Memory Management

The CPU views memory as uniquely addressable locations.

The CPU generates an address to memory but never generates an address to I/O devices. The generation of I/O addresses is left to the I/O device controller.

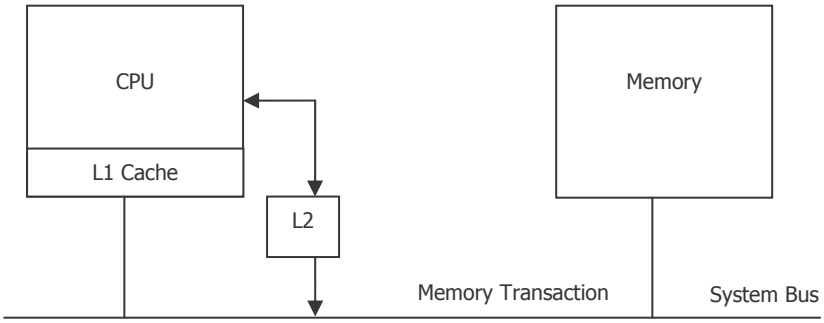
A register access is completed in one clock cycle.
HC11: ~ 3 clock cycles

Main memory access time is still much much slower than registers.

Memory Stall

Occurs when the CPU attempts to access a data variable that isn't in the registers.
Requires that the CPU go get that from memory. That memory access stalls the execution of the program.

In classic computers, memory access times are on the order of hundreds of nanoseconds. Now, we're on the order of 5 or 10 ns.



Cache Memories

Buffer memory holding recently used data / instructions. It is static ram and therefore requires less time to access. The memory penalty is not as severe.

Increasing number of on-CPU registers also helps to mitigate the issue of memory stalling

Types of cache memories:

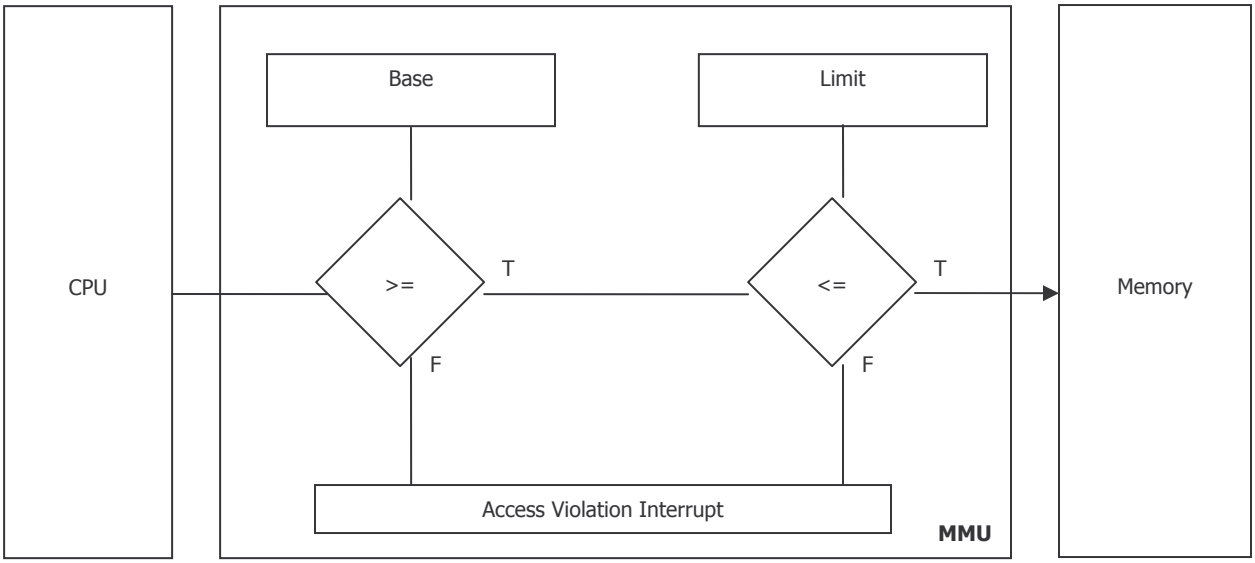
- **L1 Cache**
  - On the CPU. There is no pin connection penalty.
- **L2 Cache**

The goal of the modern OS

To have separate, private, protected images / regions of memory for each process. This is accomplished with the MMU.

Memory Management Unit (MMU)

Hardware devices that support protected memory images



Only if both conditions are true (address is in the valid range), memory is accessed. If either is false, an **access violation** occurs and an access violation interrupt occurs.

In older versions of Microsoft Windows, this would be handled with a blue screen of death.

Address Binding

Assigning addresses to the program.

Classic Binding Techniques:

1. Compile Time Binding
    a. All addresses (not just the code, but also the variables) are fixed by the compiler at compile time and cannot be changed. This implies that the application must be loaded into that exact memory location.
    b. This is a very limited binding technique, but it is the original way things were done.
    c. Example: MSDOS.com
2. Load Time Binding
    a. Makes a small improvement on compile time binding. The compiler generates relocatable / relative addresses. The address are relocated / bound at load time by the loader by adding the base.
3. Execution Time Binding

a. Compiler generates relocatable code, but addresses are bound as instructions run. (MMU) This allows the program to be sent to disk.

Compile	Load Time	Execution	
8000	0000	0000	`LDAA #_`
8002	0002	0002	`LDX #_ _`
8005	0005	0005	`DEX`
8009	0009	0009	`BEQ LABEL`