Computer Architecture & Organization

BCA 2^{nd} Sem. Study Material
Paper: C4T

Memory Unit

Anupam Pattanayak[1]
Assistant Professor,
Department of Computer Science,
Raja N. L. Khan Women's College (Autonomous),
Midnapore, West Bengal

April 18, 2020

[1]anupam.pk@gmail.com

Contents

1

Memory Unit

In previous study material, we have discussed control unit. Hopefully, you have understood the discussions there. In case of any difficulty, you can send me your questions[1].

In this study matrial, we will try to cover the entire memroy unit of a computer system. We will cover the principle of locality, memory hierarchy, cache memory, primary memory, secondary memory, and virtual memory in brief. We will refer the books by Hennessy and Patterson[2] which explains cache memory very nicely , and book by Pacheco[3] for few selected topics. You are encouraged to read any text book you have with you at this time or the e-books that I have sent to you by e-mail.

1.1 Introduction

In previous semester, you must have studied basics of memory, primary memory and secondary memory. We can perform two basic operations in memory: *read* and *write*. Any memory system must have two associated reisters with it: *address register*, and *data register*. When we perform read operation, CPU provides the meory location whic is to be read in address register and after read operation the content of that memory address is available in data register. Similarly, in write operation, the data CPU wants to write in a particular memory address is put into data register an the address is put into address register. During write operation the content of data register is written into the specified memory location.

Primary memory is the memory which is compulsory for a computer

[1] anupam.pk@gmail.com

[2] Computer Architecture by Hennessy and Patterson, Morgan Kaufmann Publication

[3] An Introduction to Parallel Programming by Pacheco, Morgan Kaufmann Publication

system to have. That is why the term *primary*. Primary memory is *volatile*. That is, memory content is lost whenever power (voltage) goes off. There are two types of primary memory.

I. Random Access Memory (RAM),

II. Read Only Memory (ROM).

Secondary memory is not *must* for a computer system to have. This is used as more backup storage for primary memory. We often use the term *hard disk* in association with our computer. Hard disk is an example of secondary memory. Secondary memory is *non-volatile*.

There are different types of memory elements used in any computer system: CPU registers, cache memory, primary memory (or main memory), and secondary memory.

1.2 Principle of Locality

Computer scientists had studied the statistics of memory accesses done by CPU. What they realized is that, if a partcular memory location is referred by CPU now, then there is high chance that this same memory location will be again referred by CPU again very soon. This property is known as *temporal locality*. *Temporal* means with respect to *time*. In simpler words, a recently accessed memory location would be accessed again very soon. This property holds because most of the time CPU spends executing codes/data of *loops* of a program.

Another property is known as *spatial locality*. *Spatial* means with respect to memory *space*. This property says, if a memory location is referred by CPU now, then there is high probability that it's nearby memory locations would be accessed very soon. Due to spatial locality and the costly memory access operation, data is transferred in terms of *memory block* in memory hierarchy which we will discuss next.

1.3 Memory Hierarchy

There is a considerable gap of speed at which CPU operates and the speed at which memory can operate. Speed of primary memory is much slower than CPU. Speed of secondary memory is much much slower than primary memory. Computer Science pioneers of initial days correctly predicted that users will want computer memory should be as large as secondary memory, and speed of memory unit should be as fast as that of CPU. But achieving

both using a large high speed primary memory is not feasible. Because if want a primary memory of bigger size, then cost of the computer system would increase much. There has to be a trade off betweeen performance of computer system and cost of the system. This demand of users is met with memory hierarchy. At the top of memory hierarchy is the CPU registers, general purpose registers, which can hold some bytes of data. Speed of CPU registers is In the next level, there is *cache memory*. Cache memory speed is much faster than primary memory and closer to the spped of CPU. But cost of cache memory is much more than primary memory. Cache memory is fabricated in the same core where processor is also fabricated. That's why we never listen of a situation that, cache memory has beeen damaged and we need to replace it! The following figure 1.1 shows the memory hierarchy, which is in pyramidical structure.

