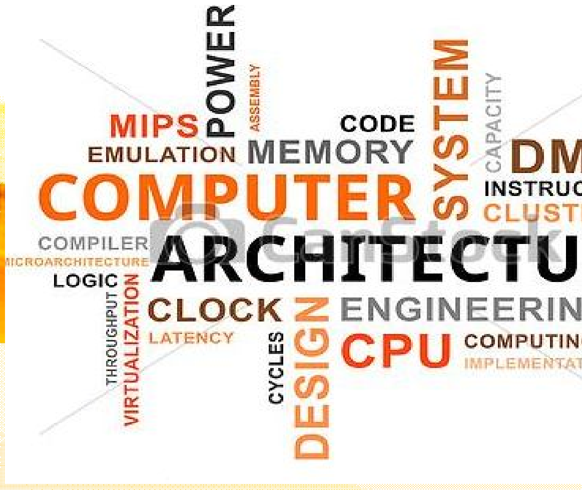
UNIT IV
MEMORY SYSTEM

Basic concepts of Semiconductor RAMs - ROMs – Speed, Size and Cost **– Cache memories – Performance consideration** – Virtual memory – Memory Management requirements – Secondary storage - Case Study: Memory Organization in Multiprocessors

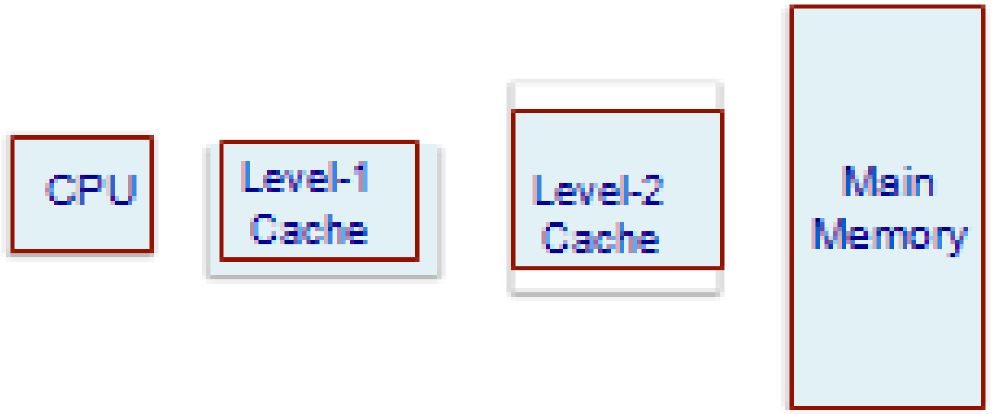


Recap the previous Class

What is Cache Memory?

A **fast memory** (possibly organized in several levels) that sits between processor and main memory.

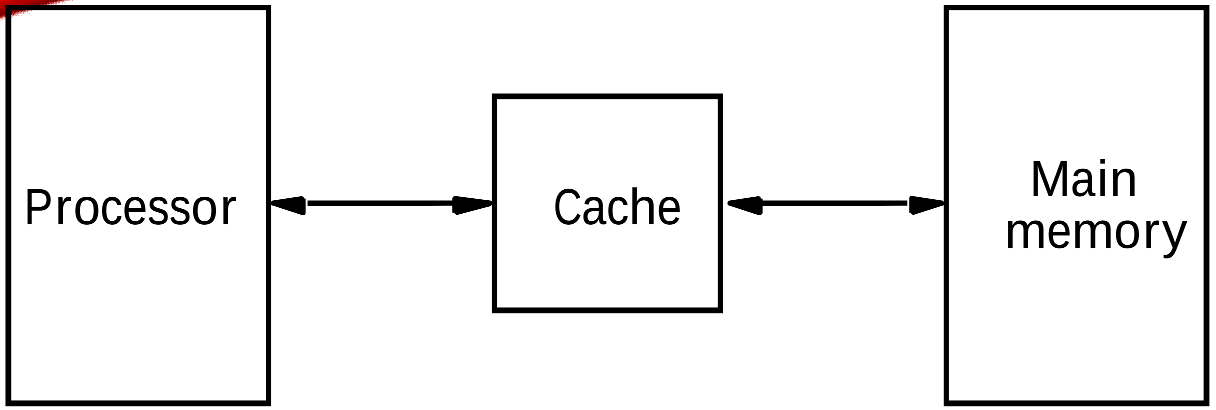
- Faster than main memory and relatively small.
- Frequently accessed data and instructions are stored here.
- Cache memory makes use of the fast SRAM technology.



