI* (average number of clock periods to execute one instruction) for the given frequencies of the instructions in the program P2 (from the table below):

| i | Instruction type | $p_i$ % Program P1 | $p_i$ % Program P2 | $CPI_i$ |
|---|---|---|---|---|
| 1 | Load / Store | 40% | 25% | 6 |
| 2 | ALE | 45% | 35% | 4 |
| 3 | Control instructions | 10% | 10% | 3 |
| 4 | I/O instructions | 5% | 8% | 7 |
| 5 | FP instructions | 0 | 22% | 8 |

$$CPI(P2) = \sum_{i=1}^{5} p_i \cdot CPI_i = 0{,}25 \cdot 6 + 0{,}35 \cdot 4 + 0{,}10 \cdot 3 + 0{,}08 \cdot 7 + 0{,}22 \cdot 8 = 5{,}52$$

- Example calculation of *CPI* (average number of clock periods to execute the instruction) for the frequency of the instructions in the program P1 (from the table below):

| i | Instruction type | $p_i$ % Program P1 | $p_i$ % Program P2 | $CPI_i$ |
|---|---|---|---|---|
| 1 | Load / Store | 40% | 25% | 6 |
| 2 | ALE | 45% | 35% | 4 |
| 3 | Control instructions | 10% | 10% | 3 |
| 4 | I / O instructions | 5% | 8% | 7 |
| 5 | FP instructions | 0 | 22% | 8 |

$$CPI(P2) = \sum_{i=1}^{5} p_i \cdot CPI_i = 0{,}25 \cdot 6 + 0{,}35 \cdot 4 + 0{,}10 \cdot 3 + 0{,}08 \cdot 7 + 0{,}22 \cdot 8 = 5{,}52$$

$CPI(P1) = ?$ $CPI(P1) = 4{,}85$

- If we know the frequency of the clock $f_{CPU}$ we can for a specific program calculate how many instructions on average the processor executes in a seconds.
- Because the result is usually in the order of a few millions (instructions/sec), we usually divide it by $10^6$ and obtain a result in millions of instructions executed by the processor in a second.
- ***MIPS*** **(Million Instructions Per Second)**

$$MIPS = \frac{f_{CPE}}{CPI \cdot 10^6}$$

$$t_{CPE} = \frac{1}{f_{CPE}} \Rightarrow MIPS = \frac{1}{CPI \cdot t_{CPE} \cdot 10^6}$$

- Case: let‘s assume:
  - the processor clock frequency $f_{CPU}$ = 1.8 GHz,
  - the average *CPI* = 5.52 clock periods (execution of the program P2)
  - then we can calculate *MIPS*:

$$MIPS(P2) = \frac{1{,}8 \cdot 10^9}{5{,}52 \cdot 10^6} = \frac{1800}{5{,}52} = 326{,}08$$

- For the same procesor we can calculate *MIPS* for the program P1:

$$MIPS(P1) = \frac{1{,}8 \cdot 10^9}{4{,}85 \cdot 10^6} = \frac{1800}{4{,}85} = 371{,}13$$

## Measuring CPU performance

- If we assume that the memory and I/O are powerful enough not to cause the CPU to wait, then the computer performance can be equated to the CPU performance.

- The only true measure of CPU performance is **program execution time** measured in seconds/program.

- If we neglect the time for I/O transfers then we can equate program execution time with the time needed by the CPU for the program.

## Measuring CPU performance

$$CPE_{čas} = number\_of\_instructions \cdot CPI \cdot t_{CPE}$$

- CPU performance depends on:
  - ***number of instructions*** into which a particular program is compiled
  - ***CPI*** - the average number of clock periods for instruction execution
  - ***$t_{CPU}$*** - clock periods or clock frequency
- These three properties are dependent and affect one another. What affects them?

## Measuring CPU performance

- Clock period $t_{CPE}$ or clock frequency is affected by:
  - The speed and number of digital circuits, with which compose the CPU
  - Structure of the control and data unit
- *CPI* - the average number of clock periods for instruction executions affects:
  - Structure of the control and data unit
  - Types and number of instructions of the processor
  - The program

## Measuring CPU performance

- The number of instructions into which a particular program is compiled is affected by:
  - Types and number of instructions of the processor
  - Compiler properties

- When comparing the performance of computers it is therefore necessary to compare all three parameters (or their product - $CPU_{time}$), not just individual parameters.

## Measuring CPU performance

- Despite this finding, the time is not the most useful parameter in practice:
    - It depends on the program
    - For the same program it also depends on the input data
- However, all other methods, which should be independent of the program and should serve better to compare different computers, proved to be worse or even misleading.

