Cache memory

The speed of main memory is very low in comparison with this speed of the processor. For good performance, the processor cannot spend much of its time waiting to access instructions and data on the main memory. A mechanism is important to reduce the time needed to access the necessary information.

If the active portion of the program and

data are placed in a fast small memory, the average memory access time can be reduced, thus reducing the total execution time of the program. Such a small memory is referred to as cache memory. It is placed between cpu and main memory. The cache memory access time is less than the access time of main memory by a factor of 5 to 10. The cache is the fastest component in the memory hierarchy and approaches the speed of cpu components.

The basic operations of the cache is as follows: when the processor needs to access memory the cache is examined. If word is found in the cache it is read from the fast memory. If the word addressed by the processor is not found in the cache the main memory is accessed to read the word. A block of words containing the one just accessed is then transferred from main memory to cache memory. The block size may vary from one word to about 16 words adjacent to the 1 just accessed.

Type text here

Hit ratio

The performance of cache memory is frequently

measured in terms of a quantity called hit ratio. When the processor refers to memory and finds the word in cache it is said to produce a hit. If the word is not found in the cache it is in the main memory and it counts as a miss. The ratio of the number of hits divided by the total cpu reference to memory is the hit ratio.

Locality of reference

The effectiveness of the cache mechanism is based on a property of computer programs called locality of reference Analysis of programs shows that most of their execution time is spent on routines in which many instructions are executed repeatedly eg: Simple loop, nested loops etc.

Many Instructions in localized areas of the program are executed repeatedly during some time period and the remainder of the program is accessed relatively infrequently. This is referred to as locality of reference. It appears in two ways - Temporal and spatial. Temporal aspect means that a recently executed instruction is likely to be executed again very soon. The spatial aspect means that instructions in close proximity to a recently executed instruction with respect to the instruction addresses are likely to be executed soon.

The temporal aspect of locality of reference suggests that whenever an information item, either an instruction or data is first needed, this item should be brought into the cache where it will remain until it is needed again. The spatial aspect suggests that instead of fetching just one item from the main memory to the cache, it is useful to fetch several items that reside at adjacent addresses as well.

(A set of contiguous address locations is referred to as a block. A cache block is known as cache line.)

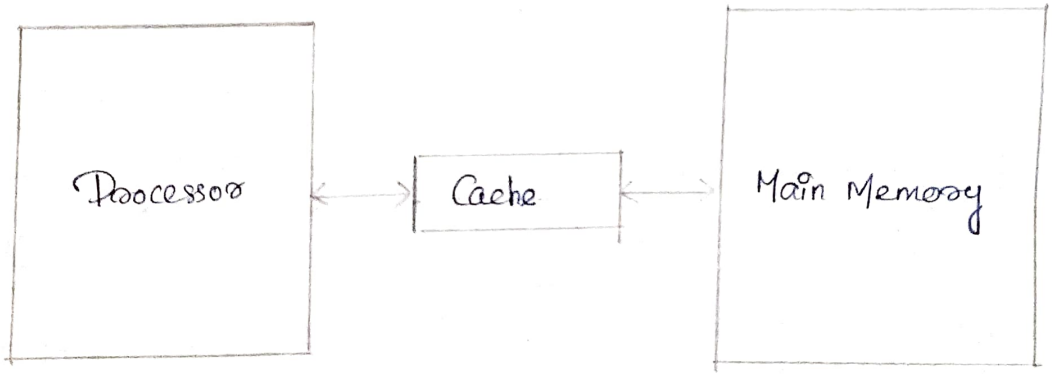


Fig: Use of Cache memory

Replacement Algorithm:

When the cache is full and a memory word (Instruction or data) that is not in the cache is referenced, the cache control hardware must decide which block

should be removed to create space for the new block that contains the referenced word. The collection of rules for making this decision constitutes the replacement algorithm.

Write through protocol and load through protocols :-

The processor does not need to know explicitly about the existence of the cache. It simply issues read and write requests using addresses that refer to locations in the memory. The cache control circuitary determines whether the requested word currently exists in the cache, if it does, the read or write operation is performed on the appropriate cache location. In this case, a read or write hit is said to have occurs. In a read operation the main memory is not involved. For write operation the system can proceed in two ways.

In the first technique called the write through protocol, the cache location and the main memory location are updated simultaneously. The second technique is to update only the cache location and to mark it as updated with an associated flag bit, called the dirty or modified bit. The main memory location of the word is updated later, when the block containing this marked word is to be removed from the cache to make room for a new block.

This technique is known as write ~~block~~ back or copy back protocol.

