Advanced Computer Architecture

(Assignment –II)

Submitted in partial fulfilment of the requirements for the degree of

Master of Technology in Information Technology

by

Vijayananda D Mohire

(Enrolment No.921DMTE0113)

Information Technology Department

Karnataka State Open University

Manasagangotri, Mysore – 570006

Karnataka, India

(2009)

Advanced Computer Architecture

CERTIFICATE

This is to certify that the Assignment-II entitled (Advanced Computer Architecture, subject code: MT12) submitted by Vijayananda D Mohire having Roll Number 921DMTE0113 for the partial fulfilment of the requirements of Master of Technology in Information Technology degree of Karnataka State Open University, Mysore, embodies the bonafide work done by him under my supervision.

Place: _____ **Signature of the Internal Supervisor**

Name

Date: _____ **Designation**

For Evaluation

Question Number	Maximum Marks	Marks awarded	Comments, if any
1	5		
2	5		
TOTAL	10		

Evaluator's Name and Signature Date

Preface

This document has been prepared specially for the assignments of M.Tech – IT I Semester. This is mainly intended for evaluation of assignment of the academic M.Tech - IT, I semester. I have made a sincere attempt to gather and study the best answers to the assignment questions and have attempted the responses to the questions. I am confident that the evaluator's will find this submission informative and evaluate based on the provide content.

For clarity and ease of use there is a Table of contents and Evaluators section to make easier navigation and recording of the marks. A list of references has been provided in the last page – Bibliography that provides the source of information both internal and external. Evaluator's are welcome to provide the necessary comments against each response, suitable space has been provided at the end of each response.

I am grateful to the Infysys academy, Koramangala, Bangalore in making this a big success. Many thanks for the timely help and attention in making this possible within specified timeframe. Special thanks to Mr. Vivek and Mr. Prakash for their timely help and guidance.

