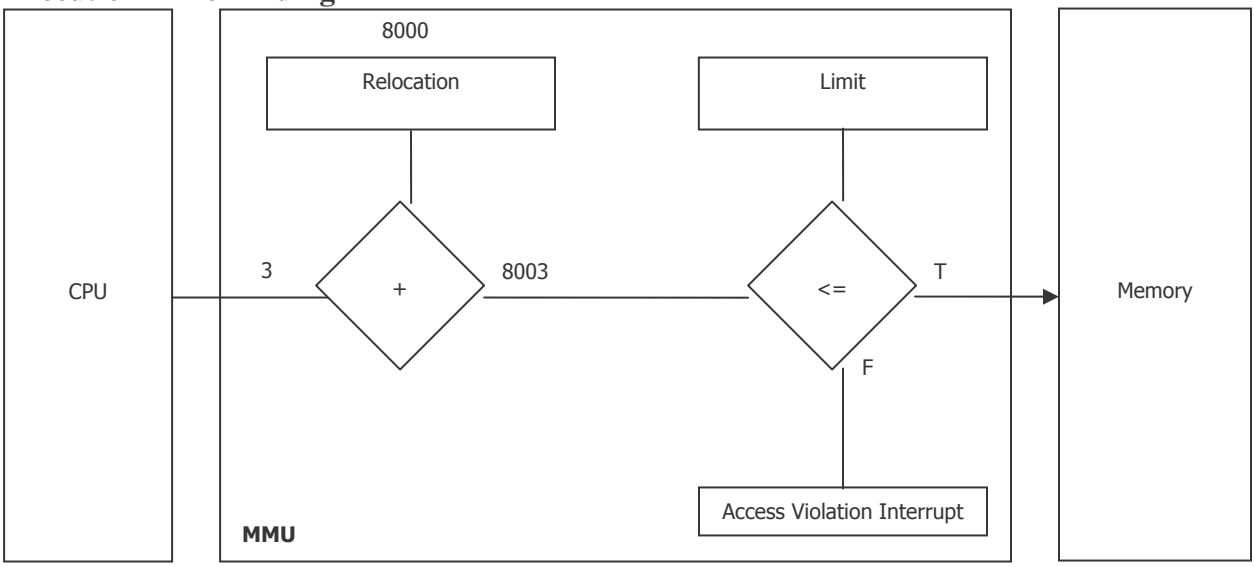
Address Binding

The addresses used in load time and execution time binding are logical addresses or virtual addresses.

In execution time, addresses are created dynamically when every instruction is fetched from memory.

Execution Time Binding



How do we actually remove routines in and out of memory?

Static Loading

All routines become part of the code and are loaded at the same time, regardless of whether you call them
This leads to bloated images of the software in memory

This was the norm until recently.

Dynamic Loading

The OS loads the routine when it is called.

Slight performance hit – when the subroutine has to be loaded, there is a stall.

Static Libraries

These libraries are humongous. The library code becomes part of your application code and is loaded every time the application is loaded.

Dynamically Linked Libraries

Every place in your code that you call a subroutine from an available dynamically linked library, what is actually compiled into your code is what we call a **stub**. The short little stub is just a few instructions (system call) that request that the subroutine be loaded from the operating system.

The appropriate subroutine number from the library is included in the code.

They are the standard way of doing business and help to avoid bloat.

Benefits:

- Memory bloat issue is being addressed
- If you compile your program using static libraries and the manufacturer changes the API, you have to recompile your program. With dynamically linked libraries, the names of the functions don't change, but the function names and interfaces can.
- Reuse

During the dynamic load, the OS creates a table with all the routines currently in memory. So, if two different processes call a routine, it does not need to be reloaded, but is just executed.

Swapping Mechanism (Medium-term scheduler)

- Relies on having a backing store (magnetic disk)
- Swap out of processes to swap file
- Should be avoided if at all possible, because hard drive access time is not as fast as memory

Memory Allocation to a Loading Process

Contiguous

- Process is given one linear range of memory

Noncontiguous

- Process is given multiple ranges of memory
- Fragmentation
- Paging and Segmentation