

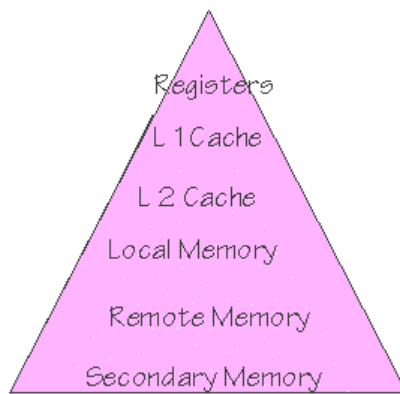
Tujuan: Memahami memory secara fisik maupun organisasi

Pokok bahasan: Memory Subsystem

Reference: Book 1 (Chapter 2)

Tugas:

- [Tugas Mandiri 1] Chapter summary tentang Virtual Memory (Reference: Buku 1 Sub Chapter 2.2 Halaman 60 - 80). Maximum 3 halaman. (Due: 1 Minggu)



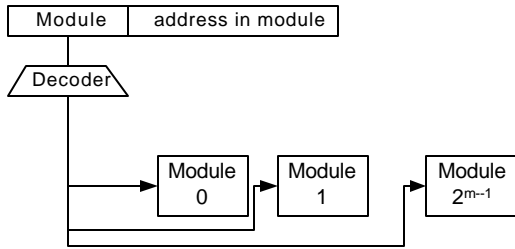
HIERARCHICAL MEMORY STRUCTURE

The **objective** for memory hierarchy is as an attempt to match the processor speed with the memory's rate of information transfer (or the bandwidth) at the lowest level possible and with reasonable cost.

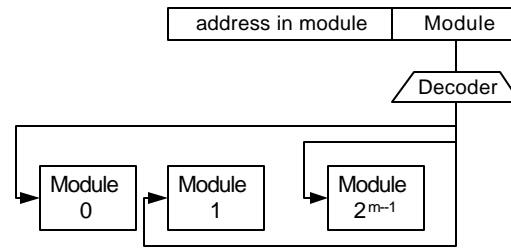
The Registers is considered as the fastest memory and Secondary memory is the slowest of all. The closer a memory is coupled to the processor, the faster its transfer rate is.

In parallel processor architecture, the main memory is a prime system resource and thus shared by all parallel processor. There are possibilities for **conflict** to occur, which is a concurrent memory request at the same level of hierarchy. More than 1 processor requesting the same memory address from the same hierarchy at the same time. To address the issues, **interleaving** is used, which is a method in partitioning the memory space into several independent memory modules and separating cache memory into several modules. There are 2 basic method in distributing the address among memory modules:

- **High-order interleaving.** The high-order m bits are used to select the module while the remaining n-m bits to select the address within the module.
- **Low-order interleaving.** The low-order m bits of the address select the module, while the remaining n-m bits to select the address within the module. This arrangement will distribute the addresses so that consecutive addresses are located within consecutive modules.



