**Memory System**

1. Microcomputer Memory

Memory is an essential component of the microcomputer system. It stores binary instructions and datum for the microcomputer. The memory is the place where the computer holds current programs and data that are in use.

Computer memory exhibits perhaps the widest range of type, technology, organization, performance and cost of any feature of a computer system. The memory unit that communicates directly with the CPU is called main memory. Devices that provide backup storage are called auxiliary memory or secondary memory.

**Characteristics of memory systems:**

The memory system can be characterised with their Location, Capacity, Unit of transfer, Access method, Performance, Physical type, Physical characteristics, Organisation.

**Location:**

- Processor memory: The memory like registers is included within the processor and termed as processor memory.
- Internal memory: It is often termed as main memory and resides within the CPU.
- External memory: It consists of peripheral storage devices such as disk and magnetic tape that are accessible to processor via i/o controllers.

**Capacity:**

- Word size: Capacity is expressed in terms of words or bytes. The natural unit of organisation
- Number of words: Common word lengths are 8, 16, 32 bits etc.

**Unit of Transfer:**

- Internal: For internal memory, the unit of transfer is equal to the number of data lines into and out of the memory module.
- External: For external memory, they are transferred in block which is larger than a word.

**Addressable unit**

- Smallest location which can be uniquely addressed
- Word internally
- Cluster on Magnetic disks

**Access Method:**

- Sequential access: In this access, it must start with beginning and read through a specific linear sequence. This means access time of data unit depends on position of records (unit of data) and previous location. e.g. tape
- Direct Access: Individual blocks of records have unique address based on location. Access is accomplished by jumping (direct access) to general vicinity plus a sequential search to reach the final location. e.g. disk
- Random access: The time to access a given location is independent of the sequence of prior accesses and is constant. Thus any location can be selected out randomly and directly addressed and accessed. e.g. RAM
- Associative access: This is random access type of memory that enables one to make a comparison of desired bit locations within a word for a specified match, and to do this for all words simultaneously. e.g. cache

**Performance:**

- Access time: For random access memory, access time is the time it takes to perform a read or write operation i.e. time taken to address a memory plus to read / write from addressed memory location. Whereas for non-random access, it is the time needed to position read / write mechanism at desired location. i.e. Time between presenting the address and getting the valid data
- Memory Cycle time: It is the total time that is required to store next memory access operation from the previous memory access operation.

Memory cycle time = access time + transient time (any additional time required before a second access can commence). i.e. Time may be required for the memory to “recover” before next access. **Note:** Cycle time is access + recovery.

---

Blog - https://anilkumarprathipati.wordpress.com/  
NEC (Autonomous)
Transfer Rate: This is the rate at which data can be transferred in and out of a memory unit. i.e. Rate at which data can be moved
- For random access, \( R = \frac{1}{\text{cycle time}} \)
- For non-random access, \( T_n = T_a + \frac{N}{R}; \) where \( T_n \) – average time to read or write \( N \) bits, \( T_a \) – average access time, \( N \) – number of bits, \( R \) – Transfer rate in bits per second (bps).

**Physical Types:**
- Semiconductor RAM
- Magnetic Disk & Tape
- Optical CD & DVD
- Others Bubble Hologram

**Physical Characteristics:**
- Decay: Information decays mean data loss.
- Volatility: Information decays when electrical power is switched off.
- Erasable: Erasable means permission to erase.
- Power consumption: how much power consumes?

