Unit 8
Memory Organization

Introduction

Memory unit is an essential component in any general purpose computer since it is needed to store programs and data. The memory unit that communicates directly with the CPU is called the **main memory** and devices that provide backup storage are called **auxiliary memory**.

NOTE: Auxiliary memory devices such as magnetic disk and tapes are used to store system programs, large data files and other backup information. Only programs and data currently needed by the processor reside in main memory. All other information is stored in main memory and transferred to main memory when needed.

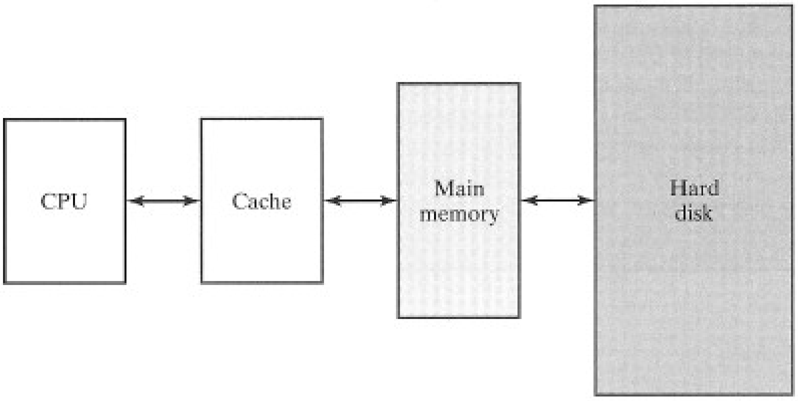
Memory Types

- **Sequential Access Memory (SAM)**: In computing, **SAM** is a class of data storage devices that read their data in sequence. This is in contrast to random access memory (RAM) where data can be accessed in any order. Sequential access devices are usually a form of magnetic memory. Magnetic sequential access memory is typically used for secondary storage in general-purpose computers due to their higher density at lower cost compared to RAM, as well as resistance to wear and non-volatility. Examples of SAM devices still in use include hard disks, CD-ROMs and magnetic tapes. Historically, drum memory has also been used.

- **Random Access Memory (RAM)**: **RAM** is a form of computer data storage. Today, it takes the form of integrated circuits that allow stored data to be accessed in any order with a worst case performance of constant time. Strictly speaking, modern types of DRAM are therefore not random access, as data is read in bursts, although the name DRAM / RAM has stuck. However, many types of SRAM, ROM and NOR flash are still random access even in a strict sense. RAM is often associated with volatile types of memory, where its stored information is lost if the power is removed. The first RAM modules to come into the market were created in 1951 and were sold until the late 1960s and early 1970s.

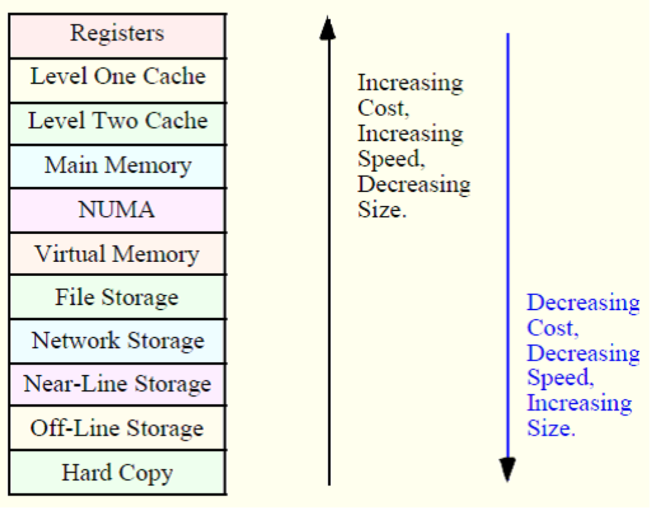
Memory hierarchy

Block diagram below shows the generic memory hierarchy.



Talking roughly, **lowest level** of hierarchy is small, fast memory called *cache* where anticipated CPU instructions and data resides. At the **next level** upward in the hierarchy is *main memory*. The main memory serves CPU instruction fetches not satisfied by cache. At the **top level** of the hierarchy is the *hard disk* which is accessed rarely only when CPU instruction fetch is not found even in main memory.