•Case: For a particular program compiled and run on two different computers R1 and R2 we measure the following:

| Measurement | Computer R1 | Computer R2 |
|---|---|---|
| Number of machine instructions | $10 \times 10^9$ | $8 \times 10^9$ |
| Clock frequency | 4 GHz | 4 GHz |
| CPI | 1.0 | 1.1 |

•Which computer has the higher MIPS?
•Which computer is faster on the execution of this program?

## Measuring CPU performance

- **Clock frequency $f_{CPU}$** is inadequate measure to compare speed (performance) of different computers.

- **MIPS** is also not the best for comparing the performance of different computers:
  - It depends on the number and type of instructions, and is therefore not appropriate to compare computers with different instruction sets
  - It depends on the program (the type of instructions that are executed in the program)
  - A bigger MIPS does not always mean a more powerful computer (e.g. comparison of computer with a software implementation of FP instructions and a computer with a FP unit or comparison of RISC and CISC computers)

- MIPS – less serious interpretation „Meaningless indication of Processor Speed“

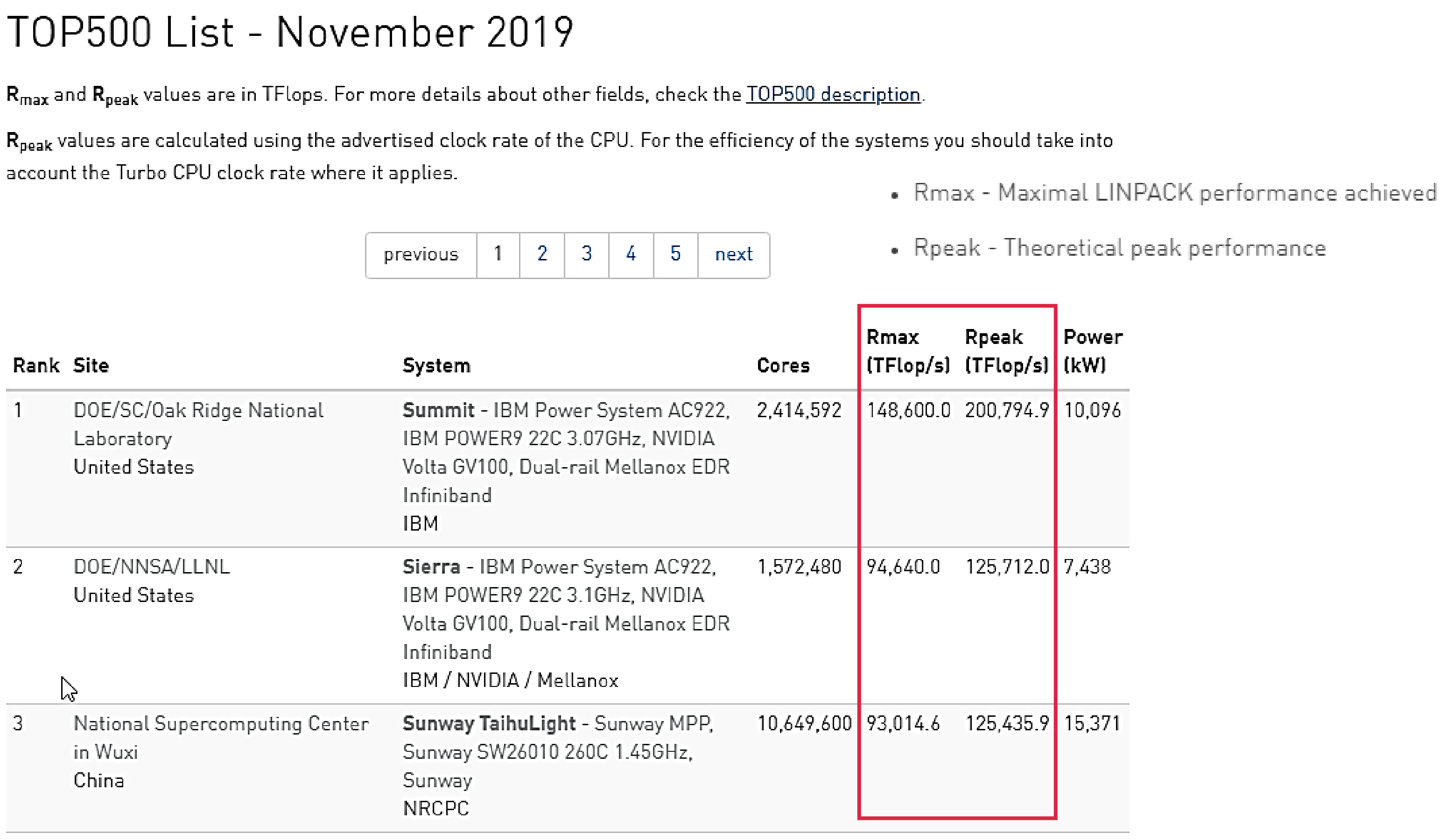
- To overcome these shortcomings, we started to **use relative MIPS,** where the program execution time is compared with the time measured on the reference computer.
  - Problems with selection of the reference computer
  - Required measurement of time, which is not necessary for MIPS

