Chapter 8: Memory Management

Background

- The main purpose of a computer system is to execute programs. These programs, together with the data they access, must be in main memory (at least partially) during execution.

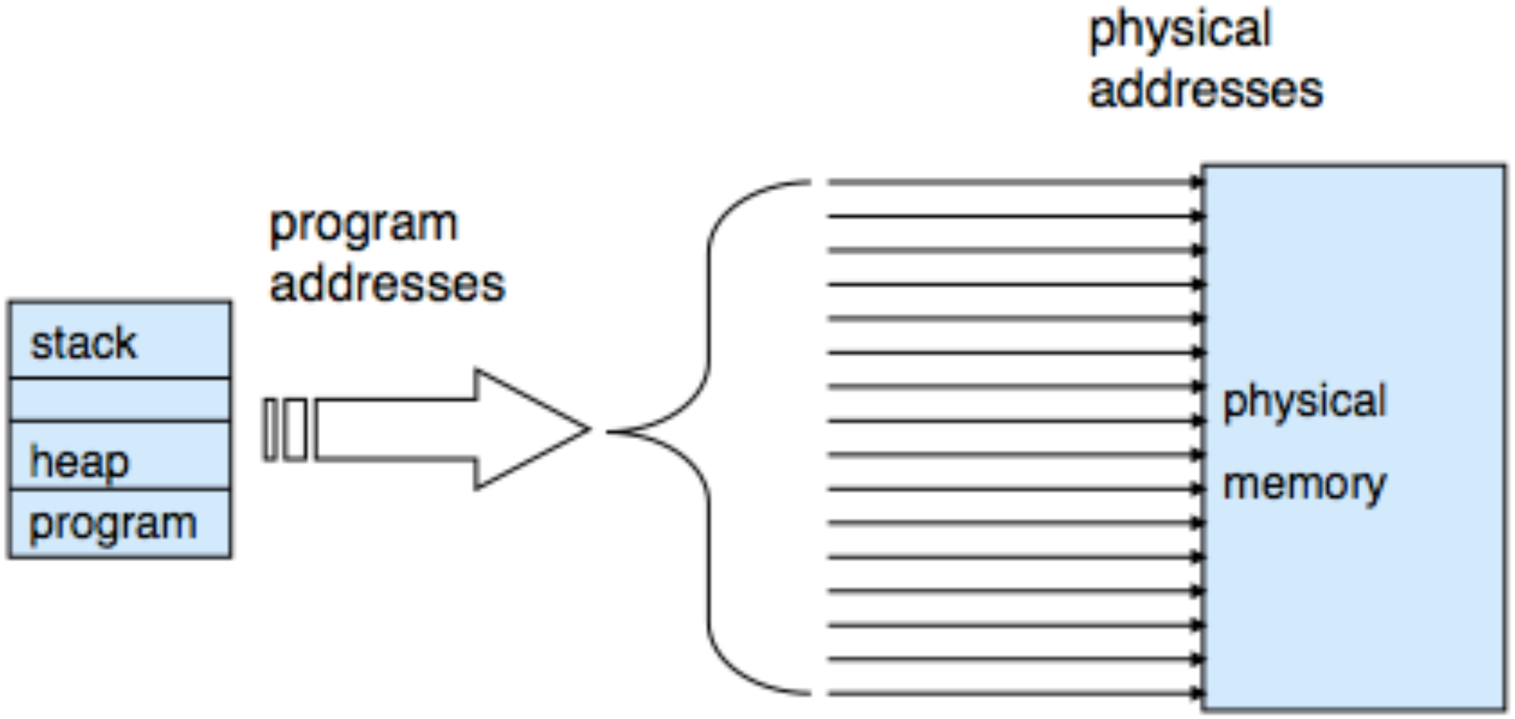
- Main memory and the registers built into the processor itself are the only storage that the CPU can access directly.

- Registers are generally accessible within one cycle of the CPU clock. The same cannot be said of main memory, which is accessed via a transaction on the memory bus.
- Memory access may take many clock cycles to complete, in which case the processor normally needs to stall, since it does not have the data required to complete the instruction that it is executing.