Example: Memory hierarchy in Intel 80x86 processor family:



- At the top level of the memory hierarchy are the CPU's general purpose registers.
- Level One on-chip Cache system is the next highest performance subsystem.
- Next is expandable level two cache
- Next, main memory (usually DRAM) subsystem. Relatively low-cost memory found in most computer systems.
- Next is NUMA (Non Uniform Memory Access)
- Next is Virtual Memory scheme that simulate main memory using storage on a disk drive.
- Next in hierarchy is File Storage which uses disk media to store program data.
- Below is Network Storage stores data in distributed fashion.
- Near-Line Storage uses the same media as Off-Line Storage; the difference is that the system holds the media in a special robotic jukebox device that can automatically mount the desired media when some program requests it.
- Next is Off-Line Storage that includes magnetic tapes, disk cartridges, optical disks, floppy diskettes and USBs.
- Hardcopy..! I don't think I have to explain it.

Primary and Secondary Memory

Primary (Main) Memory

It is a relatively large and fast memory used to store programs and data during the computer operation. Semiconductor integrated circuit is the principle technology used for main memory.

Random Access Memory (RAM): RAM chips are available in two possible modes, *static* and *dynamic*.

Static RAM: consists of internal flip-flops to store binary information. It is easier to use and has shorter read/write cycles.

Dynamic RAM: stores binary information in the form of electric charges in capacitors. The stored charge tends to discharge with time, so DRAM words are refreshed every few milliseconds to restore the decaying charge. DRAM offers reduced power consumption and larger storage capacity in a single memory chip.

Read-Only Memory (ROM): Random access ROM chips are used for storing programs that are permanently resident in computer and for tables of constants that do not change once computer is manufactured. The contents of ROM remain unchanged after power is turned off and on again.

Bootstrap loader: It is initial program whose function is to start the computer operating system when power is turned on and is stored in ROM portion of the main memory.

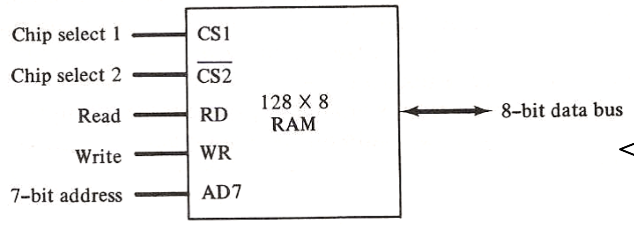
Computer startup: The startup of a computer consists of turning the power on and starting the execution of an initial program. Thus when power is turned on, the hardware of the computer sets the PC to the first address of the bootstrap loader. The bootstrap program loads the portion of the OS from the disk to main memory and control is then transferred to the OS, which prepares the computer for general use.

RAM and ROM Chips

RAM and ROM chips are available in a variety of sizes. If we larger memory for the system, it is necessary to combine a number of chips to form the required memory size.

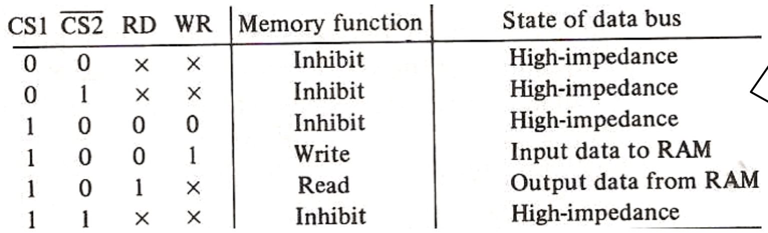
RAM Chips

A RAM chip is better suited to communicate with CPU if it has one or more control inputs that select the chip only when needed. The block diagram of a RAM chip is shown below:



- ➔ Requires 7-bit address and an 8-bit bidirectional data bus
- ➔ Chip select (CS) control inputs are for enabling the chip only when it is selected by CPU.

Fig: Typical RAM chip (128 words of eight bits each)



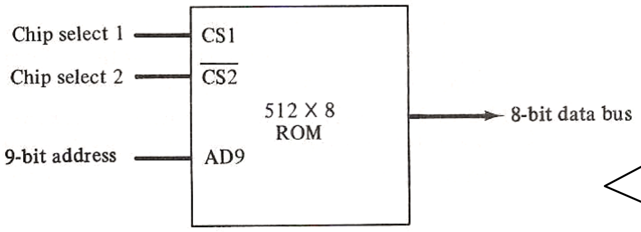
CS1	$\overline{CS2}$	RD	WR	Memory function	State of data bus
0	0	×	×	Inhibit	High-impedance
0	1	×	×	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	×	Read	Output data from RAM
1	1	×	×	Inhibit	High-impedance

- ➔ The unit is in operation only when CS1=1 and (CS2)'=0.
- ➔ High impedance state indicates open circuit i.e. output does not carry a signal and has no logic significance.