**Organization:**
- Physical arrangement of bits into words
- Not always obvious
  - e.g. interleaved

2. **The Memory Hierarchy**

Capacity, cost and speed of different types of memory play a vital role while designing a memory system for computers.
If the memory has larger capacity, more application will get space to run smoothly.
It's better to have fastest memory as far as possible to achieve a greater performance.
Moreover for the practical system, the cost should be reasonable.
There is a trade-off between these three characteristics cost, capacity and access time. One cannot achieve all these quantities in same memory module because
If capacity increases then access time increases (slower) and due to which cost per bit decreases.
If access time decreases (faster), capacity decreases and due to which cost per bit increases.
The designer tries to increase capacity because cost per bit decreases and the more application program can be accommodated. But at the same time, access time increases and hence decreases the performance. **So the best idea will be to use memory hierarchy.**
Memory Hierarchy is to obtain the highest possible access speed while minimizing the total cost of the memory system.
Not all accumulated information is needed by the CPU at the same time.
Therefore, it is more economical to use low-cost storage devices to serve as a backup for storing the information that is not currently used by CPU
The memory unit that directly communicate with CPU is called the **main memory**
Devices that provide backup storage are called **auxiliary memory**
The memory hierarchy system consists of all storage devices employed in a computer system from the slow by high-capacity auxiliary memory to a relatively faster main memory, to an even smaller and faster cache memory
The main memory occupies a central position by being able to communicate directly with the CPU and with auxiliary memory devices through an I/O processor
A special very-high-speed memory called **cache** is used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate.
CPU logic is usually faster than main memory access time, with the result that processing speed is limited primarily by the speed of main memory.

The cache is used for storing segments of programs currently being executed in the CPU and temporary data frequently needed in the present calculations.

The memory hierarchy system consists of all storage devices employed in a computer system from slow but high capacity auxiliary memory to a relatively faster cache memory accessible to high speed processing logic. The figure below illustrates memory hierarchy.

![Memory Hierarchy Diagram](image)

**Fig: Memory Hierarchy**

As we go down in the hierarchy:
- Cost per bit decreases
- Capacity of memory increases
- Access time increases
- Frequency of access of memory by processor also decreases.

**Hierarchy List**
- Registers
- L1 Cache
- L2 Cache
- Main memory
- Disk cache
- Disk
- Optical
- Tape

### 3. Internal (or) Main Memory
- The main memory is the central unit of the computer system. It is relatively large and fast memory to store programs and data during the computer operation. These memories employ semiconductor integrated circuits. The basic element of the semiconductor memory is the memory cell.
- The memory cell has three functional terminals which carries the electrical signal.
  - The select terminal: It selects the cell.
  - The data in terminal: It is used to input data as 0 or 1 and data out or sense terminal is used for the output of the cell's state.
  - The control terminal: It controls the function i.e. it indicates read and write.
- Most of the main memory in a general purpose computer is made up of RAM integrated circuits chips, but a portion of the memory may be constructed with ROM chips.
RAM—Random Access memory:
- Memory cells can be accessed for information transfer from any desired random location.
- The process of locating a word in memory is the same and requires of locating a word in memory is the same and requires an equal amount of time no matter where the cells are located physically in memory thus named 'Random access'.
- Integrated RAM are available in two possible operating modes, Static and Dynamic.

SRAM versus DRAM
- Both volatile
  - Power needed to preserve data
- Static RAM
  - Uses flip flop to store information
  - Needs more space
  - Faster, digital device
  - Expensive, big in size
  - Don't require refreshing circuit
  - Used in cache memory
- Dynamic RAM
  - Uses capacitor to store information
  - More dense i.e. more cells can be accommodated per unit area
  - Slower, analog device
  - Less expensive, small in size
  - Needs refreshing circuit
  - Used in main memory, larger memory units

ROM—Read Only memory:
- Read only memory (ROM) contains a permanent pattern of data that cannot be changed.
- A ROM is non-volatile that is no power source is required to maintain the bit values in memory.
- While it is possible to read a ROM, it is not possible to write new data into it.
- The data or program is permanently presented in main memory and never be loaded from a secondary storage device with the advantage of ROM.
- A ROM is created like any other integrated circuit chip, with the data actually wired into the chip as part of the fabrication process.
- It presents two problems
  - The data insertion step includes a relatively large fixed cost, whether one or thousands of copies of a particular ROM are fabricated.
  - There is no room for error. If one bit is wrong, the whole batch of ROM must be thrown out.