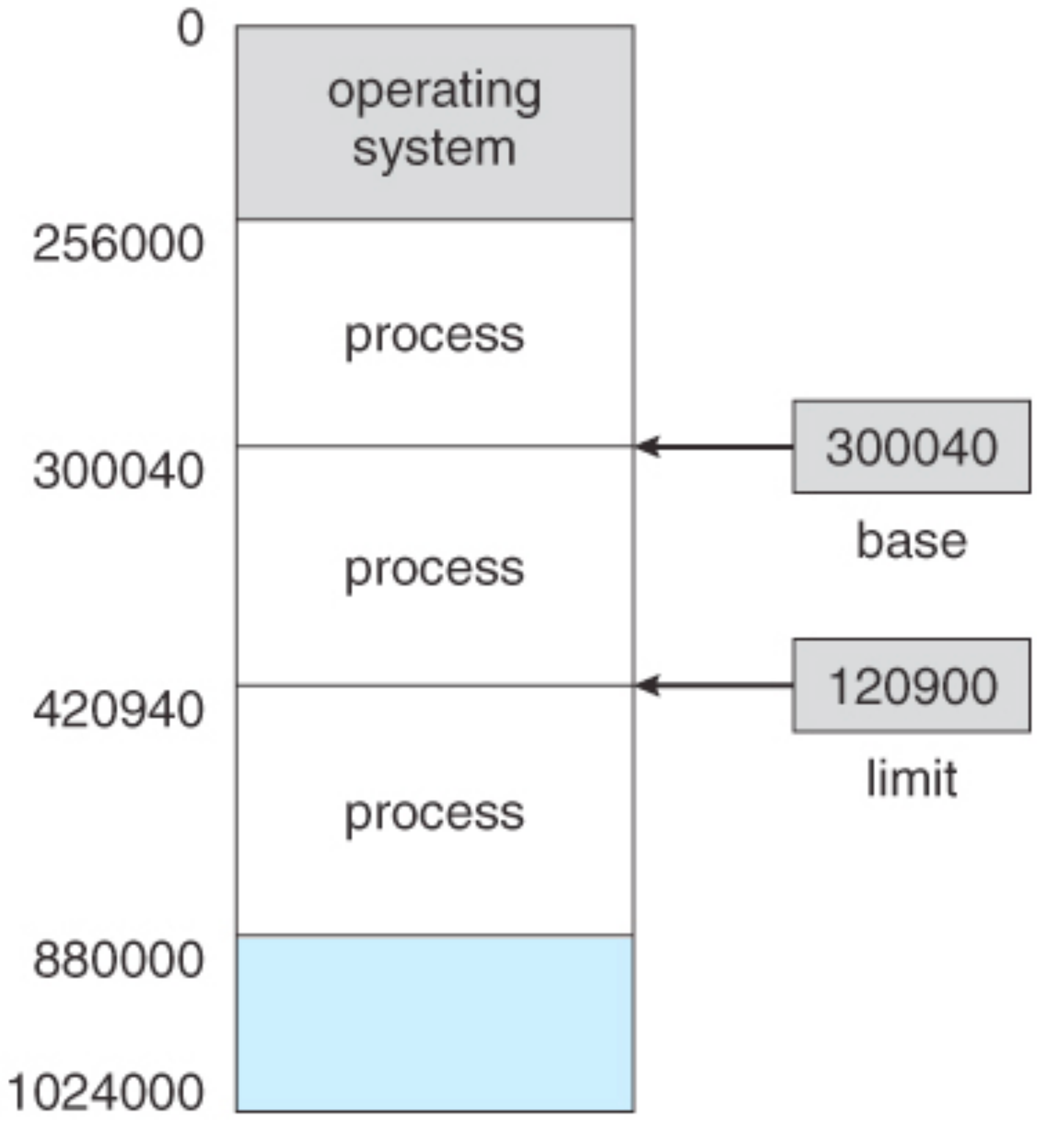
Address Binding

- Usually, a program resides on a disk as a binary executable file. To be executed, the program must be brought into memory.
- **The process of associating program instructions and data to physical memory addresses is called address binding, or relocation**.
- Addresses in the source program are generally relocatable addresses generated by the compiler.
- Loader will in turn bind the relocatable addresses to absolute addresses.
- Each binding is a mapping from one address space to another.

