Memory Hierarchy and Cache Design

Introduction

In Computer Architecture, Memory Hierarchy is the organization of computer storage into various levels, prioritizing speed and quick access to allow for the least amount of time needed to access that piece of memory. It was made based on a computer science principle known as "*Locality of References*", which is the tendency of a computer program to access the same memory locations repeatedly over a relatively short period of time.

The importance of Memory Hierarchy lies in its optimization of the available computer memory. In the typical computer, the memory consists of various levels, with each level comprising of a different size, cost and type. The way data is accessed at each level varies greatly, and by understanding which levels can be accessed faster, or which levels hold the greatest storage, we can learn how to more efficiently use and manage the memory space.

So, the purpose of this case study will be to explore the significance of Memory Hierarchy in relation to Computer Architecture by highlighting key aspects of its function/role in the computer and providing key examples of its application and utilization within the computer. I will also discuss cache design in relation to its role in Memory Hierarchy.

Literature Review and Analysis

In the Memory Hierarchy, we have external and internal memory. External memory usually is in the form of peripheral or outer devices, such as magnetic or optical disks, that can be connected to the CPU. Internal memory consists of the ***CPU Registers, Cache Memory*** and the ***Main Memory,*** which are all directly accessible by the computer's processor.

Displayed below is the order in which each level appears in the Memory Hierarchy:

- ***CPU Registers (Level 0)***
- ***Cache Memory (Level 1)***
- ***Main Memory (Level 2)***
- ***Disk Storage/Magnetic Disk (Level 3)***
- ***Optical Disk (Level 4)***
- ***Magnetic Tape (Level 4)***

As you go further down each level, the access speed decreases while the storage capacity increases. I will now break down and explain the function of each component/level:

CPU Registers: The registers are small in storage capacity, but high in speed and are used to hold the data and instructions that are most repeatedly requested by the user/system, mainly ***unprocessed data,*** prior to or during the processing of the information. There are different types of register within the CPU (*Accumulator, Memory Address Registers, General Purpose Registers, Program Counter, Instruction Register, Stack Pointer, Flag Register, Condition Code Register*), each with a different and specific function.

Registers come in four different sizes: *8-bit, 16-bit, 32-bit* and *64-bit*, with majority of modern-day computers using the 32- and 64-bit types. Some processors hold 128-bit and 256-bit, for handling larger data and more complex processing tasks.

With the register, its main role is to provide fast access to key data in the system. It allows the computer to retrieve important information quickly, and is vital to tasks such as calculations, or holding instructions that are necessary for the CPU to load before being executed.

Cache Memory: Like the *CPU Registers*, the cache has small storage capacity but very quick speeds. The cache tends to store processed data/instructions that the current executing CPU program needs to function. It's storage and location close to the CPU allows the cache to increase the processing time and enable operations to be done quickly. I will elaborate more, later in this paper, about the design of the cache and its application.

Main Memory: This can also be referred to as the *RAM (Random Access Memory)*. It is volatile, meaning that when there is no electricity in the system all the data is lost. RAM is a key component of the CPU and has larger storage capacity than the cache, but lower speed. RAM is utilized in storing data that is in use by the computer.

When working with the Main Memory, the concept of *Memory Management* is very important. *Memory Management* is a function in the Operating System that manages processes that occur between the main memory and the disk during the execution of these processes. We use memory management to effectively and efficiently manage how the memory is used. Memory management is important because:

- It allows one to monitor the amount of memory being used by the processes.
- It enables the maintenance of data integrity when executing a process.
- It ensures that the main memory is properly utilized.
- It allows for allocation and de-allocation of memory space prior to, and after execution of the process.
- It reduces ***fragmentation,*** which is the scattering of free memory space after a process has been loaded and removed, that ultimately becomes unusable and wasted.

Other key concepts when it comes to Memory Management are ***Logical & Physical Address Spaces, Loading, Linking*** and ***Swapping,*** which I will examine briefly below.

Logical & Physical Address Spaces:

An address refers to a location in the computer's memory where data and instructions are saved. The ***Logical Address Space***, also known as the "Virtual" address, can be referred to as the memory size of a process, and is made by the CPU during the execution of a program. A logical address is used by processes to access memory, and this is interpreted by the operating system into the ***Physical Address.*** The Logical Address will be used by the computer as a reference to the actual location in memory, and its location within memory can be changed.

In contrast, the ***Physical Address Space*** has a "fixed" or constant location within memory. The physical address is the actual location in the memory where the data is saved.

To simplify their roles, think of a GPS system, or a map. The Logical Address Space can be referred to as directions, and the Physical Address Space is the location one seeks to arrive at. Both are necessary for a computer program to function properly.

Loading:

This is when a program is taken from secondary memory and sent to the main memory. It is done by a ***Loader***, which takes information from an executable file, puts it in the main memory, and prepares the program for execution by the CPU. There is ***Static Loading*** (putting the whole program into the main memory prior to the beginning of its execution) and ***Dynamic Loading*** (putting the program into the main memory as soon as it's requested).