Candidate's Name and Signature Date

Table of Contents

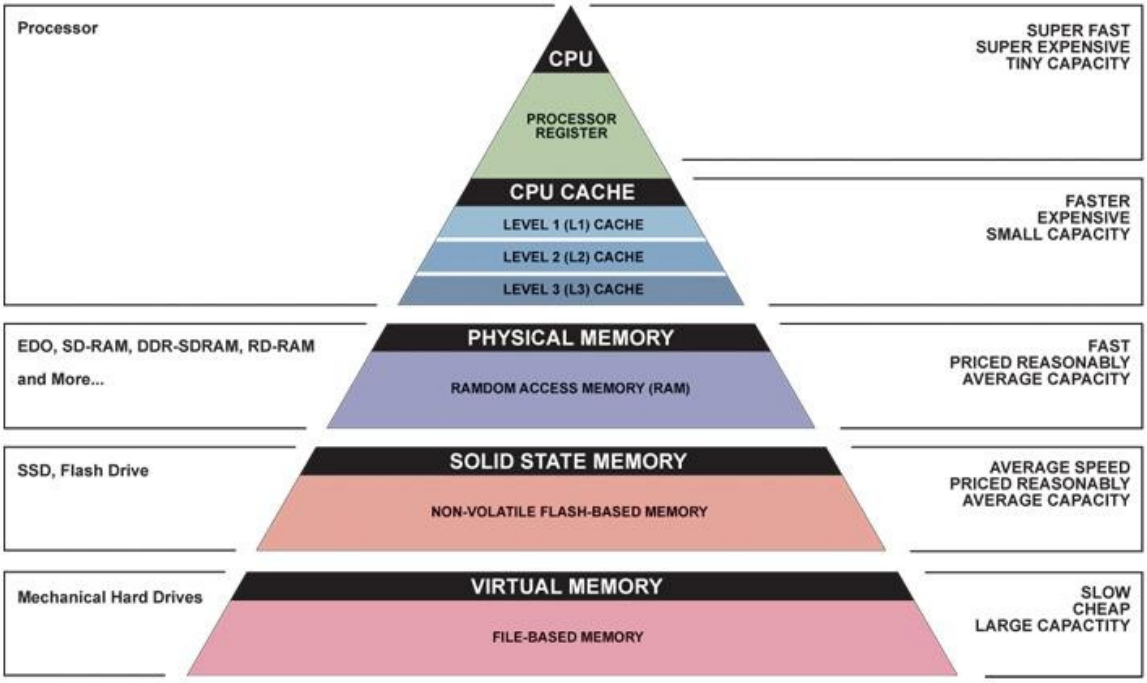
Table of Figures

ADVANCED COMPUTER ARCHITECTURE

RESPONSE TO ASSIGNMENT - II

Question 1 What is memory hierarchy? Explain the different levels of memory hierarchy?

Answer 1



▲ Simplified Computer Memory Hierarchy

Figure 1 Memory Hierarchy

A ranking of computer memory devices, with devices having the fastest access time at the top of the hierarchy, and devices with slower access times but larger capacity and lower cost at lower levels.

We classify memory based on its "distance" from the processor, with distance

measured by the number of machine cycles required for access. The closer memory is to the processor, the faster it should be. As memory gets further from the main processor, we can afford longer access times. Thus, slower technologies are used for these memories, and faster technologies are used for memories closer to the CPU. The better the technology, the faster and more expensive the memory becomes. Thus, faster memories tend to be smaller than slower ones, due to cost.

Memory hierarchy refers to a CPU-centric latency (delay)—the primary criterion for *designing* a placement in storage in a memory hierarchy—that fits the storage device into the design considerations concerning the operators of the computer. It is used only incidentally in operations, artifactually. It's primary use is in thinking about an abstract machines.

The levels of memory in a computer (Hyde, 2008). From fastest to slowest speed, they are:

1. CPU registers
2. L1 cache
3. L2 cache
4. Main memory
5. Virtual memory

6. Disk

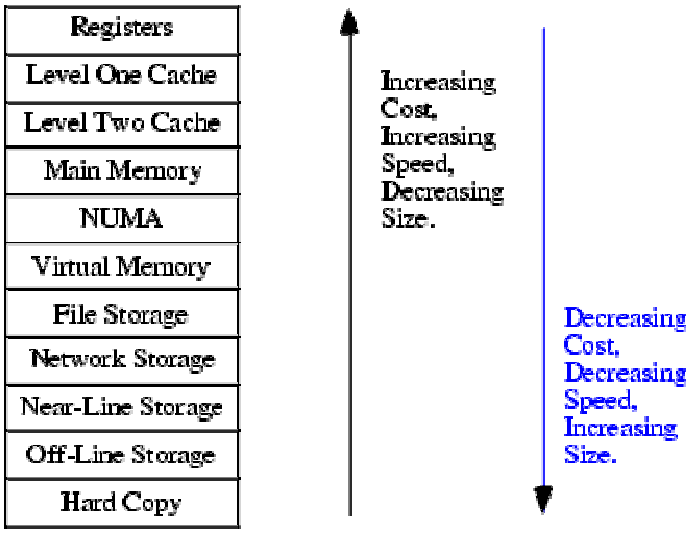


Figure 2 Levels of Memory (Hyde, 2008)

At the top level of the memory hierarchy are the CPU's general purpose registers. The registers provide the fastest access to data possible on the 80x86 CPU. The register file is also the smallest memory object in the memory hierarchy (with just eight general purpose registers available). By virtue of the fact that it is virtually impossible to add more registers to the 80x86, registers are also the most expensive memory locations. Note that we can include FPU, MMX, SIMD, and other CPU registers in this class as well. These additional registers do not change the fact that there are a very limited number of registers and the cost per byte is quite high (figuring the cost of the CPU divided by the number of bytes of register available).

Working our way down, the Level One Cache system is the next highest performance subsystem in the memory hierarchy. On the 80x86 CPUs, the Level One Cache is provided on-chip by Intel and cannot be expanded. The size is usually quite small (typically between 4Kbytes and 32Kbytes), though much larger than the registers available on the CPU chip. Although the Level One Cache size is fixed on the CPU and you cannot expand it, the cost per byte of cache memory is much lower than that of the registers because the cache contains far more storage than is available in all the combined registers.