When the addressed word in a read operation is not in the cache, a read miss occurs. The block of words that contains the requested word is copied from the main memory into the cache. After the entire block is loaded into the cache the particular word requested is forwarded into the processor. This word may be sent to the processor as soon as it is read from the main memory. This approach is called load through or early restart.

During a write operation, if the addressed word is not in the cache, write miss occurs. Then the write through protocol or write back protocol can be used. If the write through protocol is used the information is written directly into the main memory. In the write back protocol, the block containing the addressed word is first brought into the cache and then the desired word in the cache is overwritten with the new information. later to memory also

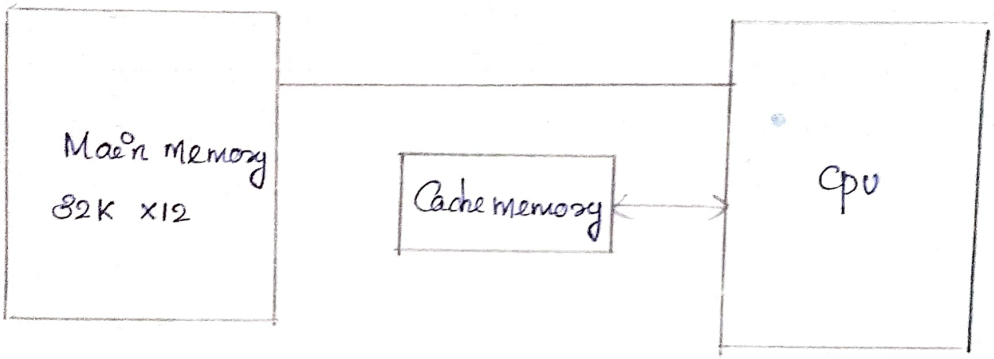
Mapping functions

The basic characteristic of cache memory is its fast access time. Therefore, very little or no time must be wasted when searching for words in the Cache.

The transformation of data from main memory to Cache memory is refered to as a mapping process. 3 types of mapping procedures are these

1. Associative mapping
2. Direct mapping
3. Set associative mapping

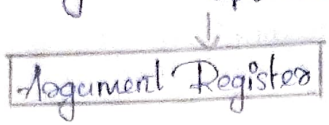
Consider the following memory Organization.



The main memory can store 32 k words of 12 bits each. The Cache is capable of storing 512 of these words at any given time. For every word stored in Cache, there is duplicate copy in the main memory. The cpu communicate with both memories. The processor first send the 15 bit address to the cache. If there is a hit, the cpu accept the 12 bit data from the Cache. If there is a miss, the cpu reads the word from the main memory and the word is then transfered to the Cache

Associative Mapping

The fastest and most flexible cache organization uses an associative memory. The organization is shown in the following figure.



Associative mapping cache

← Address →	← Data →
01000	3450
02777	6710
22345	1234

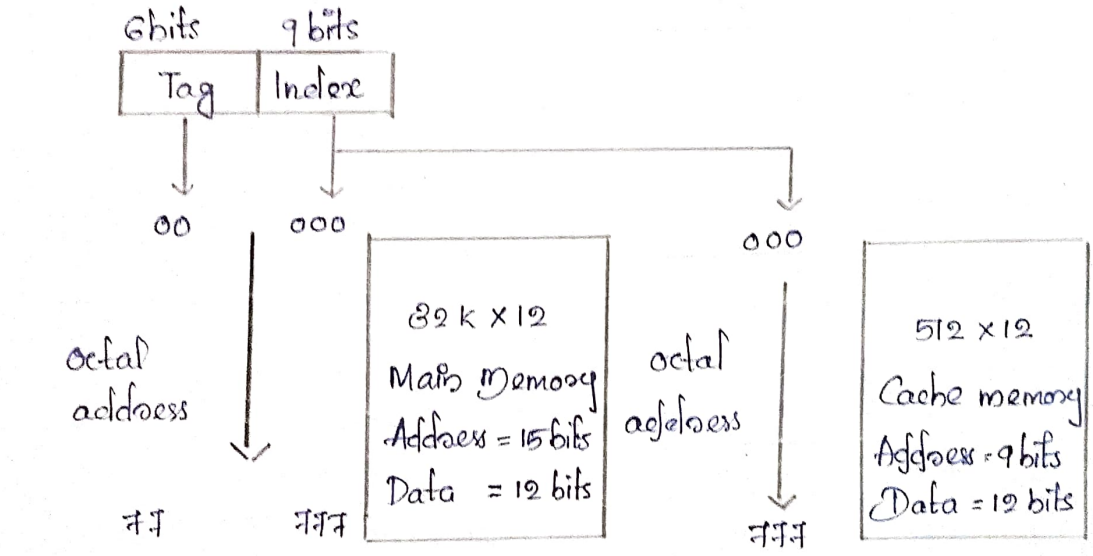
The associative memory stores both the address and content (data) of the memory word. This permits any location in cache to store any word from the memory. The figure shows three words presently stored in the cache. The address value of 15 bits is shown as a five digit octal number, and its corresponding 12 bit word is shown as a four digit octal number. A CPU address of 15 bits is placed in the argument register and the associative memory is searched for a matching address. If the address is found, the corresponding 12 bit data is read and sent to the CPU. If no match occurs, the main memory