Fig: Function table for RAM chip

ROM Chips

Since a ROM chip can only read, data bus is unidirectional (output mode only).



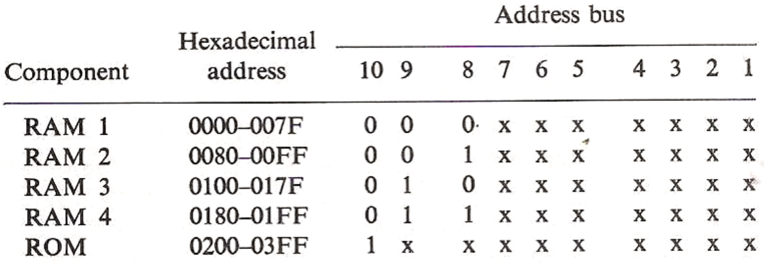
- ➔ 9 address lines to address 512 bytes
- ➔ Two chip select (CS) inputs CS1=1 and (CS2)'=0 for the unit to operate, otherwise the data bus is in high-impedance state.
- ➔ No need for read or write control since the unit can only read.

Fig: Typical ROM chip (512 byte ROM)

Memory Address Map

The addressing of memory can be established by means of a table that specifies the memory address assigned to each RAM or ROM chip. This table is called memory address map and is a pictorial representation of assigned address space for particular chip.

Example: Suppose computer system needs 512 bytes of RAM and 512 bytes of ROM.

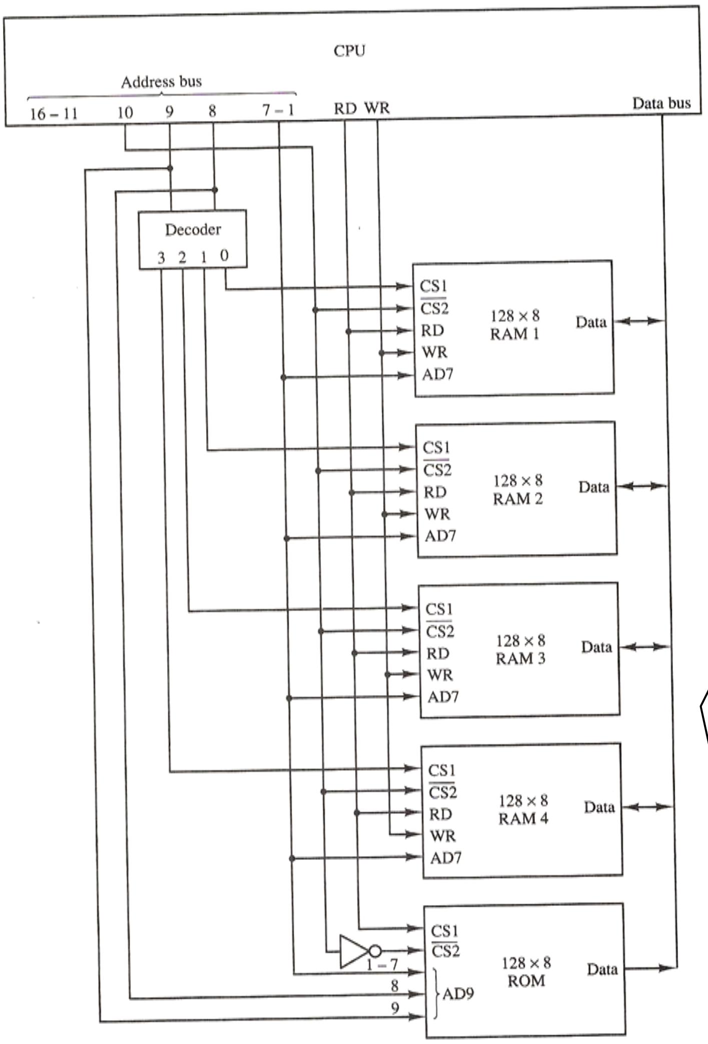


Component	Hexadecimal address	Address bus									
		10	9	8	7	6	5	4	3	2	1
RAM 1	0000–007F	0	0	0	x	x	x	x	x	x	x
RAM 2	0080–00FF	0	0	1	x	x	x	x	x	x	x
RAM 3	0100–017F	0	1	0	x	x	x	x	x	x	x
RAM 4	0180–01FF	0	1	1	x	x	x	x	x	x	x
ROM	0200–03FF	1	x	x	x	x	x	x	x	x	x