- Processor is much faster than the main memory.
  - As a result, the processor has to spend much of its time waiting while instructions and data are being fetched from the main memory.
  - Major obstacle towards achieving good performance.
- Cache memory is an architectural arrangement which makes the main memory appear faster to the processor than it really is.
- Cache memory is based on the property of computer programs known as **“locality of reference”.**

Locality of Reference

- Analysis of programs indicates that many **instructions in localized areas of a program are executed repeatedly** during some period of time, while the others are accessed relatively less frequently.
  - These instructions may be the **ones in a loop, nested loop or few procedures** calling each other repeatedly. This is called **“locality of reference”.**
- **Temporal locality of reference:**
  - Recently executed instruction is likely to be executed again very soon.
- **Spatial locality of reference:**
  - Instructions with addresses close to a recently instruction are likely to be executed soon.



- Processor issues a Read request, a block of words is transferred from the main memory to the cache, one word at a time.
- Subsequent references to the data in this block of words are found in the cache.
- At any given time, only some blocks in the main memory are held in the cache. Which blocks in the main memory are in the cache is determined by a **"mapping function".**
- When the cache is full, and a block of words needs to be transferred from the main memory, some block of words in the cache must be replaced. This is determined by a **"replacement algorithm".**

Cache hit

- Existence of a cache is transparent to the processor. The processor issues Read and Write requests in the same manner.
- If the data is in the cache it is called a **<u>Read or Write hit</u>**.
- **Read hit:**
  - The data is obtained from the cache.

Cache hit

- **Write hit:**
  - Cache has a replica of the contents of the main memory.
  - Contents of the cache and the main memory may be updated simultaneously. This is the **write-through protocol**.
  - Update the contents of the cache, and mark it as updated by setting a bit known as the **dirty bit or modified** bit. The contents of the main memory are updated when this block is replaced. This is **write-back or copy-back protocol.**

Cache miss

- If the data is not present in the cache, then a **Read miss or Write miss** occurs.
- **Read miss:**
  - Block of words containing this requested word is transferred from the memory.
  - After the block is transferred, the desired word is forwarded to the processor.
  - The desired word may also be forwarded to the processor as soon as it is transferred without waiting for the entire block to be transferred. This is called **load-through or early-restart.**

Cache miss

- **Write-miss:**
  - Write-through protocol is used, then the contents of the main memory are updated directly.
  - If write-back protocol is used, the block containing the addressed word is first brought into the cache. The desired word is overwritten with new information.

Cache Coherence Problem

- A bit called as **“valid bit”** is provided for each block.
- If the block **contains** valid data, then the bit is **set to 1**, else it is 0.
- Valid bits are set to 0, when the power is just turned on.
- When a block is loaded into the cache for the first time, the valid bit is set to 1.
- Data transfers between main memory and disk occur directly bypassing the cache.
- When the data on a disk changes, the main memory block is also updated.
- However, if the data is also resident in the cache, then the valid bit is set to 0.