- **M(GT)FLOPS** (.. Floating Point Operations Per Second)
  - Also depends on the program
  - Makes sense only for processors which have a floating-point operations
  - Real performance in MFLOPS is often only 10% of the theoretical performance declared by manufacturers

Improvement: Use of specially written programs for performance measurement - **synthetic benchmarks.**

## Measuring CPU performance

- **M(GT)FLOPS** (.. Floating Point Operations Per Second) – Case:



TOP500 List - November 2019

**R**$_{max}$ and **R**$_{peak}$ values are in TFlops. For more details about other fields, check the TOP500 description.

**R**$_{peak}$ values are calculated using the advertised clock rate of the CPU. For the efficiency of the systems you should take into account the Turbo CPU clock rate where it applies.

- Rmax - Maximal LINPACK performance achieved
- Rpeak - Theoretical peak performance

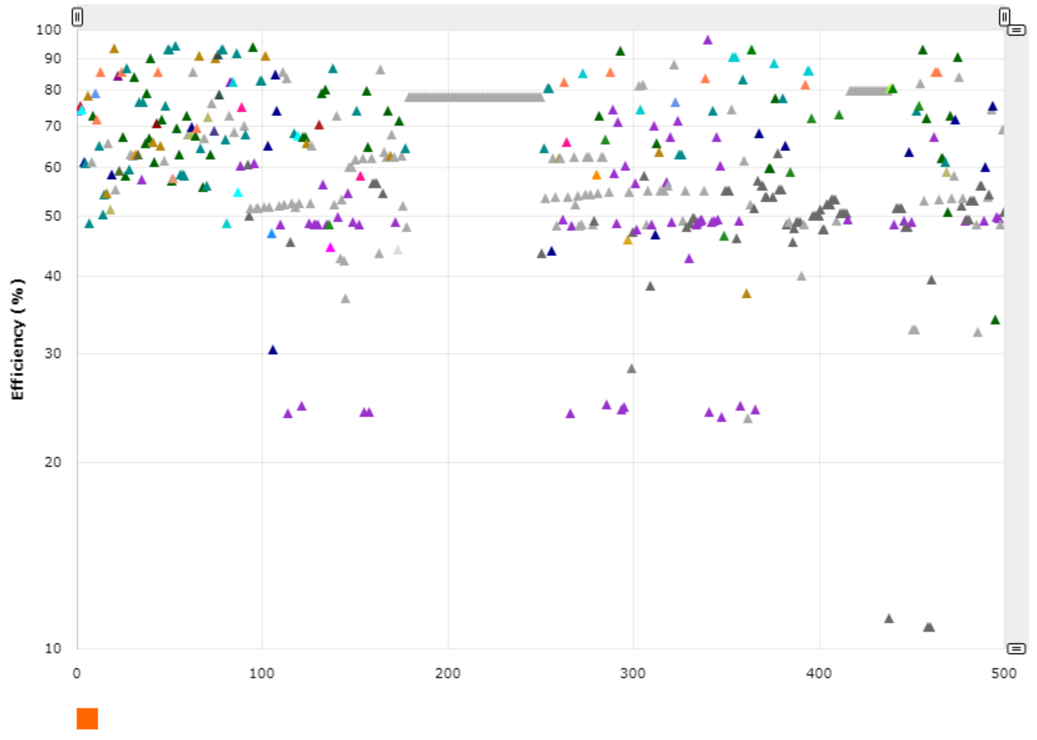
| Rank | Site | System | Cores | Rmax (TFlop/s) | Rpeak (TFlop/s) | Power (kW) |
|---|---|---|---|---|---|---|
| 1 | DOE/SC/Oak Ridge National Laboratory<br>United States | **Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband<br>IBM | 2,414,592 | 148,600.0 | 200,794.9 | 10,096 |
| 2 | DOE/NNSA/LLNL<br>United States | **Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband<br>IBM / NVIDIA / Mellanox | 1,572,480 | 94,640.0 | 125,712.0 | 7,438 |
| 3 | National Supercomputing Center in Wuxi<br>China | **Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway<br>NRCPC | 10,649,600 | 93,014.6 | 125,435.9 | 15,371 |

https://www.top500.org/list/2019/11/

- Rmax - Maximal LINPACK performance achieved
- Rpeak - Theoretical peak performance

