**Logo

Description automatically generated San Francisco Bay University**

**EE488 - Computer Architecture**

**Homework Assignment #2**

**Due day: 2/22/2023**

Name: Melvin Divine Pritchard Student ID: 19857

**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 example. And compare the pros and cons between stack-based virtual machine and register-based virtual machine (1.5~2 pages)

Stack architecture, also known as stack-based architecture or stack machines, is a sort of computer architecture in which the majority of operations are performed on a stack. The fundamental premise of a stack computer is based on the Last In, First Out (LIFO) access strategy, which is utilized by the majority of CPUs. In this architecture, data is loaded onto a stack, and operations are executed by popping data from the stack, executing the required computation, and then pushing the result back onto the stack. The stack pointer is a register that contains the addresses of the elements at the top of the stack.

Charles Forgy created the first stack-based computer architecture in 1963 at the Massachusetts Institute of Technology (MIT). Forgy created the FORTH programming language, which was meant to run on the FORTH Engine stack computer. FORTH was utilized in numerous applications, such as control systems, real-time data gathering, and scientific computing. In the 1970s and 1980s, the development of microprocessors led to the widespread adoption of stack-based systems. The simplicity of the stack-based technique made it excellent for compact, low-power microcontrollers, and embedded systems and scientific calculators utilized it extensively. The stack design is founded on the Postfix notation, commonly referred to as Reverse Polish Notation (RPN). In this notation, operators are placed after their operands, as opposed to between them as in infix notation. This enables easier expression parsing and execution and is particularly suited to stack-based designs.

In recent years, stack-based architectures have found new applications in high-performance computing in addition to their continuous use in embedded systems and low-power devices. The GreenArrays GA144 is a famous example of a massively parallel stack-based architecture designed for low-power computing. MPE's Forth-optimized CPUs are yet another illustration. These processors are designed to execute FORTH code natively and can perform a great deal of work while consuming little energy.

Push and pop are the two essential operations of a computer with a stack architecture. The push action adds data to the stack's top, whereas the pop operation removes data from the stack's top. These two actions are fundamental in a computer with a stack design and allow for efficient data storage and retrieval. A push operation is utilized to add data to the stack's top. As an item is pushed onto a stack, it replaces the previous item at the top. The procedure to push requires two steps: Initially, the size of the data to be added is added to the memory address at the top of the stack. The information is then written to the new memory location. Pop is used to remove data from the stack's top position. When an item is popped from the stack, it is removed from the top of the stack and the top memory location is decremented. The pop procedure consists of two steps: The data at the top of the stack are read first. Second, the memory address of the stack's topmost element is decremented.

**Example of a PUSH program in python**: In the below example, the stack pointer is initially configured to refer to the top of the stack. The instruction PUSH is then used to push the value 42 onto the stack. The PUSH instruction decrements the stack pointer and stores the value on the new stack's top.

; Set up stack pointer

MOV SP, #0x1000 ; Set stack pointer to 0x1000

; Push value onto stack

PUSH #42 ; Push the value 42 onto the stack

; End of program

HALT ; Halt the computer

**Example of a pop operation.** In the below example, the pop operation results in deleting one operand from the top of the stack and increasing the stack pointer register.

//transfer the content of SP (i.e, at top most data)

//into specified memory location

(memory address) <-- SP

//increment SP by 1

SP <-- SP + 1

**Pros and Cons between stack-based virtual machine and register-based virtual machine**

Unlike a stack-based virtual machine, which operates using PUSH and POP, a register-based virtual machine stores and manipulates values using a series of registers. In other words, the data structure in which the operands are stored is based on the CPU's registers. The stack-based virtual machine employs a stack pointer to point to the operand, whereas the register-based virtual machine addresses the operands explicitly in the instruction. For instance, an addition operation in a stack-based virtual machine would take four lines of instructions to do the push and pop operation, whereas a similar program in a register-based virtual machine would only require one line. The difference between the two approaches is in the mechanism used for storing and retrieving operands and their results.

Below are some pros and cons of stack-based virtual machine and register-based virtual machine.

|  |  |
| --- | --- |
| **Pros** | |
| **Stack-based virtual machine** | **Register-based virtual machine** |
| **Flexibility:** A stack-based virtual machine can be used to implement a wide variety of programming languages and programming paradigms, including functional and object-oriented programming. | Smaller memory footprint: A register-based virtual machine typically requires less memory than a stack-based virtual machine, which can be important in resource-constrained environments. |
| **Simple and efficient bytecode**: A stack-based virtual machine can use a simple and efficient bytecode that operates directly on the stack. This means that it can be faster and more memory-efficient than a register-based virtual machine. | Faster performance: Because a register-based virtual machine operates directly on registers, it can be faster than a stack-based virtual machine in some cases. This is because it requires less memory access. |
| **Easier implementation**: Implementing a stack-based virtual machine is usually simpler than implementing a register-based virtual machine. This is because stack-based virtual machines require less bookkeeping code to keep track of registers. | Better optimization opportunities: A register-based virtual machine can be optimized more easily than a stack-based virtual machine. This can lead to faster performance and more efficient memory usage. |
| **Cons** | |
| **Reduced optimization opportunities**: Because a stack-based virtual machine uses a stack data structure, it can be more difficult to optimize code than in a register-based virtual machine. This can lead to slower performance in some cases. | **More complex bytecode**: A register-based virtual machine requires a more complex bytecode than a stack-based virtual machine. This is because the bytecode needs to explicitly specify the registers used in each operation.  **Harder to implement**: Implementing a register-based virtual machine can be more challenging than implementing a stack-based virtual machine. This is because a register-based virtual machine requires more bookkeeping code to keep track of registers. |
| **Increased memory usage**: A stack-based virtual machine typically requires more memory than a register-based virtual machine. This is because each operation on the stack requires additional memory to store the operands and results. | **Less flexible**: A register-based virtual machine may not be as flexible as a stack-based virtual machine in some cases. This is because it may be optimized for a specific programming language or paradigm, and may not be as easily adaptable to other languages or paradigms. |

