Paper Name : Operating System

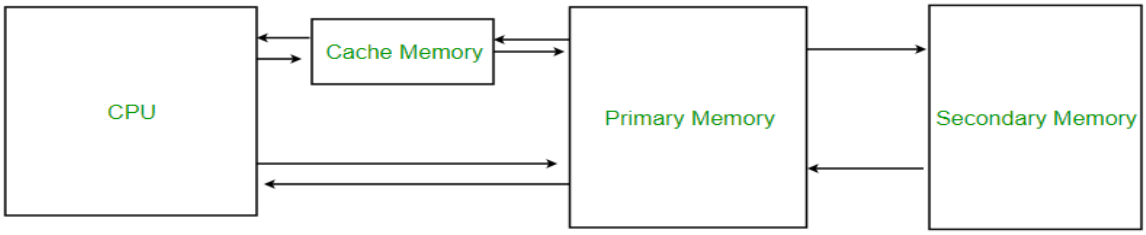
Prepared by : Debanjali Jana

Memory Management :

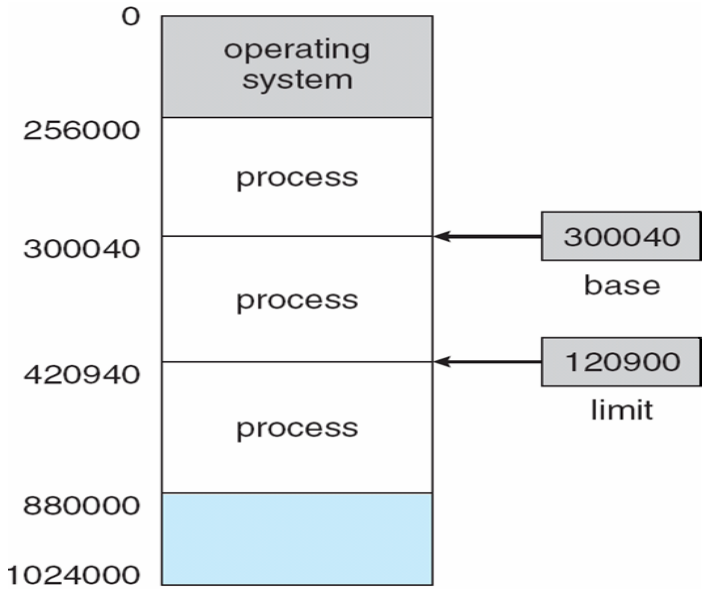
The main motivation for management of main memory comes from the support for multiprogramming. Several executables processes reside in main memory at any given time. In other words, there are several programs using the main memory as their address space. Also, programs move into, and out of, the main memory as they terminate, or get suspended for some IO, or new executables are required to be loaded in main memory. So, the OS has to have some strategy for main memory management. In this chapter we shall discuss the management issues and strategies for both main memory and secondary memory.

- Program must be brought (from disk) into memory and placed within a process for it to be run
- Main memory and registers built into the processor are only storage CPU can access directly

  Therefore, any instructions in execution, and any data being used by the instructions, must be in one of these direct-access storage devices. If the data are not in memory, they must be moved there before the CPU can operate on them.
- Memory unit only sees a stream of addresses + read requests, or address + data and write requests
- Register access in one CPU clock (or less)
- Main memory can take many cycles, causing a stall
- Fast memory cache sits between main memory and CPU registers
- Protection of memory required to ensure correct operation



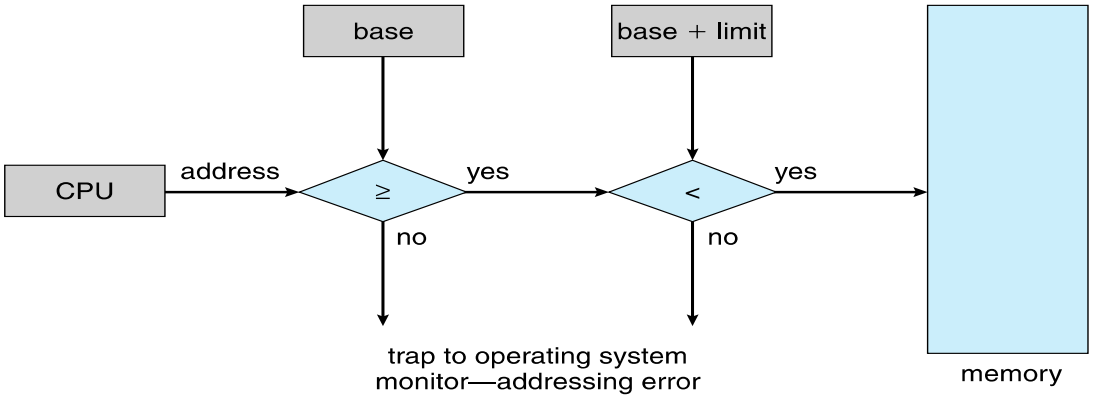
Base and Limit Registers



- Base register - holds the value of smallest physical memory address
- Limit register - specifies the size of the range.
  For example, if the base register holds 300040 and the limit register is 120900, then the program can legally access all addresses from 300040 through 420939 (inclusive).