## Measuring CPU performance

- **M(GT)FLOPS** (Floating Point Operations Per Second)
  - Real performance $R_{max}$ v M(GT)FLOPS is often lower (sometimes even under 10%) of theoretical peak performance $R_{peak}$, specified by manufacturers



Legend:

IBM, Cray/HPE, Nvidia, NEC/HPE, Cray Inc./Hitachi, Huawei, T-Platforms, Dell EMC, Sugon, T-Platforms, Intel, Dell, NEC/MEGWARE, Fujitsu / Lenovo / Xenon, Atos, Cray Inc./T-Platforms, NTT Comm. / NTT PC Comm., NUDT, Dell EMC / IBM-GBS, Lenovo, NEC, Intel, IBM / NVIDIA / Mellanox, ClusterVision / Hammer, Self-made, IBM/Lenovo, Fujitsu, Penguin

https://www.top500.org/statistics/efficiency-power-cores/

- **Synthetic benchmark programs** are programs with which we want to compare the performance of computers:
  - Whetstone (Algol 60, FORTRAN, Pascal, ...)
  - Linpack (FORTRAN)
  - Livermore Loops (FORTRAN)
  - Dhrystone (does not contain the FP operations, Ada, Pascal, C)
  - Quicksort (sorting)
  - Sieve (finding primes)
  - Puzzle

## Measuring CPU performance

- Benchmark programs were known to computer manufacturers, therefore they optimized operation (compilers) only for these programs!

- The best solution today is to use a large number of benchmark programs and calculate the arithmetic or geometric mean value of the measured results.

- **SPEC:** In 1988 some companies set up a non-profit organization **SPEC** (Standard performance Evaluation Corporation).

## Measuring CPU performance

- Organization chooses a standard set of benchmark programs to measure CPU performance.
  - The first set of programs was published in 1989: SPECmark 89 (arithmetic mean of time for 10 programs in relation with the times measured on a VAX-11/780)
  - Fifth generation published in 2006: SPEC CPU2006
    - CINT2006 for integer operations in fixed-point arithmetic - 12 programs,
    - CFP2006 for floating-point operations - 17 programs
    - The reference time for each program is divided by the measured program time, the result is "SPEC ratio“ of individual program ⇒ higher score indicates better performance
    - The end result is the geometric mean of individual performances

## 2017: the latest, the sixth generation:

### Benchmark Description:

The **SPEC CPU® 2017** benchmark package contains SPEC's next-generation, industry-standardized, CPU intensive suites for measuring and comparing compute intensive performance, stressing a system's processor, memory subsystem and compiler.

SPEC designed these suites to provide a comparative measure of compute-intensive performance across the widest practical range of hardware using workloads developed from real user applications. The benchmarks are provided as source code and require the use of compiler commands as well as other commands via a shell or command prompt window.

http: //www.intel.com/performance

https://www.spec.org/cpu2017/

Search across all the SPEC CPU® 2017 benchmark results
in SPEC's online result database.

## Measuring CPU performance

- These sets are no longer a synthetic programs (designed specifically for measuring the performance), but real applications that are updated and renewed every few years.

- These programs allow evaluation of CPU performance, memory architecture and compiler.

http://www.spec.org

## SPEC2017 – practical aspects :

**Q16. What are "base" and "peak" metrics?**

- The base metrics (such as SPECspeed2017_int_base) require that all modules of a given language in a suite must be compiled using the same flags, in the same order.

- The optional peak metrics (such as SPECspeed2017_int_peak) allow greater flexibility. Different compiler options may be used for each benchmark, and feedback-directed optimization is allowed.

