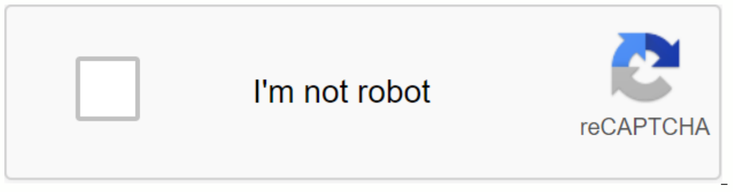
Instructions per cycle microprocessor



Continue

This article requires additional citations for verification. Help improve this article by adding quotes to reliable sources. Uns out-of-commissioned materials may be challenged and removed. Find sources: Cycles on instruction – news · newspapers · books · scholar · JSTOR (December 2009) (Learn how and when to delete this template message) In a computer architecture, cycles on instructions (aka clock cycles per statement, clocks on instructions, or CPI) is one aspect of processor performance: the average number of clock cycles per instruction for a program or program fragment. [1] This is a multiplication of the inverse of statements per cycle. The definition of Average cycles per statement in a given process is defined by the following: C P I = Σ i (I C i) (C C i) I C {\displaystyle CPI={\frac {\Sigma _{i}(IC_{i})(CC_{i})}{IC}}} Where I C and {\displaystyle IC_ {i}} is the number of statements for this type of statement and {\displaystyle i} , C and {\displaystyle CC_{i}} are the clock cycles for this type of statement, and I C = Σ and (I C i) {\displaystyle IC=\Sigma _{i}(IC_{i})} is the total number of statements. Summation sums all types of statements for a given benchmarking process. Explanation Suppose you have a classic RISC pipeline, with the following five steps: Statement Download Cycle (IF). Instructions for decoding/recording the download cycle (ID). Execution/effective address cycle (EX). Memory Access (MEM). Write-off cycle (WB). Each stage requires one clock cycle, and the statement goes through the steps sequentially. Without pipelining, in a multi-speed processor, the new statement is retrieved in step 1 only after the previous statement is completed at step 5, so the number of clock cycles needed to execute the instruction is five (CPI = 5 > 1). In this case, the processor is said to be subscalar. With pipelining, a new statement is retrieved every clock cycle, utilizing parallelism at the instruction level, therefore, because you can theoretically have five statements in five pipeline stages at a time (one statement per stage), another statement will complete stage 5 in each clock cycle, and the average number of clock cycles needed to execute the statement is 1 (CPI=1). In this case, the processor is said to be scalable. For a processor with one unit, the best achievable CPI is 1. However, for a processor with multiple execution units, even better CPI (CPI < 1) can be achieved. In this case, the processor is said to be a superscalar. To get better CPI values without a pipeline, the number of runtimes must be greater than the number of stages. For example, with six execution units, six new statements are retrieved in step 1 only after the six previous statements complete at stage 5, therefore the average number of clock cycles needed to execute the statement is 5/6 (CPI = 5/6 < 1). To better CPI values with pipelining, there must be at least two execution units. For example, with two execution units, two new statements are retrieved every clock cycle, utilizing parallelism at the instruction level, therefore two different statements will complete stage 5 in each clock cycle, and the average number of clock cycles needed to execute the statement is 1/2 (CPI = 1/2 < 1). Example 1 For multicycle MIPS there are five kinds of instructions: Loading (5 cycles) Storage (4 cycles) R-type (4 cycles) Branch (3 cycles) Jump (3 cycles) If the program has: 50% load statements 25% store instructions 15% instructions type R 8% branch instructions 2% jump instructions then, CPI is: CPI = 5 × 50 + 4 × 25 + 4 × 15 + 3 × 8 + 3 × 2 100 = 4.4 {\displaystyle {\text{CPI}}={\frac {5\times 50+4\times 25+4\times 15+3\times 8+8 3\times 2}{100}}=4.4} Example 2 [2] A 400 MHz processor was used to execute a comparison program with the following combination of statements and clock cycle number: TYPE Statement Number of statements Number of statements Clock Cycle Integer Arithmetic 45000 1 Data Transfer 32000 2 Floating Point 15000 2 Control Transfer 8000 2 Determination of Effective CPI , MIPS (Millions of instructions per second) speed and run time for this program. CPI = 45000 × 1 + 32000 × 2 + 15000 × 2 + 8000 × 2 100000 = 155000 100000 = 1.55 {\displaystyle {\text{CPI}}={\frac {45000\times 1+32000\times 2+15000\times 2+8000\times 2}{100000}}={\frac {155000}{100000}}=1.55} 400 MHz = 400 , 000 , 000 Hz {\displaystyle 400\,{\text{MHz}}=400,000,000\,{\text{Hz}}} since: MIPS ∝ 1 / CPI {\displaystyle {\text{MIPS}}\propto 1/{\text{CPI}}} and MIPS ∝ clock frequency {\displaystyle {\text{MIPS}}\propto {\text{clock frequency}}} Effective processor performance = MIPS = clock frequency CPI × 1 1 Million = 400 , 000 , 000 1.55 × 1000000 = 400 1.55 = 258 MIPS {\displaystyle {\text{Effective processor performance}}={\text{MIPS}}={\frac {\text{clock frequency}} {\text{CPI}}}\times {\frac {1}{\text{1 Million}}}={\frac {400,000,000}{1.55\times 