Types of ROM
- Programmable ROM (PROM)
  - It is non-volatile and may be written into only once. The writing process is performed electrically and may be performed by a supplier or customer at a time later than the original chip fabrication.
- Erasable Programmable ROM (EPROM)
It is read and written electrically. However, before a write operation, all the storage cells must be erased to the same initial state by exposure of the packaged chip to ultraviolet radiation (UV ray). Erasure is performed by shining an intense ultraviolet light through a window that is designed into the memory chip. EPROM is optically managed and more expensive than PROM, but it has the advantage of the multiple update capability.

- Electrically Erasable programmable ROM (EEPROM)
  - This is a read mostly memory that can be written into at any time without erasing prior contents, only the byte or byte addresses are updated. The write operation takes considerably longer than the read operation, on the order of several hundred microseconds per byte. The EEPROM combines the advantage of non-volatility with the flexibility of being updatable in place, using ordinary bus control, addresses and data lines. EEPROM is more expensive than EPROM and also is less dense, supporting fewer bits per chip.

- Flash Memory
  - Flash memory is also the semiconductor memory and because of the speed with which it can be reprogrammed, it is termed as flash. It is interpreted between EPROM and EEPROM in both cost and functionality. Like EEPROM, flash memory uses an electrical erasing technology. An entire flash memory can be erased in one or a few seconds, which is much faster than EPROM. In addition, it is possible to erase just blocks of memory rather than an entire chip. However, flash memory doesn’t provide byte level erasure, a section of memory cells are erased in an action or 'flash'.

Now, we can assume RAM and ROM Chips sizes are

- RAM and ROM Chips
- Typical RAM chip, as shown in below figure.
- 128 X 8 RAM : \(2^7 = 128\) (7 bit address lines)
- Typical ROM chip, as shown in below figure.
- 512 X 8 ROM : \(2^9 = 512\) (9 bit address lines)

---

**Fig: RAM chip**
If we want to construct the Main memory size as 1024 x 8, then the Memory Configuration:

**512 bytes RAM + 512 bytes ROM**

1 x 512 byte ROM + 4 x 128 bytes RAM  

Memory Address Map can be showed as below:

<table>
<thead>
<tr>
<th>Component</th>
<th>Hexadecimal address</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM 1</td>
<td>0000–007F</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>RAM 2</td>
<td>0080–00FF</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>RAM 3</td>
<td>0100–017F</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>RAM 4</td>
<td>0180–01FF</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ROM</td>
<td>0200–03FF</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Memory Connection to CPU uses 2 x 4 Decoder: RAM select (CS1)

The circuit as shown below:

![Main memory diagram](image-url)
4. Associative Memory

Many data-processing applications require the search of items in a table stored in memory. An assembler program searches the symbol address table in order to extract the symbol's binary equivalent. An account number may be searched in a file to determine the holder's name and account status.

The established way to search a table is to store all items where they can be addressed in sequence. The search procedure is a strategy for choosing a sequence of addresses, reading the content of memory at each address, and comparing the information read with the item being searched until a match occurs.

The number of accesses to memory depends on the location of the item and the efficiency of the search algorithm. Many search algorithms have been developed to minimize the number of accesses while searching for an item in a random or sequential access memory.

The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address. A memory unit accessed by content is called content addressable an associative memory or content addressable memory (CAM).

This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location. When a word is written in an associative memory, no address is given. The memory is capable of finding an empty unused location to store the word. When a word is to be read from an associative memory, the content of the word, or part of the word, is specified. The memory locates all words which match the specified content and marks them for reading.

Because of its organization, the associative memory is uniquely suited to do parallel searches by data association. Moreover, searches can be done on an entire word or on a specific field within a word. An associative memory is more expensive than a random access memory because each cell must have storage capability as well as logic circuits for matching its content with an external argument.

**Note:** Associative memories are used in applications where the search time is very critical and must be very short.

**Hardware Organization:**

The block diagram of an associative memory is shown in below figure. It consists of a memory array and logic for m-words with n-bits per word. The argument register A and key register K each have n-bits, one for each bit of a word. The match register M has m-bits, one for each memory word. The words that match the bits of the argument register set a corresponding bit in the match register. After the matching process, those bits in the match register that have been set indicate the fact that their corresponding words have been matched.