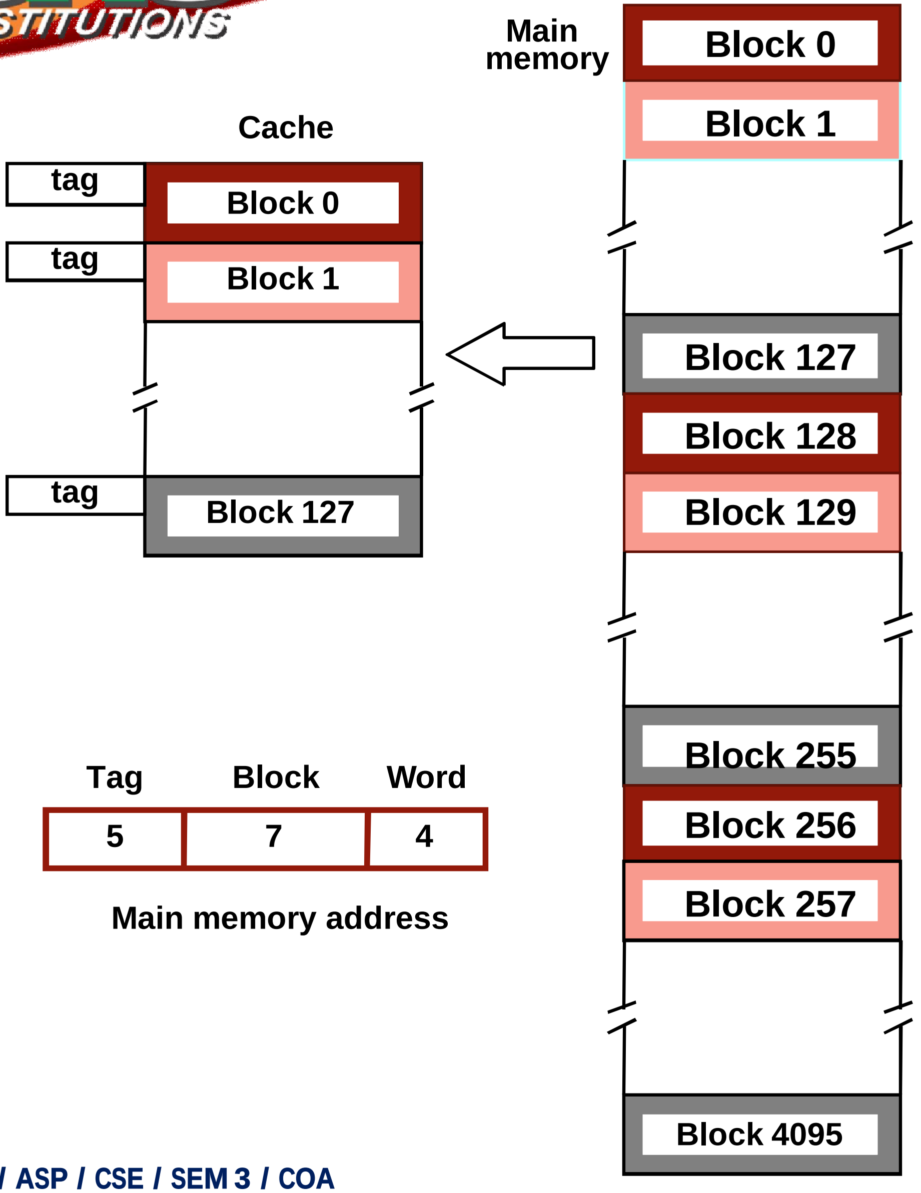
Cache Coherence Problem

- What happens if the data in the disk and main memory changes and the write-back protocol is being used?
- In this case, the data in the cache may also have changed and is indicated by the dirty bit.
- The copies of the data in the cache, and the main memory are different. This is called the **cache coherence problem**.
- One option is to force a write-back before the main memory is updated from the disk.

Mapping functions

- Mapping functions determine how memory blocks are placed in the cache.
- A simple processor example:
  - Cache consisting of 128 blocks of 16 words each.
  - Total size of cache is 2048 (2K) words.
  - Main memory is addressable by a 16-bit address.
  - Main memory has 64K words.
  - Main memory has 4K blocks of 16 words each.
- **Three mapping functions:**
  - Direct mapping
  - Associative mapping
  - Set-associative mapping.

Direct mapping



- Block j of the main memory maps to j modulo 128 of the cache. 0 maps to 0, 129 maps to 1.
- More than one memory block is mapped onto the same position in the cache.
- May lead to contention for cache blocks even if the cache is not full.

Direct mapping

•Resolve the contention by allowing new block to replace the old block, leading to a trivial replacement algorithm.