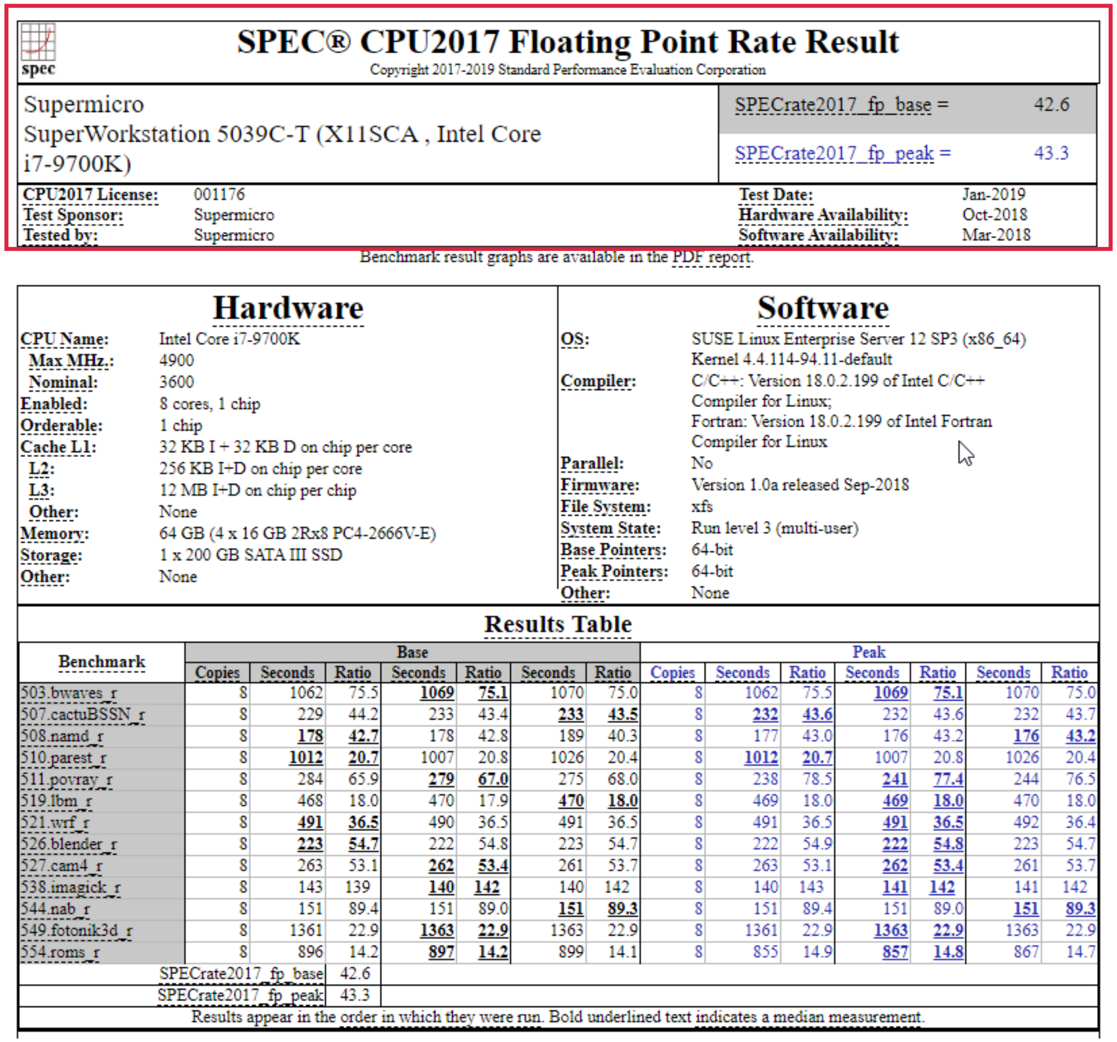
**Q17. Which SPEC CPU 2017 metric should I use?**

Examples:

- A single user running a variety of generic desktop programs may, perhaps, be interested in
  - SPECspeed2017_int_base.
- A group of scientists running customized modeling programs may, perhaps, be interested in
  - SPECrate2017_fp_peak.

Z naslova <https://www.spec.org/cpu2017/Docs/overview.html#Q16>

## SPEC2017 – Example of searched database entry :



**SPEC® CPU2017 Floating Point Rate Result**

Copyright 2017-2019 Standard Performance Evaluation Corporation

| | |
|---|---|
| Supermicro SuperWorkstation 5039C-T (X11SCA , Intel Core i7-9700K) | SPECrate2017_fp_base = 42.6<br>SPECrate2017_fp_peak = 43.3 |

| | | | |
|---|---|---|---|
| CPU2017 License: | 001176 | Test Date: | Jan-2019 |
| Test Sponsor: | Supermicro | Hardware Availability: | Oct-2018 |
| Tested by: | Supermicro | Software Availability: | Mar-2018 |

Benchmark result graphs are available in the PDF report.

