# San Francisco Bay University

## EE488 - Computer Architecture
## Homework Assignment #2

**Due day: 2/25/2025**

## Instruction: Yunisha Basnet

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

   ⇨ Answer:

   Stack Architecture in Computers.

   A stack is a storage device in which the information or item stored last is retrieved first. It is a type of processor that uses a last-in, first-out (LIFO). A stack for instruction execution and operand storage instead of a traditional register. A computer system follows a memory stack organization and works accordingly.

   A portion of memory is assigned to a stack operation to implement the stack in the CPU. Here, the processor register is used as a Stack Pointer. The pointer of computer memory is divided into three segments, which are program instructions, data, and stack.

   The program Counter is a register that points to the address of the next instruction that is going to be executed in the

program. The address register points at the collection of data and is used during the execution phase to read an operand.
The stack pointer helps to point at the top of the stack. It is also used to push or pop the data items in or from the stack which is mostly used in embedded systems and virtual machines, such as the Java Virtual Machine and the Forth programming language.

The key example of stack-based computation is the evaluation of the arithmetic expressions using Reverse Polish Notation. Unlike the conventional infix notation, where the operators are placed between operands [e.g., 3+4], Reverse Polish Notation is a mathematical notation where the number is first, then the operation to be performed on them, eliminating the need for parentheses to indicate the order of operations. [e.g., 3 4 + ]. Which eliminates the need for parentheses and follows a straightforward execution process using a stack.
For example, consider the expression $(3 + 4) \times 5.$
In RRP, it is written as $3\ 4 + 5 \times$
Execution using a stack:

1. Push **3.**
2. Push **4.**
3. Apply **+**: Pop **3** and **4,** compute $3 + 4 = 7,$ push **7.**
4. Push **5**
5. Appy $\times$ : Pop **7** and **5,** compute $7 \times 5 = 35$
6. The result **35** remains on the stack.

## Stack-Based vs. Register-Based Virtual Machine.

A virtual machine (VM) is a computer resource that uses software instead of a physical computer to run programs and deploy apps. It can be categorized as either stack-based or

register-based, each with its own advantages and disadvantages.

Stack-Based Virtual Machine

A stack-based virtual machine is a computer simulation that uses a stack to store and process data.

Pros:

1. Stack-based VMs have a simpler instruction set, reducing the complexity of the compiler and making them easier to implement.
2. Stack-based VMs don't rely on a fixed number of registers; they are highly portable across different hardware architectures.
3. Instructions tend to be shorter because they do not need to specify register locations explicitly.

Cons:

1. More instructions are required for basic operations since operations since operands must be pushed and popped from the stack, increasing execution time.
2. The need for explicit stack management leads to a greater number of instructions compared to register-based VMs.
3. Optimized opportunities are restricted because operations are strictly dependent on the stack order.

Register-Based Virtual Machine.

A register-based virtual machine is a type of virtual machine where the primary data storage mechanism for the operands during computation is a set of virtual registers. This architecture is like real-world CPU design, where instructions operate directly on registers, leading to improved performance and efficiency.

Pros:

1. Register-based VMs reduce memory access overhead by directly using registers for computation, improving the execution speed.
2. Compilers can optimize instruction sequences more effectively since operands are directly accessed.
3. The number of instructions is lower than in stack-based VMs, as operations directly reference registers.

Cons:

1. The instruction set architecture (ISA) of register-based VMs is more complex, requiring sophisticated compiler support.
2. Different processors have varying numbers of registers, so adapting a register-based VM to different architectures can be more challenging.
3. Instructions tend to be longer due to the need for explicitly referencing registers, leading to increased memory usage.

Stack-based architectures and VMs provide simplicity and portability, making them suitable for interpreted languages like Java and Python. However, they suffer from performance overhead due to excessive memory operations. Register-based VMs, used in modern runtime environments like Android's Dalvik VM and LuaJIT, offer improved execution efficiency at the cost of increased complexity and lower portability. The choice between these architectures depends on the specific requirements of the application and system constraints.