The Level Two Cache is present on some CPUs, on other CPUs it is the system designer's task to incorporate this cache (if it is present at all). For example, most Pentium II, III, and IV CPUs have a level two cache as part of the CPU package, but many of Intel's Celeron chips do not[1]. The Level Two Cache is generally much larger than the level one cache (e.g., 256 or 512KBytes versus 16 Kilobytes). On CPUs where Intel includes the Level Two Cache as part of the CPU package, the cache is not expandable. It is still lower cost than the Level One Cache because we amortize the cost of the CPU across all the bytes in the Level Two Cache. On systems where the Level Two Cache is external, many system designers let the end user select the cache size and upgrade the size. For economic reasons, external caches are actually more expensive than caches that are part of the CPU

package, but the cost per bit at the transistor level is still equivalent to the in-package caches.

Below the Level Two Cache system in the memory hierarchy falls the main memory subsystem. This is the general-purpose, relatively low-cost memory found in most computer systems. Typically, this is DRAM or some similar inexpensive memory technology.

Below main memory is the NUMA category. NUMA, which stands for NonUniform Memory Access is a bit of a misnomer here. NUMA means that different types of memory have different access times. Therefore, the term NUMA is fairly descriptive of the entire memory hierarchy. In Fig.2, however, we'll use the term NUMA to describe blocks of memory that are electronically similar to main memory but for one reason or another operate significantly slower than main memory. A good example is the memory on a video display card. Access to memory on video display cards is often much slower than access to main memory. Other peripheral devices that provide a block of shared memory between the CPU and the peripheral probably have similar access times as this video card example. Another example of NUMA includes certain slower memory technologies like Flash Memory that have significant slower access and transfers times than standard semiconductor RAM. We'll use the term NUMA in this chapter

to describe these blocks of memory that look like main memory but run at slower speeds.

Most modern computer systems implement a Virtual Memory scheme that lets them simulate main memory using storage on a disk drive. While disks are significantly slower than main memory, the cost per bit is also significantly lower. Therefore, it is far less expensive (by three orders of magnitude) to keep some data on magnetic storage rather than in main memory. A Virtual Memory subsystem is responsible for transparently copying data between the disk and main memory as needed by a program.

File Storage also uses disk media to store program data. However, it is the program's responsibility to store and retrieve file data. In many instances, this is a bit slower than using Virtual Memory, hence the lower position in the memory hierarchy.

Below File Storage in the memory hierarchy comes Network Storage. At this level a program is keeping data on a different system that connects the program's system via a network. With Network Storage you can implement Virtual Memory, File Storage, and a system known as Distributed Shared Memory (where processes running on different computer systems share data in a common block

of memory and communicate changes to that block across the network).

Virtual Memory, File Storage, and Network Storage are examples of so-called *on-line memory subsystems*. Memory access via these mechanism is slower than main memory access, but when a program requests data from one of these memory devices, the device is ready and able to respond to the request as quickly as is physically possible. This is not true for the remaining levels in the memory hierarchy.

The Near-Line and Off-Line Storage subsystems are not immediately ready to respond to a program's request for data. An Off-Line Storage system keeps its data in electronic form (usually magnetic or optical) but on media that is not (necessarily) connected to the computer system while the program that needs the data is running. Examples of Off-Line Storage include magnetic tapes, disk cartridges, optical disks, and floppy diskettes. When a program needs data from an off-line medium, the program must stop and wait for a someone or something to mount the appropriate media on the computer system. This delay can be quite long (perhaps the computer operator decided to take a coffee break?). Near-Line Storage uses the same media as Off-Line Storage, the difference is that the system holds the media in a special robotic jukebox device that can automatically mount the desired media when some program requests it. Tapes and removable

media are among the most inexpensive electronic data storage formats available. Hence, these media are great for storing large amounts of data for long time periods.

