Memory Hierarchy Design

→ Principle of locality

For relatively long periods of time, a program tends to reference a very definite fraction of its addressing space.

- spatial locality: when a memory word is referenced once, another one stored nearby may be referenced soon.
- temporal locality: when a memory word is referenced once, it may be referenced soon.

→ Memory hierarchy

Hierarchy in different levels

- register bank: internal memory to the processor
- cache: external memory to the processor, access is controlled by instruction fetch and data transfer, consists of several levels of static RAM (first level inside processor chip).
- main memory: implements the stored-program concept with instructions and data of an executing program are stored (dynamic RAM).
- swapping area: non-volatile memory located in mass storage (OS), usually a virtual memory paged-architecture.

Although memory hierarchy may consist of several levels, data transfer usually takes place between two adjacent levels at a time.

hit → data requested by processor appears in some block at upper level

miss → lower level is accessed to retrieve the block containing the requested data.

hit rate / ratio → fraction of memory references to date found in the upper level over all memory references.

miss rate / ratio → complement of 1 of hit ratio.

→ Cache

memory addrs

block addr. (s bits)	offset (w bits)

num blocks in main mem = 2^s

Name given to levels of memory hierarchy between the processor and main memory.

Cache (number of bytes stored is a power of 2) can be organized as:

- direct mapped: single place where a block may be placed.
- fully associative: a block can be placed anywhere
- set associative: aggregate of places, set, where a block may be placed.

thrashing is a phenomenon that arises when the fraction of the addressing space referenced by the processor in an extended period of time contains groups of two or more addresses that map into the same cache lines.

Direct mapping:

block address.

number of lines

tag - field (s-r bits)	index field (r bits)

in cache = 2^r
cache size = 2^{r+w}

Number of lines per set is 1, number of sets is equal to the number of lines. The tag field contained in each line has minimal length.

Fully associativity:

block address

tag field (s bits)

Number of lines per set is equal to the number of lines in the cache and number of sets is 1 and tag field in each line has maximum length.

Set associativity

tag field (s-d bits)	index field (d bits)

There are k lines where a particular block may be stored. The number of lines per set is k and number of sets is v. ($v = 2^d$) ($k = m/v = 2^{r-d}$)

↳ Cache misses

What line whose block will be replaced is selected when a cache miss occurs?

In direct mapping, the line checked is the

are modified.

In associative organization there are some lines to be considered (if validation bit is used, one of them will be selected, but often some time all will be valid and a decision must be taken)

strategies:

random – pseudo-random generator

least recently used (LRU) – relies on past to predict the future (not influenced for the longest time)

first in first out (FIFO) – line whose contents remained for the longest time in the cache.

When a cache miss occurs, the block address is loaded onto the system bus and data is returned to both the cache and the processor.

In reads, the tag of the candidate are compared to the tag field of the block address to see if there is a hit. If so, the requested part of the block is passed to the processor. If not, the block transfer from the lower level is initiated and the read operation is stalled until the transfer is complete.

In writes, modifying blocks only happen after the tag is checked for a hit. Two write policies:

write-through: data are written to both the cache line and the block in the lower level. Its simpler to

implement, updated block contents is always present at a lower level. Important note in multi-level caches (for upper levels it only needs to propagate to next lower level)

write-back: data are written to the cache line. The line block contents are only written to the lower level when the block is replaced. Write operations for hits occur at the speed of the cache. Less memory bandwidth. Power is saved.

A write miss may be dealt in the following ways:

- write allocate: a line is allocated whenever a miss occurs, then the write operation takes place.
- non write allocate: the cache is unaffected by misses, the write operation only takes place at a lower level.