**Hardware**

| | |
|---|---|
| CPU Name: | Intel Core i7-9700K |
| Max MHz.: | 4900 |
| Nominal: | 3600 |
| Enabled: | 8 cores, 1 chip |
| Orderable: | 1 chip |
| Cache L1: | 32 KB I + 32 KB D on chip per core |
| L2: | 256 KB I+D on chip per core |
| L3: | 12 MB I+D on chip per chip |
| Other: | None |
| Memory: | 64 GB (4 x 16 GB 2Rx8 PC4-2666V-E) |
| Storage: | 1 x 200 GB SATA III SSD |
| Other: | None |

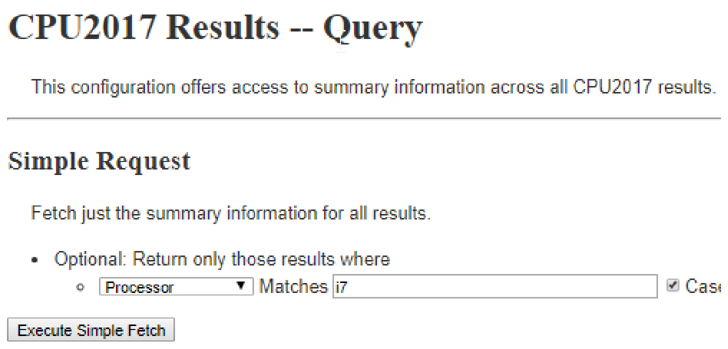
**Software**

| | |
|---|---|
| OS: | SUSE Linux Enterprise Server 12 SP3 (x86_64)<br>Kernel 4.4.114-94.11-default |
| Compiler: | C/C++: Version 18.0.2.199 of Intel C/C++ Compiler for Linux;<br>Fortran: Version 18.0.2.199 of Intel Fortran Compiler for Linux |
| Parallel: | No |
| Firmware: | Version 1.0a released Sep-2018 |
| File System: | xfs |
| System State: | Run level 3 (multi-user) |
| Base Pointers: | 64-bit |
| Peak Pointers: | 64-bit |
| Other: | None |

**Results Table**

| Benchmark | Base | | | | | | | Peak | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Copies | Seconds | Ratio | Seconds | Ratio | Seconds | Ratio | Copies | Seconds | Ratio | Seconds | Ratio | Seconds | Ratio |
| 503.bwaves_r | 8 | 1062 | 75.5 | **1069** | **75.1** | 1070 | 75.0 | 8 | 1062 | 75.5 | **1069** | **75.1** | 1070 | 75.0 |
| 507.cactuBSSN_r | 8 | 229 | 44.2 | 233 | 43.4 | **233** | **43.5** | 8 | **232** | **43.6** | 232 | 43.6 | 232 | 43.7 |
| 508.namd_r | 8 | **178** | **42.7** | 178 | 42.8 | 189 | 40.3 | 8 | 177 | 43.0 | 176 | 43.2 | **176** | **43.2** |
| 510.parest_r | 8 | **1012** | **20.7** | 1007 | 20.8 | 1026 | 20.4 | 8 | **1012** | **20.7** | 1007 | 20.8 | 1026 | 20.4 |
| 511.povray_r | 8 | 284 | 65.9 | **279** | **67.0** | 275 | 68.0 | 8 | 238 | 78.5 | **241** | **77.4** | 244 | 76.5 |
| 519.lbm_r | 8 | 468 | 18.0 | 470 | 17.9 | **470** | **18.0** | 8 | 469 | 18.0 | **469** | **18.0** | 470 | 18.0 |
| 521.wrf_r | 8 | **491** | **36.5** | 490 | 36.5 | 491 | 36.5 | 8 | 491 | 36.5 | **491** | **36.5** | 492 | 36.4 |
| 526.blender_r | 8 | **223** | **54.7** | 222 | 54.8 | 223 | 54.7 | 8 | 222 | 54.9 | **222** | **54.8** | 223 | 54.7 |
| 527.cam4_r | 8 | 263 | 53.1 | **262** | **53.4** | 261 | 53.7 | 8 | 263 | 53.1 | **262** | **53.4** | 261 | 53.7 |
| 538.imagick_r | 8 | 143 | 139 | **140** | **142** | 140 | 142 | 8 | 140 | 143 | **141** | **142** | 141 | 142 |
| 544.nab_r | 8 | 151 | 89.4 | 151 | 89.0 | **151** | **89.3** | 8 | 151 | 89.4 | 151 | 89.0 | **151** | **89.3** |
| 549.fotonik3d_r | 8 | 1361 | 22.9 | **1363** | **22.9** | 1363 | 22.9 | 8 | 1361 | 22.9 | **1363** | **22.9** | 1363 | 22.9 |
| 554.roms_r | 8 | 896 | 14.2 | **897** | **14.2** | 899 | 14.1 | 8 | 855 | 14.9 | **857** | **14.8** | 867 | 14.7 |
| SPECrate2017_fp_base | | | 42.6 | | | | | | | | | | | |
| SPECrate2017_fp_peak | | | 43.3 | | | | | | | | | | | |

Results appear in the order in which they were run. Bold underlined text indicates a median measurement.