is accessed for the word. The address data pair is then transferred to the associative cache memory. If the cache is full, an address data pair must be displaced to make room for a pair that is needed and not presently in the cache. This decision is made by replacement algorithms.

Direct Mapping

Instead of using associative memory for cache, there is the possibility of using a random access memory for the cache.

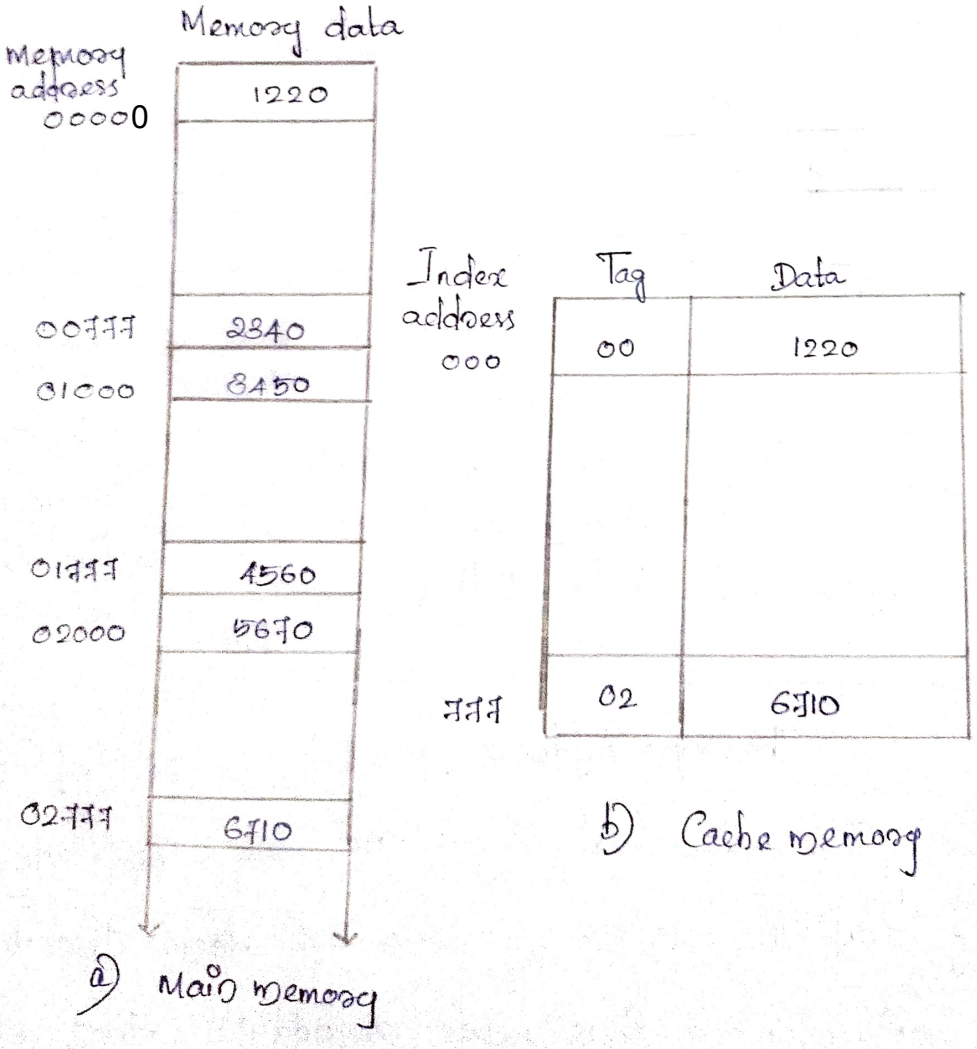
Addressing relationship between main memory and cache memory.



The cpu address of 15 bits is divided into two fields. The nine least significant bits constitute the index field and the remaining six bits form the tag field. The main memory thus needs an address that includes

both the tag and index bits. The number of bits in the index field is equal to the number of address bits required to access the cache memory.

In general, there are 2^k words in the cache memory and 2^n words in the main memory. The n-bit memory address is divided into two fields: k-bits for the index field and n-k bits for the tag field. The direct mapping cache organization uses the n-bit address to access the main memory and the k bit index to access the cache. The internal organization of the words in the cache memory is shown in fig(b).