•**Memory address is divided into three fields**:

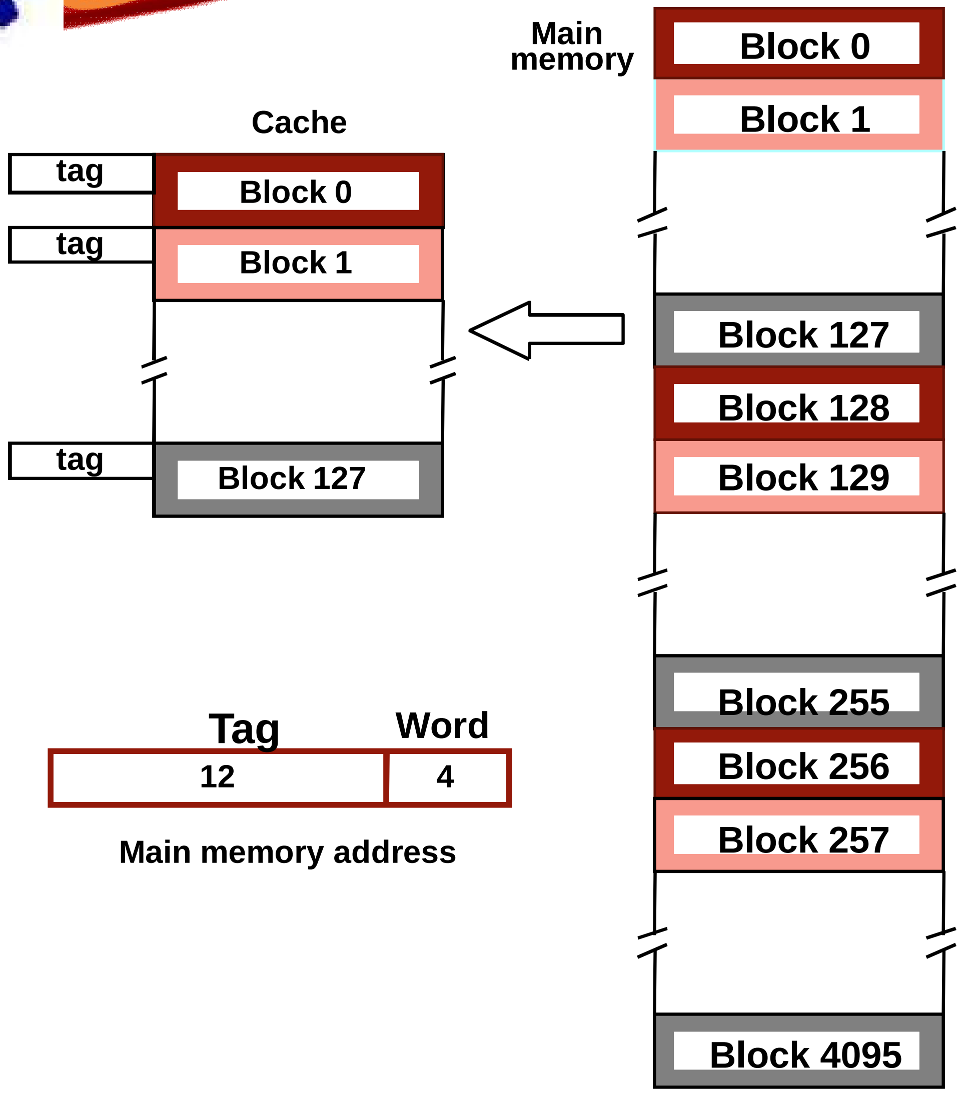
- Low order 4 bits determine one of the 16 words in a block.

- When a new block is brought into the cache, the next 7 bits determine which cache block this new block is placed in.

- High order 5 bits determine which of the possible 32 blocks is currently present in the cache. These are **tag bits**.

• **Simple** to implement but not very flexible.

Associative mapping



•Main memory block can be placed into any cache position.