Hardware Address Protection:

Protection of memory space is accomplished by having the CPU hardware compare every address generated in user mode with the registers. Any attempt by a program executing in user mode to access operating-system memory or other users' memory results in a trap to the operating system, which treats the attempt as a fatal error (Figure 8.2). This scheme prevents a user program from (accidentally or deliberately) modifying the code or data structures of either the operating system or other users.
The base and limit registers can be loaded only by the operating system, which uses a special privileged instruction. Since privileged instructions can be executed only in kernel mode, and since only the operating system executes in kernel mode, only the operating system can load the base and limit registers. This scheme allows the operating system to change the value of the registers but prevents user programs from changing the registers' contents.



Address binding of instructions and data to memory addresses can happen at three different stages

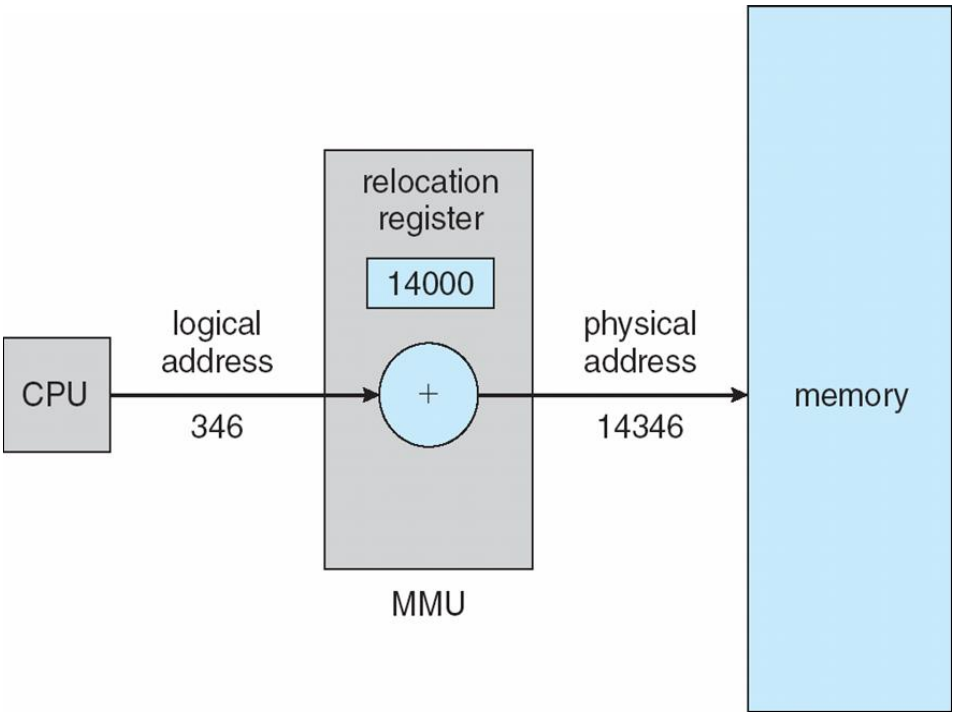
- Compile time: If you know at compile time where the process will reside in memory, then absolute address can be generated. For example, if you know that a user process will reside starting at location R, then the generated compiler code will start at that location and extend up from there. If, at some later time, the starting location changes, then it will be necessary to recompile this code.
- Load time: If it is not known at compile time where the process will reside in memory, then the compiler must generate relocatable address. In this case, final binding is delayed until load time. If the starting address changes, we need only reload the user code to incorporate this changed value.
- Execution time: If the process can be moved during its execution from one memory segment to another, then binding must be delayed until run time.
  - Need hardware support for address maps (e.g., base and limit registers)

Logical versus Physical Address Space

- **Logical address** -An address generated by the CPU is commonly referred to as a logical address. The logical address is virtual address as it does not exist physically, therefore, it is also known as Virtual Address. This address is used as a reference to access the physical memory location by CPU. The hardware device called Memory-Management Unit is used for mapping logical address to its corresponding physical address.
- **Physical address**- An address seen by the memory unit-that is, the one loaded into the of the memory-is commonly referred to as a physical address. Physical Address identifies a physical location of required data in a memory.
- **Logical address space** - Set of all logical addresses generated by a program
- **Physical address space** - Set of all physical addresses generated by a program

Thus the logical address does not exist physically in the memory whereas physical address is a location in the memory that can be accessed physically.

The run-time mapping from virtual to physical addresses is done by a hardware device called the Memory-Management Unit (MMU).



The base register is now called a relocation register. The value in the relocation register is added to every address generated by a user process at the time the address is sent to memory (see Figure above). For example, if the base is at 14000, then an attempt by the user to address location 0 is dynamically relocated to location 14000; an access to location 346 is mapped to location 14346.