**CPU2017 Results -- Query**

This configuration offers access to summary information across all CPU2017 results.

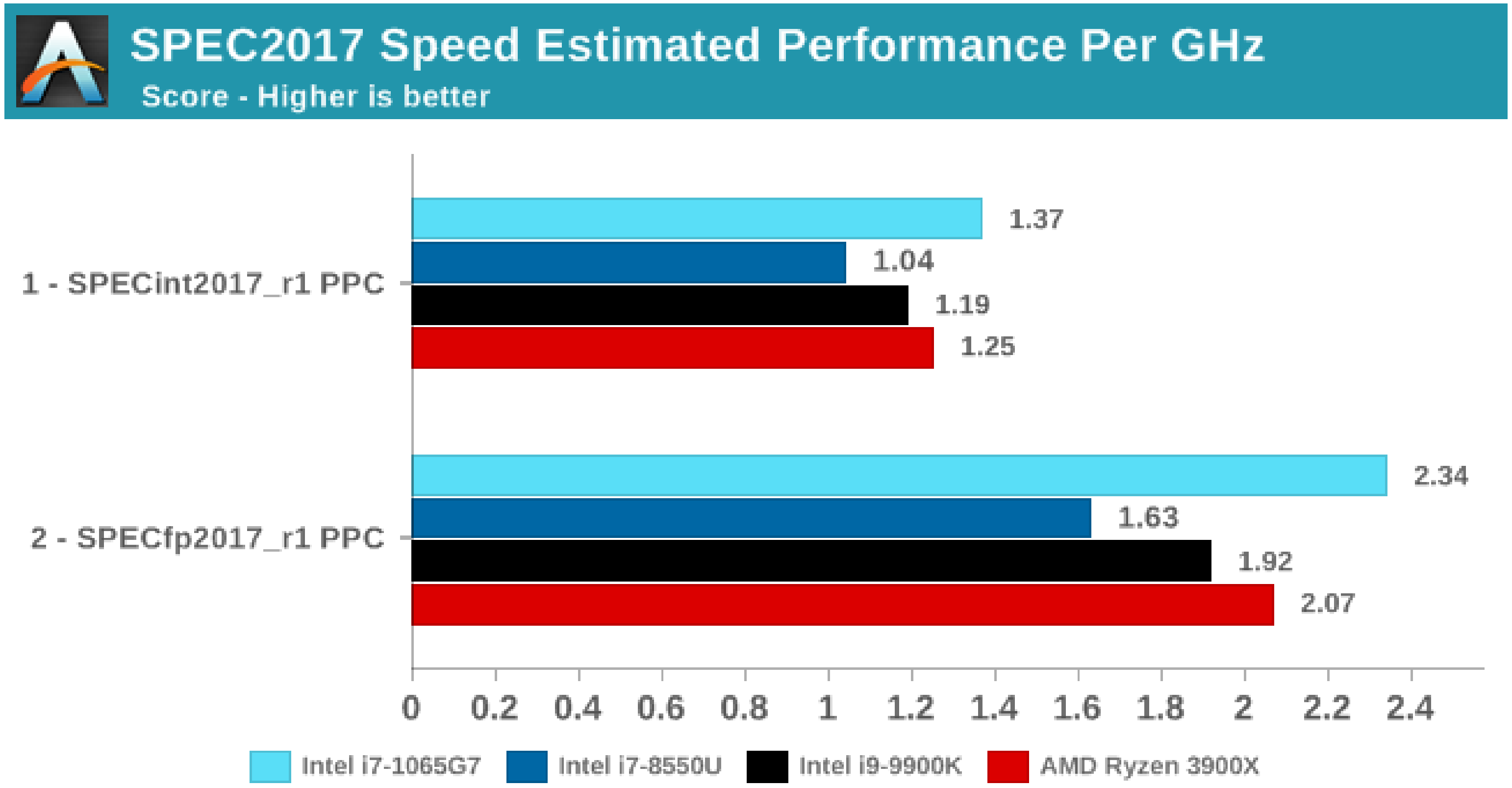
**Simple Request**

Fetch just the summary information for all results.