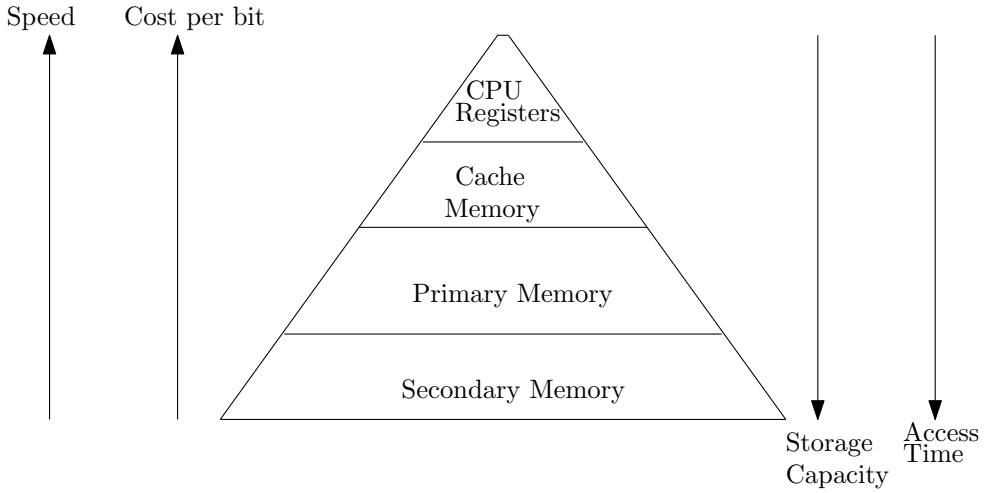

Figure 1.1: Memory Hierarchy of a Computer System

CPU register is at the top of thy memory hierarchy. Secondary memory is at the bottom of memory hierarchy. The arrows point out the different characteristics of different memory units. For example, storage capacity memory units increases starting from CPU registers towards the bottom and secondary memory is the largest in memory size. Access time also increases as we go down in the hierarchy. Cost per bit of storage increases as we go up in the memory hierarchy.

It can be pointed that some authors do not prefer to show CPU registers in memory hierarchy. It is consistent to include all the storage units of CPU including CPU registers in the memory hierarchy.

Memory hierarchy exploits principle of locality. It encashes temporal locality by maintaining recently accessed memory ontents closer to CPU. It exploits spatial locality by bringing in whole memory block to up in memory hierarchy level.

1.4 Cache Memory

Remember the concept of von Neumann Architecture where both code and data are kept in same memory which is shown in figure 1.2.

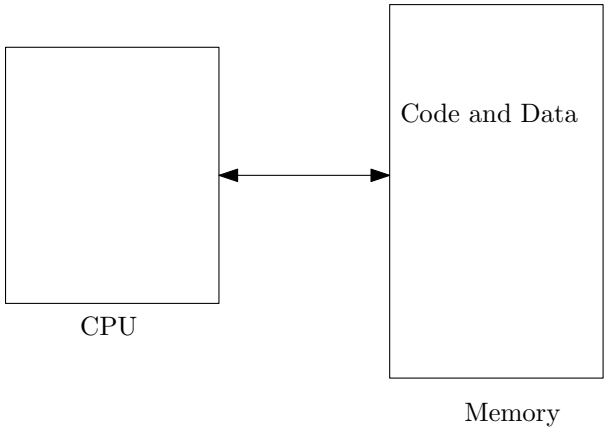

Figure 1.2: von Neumann Architecture

This architecture consists of CPU, primary memory (or main memory), and an *interconnection* between CPU and memory. This interconnection is generally a *bus*. Now the the data transfer speed between CPU and main memory is limited by the speed of this bus. This separation of CPU and memory is called *von Neumann bottleneck*. To overcome limitations caused by von Neumann bottleneck few modifications to the basic von Neumann architecture has been propose. One such proposal is to use *cache memory* between CPU and main memory.