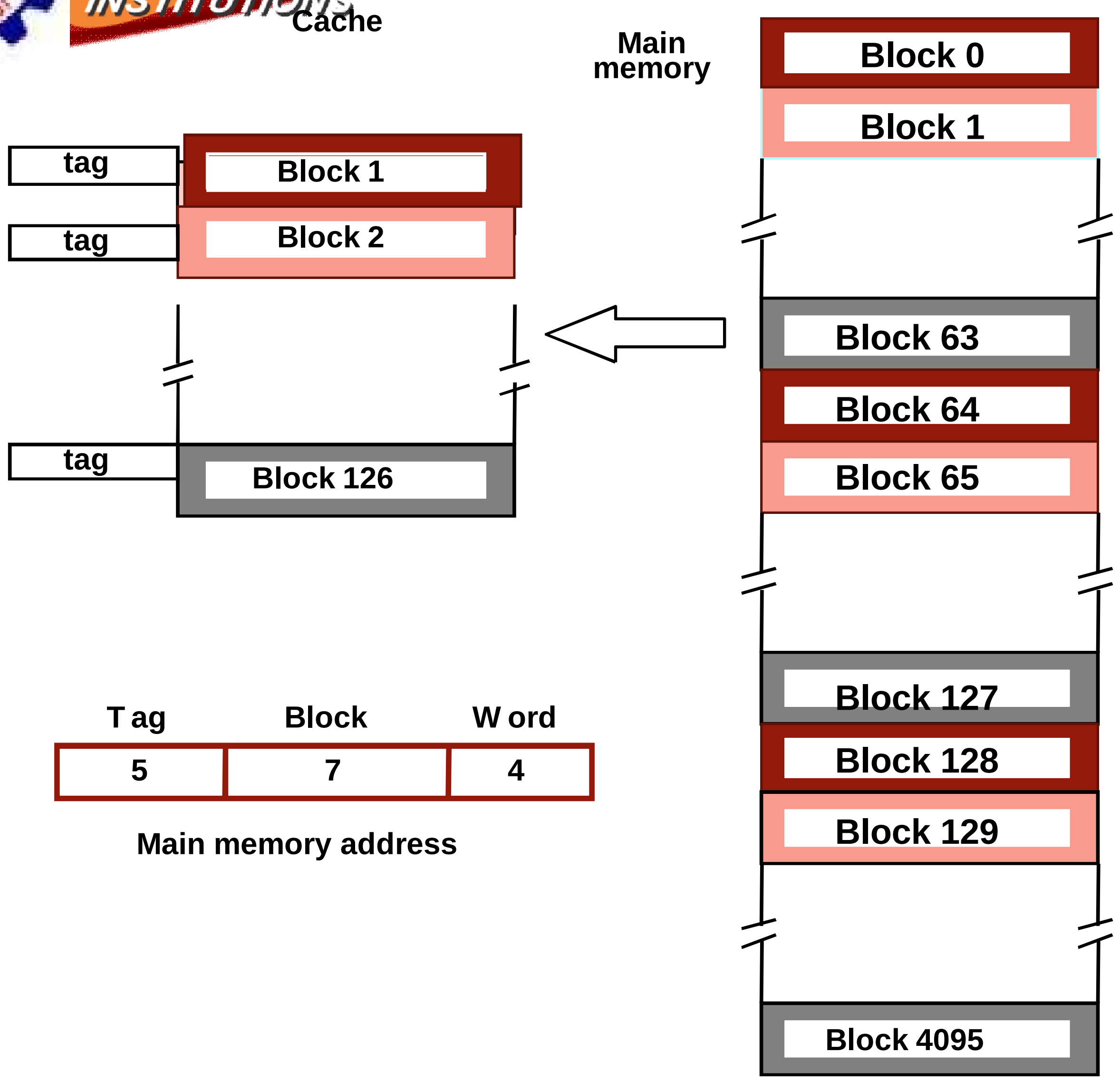
•Memory address is divided into **two fields**:

- Low order 4 bits identify the word within a block.
- High order 12 bits or tag bits identify a memory block when it is resident in the cache.

Associative mapping

- **Flexible,** and uses cache space efficiently.
- Replacement algorithms can be used to replace an existing block in the cache when the cache is full.
- Cost **is higher than direct-mapped cache** because of the need to search all 128 patterns to determine whether a given block is in the cache.

Set-Associative mapping



- Blocks of cache are grouped into **sets**.
- Mapping function allows a block of the main memory to reside in any block of a specific set.
- Divide the cache into 64 sets, with two blocks per set.
- Memory block 0, 64, 128 etc. map to block 0, and they can occupy either of the two positions.