Reading is accomplished by a sequential access to memory for those words whose corresponding bits in the match register have been set.

![Block diagram of Associative Memory](image-url)
To illustrate with a numerical example, suppose that the argument register A and the key register K have the bit configuration shown below. Only the three leftmost bits of A are compared with memory words because K has 1's in these positions.

A  101 111100
K  111 000000
Word 1 100 111100 no match
Word 2 101 000001 match

Word 2 matches the unmasked argument field because the three leftmost bits of the argument and the word are equal.

The relation between the memory array and external registers in an associative memory is shown in the below figure. The cells in the array are marked by the letter C with two subscripts. The first subscript gives the word number and the second specifies the bit position in the word. Thus cell Cij is the cell for bit j in word i. A bit Aj in the argument register is compared with all the bits in column j of the array provided that Kj = 1. This is done for all columns j = 1, 2, ..., n. If a match occurs between all the unmasked bits of the argument and the bits in word i, the corresponding bit Mi in the match register is set to 1. If one or more unmasked bits of the argument and the word do not match, Mi is cleared to 0.

**5. External Memory**

- The devices that provide backup storage are called external memory or auxiliary memory. It includes serial access type such as magnetic tapes and random access type such as magnetic disks.

**Magnetic Tape:**

- A magnetic tape is the strip of plastic coated with a magnetic recording medium. Data can be recorded and read as a sequence of character through read / write head. It can be stopped, started to move forward or in reverse or can be rewound.
- Data on tapes are structured as number of parallel tracks running length wise. Earlier tape system typically used nine tracks. This made it possible to store data one byte at a time with additional parity bit as 9th track. The recording of data in this form is referred to as parallel recording.

**Magnetic Disk:**

- A magnetic disk is a circular plate constructed with metal or plastic coated with magnetic material often both side of disk are used and several disk stacked on one spindle which Read/write head available on each surface. All disks rotate together at high speed. Bits are stored in magnetize surface in spots along concentric circles called tracks. The tracks are commonly divided into sections called sectors. After the read/write head are positioned in
specified track the system has to wait until the rotating disk reaches the specified sector under read/write head.

![Magnetic Disk](Image)

- Information transfer is very fast once the beginning of sector has been reached. Disk that are permanently attached to the unit assembly and cannot be used by occasional user are called hard disk drive with removal disk is called floppy disk.

### 6. Cache memory

**Principles:**

- Intended to give memory speed approaching that of fastest memories available but with large size, at close to price of slower memories
- Cache is checked first for all memory references.
- If not found, the entire block in which that reference resides in main memory is stored in a cache slot, called a line
- Each line includes a tag (usually a portion of the main memory address) which identifies which particular block is being stored
- Locality of reference implies that future references will likely come from this block of memory, so that cache line will probably be utilized repeatedly.
- The proportion of memory references, which are found already stored in cache, is called the hit ratio.
- Cache memory is intended to give memory speed approaching that of the fastest memories available, and at the same time provide a large memory size at the price of less expensive types of semiconductor memories. There is a relatively large and slow main memory together with a smaller, faster cache memory contains a copy of portions of main memory.
- When the processor attempts to read a word of memory, a check is made to determine if the word is in the cache. If so, the word is delivered to the processor. If not, a block of main memory, consisting of fixed number of words is read into the cache and then the word is delivered to the processor.
- The locality of reference property states that over a short interval of time, address generated by a typical program refers to a few localized area of memory repeatedly. So if programs and data which are accessed frequently are placed in a fast memory, the average access time can be reduced. This type of small, fast memory is called cache memory which is placed in between the CPU and the main memory.

![Positions of the Cache Memory](Image)
When the CPU needs to access memory, cache is examined. If the word is found in cache, it is read from the cache and if the word is not found in cache, main memory is accessed to read word. A block of word containing the one just accessed is then transferred from main memory to cache memory.