Linking:

Linking is when the connections between all functions of a program are established for it to be executed. It is the collection and maintenance of various pieces of data and code into a single file. Linking is done when a program is *compiling* (the translation of source code into machine-readable code) and *loading*. Like with loading, there is ***Static Linking*** and ***Dynamic Linking*.**

Swapping:

Also referred to as *Memory Compaction,* swapping is when a process is brought to memory and temporarily copied to the disk after having been run. It allows us to access data on the hard disk when it's unavailable on the RAM, and it allows for larger and multiple processes to be executed simultaneously, while sacrificing system performance.

As mentioned earlier, ***fragmentation*** is a serious problem that can affect memory use in the CPU. There are two types of fragmentation:

- ***External Fragmentation***, which is when there is free memory space that cannot be assigned to because it is not contiguous. It can be overcome with ***Swapping*** (which we have explored above)***,*** as well as ***Paging*** (dividing the physical memory into split partitions of a fixed size and allowing the address space to be non-contiguous).
- ***Internal Fragmentation,*** which is when the memory allocation for a process is larger than the memory that has been requested. Proper ***Memory Allocation*** can be utilized to ensure that the size of requested memory best fits with the available memory space as much as possible.

Application

The main goal of using Memory Hierarchy is to carry out the optimal and most efficient use of memory within the CPU, by careful organization of each level. When done properly, it vastly reduces the time needed to access information/data and increases the speed with which operations are executed.

As elaborated earlier, the positioning of each memory level is important. The CPU Registers, being the fastest level to read and write information to, are placed at the top of the hierarchy, with the caveat of having the lowest storage space while in contrast devices like magnetic tapes have the largest storage space but are slow to access and are therefore at the bottom.

Arranging a memory hierarchy properly allows for an ***even distribution*** of the memory and enables data to be shared among different types of memory to further reduce access time and make it readily available. ***System Performance and responsivity*** is enhanced as a result, with the added benefit of ***lowering power consumption*** (the higher levels of the hierarchy use up more power, and by visiting the smaller/faster level first we can save on power by reducing the use of higher levels when they are not needed) and ***optimizing the cost efficiency*** of the memory.

Conclusion

With this case study, I have highlighted the need for and importance of the role that Memory Hierarchy plays within Computer Architecture. My emphasis has been on the priority of placement of each memory level, the function that each memory level performs as well as indicating their storage capacity and operation speeds, the types of memory and key processes in *Memory Management* such as *Loading, Linking and Swapping*.

I also discussed problems such as *fragmentation* and how best to deal with it via *Memory Allocation* and *Paging,* depending on the type of fragmentation.

Lastly, I discussed how best Memory Hierarchy can be applied and highlighted the advantages that come from its proper utilization.

Sources for the Literature Review/References:

Memory hierarchy design and its characteristics (2023) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/memory-hierarchy-design-and-its-characteristics/ (Accessed: 15 July 2024).

Chapter 6 - The Memory Hierarchy. Available at: https://csapp.cs.cmu.edu/2e/ch6-preview.pdf. (Accessed: 16 July 2024).

Unit-IV memory hierarchy design and its characteristics. Available at: https://annamalaiuniversity.ac.in/studport/download/engg/it/resources/Unit-IV%20&%20V%20Course%20Material.pdf (Accessed: 16 July 2024).

Awati, R. and Wigmore, I. (2022) *What is hierarchy (memory hierarchy)?, WhatIs*. Available at: https://www.techtarget.com/whatis/definition/hierarchy (Accessed: 16 July 2024).

Mehar, A. (2024) *What is memory hierarchy? design and characteristics, AlmaBetter*. Available at: https://www.almabetter.com/bytes/articles/what-is-memory-hierarchy (Accessed: 16 July 2024).

Locality of reference and cache operation in Cache Memory (2023) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/locality-of-reference-and-cache-operation-in-cache-memory/ (Accessed: 17 July 2024).

Cache memory in computer organization (2024) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/cache-memory-in-computer-organization/ (Accessed: 17 July 2024).

Memory management in operating system (2023) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/memory-management-in-operating-system/ (Accessed: 17 July 2024).

Logical and physical address in operating system (2024) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/logical-and-physical-address-in-operating-system/ (Accessed: 17 July 2024).

GeeksforGeeks (2023) *Difference between loading and linking in operating system, GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/difference-between-loading-and-linking/ (Accessed: 17 July 2024).

GeeksforGeeks (2023) *Swapping in operating system, GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/swapping-in-operating-system/ (Accessed: 17 July 2024).

GeeksforGeeks (2023) *Difference between internal and external fragmentation, GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/difference-between-internal-and-external-fragmentation/ (Accessed: 17 July 2024).

Paging in operating system (2024) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/paging-in-operating-system/ (Accessed: 17 July 2024).

Memory hierarchy (2024) *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Memory_hierarchy (Accessed: 18 July 2024).

Locality of Reference (2023) *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Locality_of_reference (Accessed: 18 July 2024).