1. Processor is one of most important components in computing system. Its performance can make big impact to the whole system. Discuss about processor design metrics and benchmarking tools (1.5~2 pages)

Previously, the performance of a processor was assessed primarily by its clock speed. When clock rates approached physical constraints, however, processor manufacturers shifted their focus to multi-core architectures and IPC enhancements to boost performance. In recent years, the importance of power consumption and efficiency has increased, leading to the development of low-power and mobile processors. The rise of machine learning and artificial intelligence has also spurred the creation of specialized processors, such as graphics processing units (GPUs) and tensor processing units, for these applications (TPUs).

In a computer system, the processor, also known as the Central Processor Unit (CPU), executes instructions and performs calculations. The CPU is the computer's brain, and all other components depend on it to work efficiently. Without a processor, a computer is completely inoperable. The processor is also an important aspect in affecting the power consumption and thermal management of the system. Processors with greater clock rates and more cores use more power and emit more heat, which might impair the system's overall stability and lifespan. Its performance has a considerable impact on the computer system's overall performance. A faster CPU can handle more complicated operations and process data more quickly, resulting in faster application load times, smoother multitasking, and quicker response times, so enhancing the user experience. On the other hand, a slower processor can result in excruciatingly lengthy waits for tasks to complete.

Performance implies different things to different individuals, and the criteria for evaluating performance vary across users and designers. Performance is essential from both the buyer's and designer's perspectives. In terms of purchasing, you must be able to decide which machine has the best performance, the lowest cost, and the best cost-to-performance ratio, given a collection of machines. Similarly, from the standpoint of a designer, you are presented with a variety of design possibilities, such as those with the greatest performance improvement, the lowest cost, and the best cost/performance ratio. Without you having some understanding of the performance measures, you will not be able to determine which of the potential performance improvements is the greatest, which will result in the lowest cost, and which will provide the best cost-performance ratio. Hence, regardless of who the prospective client is, both the buyer and the designer must be familiar with the performance measures.

A computer system's performance can be affected by a variety of factors, including the instructions used and how they are implemented, the memory hierarchy, and how I/O is handled. Response time, which is the amount of time between the initiation and completion of an activity, has been regarded as an essential key aspect for evaluating computer performance. Throughput, which is the overall amount of work completed in a given amount of time, is an additional crucial factor to consider. While reaction time is essential for users, throughput (bandwidth) is crucial for data center administrators. As execution time is inversely proportional to performance, if we wish to maximize performance, we must obviously reduce our execution time.

Research and technological developments will continue to result in advancements in computer architecture. Nonetheless, ongoing evaluation of existing systems is necessary. While examining current systems for bottlenecks, measure performance, develop summaries, and report performance. Knowledge of performance measurements and benchmarks is necessary in order to make sensible decisions on the computer systems you wish to acquire. Knowing performance measurements is also essential to comprehending the underlying organizational motivation; based on what motivates users, these alterations are made in an effort to increase performance. While examining the computer industry, there may be several classes of computer systems and performance criteria that are essential for certain types of applications but less so for others. One should be able to determine which characteristics are crucial for which processor type. To make the correct choice, one must comprehend the internal architecture of the CPU.

Recent revisions to processor design criteria and benchmarking instruments have centered on gauging performance in specific applications like as machine learning and artificial intelligence. To evaluate performance in these domains, standards such as MLPerf and HPCG have been developed. In addition, the importance of power consumption and efficiency has increased, resulting in the development of low-power and mobile processors. As a result, benchmarks such as SPEC Power have been created to quantify power usage and efficiency. In general, processor design criteria and benchmarking tools continue to evolve to satisfy the ever-changing demands of the computer industry.

Processor design metrics and benchmarking tools are important for measuring the performance, power consumption, and efficiency of processors.

**Processor Design Metrics:**

**Clock Speed**: Clock speed is the number of cycles per second at which a processor can execute instructions. It is measured in GHz (gigahertz) or MHz (megahertz). Historically, clock speed was the most significant metric used to measure processor performance.

**Number of Cores**: The number of cores in a processor is the number of independent processing units it contains. Multiple cores enable the processor to handle more tasks simultaneously, leading to better performance.

**Instruction Per Cycle (IPC):** IPC is the number of instructions a processor can execute per clock cycle. Higher IPC leads to better performance.

**Cache Size**: Cache is a small amount of memory that stores frequently accessed data to improve performance. The larger the cache, the better the performance.

**Power Consumption**: Power consumption is the amount of power a processor uses to operate. Lower power consumption leads to longer battery life and more environmentally friendly computing.

**Benchmarking Tools:**

**SPEC CPU**: SPEC CPU is a benchmark suite that measures the performance of CPUs using a variety of applications, including scientific computing, image processing, and database operations. It has been widely used since the 1990s and is considered one of the most reliable benchmarks for CPU performance.

**Geekbench**: Geekbench is a cross-platform benchmarking tool that measures CPU and GPU performance. It is popular among consumers and professionals for comparing the performance of different systems.

**Cinebench**: Cinebench is a benchmarking tool that measures CPU and GPU performance using 3D rendering tasks. It is widely used in the media and entertainment industry to evaluate hardware performance.