Hard Copy storage is simply a print-out (in one form or another) of some data. If a program requests some data and that data is present only in hard copy form, someone will have to manually enter the data into the computer. Paper (or other hard copy media) is probably the least expensive form of memory, at least for certain data types.

Evaluator's Comments if any:

Question 2 Explain the different Flynn's Taxonomy of computer architecture

Answer 2

Over the years, several attempts have been made to find a satisfactory way to categorize computer architectures. Although none of them are perfect, today's

most widely accepted taxonomy is the one proposed by Michael Flynn in 1972. *Flynn's taxonomy* considers two factors: the number of instructions and the number of data streams that flow into the processor. A machine can have either one or multiple streams of data, and can have either one or multiple processors working on this data. This gives us four possible combinations: *SISD* (*single instruction stream, single data stream*), *SIMD* (*single instruction stream, multiple data streams*), *MISD* (*multiple instruction streams, single data stream*), and *MIMD* (*multiple instruction streams, multiple data streams*). (Anonymous, Flynn's_taxonomy, 2009)

	Single Instruction	**Multiple Instruction**
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

Figure 3 Computer classifications defined by Flynn (Anonymous, Flynn's_taxonomy, 2009)

The four classifications defined by Flynn are based upon the number of concurrent instruction (or control) and data streams available in the architecture:

Single Instruction, Single Data stream (SISD)

A sequential computer which exploits no parallelism in either the instruction or data streams. Examples of SISD architecture are the traditional uniprocessor machines like a PC or old mainframes.

Single Instruction, Multiple Data streams (SIMD)

A computer which exploits multiple data streams against a single instruction stream to perform operations which may be naturally parallelized. For example, an array processor or GPU.

Multiple Instruction, Single Data stream (MISD)

Multiple instructions operate on a single data stream. Uncommon architecture which is generally used for fault tolerance. Heterogeneous systems operate on the same data stream and must agree on the result. Examples include the Space Shuttle flight control computer.

Multiple Instruction, Multiple Data streams (MIMD)

Multiple autonomous processors simultaneously executing different instructions on different data. Distributed systems are generally recognized to be MIMD architectures; either exploiting a single shared memory space or a distributed memory space.

Visually, these four architectures are shown below where each "PU" is a processing unit:

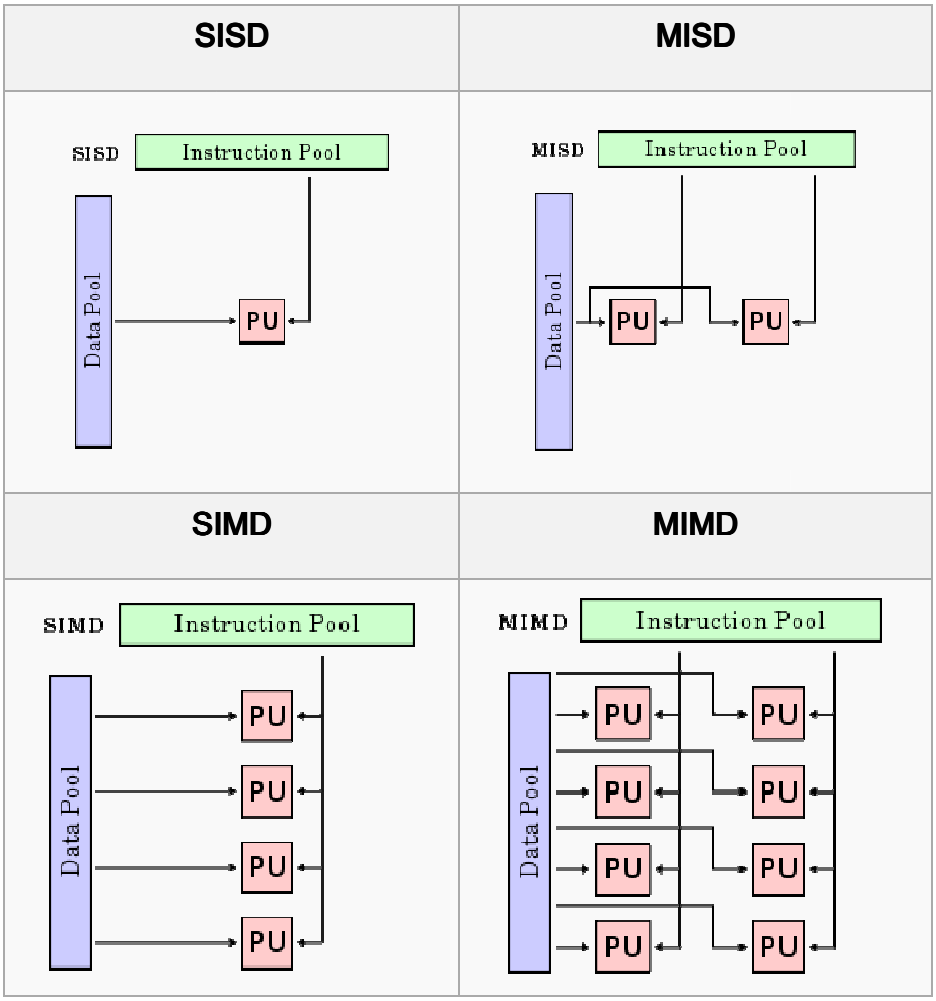


Figure 4 Pictorials of Flynn's Taxonomy (Anonymous, Flynn's_taxonomy, 2009)

At a level above where Flynn begins his taxonomy, we need to add one more characteristic, and that is whether the architecture is instruction driven or data driven. The classic von Neumann architecture is instruction driven. All processor activities are determined by a sequence of program code. Program instructions act *on* the data. Data driven, or *dataflow*, architectures do just the opposite. The characteristics of the data determine the sequence of processor

events.

With the addition of dataflow computers and some refinements to the MIMD classification, we obtain the taxonomy shown below .

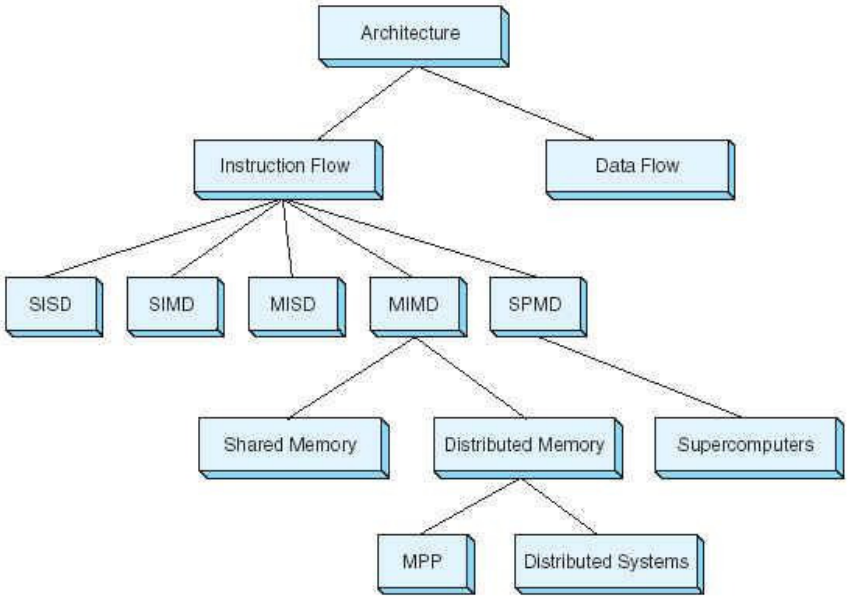


Figure 5 Refined Flynn's taxonomy

Evaluator’s Comments if any:

Bibliography

Anonymous. (2009). *Flynn's_taxonomy*. Retrieved 2009, from Wikipedia: http://en.wikipedia.org/wiki/Flynn's_taxonomy

Hyde, R. (2008). *Chapter Six Memory Architecture* . Retrieved from Art of Assembly Language by Webster: http://webster.cs.ucr.edu/AoA/Windows/HTML/MemoryArchitecture.html