Cache memory is a fast memory that resides in memory hierarchy between CPU registers and primary memory. CPU generates physical memory address of primary memory. This address is first searched in cache memory. If the corresponding memory content is available in cache memory then that is referred as *cache hit*. Otherwise, if the memory content is not found in cache memory, it is referred as *cache miss*. When cache miss occurs, then CPU looks for the data in primary memory. If the data is found there, then

copy of this data is kept in cache memory as well, with the hope that immediate access to this data will be served from cache memory. As we mentioned earlier, a cache miss causes a data block to be brought into cache instead of just one data word. So, *memory block* is the unit of data that is transerred from/to upper level to/from lower level of memory hierarchy. Typically, block size can be 64 bytes or 128 bytes, or so. *Hit rate* is the fraction of memory accesses that results cache hit. That is,

$$\text{Hit Rate} = \frac{\text{Number of times data found in cache}}{\text{Total number of memory accesses by CPU}} .$$

So, $\text{Miss Rate} = 1 - \text{Hit Rate}$
. In general, cache hit rate and hit time is specifed with the system speci-

fications. When cache miss occurs, CPU looks for the missed data item in primary memory. For the time being, we assume that on a cache miss, data referreed by CPU will always be found in primary memory. This *miss penalty* is time consuming. So the less cache miss the better system perfomance we get.

If we are given two choices to made: one is a cache memory with hit rate 40% or 0.4, another cache memory with cache hit rate of 60% or 0.6, then which one is preferable? Obviously the second one. If the cache hit rate (or miss rate) and hit time (or miss penalty), then we can calculate the *average memory access time*. Average memory access time is also called *effective memory access time*. As stated earlier, we assume for the time being that miss penalty=main memory access time.

$$\text{Avg Mem Access Time} = \text{Hit Rate} \times \text{Hit Time} + \text{Miss Rate} \times \text{Miss Penalty}$$

Some problems can be set using these formulas. For example, it is given that

miss rate=0.3, hit time=20 μ *seconds*, primary memory access time=150 *mili seconds*, then average memory access time, *Avg M. A. Time* is computed as

Avg M. A. Time=Hit Rate×Hit Time+Miss Rate×Main memory access time
$= (1 - 0.3) \times 20\ \mu s + 0.3 \times 150\ ms$
$= 0.7 \times 20\ \mu s + 45.0\ ms$
$=1.4\ \mu s + 45.0\ ms$
$=1.4\ \mu s + 45.0\ \times 1000 \mu s$
$=1.4\ \mu s + 45000.0\ \mu s$
$=45001.4\ \mu s$

So, cache memory data acts like local copy of data to CPU. The corre-

sponding global data copy is still maintained in main memory. There are two policies to keep synchronization between this local copy and global copy of data. First approach is called *write back cache*. In this type of cache, data is updated in main memory when the correponding block is getting replaced in cache memory. Another approach is called *write through cache*. Here, the data in main memory is updated whenever the corresponding data in cache memory gets updated.

1.5 Cache Memory Organization

One question may pop up in your mind: where a block is placed in cache memory? Is this done randomly? No, this block placement is decided by the underlying cache organization. This is very important topic. However, we will briefly cover this topic. Refer the book by Hennessy and Patterson[4] for in-depth study on this topic.

There are three types of cache memory organizations:

I. Direct-mapped cache,

II. Fully associative cache, and

III. Set-associative cache.

As, the term *direct* suggest, in direct-mapped cache, a memory block is mapped to directly one particular block in cache memory. Following formula can be used to find the cache block address in direct-mapped cache:

Cache block address = Main memory block addr. % No. of cache blocks
So, here one disadvantage is that a recently used cache memory block can be replaced by another main memory block by the mapping even if some other cache memory blocks are free for use or blocks which are not used recently are there in cache.

To overcome this disadvantage, second type of cache organization, *fully associative cache* organization is used. In this organization a main memory block can be placed anywhere in the cache memory. However, it has a disadvantage that searching in cache memory increases for a possible cache hit.