Cache connects to the processor via data control and address line. The data and address lines also attached to data and address buffer which attached to a system bus from which main memory is reached.

When a cache hit occurs, the data and address buffers are disabled and the communication is only between processor and cache with no system bus traffic. When a cache miss occurs, the desired word is first read into the cache and then transferred from cache to processor. For later case, the cache is physically interposed between the processor and main memory for all data, address and control lines.

1. CPU generates the receive address (RA) of a word to be moved (read).
2. Check a block containing RA is in cache.
3. If present, get from cache (fast) and return.
4. If not present, access and read required block from main memory to cache.
5. Allocate cache line for this new found block.
6. Load block for cache and deliver word to CPU

Cache includes tags to identify which block of main memory is in each cache slot

Locality of Reference:

- The reference to memory at any given interval of time tends to be confined within a few localized area of memory. This property is called locality of reference. This is possible because the program loops and subroutine calls are encountered frequently. When program loop is executed, the CPU will execute same portion of program repeatedly. Similarly, when a subroutine is called, the CPU fetched starting address of subroutine and executes the subroutine program. Thus loops and subroutine localize reference to memory.

- This principle states that memory references tend to cluster over a long period of time, the clusters in use changes but over a short period of time, the processor is primarily working with fixed clusters of memory references.

Spatial Locality:

- It refers to the tendency of execution to involve a number of memory locations that are clustered.
- It reflects tendency of a program to access data locations sequentially, such as when processing a table of data.

Temporal Locality:

- It refers to the tendency for a processor to access memory locations that have been used frequently. For e.g. Iteration loops executes same set of instructions repeatedly.

Cache Memory Mapping Functions:

- The transformation of data from main memory to cache memory is referred to as memory mapping process.
- Because there are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines.
- There are three different types of mapping functions in common use and are direct, associative and set associative.

Example of Cache and Main memory sizes are:

- Main memory: 32 K x 12 bit word (15 bit address lines)
- Cache memory: 512 x 12 bit word

  - CPU sends a 15-bit address to cache
  - Hit : CPU accepts the 12-bit data from cache
  - Miss : CPU reads the data from main memory (then data is written to cache).
1) Direct Mapping:

It is the simplex technique, maps each block of main memory into only one possible cache line i.e. a given main memory block can be placed in one and only one place on cache.

\[ i = j \mod m \], Where \( i = \) cache line number; \( j = \) main memory block number; \( m = \) number of lines in the cache

The mapping function is easily implemented using the address. For purposes of cache access, each main memory address can be viewed as consisting of three fields.

The least significant \( w \) bits identify a unique word or byte within a block of main memory. The remaining \( s \) bits specify one of the \( 2^s \) blocks of main memory.

The cache logic interprets these \( s \) bits as a tag of \((s-r)\) bits most significant position and a line field of \( r \) bits. The latter field identifies one of the \( m = 2^r \) lines of the cache.

![Fig: Addressing Relations b/w Main and Cache memory](image1)

![Fig: Direct Mapping](image2)
2) **Associative mapping:**

It overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of cache.

Cache control logic interprets a memory address simply as a tag and a word field. Tag uniquely identifies the block of memory.

Cache control logic must simultaneously examine every line’s tag for a match, which requires fully associative memory. Very complex circuitry, complexity increases exponentially with size cache searching gets expensive.

Address Data Cache memory

Tag field (n - k) and Index field (k)

\[ 2^k \] words cache memory + \( 2^n \) words main memory

Tag = 6 bit (15 - 9), Index = 9 bit

3) **Set-Associative Mapping:**

It is a compromise between direct and associative mappings that exhibits the strength and reduces the disadvantages.

Cache is divided into \( v \) sets, each of which has \( k \) lines; number of cache lines = \( v k \)

\[ M = v X k \]

\[ I = j \mod v \]

Where, \( i \) = cache set number; \( j \) = main memory block number; \( m \) = number of lines in the cache. So a given block will map directly to a particular set, but can occupy any line in that set (associative mapping is used within the set).