1000000}}={\frac {400}{1.55}}=258\,{\text{MIPS}}} Therefore : Execution time (T) = CPI instruction counter × × clock time = CPI instruction counter frequency × = 1.55 × 100000 400 × 10000000 = 1.55 4000 = 0.0 003875 s = 0.3875 ms {\displaystyle {\text{{Time}}(T)={\text{CPI}}\times {\text{Instruction count}}\times {\text{clock time}}={\frac {{CPI}\times {\text{Number statement}}}{\text{frequency}}={\frac {1.55\times 100000}{400\times 1000000}}={\frac {1.55}{4000}}=0.0003875 \,{\text{sec}}=0.3875\,{\text{ms}}} See also Cycle per Second (Hz) Instructions per Cycle (IPC) Instructions per Second (IPS) Myth Megahertz MIPS This comparative article provides a useful introduction to measuring computer performance for readers interested in the topic. References ^ Patterson, David A.; Hennessy, John L. Computer and design: Hardware/software interface. ↑ Advanced Computer Architecture Kai Hwang, Chapter 1, Exercise Problem 1.1 Downloaded from This article requires additional citations for verification. Help improve this article by adding quotes to reliable sources. Uns out-of-commissioned materials may be challenged and removed. Find sources: Instructions per cycle – news · newspapers · books · scholar · JSTOR (February 2008) (Learn how and when to delete this template message) In a computer architecture, instructions per cycle (IPC), commonly called statements per clock, is one aspect of processor performance: the average number of instructions executed for each clock cycle. This is the multiplication of the inverse of cycles per statement. [1] Explanation IPC calculation IPC calculation is done by running a set of code, calculating the number of machine-level statements required to complete it, and then using high-performance timers to calculate the number of clock cycles required to complete it on actual hardware. The final result comes from dividing the number of statements by the number of cpu clock cycles. You can bring out the number of statements per second and floating-point operations per second for a processor by multiplying the number of statements per cycle at the clock frequency (cycles per second specified in hertz) of the processor. The number of statements per second is an approximate indicator of likely CPU performance. The number of statements executed per clock is not constant for a given processor; it depends on how the software runs interacts with the processor and even the entire computer, specifically the memory hierarchy. However, some processor features typically lead to projects that have higher than average IPC values; the presence of multiple arithmetic logic units (ALU is a subsystem of the processor that can perform basic arithmetic and logical operations) and short pipelines. When comparing different sets of statements, a simpler set of instructions can lead to a higher number of IPC than implementing a more complex set of instructions using the same chip technology; however, a more complex set of statements may be able to achieve more useful work with fewer statements. Therefore, comparing IPC data between different sets of statements (for example, x86 vs ARM) is usually irrelevant. IPC ruling factors No sources are cited in this section. Help improve this section by adding citations to reliable sources. Uns out-of-commissioned materials may be challenged and removed. (July 2017) (Learn how and when to delete this template message) The given instruction level per second can be achieved with high IPC and low clock speed (like AMD Athlon and early Intel's Core or with low IPC and high clock speed (like Intel Pentium 4 and to a lesser extent AMD Bulldozer). Both are valid processor designs, and the choice between them is often dictated by history, engineering limitations, or marketing pressure. [original research?] However, high-frequency IPC always gives you the best performance. Computer speed Useful work that can be done with any computer depends on many factors in addition to the speed of the processor. These factors include the architecture of the instruction set, the processor microarchitecture, and the organization of the computer system (such as disk storage system design and the capabilities and performance of other connected devices), operating system performance, and most importantly, the design of high-level application software used. For users and buyers of a computer system, instructions on the clock are not a particularly useful indicator of the performance of their system. For an accurate measure of performance that is relevant to them, application indicators are much more useful. Awareness of its existence is useful because it provides an easy-to-capture example of why clock speed is not the only factor that is important for computer performance. See also Instructions per Second Cycles on the instruction FLOPS Megahertz mit Benchmark (computer science) Testimonials ^ John L. Hennessy, David A. Patterson. Computer architecture: quantitative approach. 2007. Source :