- Optional: Return only those results where
  - Processor Matches i7 ☑ Case

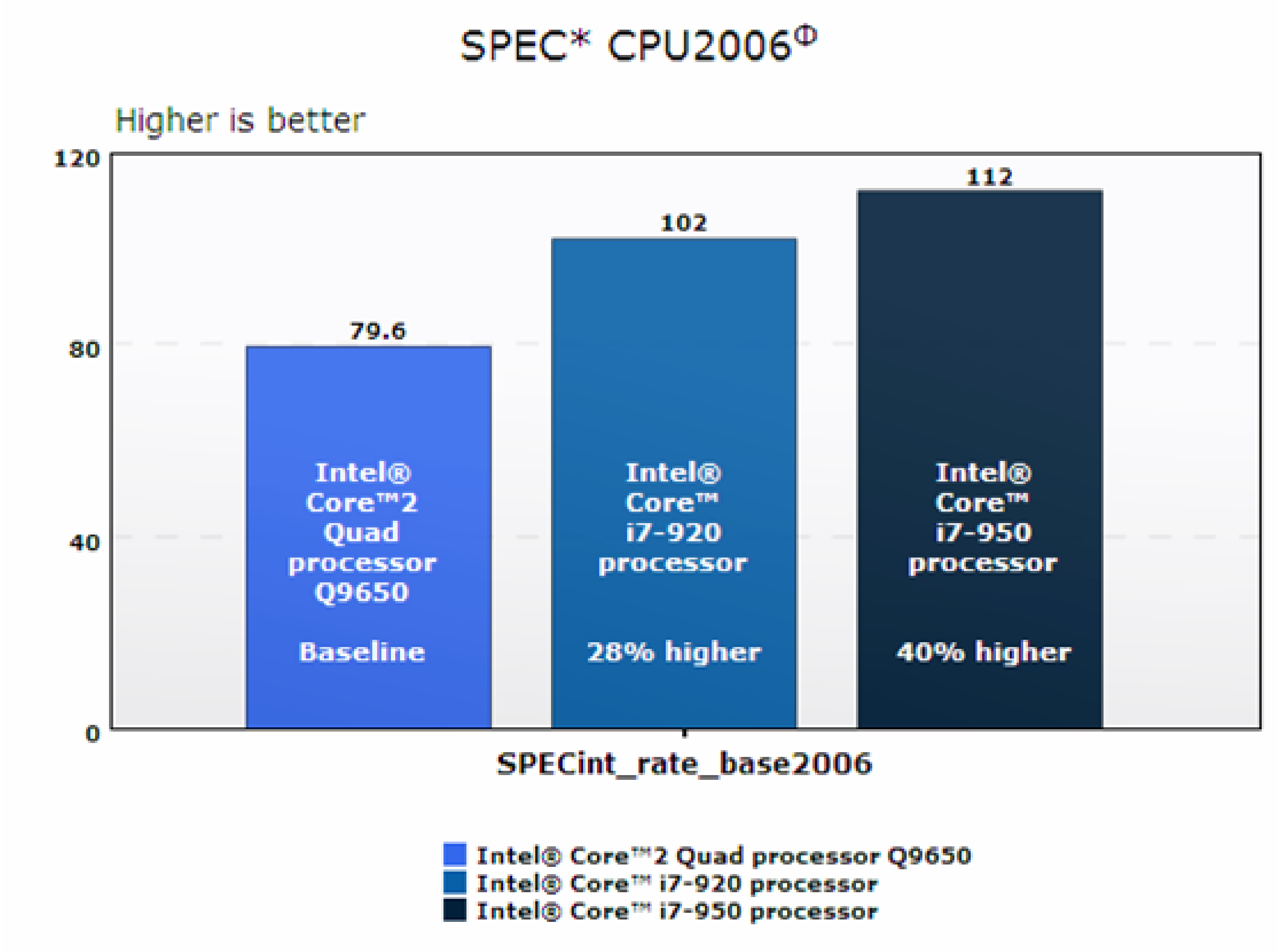
Z naslova <https://www.spec.org/cgi-bin/osgresults?conf=cpu2017>

## Case of practical usage of SPEC2017 results:



https://www.anandtech.com/show/14664/testing-intel-ice-lake-10nm/4

## Case: Comparison of Intel processors performance



- SPEC organization publishes benchmark programs and the results of the tests in addition to CPU also for:
  - Graphics and Workstation
  - Java clients and servers
  - High performance computing (OpenMP, MPI)
  - Mail servers
  - Network file systems
  - Measurement of consumption (for large number of servers)
  - SOA (Service Oriented Architectures)
  - SIP servers (Session Initiation Protocol)
  - Virtualization
  - Web servers