Cache control logic interprets a memory address simply as three fields, tag, set, and word. The \( d \) set bits specify one of \( v = 2^d \) sets. Thus \( s \) bits of tag and set fields specify one of the \( 2^s \) block of main memory.
The most common set associative mapping is 2 lines per set, and is called two-way set associative. It significantly improves hit ratio over direct mapping, and the associative hardware is not too expensive.

<table>
<thead>
<tr>
<th>Index</th>
<th>Tag</th>
<th>Data</th>
<th>Tag</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>01</td>
<td>3450</td>
<td>02</td>
<td>5670</td>
</tr>
<tr>
<td>777</td>
<td>02</td>
<td>6710</td>
<td>00</td>
<td>2340</td>
</tr>
</tbody>
</table>

Fig: Set-Associative Memory Mapping

7. Virtual Memory

In a memory hierarchy system, programs and data are first stored in auxiliary memory. Portions of a program or data are brought into main memory as they are needed by the CPU. Virtual memory is a concept used in some large computer systems that permit the user to construct programs as though a large memory space were available, equal to the totality of auxiliary memory.

Each address that is referenced by the CPU goes through an address mapping from the so-called virtual address to a physical address in main memory.

A virtual memory system provides a mechanism for translating program-generated addresses into correct main memory locations.

This is done dynamically, while programs are being executed in the CPU. The translation or mapping is handled automatically by the hardware by means of a mapping table.

Address space memory space:

An address used by a programmer will be called a virtual address, and the set of such addresses the address space. An address in main memory is called a location or physical address. The set of such locations is called the memory space.

In most computers the address and memory spaces are identical. The address space is allowed to be larger than the memory space in computers with virtual memory.

As an illustration, consider a computer with a main-memory capacity of 32K words (K = 1024). Fifteen bits are needed to specify a physical address in memory since 32K = 2^{15}. Suppose that the computer has available auxiliary memory for storing 2^{20} = 1024K words. Thus auxiliary memory has a capacity for storing information equivalent to the capacity of 32 main memories. Denoting the address space by N and the memory space by M, we then have for this example N = 1024K and M = 32K.

Program 1 and a portion of its associated data are moved from auxiliary memory into main memory as shown in the below figure. Portions of programs and data need not be in contiguous locations in memory since information is being moved in and out, and empty spaces may be available in scattered locations.

In our example, the address field of an instruction code will consist of 20 bits but physical memory addresses must be specified with only 15 bits. Thus CPU will reference instructions and data with a 20-bit address, but the information at this address must be taken from physical memory because access to auxiliary storage for individual words will be prohibitively long. (Remember that for efficient transfers, auxiliary storage moves an entire record to the main memory.)
A table is then needed, as shown in below, to map a virtual address of 20 bits to a physical address of 15 bits. The mapping is a dynamic operation, which means that every address is translated immediately as a word is referenced by CPU.

The mapping table may be stored in a separate memory as shown in below figure or in main memory. In the first case, an additional memory unit is required as well as one extra memory access time. In the second case, the table takes space from main memory and two accesses to memory are required with the program running at half speed. A third alternative is to use an associative memory as explained below.

Address Mapping Using Pages:
The table implementation of the address mapping is simplified if the information in the address space and the memory space are each divided into groups of fixed size. The physical memory is broken down into groups of equal size called blocks, which may range from 64 to 4096 words each. The term page refers to groups of address space of the same size.

For example, if a page or block consists of 1K words, then, using the previous example, address space is divided into 1024 pages and main memory is divided into 32 blocks. Although both a page and a block are split into groups of 1K words.

Note: The programs are also considered to be split into pages.

Consider a computer with an address space of 8K and a memory space of 4K. If we split each into groups of 1K words we obtain eight pages and four blocks as shown in below figure. At any given time, up to four pages of address space may reside in main memory in anyone of the four blocks.