To keep balance between these two extreme cache organizations, *set-associative cache* is used. Here, a block can be placed in a restricted set

[4]Computer Architecture by Hennessy and Patterson, Morgan Kaufmann Publication.

of cache blocks. This set is generally of two blocks, or four blocks, or eight blocks. When, a main memory block is mapped, it's mapped to a fixed set. Then inside that set, it can be placed anywhere. If there are 2 blocks in every set, then that cache is referred as *2-way set associative cache*. Similarly, if there are 4 blocks in every set, then that cache is referred as *4-way set associative*. Following formula are used to determine the set address in the set-associative cache:

Cache set address = Main memory block addr. % No. of sets in the cache

1.6 Main Memory Organization

RAM and ROM are the examples of main memory or primary memory. We will here briefly discuss the RAM organization. There are two different types of organizations of RAM:

I. Static RAM (SRAM), and

II. Dynamic RAM (DRAM).

SRAM uses transistors to store a single bit. SRAM uses very minimum power to hold the charge of transistors when the memory is not used for a long time or goes into standby mode. Data access is very fast in SRAM. That is why, cache memory is generally built by SRAM technology. However, the cost per bit is much more in SRAM.

DRAM packs more bits per chip than SRAM. It uses capacitors to store a bit. However, as the capacitor tends to decay charge over the time, DRAM needs to be periodically *refreshed* to retain the charge. All the bits in same row can be refreshed at the same time. It reduces cost per bit. Main memory is generally built using DRAM technology.

In the following table 1.1, the difference between SRAM and DRAM are given.

Table 1.1: Difference between SRAM and DRAM

SRAM	DRAM
It stands for Static RAM.	It stands for Dynamic RAM.
Transistor is used to store a bit.	Capacitor is used to store a bit.
Bit charge does not decay over time.	Bit charge decays over time.
No need to refresh memory cells.	Periodic refresh of memory cells are required.
Very fast speed	Speed is slower.
Cost per bit is high	Cost per bit is much lower.
Less density of storage in IC packaging.	High density of storage in IC packaging.
Used to manufacture cache memory.	Used to bulid main memory.

1.7 Secondary Memory Organization

The *hard disk* used in our systems is the example secondary memory. It provides bulk storage capacity. When a data referred by CPU is not found in main memory, it is known as *page fault*. When a page fault occurs, the data is searched for in secondary memory.

Hard disks uses magnetic disk technology. It has many circular shaped plates inside it kept fixed by a rod in the centre. Two surfaces of plates are coated with magnetic material. Data is recorded on the surfaces. The disks rotates, say 120 revolutions per second, when in use. There is a RD/WR head placed over both the surface of the plates. This RD/WR head movement is mechanical process involved in magenttic disk. Due to the presence of this mechanical component, magnetic disks are so slow compare to main memory speed.

The RD/WR heads on all the plates rotate together. As a result, when the head on one plate is over the i^{th} track, the heads on remaining plates are also over their i^{th} tracks. That is why, all the i^{th} tracks of all plates form a logical storage unit called the i^{th} *cylinder*.

The disk surface is logically divided into several *tracks*. Every track is subdivided into several *sectors*. A sector is typically of 512-bytes.

To access a particular sector, the RD/WR head must be first placed on the correct track. This time to repositioning the RD/WR head is called *seek time*. After the RD/WR head is place onto the track, it must wait for the plate to rotate, so that desired sector comes under the RD/WR head. This time of disk rotation is called *latency time*. *Access time* is the sum of seek time and latency time.

1.8 Virtual Memory

Dictionary meaning of virtual is *not physical*. *Virtual memory* is a technique that enables main memory to be treated as cache for secondary memory. Virtual memory accommodates only the active portions of several executing programs in main memory.

Here, the unit of data transfer between main memory and secondary memory is called *page*. If a page is not found in main memory, then that event is called *page fault*. In case of page fault, the page is searched for in secondary memory. A *page table* keeps the address mapping of page from secondary memory to main memory. If the page table is maintained in main memory, then we need two main memory accesses for accessing a page. To avoid this, page table is keppt in a fast cache memory called *translation look-ahead buffer* (TLB). You will study virtual memory in much more detail when *operating system* will be taught in some later semesters of your course. There some other topics like memory block replacement strategies, disk addresses scheduling are covered, which is not in the scope of currrent syllabus. But this knowledge will definitely help you there.