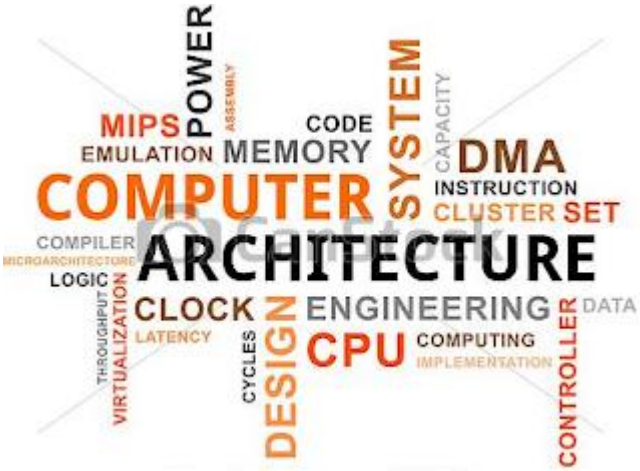
Date:6/16/24

EE488 - Computer Architecture

HW Assignment 2



NAME : S A SABBIRUL MOHOSIN NAIM

ID : 20176

ANSWER TO THE QUESTION NO 1 :

Stack Architecture Computer:

In a stack architecture computer, the operands of arithmetic operations are pushed onto a stack, and the operations are performed on the operands at the head of the stack. Typically, the most commonly known is RPN (Reverse Polish Notation) as before mentioned = postfix notation and is used in stack-based architectures.

In RPN, Operators are placed after their operands For instance, to calculate the Answer to the expression 5 + 6 * 7 in Reverse Polish Notation:-

5 6 7 * +

Here, this equation can evaluated on a stack system like,

- **Push 5 onto stack.**
- **Push 6 onto stack.**
- **Push 7 onto the stack.**
- **Multiplying top two elements on stack at the end and resulting 6*7=42.**
- **Now push 42 onto the stack.**
- **Now add 5 and 42 on the stack and final result would be 45 .**

Pros of Stack-based Virtual Machines:

Easy Implementation: In comparison to register-based architectures, stack-based structures are typically easier for developers to implement in hardware or software. This simplicity may result in less complexity and possibly lower development and maintenance expenses.

Flexibility: Since instructions usually only need to reference the top few pieces of the stack, stack-based designs are better suited to manage variable-length instructions than register-based architectures.

Cons of Stack-based Virtual Machines:

Efficiency: When it comes to execution speed, register-based architectures may be more efficient than stack-based structures. The requirement to continuously push and pop data onto and off the stack, which might impose overhead, is the source of this inefficiency.

Restricted Parallelism: Because instructions frequently rely on the outcomes of earlier instructions, stack-based architectures may not be able to provide large-scale instruction parallelism.

Register-based Virtual Machine:

In a register virtual machine, the operands are placed in named registers and the operations are performed on these registers directly. Example, evaluating the expression "a = b + c *d", in a register-based virtual machine:

```
LOAD b

LOAD c

LOAD d

MUL

ADD

STORE a
```

Although, this is illustrated in preparation for a register-based system:

- **Load b into a register**
- **Load register with value of c**
- **Place the value d in a register.**
- **Multiply the value in the register 'c' with 'd'.**
- **Add those values from the b registers to the result of the multiplication**
- **Finally, store the result in the register for 'a'.**

Pros of Register-based Virtual Machines:

Efficiency: When it comes to execution speed, register-based architectures may be more effective than stack-based structures. Operands don't need to be repeatedly pushed and popped off the stack because they are kept in named registers.

Better Support for Parallelism: Since instructions can run independently on various registers, register-based architectures frequently offer greater support for instruction-level parallelism. This is because there are more options for parallel execution.

Cons of Register-based Virtual Machines:

Complexity: **Compared to stack-based architectures, register-based systems are typically more difficult to implement in hardware or software. It can be difficult to oversee several registers and make sure they are used effectively.**

In conclusion, there are benefits and drawbacks to both register-based and stack-based virtual machines. Stack-based architectures can be more flexible and simple, but they may also be less efficient and have less parallelism. Conversely, register-based architectures have the advantage of efficiency, improved support for parallelism, and potential for optimization, but they can also be more difficult to construct and call for explicit memory management. The application's particular requirements and the trade-offs between simplicity, efficiency, and performance determine whether architecture—stack-based or register-based—to choose.

ANSWER TO THE QUESTION NO 2:

Without a doubt, processor performance is greatly influenced by them, and accurately evaluating their capabilities requires an awareness of their design criteria and evaluation techniques.

Processor Design Metrics:

Clock Speed: **The clock speed of the processor, often listed in gigahertz (GHz). This typically correlates with a quicker execution of instructions.**

Instructions Per Cycle (IPC): **IPC is an average value of executed instructions per single clock cycle. A processor that can accomplish more work in each clock cycle - usually CPUs with a higher IPC - will perform better.**

Cache Size and Hierarchy: **Cache memory is utilized for holding the most widely recognized guidelines and data for minimizing access to primary memory that is moderate contrasted with cache memory. Cache Size and Hierarchy The cache size (L1 *, L2, L3) and hierarchy has a lot to do with the speed of the processor pulling in data.**

Pipeline Depth: To increase throughput, modern processors employ pipelines to overlap the execution of instructions. The number of stages in a pipeline is referred to as pipeline depth; deeper pipelines may be able to support greater clock rates but may also result in longer latencies.

Support for Parallelism: Processors can support multiple parallelism models, such as data-level parallelism (DLP), thread-level parallelism (TLP), and instruction-level parallelism (ILP). Supporting parallel processing has the potential to improve performance dramatically.

Power Efficiency: As the focus on energy efficiency grows, CPU design must take into account power consumption measures like performance-per-watt and thermal design power (TDP).

Benchmarking Tools:

SPEC CPU BenchMark: Benchmark suites like SPEC CPU, which are composed of standardized tests to assess CPU performance across various workloads, including integer and floating-point operations, are offered by the Standard Performance Evaluation Corporation (SPEC).

Geekbench: Known for its cross-platform testing capabilities, Geekbench evaluates CPU and memory performance through a battery of simulated tests, yielding scores for both single- and multi-core performance.

Cinebench: Using real-world rendering tasks, MAXON's Cinebench assesses CPU and GPU performance. It offers information on both single- and multi-core performance, which is especially useful for workstation and multimedia applications.

AIDA64: This all-in-one program for system diagnostics and benchmarking provides thorough insights into system performance with its cache and memory latency measures, CPU stress tests, and memory benchmarks.

PCMark: UL's benchmarking suite simulates real-world operations including online browsing, video conferencing, and content production to assess overall system performance. Together with other system elements, it evaluates the

CPU's performance.

Sysbench: With benchmarks for CPU, memory, and file I/O, Sysbench is a flexible benchmarking tool. It provides Linux-based systems with command-line applications for doing performance assessments and stress tests.

In conclusion, benchmarking techniques and processor design metrics are critical for assessing and contrasting CPU performance across various workloads and architectures. Accurate evaluation of a processor's capabilities is facilitated by knowledge of critical parameters including clock speed, IPC, cache hierarchy, and power efficiency. To measure CPU performance methodically in the interim, benchmarking programs offer synthetic workloads and defined tests. Consumers, researchers, and business experts can choose and optimize processor designs for a range of uses and use cases by utilizing these measurements and tools.

THANK YOU!