High-order interleaving

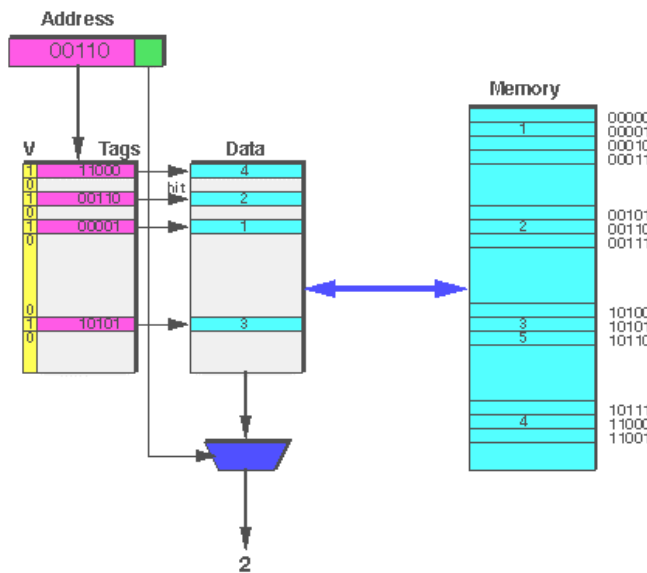


Low-order interleaving

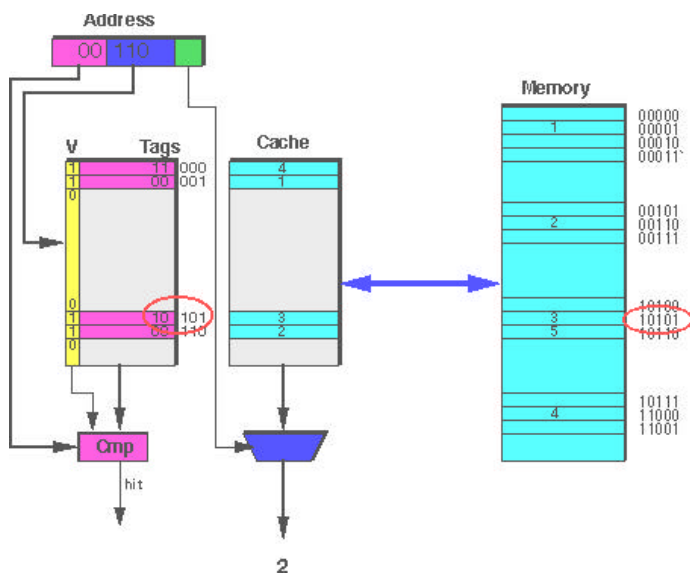
CACHE ORGANIZATION

Fully Associative

How does the cache know if a requested data item is present in the cache? It uses a **tag** associated with each entry in the cache containing the memory address of the data item currently residing in this cache line.



When the processor provides an address to the cache, the tag bits of the address are compared against all tags stored in the cache. If one matches, it is a hit, and the corresponding data item is sent to the processor. If there is a miss, an available location in the cache is selected and the data is retrieved from memory. The data, its tag, and the valid bit are written in the cache



This cache is called fully associative because all cache **tags** are searched in parallel to find a hit. Data items, if they are present, could be anywhere in the cache. However, this can be expensive

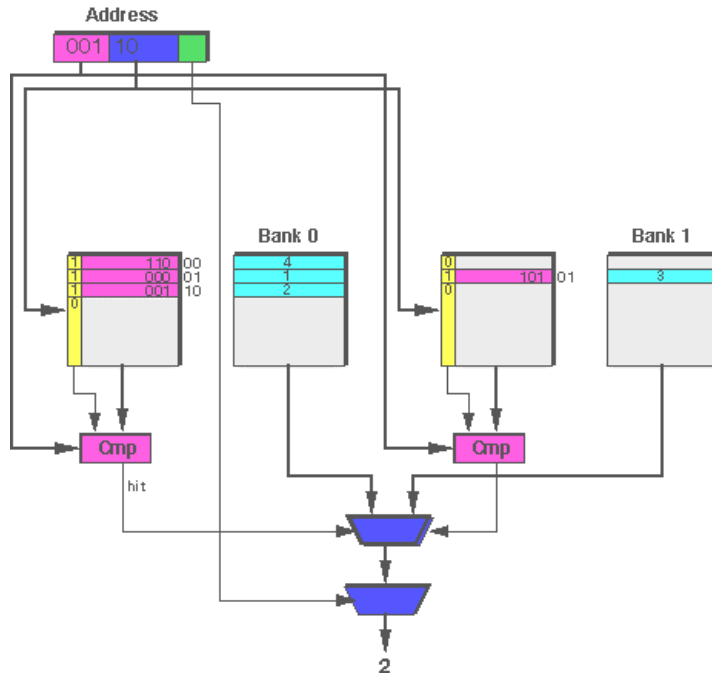
Direct Mapped

In direct mapped cache, if a data item is in the cache, there is exactly one place where they could be, there is a definite location in the cache memory allocated for each address in the main memory.