References:

Hennessy, J. L., & Patterson, D. A. (2017). Computer Architecture: A Quantitative Approach. Morgan Kaufmann. Retrieved from https://www.sciencedirect.com/book/9780128119051/computer-architecture

Tanenbaum, A. S. (2016). Structured Computer Organization. Pearson. Retrieved from https://www.pearson.com/en-us/subject-

catalog/p/structured-computer-organization/P200000008088/9780132916523

Smith, J. E., & Nair, R. (2005). Virtual Machines: Versatile Platforms for Systems and Processes. Elsevier. Retrieved from https://www.sciencedirect.com/book/9781558609105/virtual-machines

Lindholm, T., & Yellin, F. (2014). The Java Virtual Machine Specification. Addison-Wesley. Retrieved from https://docs.oracle.com/javase/specs/jvms/se7/html/

GeeksforGeeks. (n.d.). Memory Stack Organization in Computer Architecture. Retrieved from https://www.geeksforgeeks.org/memory-stack-organization-in-computer-architecture/

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)

   ⇨ Answer:

   Processors are the soul of computing systems, and their efficiency, speed, responsiveness of applications, and power are characterized by them. Their design involves many trade-offs to realize speed, power, cost, and design complexity. It needs absolute measures and benchmarking tools for efficiency accounting in real-world applications.

   The parameter that defines the efficiency and effectiveness of a processor are:

1. Clock speed in gigahertz (GHz) specifies the number of times the processor clocks every second. The more clocking, the more performance is at the expense of increased power dissipation and heat.

2. The Instruction Per Cycle (IPC) is a metric that quantifies the number of instructions executed per clock cycle. The higher the IPC, the more instructions the processor executes in more efficient ways, and thus more benefit can be derived from higher clock speed.

3. Efficiency matters, especially for data centers and phone chips. Performance per Watt (PPW) is a measure of how much processing the processor does compared to each unit of electricity it consumes.

4. Processor cache memory (L1, L2, L3) matters to enhance performance by not reading data from slower RAM. Bigger and quicker caches result in quicker execution, particularly for applications that have plenty of data.

5. Modern processors use more than one core to enhance efficiency by using parallel processing. Multi-threaded software takes advantage of higher core counts, but single thread performance continues to depend upon clock speed and IPC.

6. Pipelining divides instruction execution into stages, enhancing throughput. Superscalar designs allow the

parallel execution of instructions, enhancing processing efficiency.

7. Thermal Design Power (TDP) is the maximum amount of heat produced by a processor under load. It is related to system design and cooling, particularly for high-end computing.

Benchmarking Tools.

Benchmarking tools are software programs that evaluate the performance of processors by running standardized tests. These tools provide measurable performance scores that help compare different processors based on workload efficiency.

1. Standard Performance Evaluation Corporation Benchmarks. It measures single-threaded and multi-threaded CPU performance in SPEC CPU. Evaluate integer computation performance in SPECint. It assesses floating-point computation performance in SPECfp.

2. Geekbench tests single-core and multi-core performance for general workloads, which provides cross-platform comparisons across Windows, macOS, Linux, and mobile devices.

3. Cinebench focuses on rendering performance using real-world 3D rendering tasks. It is also useful for evaluating multi-core performance in creative workloads.

4. PassMark PerformanceTest measures CPU speed across various tasks, including integer math, floating-point math,

and compression. It also provides a comparative ranking of processors.

5. PCMark evaluates overall system performance based on real-world applications like web browsing, document editing, and video conferencing. 3DMark Focuses on gaming and graphics-intensive applications, testing CPU and GPU collaboration.

6. Linpack and High-Performance Linpac measure floating-point performance, which is commonly used for supercomputers. They also determine peak FLOPS (floating-point operations per second) capability.

Processor design involves balancing multiple metrics to optimize speed, efficiency, and cost. Benchmarking tools are crucial for assessing real-world performance across various workloads. By analyzing these design metrics and benchmarks, users and engineers can make informed decisions about processor selection for specific applications.

References

Dongarra, J. (2020). High-performance computing and benchmarking tools. Netlib. Retrieved from https://www.netlib.org/benchmark/hpl/

Maxon. (2023). Cinebench benchmark tool. Retrieved from https://www.maxon.net/en/cinebench

PassMark Software. (2023). PerformanceTest benchmarking tool. Retrieved from https://www.passmark.com/products/performancetest/index.php

Primate Labs. (2023). Geekbench benchmark tool. Retrieved from https://www.geekbench.com/

SPEC. (2023). SPEC CPU benchmark suite. Retrieved from https://www.spec.org/

Stallings, W. (2017). Computer organization and architecture: Designing for performance (11th ed.). Pearson.

UL Benchmarks. (2023). PCMark and 3DMark benchmark tools. Retrieved from https://benchmarks.ul.com/