- ➔ Component column specifies RAM or ROM chip. We use four 128 words RAM to make 512 byte size.
- ➔ Hexadecimal address column assigns a range of addresses for each chip.
- ➔ 10 lines in address bus column: lines 1 through 7 for RAM and 1 through 9 for ROM. Distinction between RAM and ROM chip is made by line 10. When line 10 is 1, it selects ROM and when it is 0, CPU selects RAM.
- ➔ X's represents a binary number ranging from all-0's to all-1's.

Memory-CPU Connection

RAM and ROM chips are connected to CPU through data and address buses.



Example gives an indication of the interconnection complexity that can exist between memory chips and CPU. More the chips, more external decoders are required for selection among the chips.

- ➔ This configuration gives 512 bytes of RAM and 5112 bytes of ROM
- ➔ Each RAM receives 7 low-order bits of the address bus to select a byte.
- ➔ RAM chips are selected with decoder with selection input of line 8 and 9.
- ➔ The selection between RAM and ROM is done by line 10. When 0, RAMs are selected and when 1 ROM get selected.

Auxiliary (Secondary) Memory

The most common auxiliary memory devices used in computer systems are **magnetic disks**, **magnetic tapes** and **optical disks**. To understand fully the physical mechanism of auxiliary memory devices, we should have knowledge of magnetics, electronics and electromechanical systems.

HEY! Read yourself about these three devices… I hope u guys have studied in your OS course.

Virtual Memory

- A virtual memory system attempts to optimize the use of the main memory (the higher speed portion) with the hard disk (the lower speed portion). In effect, **virtual memory** is a technique for using the secondary storage to extend the apparent limited size of the physical memory beyond its actual physical size. It is usually the case that the available physical memory space will not be enough to host all the parts of a given active program.
- Virtual memory gives programmers the illusion that they have a very large memory and provides mechanism for dynamically translating program-generated addresses into correct main memory locations. The translation or mapping is handled automatically by the hardware by means of a mapping table.

Address space and Memory Space

An address used by the programmer is a virtual address (virtual memory addresses) and the set of such addresses is 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**. Thus the address space is the set of addresses generated by the programs as they reference instructions and data; the memory space consists of actual main memory locations directly addressable for processing. Generally, the address space is larger than the memory space.

Example: consider main memory: 32K words (K = 1024) = 2^{15} and auxiliary memory 1024K words = 2^{20}. Thus we need 15 bits to address physical memory and 20 bits for virtual memory (virtual memory can be as large as we have auxiliary storage).

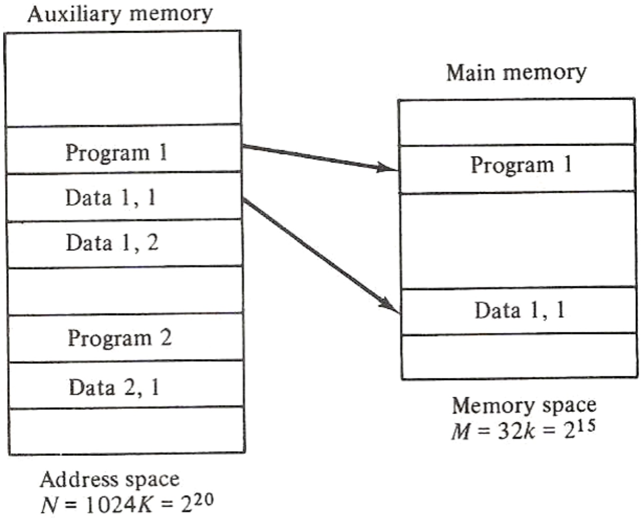


Fig: Relation between address and memory space in a virtual memory system

- ➔ Here auxiliary memory has the capacity of storing information equivalent to 32 main memories.
- ➔ Address space N = 1024K
- ➔ Memory space M = 32K
- ➔ In multiprogram computer system, programs and data are transferred to and from auxiliary memory and main memory based on the demands imposed by the CPU.