Direct mapping Cache Organization

Each word in the cache consists of the data word and its associated tag. When a new word is first brought into the cache, the tag bits are stored alongside the data bits. When the cpu generates a memory request, the index field is used for the address to access the cache. The tag field of the cpu address is compared with the tag in the word read from the cache. If the two tags match, there is a hit and the desired data word is in cache. If there is no match, there is a miss and the required word is read from the main memory. It is then stored in the cache together with the new tag.

The disadvantage of direct mapping is that the hit ratio can drop considerably if two or more words whose addresses have the same index but different tags are accessed repeatedly.

operation of direct mapping organization is as follows: Consider the above example figures fig a & fig b. The word at address zero is presently stored in the cache (index = 000, tag = 00, data = 1220) Suppose that the cpu now wants to access the word at address 02000. The index address is 000, so it is used to access the cache. The two tags are then compared. The cache tag is 00, but the address tag is 02, which does not produce a match. Therefore the main memory is accessed and cache word at index address 000 is then replaced with a tag of 02 and data of 5670

Set Associative Mapping

The disadvantage of direct mapping is that two words with the same index in their address but with different tag values cannot reside in the cache memory at the same time. So there are another mapping, set associative mapping. In this organization, each word of cache can store two or more words of memory under the same index address. Each data word is stored together with its tag and number of tag data items in one word of cache is said to form a set. An example of a set associative cache organization for a set size of two is shown below.

Index	Tag	Data	Tag	Data
000	01	3450	02	5670
777	02	6710	00	2340

Two way set associative mapping cache

Each index address refers to two data words and their associated tags. Each tag requires six bits

and each data word has 12 bits, so the word length is 2(6+12) = 36 bits. An index address of nine bits can accomodate 512 words, these the size of cache memory is 512 x 36. In general a set associative cache memory of set size k will accomodate k words of main memory in each word of cache.

The operation of set associative cache is as follows. The words stored at the addresses 01000 and 02000 of the main memory are stored in cache memory at index address 000. Similarly the words of addresses 02777 and 00777 are stored in cache at index address 777. When the cpu generates a memory requests, the index value of the address is used to access the cache. The tag field of the cpu address is then compared with both tags in the cache to determine if a match occurs. The comparison is done by an associate search of the tags in the set and thus the name set-associate mapping. The hit ratio will improve as the set size increases because more words with the same index but different tags can reside in the cache.

Replacement Algorithm

When a miss occurs in n set associative cache and the set is full, it is necessary to replace one of the tag-data items with a new value. The most common replacement algorithm used are,

1. Random replacement
2. First in first out (FIFO)
3. least recently used (LRU)

With the random replacement policy, the control circuitry chooses one tag-data item for replacement at random. The FIFO selects for replacement, the item that has been in the set the longest. The LRU algorithm selects for replacement, the item that has been least recently used by the cpu.

Write through and write back methods

An important aspect of cache organization is concerned with memory write requests when the cpu finds a word in cache during a read operation, the main memory is not involved in the transfer. If the operation is a write, there are two ways that the system can proceed.

The simplest and the most commonly used procedure is to update main memory with every memory write operation with cache memory being updated in parallel if it contains the word at the specified address. This is called the write through method. This method has the advantage that main memory always contains the same

data as the cache. This characteristics is important in systems with direct memory access transfers. It ensures that the data residing in main memory are valid at all times so that an I/O device communicating through DMA would receive the most recent updated data.

The second procedure is called the write back method. In this method only the cache location is updated during a write operation. The location is then marked by a flag bit known as dirty bit or modified bit)

Modified bit, so that later when the word is removed from the cache, it is copied into the main memory. The reason for the write back method is that, during the time a word resides in the cache, it may be updated several times; however as long as the word remains in the cache, it does not matter whether the copy in main memory is out of data. It is only when the word is displaced from the cache that an accurate copy need be rewritten into main memory.

Cache Initialization

The cache is initialized when power is applied to the computer or when the main memory is loaded with a complete set of programs from auxiliary memory. After initialization the cache is considered to be empty, but in effect it

Contains some non-valid data. So it is necessary to include with each word in Cache, a valid bit to indicate whether or not the word contains valid data.

The Cache is initialized by clearing all the valid bits to 0. The valid bit of a particular cache word is set to 1, the first time this word is loaded from main memory, and stays set unless the Cache has to be initialized again. The introduction of valid bit means that a word in Cache is not replaced by another word unless the valid bit is set to 0 and a mismatch of tag occurs. If the valid bit happens to be 0, the new word automatically replaces the invalid data.