San Francisco Bay University

EE488 - Computer Architecture
Homework Assignment #2

Due day: 6/16/2024

Instruction:

1. **Push the answer sheet to Github in word file**
2. **Overdue homework submission could not be accepted.**
3. **Takes academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**

---

1. Discuss how stack architecture computer works by giving examples, such as arithmetic express in reverse polish notation. And compare the pros and cons between stack-based virtual machine and register-based virtual machine (1.5~2 pages)

Stack Architecture Computer: A stack architecture computer is a type of computer architecture where most or all computations are performed on an operand stack. It's a type of data structure where items are added or removed based on the principle of "Last In, First Out".

Reverse Polish Notation (RPN): RPN is a mathematical notation where every operator follows all of its operands. It's particularly useful in stack-based computing architectures because it eliminates the need for parentheses to indicate the order of operations.

Here's an example of how an arithmetic expression would be evaluated using RPN in a stack-based architecture:

Consider the expression "5 3 4 + * 6 2 / -":

1. Push 5 onto the stack.
2. Push 3 onto the stack.
3. Push 4 onto the stack.
4. The next symbol is +. So, we pop 3 and 4 from the stack, add them to get 7, and push 7 back onto the stack.
5. The next symbol is *. So, we pop 5 and 7 from the stack, multiply them to get 35, and push 35 back onto the stack.
6. Push 6 onto the stack.
7. Push 2 onto the stack.
8. The next symbol is /. So, we pop 6 and 2 from the stack, divide 6 by 2 to get 3, and push 3 back onto the stack.
9. The next symbol is -. So, we pop 35 and 3 from the stack, subtract 3 from 35 to get 32, and push 32 back onto the stack.

At the end of the expression, we have one number left on the stack, 32, which is the result of the expression.

This is a simplified example, but it illustrates the basic principle of how a stack architecture computer works and how it can be used to evaluate arithmetic expressions in RPN. The stack makes it easy to manage intermediate results and the RPN makes the order of operations explicit without needing parentheses. This combination is what makes stack architecture computers efficient for certain types of computations.

	Pros	**Cons**
Stack-based	1. Simpler to implement: Stack-based VMs are generally simpler to implement because they only need to manage a stack. 2. Compact code: The bytecode for stack-based VMs is usually more compact because it doesn't need to specify registers for operations. 3. No need for register allocation: Since stack-based VMs don't use registers for computations, there's no need for complex register allocation algorithms.	1. Slower execution: Stack-based VMs can be slower than register-based VMs because they need to push and pop operands from the stack for every operation. 2. More memory operations: Every operation in a stack-based VM involves memory operations (push/pop), which can be slower than register operations.
Register-based	1. Faster execution: Register-based VMs can execute code faster because they can perform operations directly on registers, which is generally faster than pushing and popping from a stack. 2. More similar to hardware: Register-based VMs are more similar to actual hardware, which can make it easier to optimize the code.	1. Complexity: Register-based VMs are generally more complex to implement because they need to manage registers and handle register allocation. 2. Larger code size: The bytecode for register-based VMs can be larger because it needs to specify registers for each operation.

---

2. Processors are one of the most important components in computing systems. Its performance can have a big impact on the whole system. Discuss about processor design metrics and benchmarking tools (1.5~2 pages)

Performance holds significant value from the designer's viewpoints. A designer is often confronted with numerous design alternatives and must determine which one provides the most substantial performance enhancement, the lowest cost, and the optimal cost-to-performance balance.

The Performance Design Metric

For a system level designer, execution time and throughput are two important performance metrics.

(1) Execution time

- Execution time is the product of the number of instructions, cycles per instruction (CPI) and the clock period

(2) Throughput

- This is the number of tasks that can be processed per unit time. For example, a camera may be able to process 4 images per second.
- Throughput (bandwidth) – the total amount of work done in a given time is important to data center managers.

(3) Latency

- This is the time between the start of the task's execution and the end. For example, processing an image may take 0.25 second.

Execution time is generally the most important measure of performance. Throughput of an application is a more important metric, especially in server systems. In servers that serve the banking industry, airline industry, or other similar business, what is important is the number of transactions that could be completed in unit time. Such servers, typically called transaction processing systems use transactions per minute (tpm) as a performance metric. MIPS (Millions of Instructions Per Second) and MFLOPS (Million of Floating Point Operations Per Second) have been very popular measures of performance in the past. Both of these are very simple and straightforward to understand and hence have been used often, however, they do not contain all three components of program execution time and hence are incomplete measures of performance. There are also several low level metrics of interest to microprocessor designers, in order to help them identify performance bottlenecks and tune their designs. Cache hit ratios, branch misprediction ratios, number of off-chip memory accesses, etc are examples of such measures.

CPU Benchmark

Benchmarks used for performance evaluation of computers should be representative of applications that are run on actual systems. Contemporary computer applications include a variety of applications, and different benchmarks are appropriate for systems targeted for different purposes. There are primarily two types of benchmarks used in CPU benchmarking: synthetic benchmarks and application benchmarks.

(1) Synthetic benchmarks are specially designed programs that simulate specific workloads on a component or system. These benchmarks generate artificial workloads to stress test individual components like CPUs, allowing for a focused analysis of their performance. They often measure raw computational power and floating-point operations, providing insights into a CPU's capabilities in different scenarios.

Benchmark	Description
3DMark	A benchmark designed to assess the performance of a CPU and GPU under gaming workloads. It includes various graphical and physics tests to evaluate the system's gaming capabilities.

SuperPi	A benchmark that calculates the value of pi to a specific number of digits. It measures a CPU's single-threaded performance and its ability to perform complex mathematical calculations.
Cinebench	A benchmark that focuses on CPU and GPU performance in cinema 4D applications. It tests a CPU's ability to render complex 3D scenes and produces scores that can be used to compare CPUs.

(2) Application benchmarks involve running real-world programs on the system to assess its performance in actual tasks. These benchmarks provide a more accurate representation of a CPU's performance in real-world workloads, taking into account factors such as memory usage, caching, and multithreading efficiency. Application benchmarks are particularly useful for evaluating a CPU's performance in specific applications or industries, such as gaming or scientific computing.

Benchmark	Description
PCMark	A benchmark that measures a CPU's performance in various real-world scenarios, including web browsing, video conferencing, and document editing. It provides scores that reflect a CPU's overall performance in everyday tasks.
PassMark PerformanceTest	A benchmark that assesses a CPU's performance in different aspects, such as CPU, memory, disk, and graphics. It provides detailed scores and results for each component, allowing for a comprehensive analysis of system performance.
Blender Benchmark	A benchmark that focuses on CPU performance in Blender, a popular 3D modeling and rendering software. It measures a CPU's ability to handle complex rendering tasks and provides scores that can be used for comparison.

Reference:

[1] Lizy, K., & John. (n.d.). Performance Evaluation: Techniques, Tools and Benchmarks. https://lca.ece.utexas.edu/pubs/john_perfeval.pdf
[2] Wieclaw, M. (2023, December 15). Benchmarking processors: A historical overview. PC Site. https://pcsite.co.uk/benchmarking-processors-a-historical-overview/
[3] Martonosi, M., Brooks, D., & Bose, P. (n.d.). Modeling and Analyzing CPU Power and Performance: Metrics, Methods, and Abstractions. Retrieved June 16, 2024, from https://www.princeton.edu/~mrm/tutorial/hpca2001_tutorial.pdf
[4] Performance Metrics – Computer Architecture. (n.d.). Www.cs.umd.edu. https://www.cs.umd.edu/~meesh/411/CA-online/chapter/performance-metrics/index.html