In virtual memory system, address field of an instruction code has a sufficient number of bits to specify all virtual addresses. In our example above we have 20-bit address of an instruction (to refer 20-bit virtual address) but physical memory addresses are specified with 15-bits. So a table is needed to map a virtual address of 20-bits to a physical address of 15-bits. Mapping is a dynamic operation, which means that every address is translated immediately as a word is referenced by CPU.

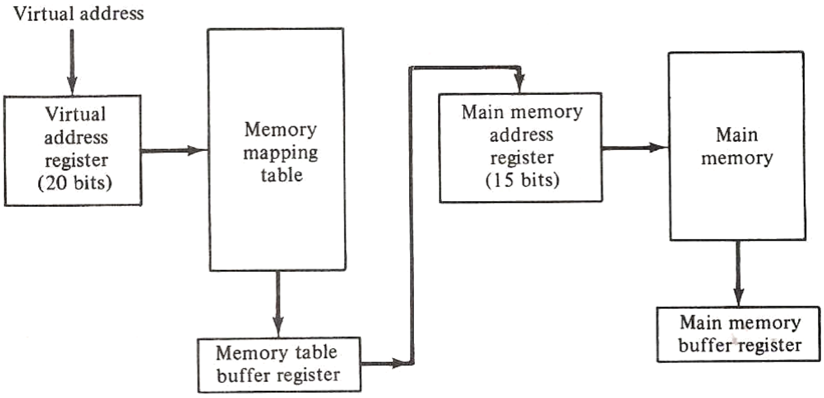

Virtual address
Virtual address register (20 bits)
Memory mapping table
Memory table buffer register
Main memory address register (15 bits)
Main memory
Main memory buffer register

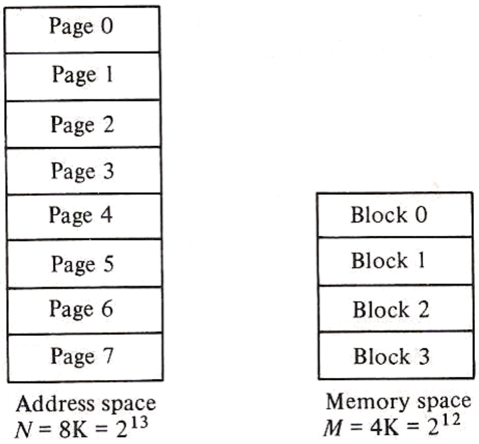
Fig: Memory table for mapping a virtual address

Address Mapping using Pages

Above memory table implementation of address mapping is simplified if the information in address space and memory space are each divided into groups of fixed size.

Blocks (or page frame): The physical memory is broken down into groups of equal size called blocks, which may range from 64 to 4096 words each.
Pages: refers to a portion of subdivided virtual memory having same size as blocks i.e. groups of address space.

Example: consider computer with address space = 8K and memory space = 4K.



- ➔ If we spit both spaces into groups of 1K words, we obtain 8 pages and 4 blocks.
- ➔ Virtual address has 13 bits. Since each page consists of 210 = 1024 words, high-order 3 bits will specify one of 8 pages and low-order 10 bits give the line address within the pages.

The mapping from address space to memory space becomes easy if virtual address is represented by two numbers: a page number address and a line with in the page. In a computer with 2^p words per page, p bits are used to specify a **line address** and remaining high-order bits of the virtual address specify the **page number**.

NOTE: line address in address space and memory space is same; only mapping required is from page number to a block number.