Set-Associative mapping

Memory address is divided into **three fields**:

- 6 bit field determines the set number.

- High order 6 bit fields are compared to the tag fields of the two blocks in a set.

Set-associative mapping combination of direct and associative mapping.

Number of blocks per set is a design parameter.

- One extreme is to have all the blocks in one set, requiring no set bits (fully associative mapping).

- Other extreme is to have one block per set, is the same as direct mapping.

Performance considerations

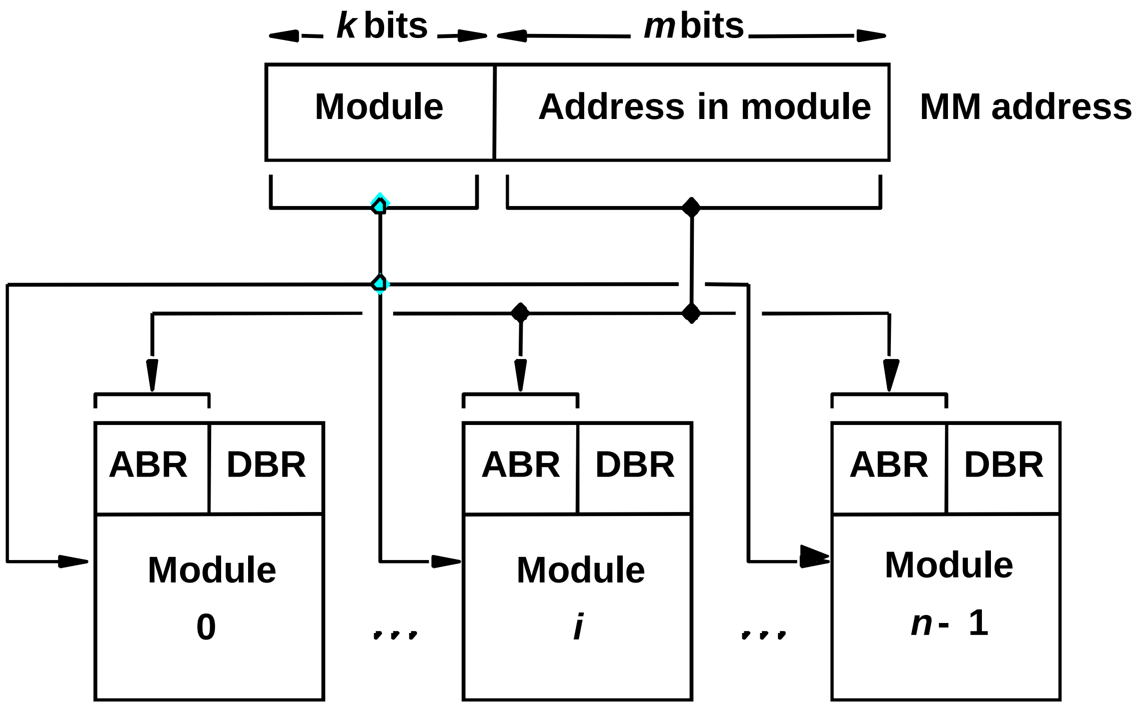
- A key design objective of a computer system is to achieve the best possible performance at the lowest possible cost.
  - Price/performance ratio is a common measure of success.
- Performance of a processor depends on:
  - How fast machine instructions can be brought into the processor for execution.
  - How fast the instructions can be executed.

Interleaving

- Divides the memory system into a number of memory modules.