There are three fields in the

address: **Tag bits**, used to check for a matching tag in the cache, **Line bits** used as the address to look up both the tag and the data for the one line possibly containing the data, and **Offset bits**, used to select one word from all words in a cache line, as before.

N-way Set Associative Organization



In this case, there are several tag-data pairs for each cache address (two in the picture) - this is called a set; hence the name set associative. The above cache is called 2-way set associative.

The address is broken into three fields: **Tag bits**, used to check for a matching tag in all of the lines in one set of the cache, **Set bits** used as the address to look up all tag-data pairs for the one set possibly containing the data, and **Offset bits**, used to select one word from all words in a cache line.

After the set bits are used to access an entire set, all of the

tags are compared in parallel. If any one matches, the corresponding data line is selected, and the offset bits are used to further select one word from the line.

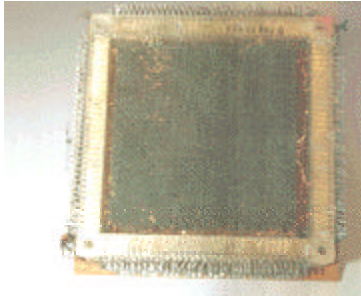
We may get a single cycle memory access only when we get a cache hit. If there is a cache miss, the processor stalls while the cache controller accesses memory to bring the desired data into the cache - at least several cycles. This extra time is called the **miss penalty**. we can classify cache misses into 3 types:

- **Compulsory Misses:** The misses occur the first time we access a piece of data, which is not yet resided in the cache memory and the cache controller need to bring it into the cache from the main memory. These type of misses may be overcome by increasing the line size, but it will make the miss penalty larger because each miss requires more data to be transferred from memory. For a fixed total size cache and associativity, increasing the line size means the number of sets in the cache decreases, thus increasing capacity misses.
- **Capacity Misses:** Misses caused by the fact that there is just not enough room in the cache for all the required data. Therefore something must be replaced and brought back into the cache later. These misses may be overcome by increasing the total cache size, but by increasing the total size we may increase the hit time (tag and data lookup time) and making the cache memory bigger in physical size and thus more expensive.
- **Conflict Misses:** Misses that are caused by conflicts in the cache; i.e. misses that occur because several data are lines map into the same place in the cache, thus requiring the old line to be replaced and brought back later when needed. These only occur for set

associative and direct mapped caches, but not the fully associative organization. These misses may be overcome by increasing associativity, but the extra logic for selecting the data can also affect the hit time and require more expensive cache.

VIRTUAL MEMORY SYSTEM

HISTORY: FERRITE CORE MEMORY BLOCK (1960S)



Ferrite is iron. This form of computer memory was widely used through the late 1950's to the early 1970's. This memory block can store up to 32 bits, or four characters. The "On" or "Off" switches are created by having an iron magnet in the shape of a donut intersected by horizontal and vertical wires. The "1" and "0" switch setting was created by alternating the direction of the positive and negative currents.

Core memory was a great step forward in memory technology in the 1960's. It was the first memory technology that had a decent quantity of memory in a practical size, power requirements and speed. Prior to core memory, the dominant technologies were magnetic drum memory, electrostatic memory, and (rarely) mercury delay line memory. Core technology was the first to allow memory to have the reliability of solid state electronics. It could be hand assembled for about a penny a bit, and retained what it stored despite power failures.

This 4K memory core block with a 22-bit word size was removed from a Daystrom-046 computer built in San Diego in 1957. This particular Daystrom was used at National Steel in Detroit, Michigan to check 300 process control points. The system would check approximately 40-150 points per hour. The Daystrom 046 computer system cost \$300,000 without peripherals.

Source: The Computer Museum of America