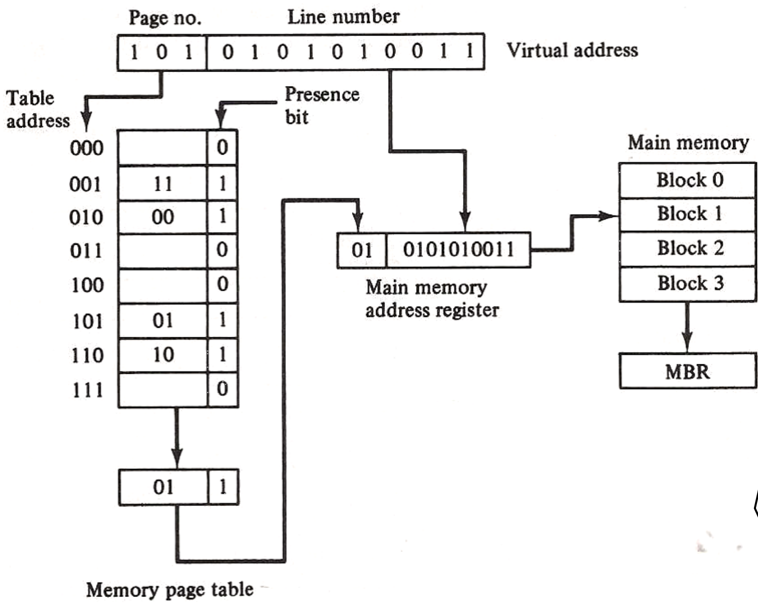
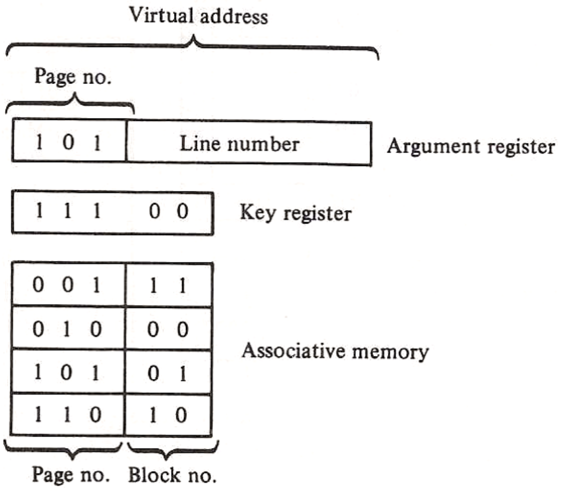


Fig: Memory table in paged system

- The memory-page table consists of 8 words, one for each page.
- The address in the page table denotes page number and the content of the word gives the block number where the page is stored in main memory.
- Presence bit when 0 indicates page is not available in main memory and when 1 says that page has been transferred to main memory.
- Table shows that pages 1,2,5 and 6 are now available in main memory in blocks 3,0,1 and 2 respectively.

Associative Memory Page table

- In above figure, we use random-access page table which is inefficient with respect to storage utilization. For example: consider address space = 1024K words and memory space = 32K words. If each page or block contains 1K words, the number of pages is 1024 and number of blocks 32. The capacity of the memory page table must be 1024 words and only 32 locations have presence bit equal to 1. At any given time, at least 992 locations will be empty and not in use.
- What about making page table with number of words equal to the number of blocks in main memory? Obviously this is an efficient approach since size of memory is reduced and each location is fully utilized.
- This method can be implemented by means of an **associative memory** in which each word in memory containing a page number with its corresponding block number.



- The page field in each **associative memory table** word is compared with page number bits in an argument register (which contains page number in the virtual address), if match occurs, the word is read form memory and its corresponding block number is extracted.
- This is the associative page table for same example in previous section where we were using random-access page table.

Page Replacement

A virtual memory system is a combination of hardware and software techniques. A memory management software system handles:

1. Which page in main memory ought to be removed to make room for a new page?
2. When a new page is to be transferred from auxiliary memory to main memory?

3. Where the page is to be placed in main memory?

Mechanism: when a program starts execution, one or more pages are transferred into main memory and the page table is set to indicate their position. The program is executed from main memory until it attempts to reference a page that is still in auxiliary memory. This condition is called **page fault**. When page fault occurs, the execution of the present program is suspended until required page is brought into memory. Since loading a page from auxiliary memory to main memory is basically an I/O operation, OS assigns this task to I/O processor. In the mean time, control is transferred to the next program in memory that is waiting to be processed in the CPU. Later, when memory block has been assigned, the original program can resume its operation.

When a **page fault occurs** in a virtual memory system, it signifies that the page referenced by the program is not in main memory. A new page is then transferred from auxiliary memory to main memory. If main memory is full, it would be necessary to remove a page from a memory block to make a room for a new page. The **policy for choosing pages to remove is determined from the replacement algorithm** that is used.

GOAL: try to remove the page least likely to be referenced by in the immediate future.

There are numerous page replacement algorithms, two of which are:

- First-in First-out (FIFO): replaces a page that has been in memory longest time.
- Least Recently Used (LRU): assumes that least recently used page is the better candidate for removal than the least recently loaded page.