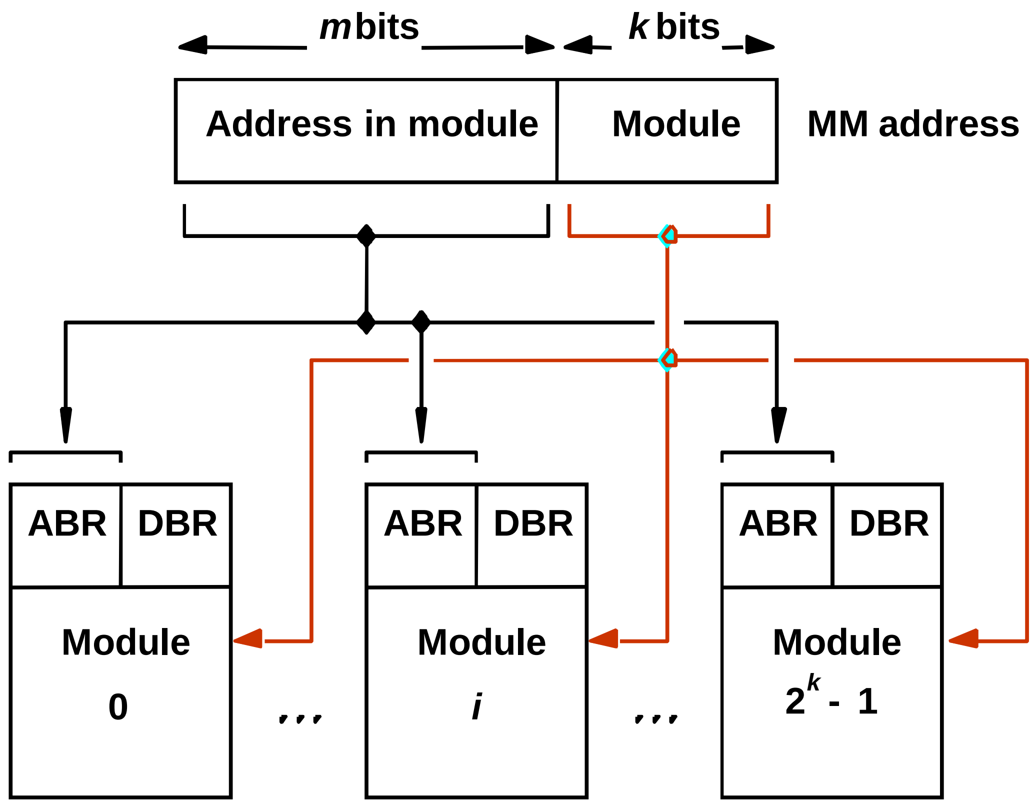
  Each module has its own **address buffer register (ABR) and data buffer register (DBR).**
- Arranges addressing so that successive words in the address space are placed in different modules.
- When requests for memory access involve consecutive addresses, the access will be to different modules.
- Since parallel access to these modules is possible, the average rate of fetching words from the Main Memory can be increased.

Methods of address layouts



- Consecutive words are placed in a module.
- High-order k bits of a memory address determine the module.
- Low-order m bits of a memory address determine the word within a module.
- When a block of words is transferred from main memory to cache, only one module is busy at a time.

Methods of address layouts



- Consecutive words are located in consecutive modules.
- Consecutive addresses can be located in consecutive modules.
- While transferring a block of data, several memory modules can be kept busy at the same time.

Hit Rate and Miss Penalty

- Hit rate
- Miss penalty
- Hit rate can be improved by increasing block size, while keeping cache size constant
- Block sizes that are neither very small nor very large give best results.
- Miss penalty can be reduced if load-through approach is used when loading new blocks into cache.

Caches on the processor chip

- In high performance processors 2 levels of caches are normally used.
- Avg access time in a system with 2 levels of caches is

$$T_{ave} = h1c1+(1-h1)h2c2+(1-h1)(1-h2)M$$

TEXT BOOK

Carl Hamacher, Zvonko Vranesic and Safwat Zaky, “Computer Organization”, McGraw-Hill, 6th Edition 2012.

REFERENCES

1. David A. Patterson and John L. Hennessey, “Computer organization and design”, MorganKauffman ,Elsevier, 5th edition, 2014.
2. William Stallings, “Computer Organization and Architecture designing for Performance”, Pearson Education 8th Edition, 2010
3. John P.Hayes, “Computer Architecture and Organization”, McGraw Hill, 3rd Edition, 2002
4. M. Morris R. Mano “Computer System Architecture” 3rd Edition 2007
5. David A. Patterson “Computer Architecture: A Quantitative Approach”, Morgan Kaufmann; 5th edition 2011

THANK YOU