Blog - https://anilkumarprathipati.wordpress.com/  
NEC (Autonomous)
The mapping from address space to memory space is facilitated if each virtual address is considered to be represented by two numbers: a page number - address and a line within the page. In a computer with $2^p$ words per page, $p$ bits are used to specify a line address and the remaining high-order bits of the virtual address specify the page number.

In the example of below figure, a virtual address has 13 bits. Since each page consists of $2^{10} = 1024$ words, the high order three bits of a virtual address will specify one of the eight pages and the low-order 10 bits give the line address within the page.

**Note:** The line address in address space and memory space is the same. So, the only mapping required is from a page number to a block number.

![Address Space and Memory Space split into groups of 1K words](https://anilkumarprathipati.wordpress.com/)

The organization of the memory mapping table in a paged system is shown in figure. The memory-page table consists of eight words, one for each page. The address in the page table denotes the page number and the content of the word gives the block number where that page is stored in main memory. The table shows that pages 1, 2, 5, and 6 are now available in main memory in blocks 3, 0, 1, and 2, respectively. A presence bit in each location indicates whether the page has been transferred from auxiliary memory into main memory. A ‘0’ in the presence bit indicates that this page is not available in main memory. If the presence bit in the word read from the page table is 0, it signifies that the content of the word referenced by the virtual address does not reside in main memory.

![Memory Table in Page System](https://anilkumarprathipati.wordpress.com/)

Blog - https://anilkumarprathipati.wordpress.com/
UNIT-IV

COMPUTER ARCHITECTURE AND ORGANIZATION

8. Replacement algorithm

When all lines are occupied, bringing in a new block requires that an existing line be overwritten.

Direct mapping

- No choice possible with direct mapping
- Each block only maps to one line
- Replace that line

Associative and Set Associative mapping

Algorithms must be implemented in hardware for speed

1) Least Recently used (LRU):
   Replace that block in the set which has been in cache longest with no reference to it
   Implementation: with 2-way set associative, have a USE bit for each line in a set. When a block
   is read into cache, use the line whose USE bit is set to 0, then set its USE bit to one and the
   other line’s USE bit to 0. Probably the most effective method

2) First in first out (FIFO):
   Replace that block in the set which has been in the cache longest Implementation: use a round-
   robin or circular buffer technique (keep up with which slot’s “turn” is next.

3) Least-frequently-used (LFU):
   Replace that block in the set which has experienced the fewest references or hits
   Implementation: associate a counter with each slot and increment when used

4) Random:
   Replace a random block in the set Interesting because it is only slightly inferior to
   algorithms based on usage

9. Write policy

When a line is to be replaced, must update the original copy of the line in main memory if
any addressable unit in the line has been changed.

If a block has been altered in cache, it is necessary to write it back out to main memory
before replacing it with another block (writes are about 15% of memory references).

Must not overwrite a cache block unless main memory is up to date

Write Through:

All write operations are made to main memory as well as to cache, so main memory is
always valid, other CPU’s monitor traffic to main memory to update their caches when needed
This generates substantial memory traffic and may create a bottleneck. Anytime a word in cache
is changed, it is also changed in main memory. Both copies always agree and generate lots of
memory writes to main memory.

Multiple CPUs can monitor main memory traffic to keep local (to CPU) cache up to date.
Lots of traffic slows down writes remember bogus write through caches.

Write back:

When an update occurs, an UPDATE bit associated with that slot is set, so when the block is
replaced it is written back first. During a write, only change the contents of the cache. Update
main memory only when the cache line is to be replaced.

Causes “cache coherency” problems - different values for the contents of an address are in
the cache and the main memory. Complex circuitry to avoid this problem and accesses by I/O
modules must occur through the cache.

Multiple caches still can become invalidated, unless some cache coherency system is used.
Such systems include:

- Bus Watching with Write Through - other caches monitor memory writes by other
  caches (using write through) and invalidates their own cache line if a match.
- Hardware Transparency - additional hardware links multiple caches so that writes to one
  cache are made to the others.
- Non-cacheable Memory - only a portion of main memory is shared by more than one
  processor, and it is non-cacheable.