Memory Management Hardware

A memory management system is a collection of hardware and software procedures for managing various programs (effect of multiprogramming support) residing in memory. Basic components of memory management unit (MMU) are:

- A facility for dynamic storage relocation that maps logical memory references into physical memory addresses.
- A provision for sharing common programs by multiple users
- Protection of information against unauthorized access.

The dynamic storage relocation hardware is a mapping process similar to paging system.

Segment: It is more convenient to divide programs and data into logical parts called segments despite of fixed-size pages. A **segment** is a set of logically related instructions or data elements. Segments may be generated by the programmer or by OS. Examples are: a subroutine, an array of data, a table of symbols or user's program.

Logical address: The address generated by the segmented program is called a *logical address*. This is similar to virtual address except that logical address space is associated with variable-length segments rather than fixed-length pages.

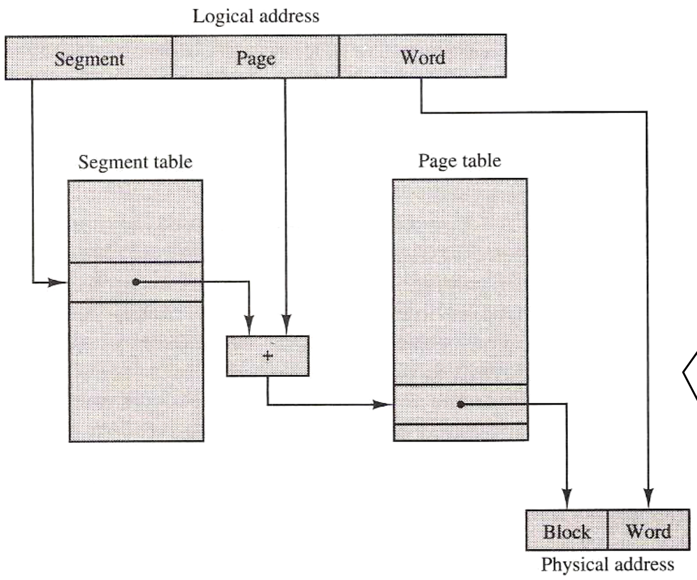
Segmented-Page Mapping

The length of each segment is allowed to grow and contract according to the needs of the program being executed. One way of specifying the length of a segment is by associating with it a number of equal-sized pages.

Consider diagram below:

Logical address = Segment + page + Word

Where **segment** specifies segment number, **page** field specifies page with in the segment and **word** field specifies specific word within the page.



- → Mapping of logical address to physical address is done by using two tables: segment and page table.
- → The entry in the segment table is a pointer address for the page table base, which is then added to page number (given in logical address). The sum point to some entry in page table and content of that page is the address of physical block. The concatenation of block field with the word field produces final **physical mapped address**.

Fig: Logical to physical address mapping

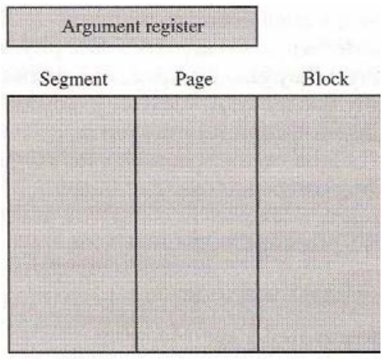


Fig: Associative memory Translation Look-aside Buffer (TLB)

- → This is a fast associative memory (TLB) and holds most recently referenced entries.

(Alternatively we could store above two tables: segment table and page table, in two separate small memories which really increases the CPU access time)

HEY..! *See Numerical example to clear the concept of MMU (page no. 497, Morris Mano 3rd edition Computer System Architecture)*

Memory Protection

- → Memory protection is concerned with protecting one program from unwanted interaction with another and preventing the occasional user performing OS functions.
- → Memory protection can be assigned to the physical address or the logical address.
  - o <u>Through physical address</u>: assign each block in memory a number of protection bits.
  - o <u>Through logical address</u>: better idea is to apply protection bits in logical address and can be done by including protection information within the segment table or segment register.

Base Address	Length	Protection

Fig: format of typical segment descriptor

Where

- ▪ **Base address field** gives the base of the page table address in segmented-page organization.

- **Length field** gives the segment size (in number of pages)
- The **protection field** specifies access rights available to a particular segment. The protection information is set into the descriptor by the master control program of the OS. Some of the access rights are:
    - Full read and write privileges
    - Read only (Write protection)
    - Execute only (Program protection)
    - System only (OS protection)