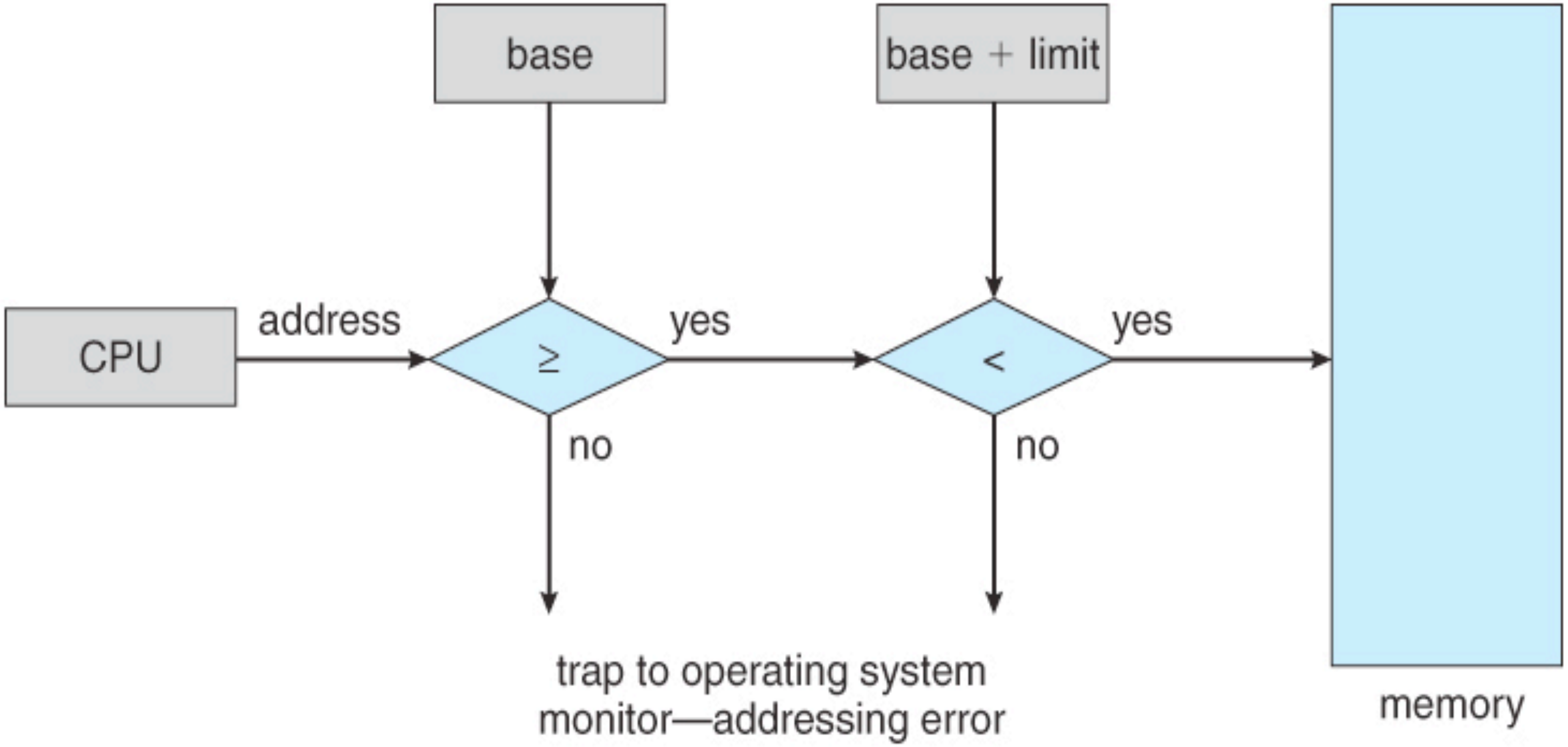
Address Binding (cntd)



- User processes must be restricted so that they only access memory locations that "belong" to that particular process.
- This is usually implemented using a **base register** and a **limit register** for each process.
- Every memory access made by a user process is checked against these two registers, and if a memory access is attempted outside the valid range, then a fatal error is generated.



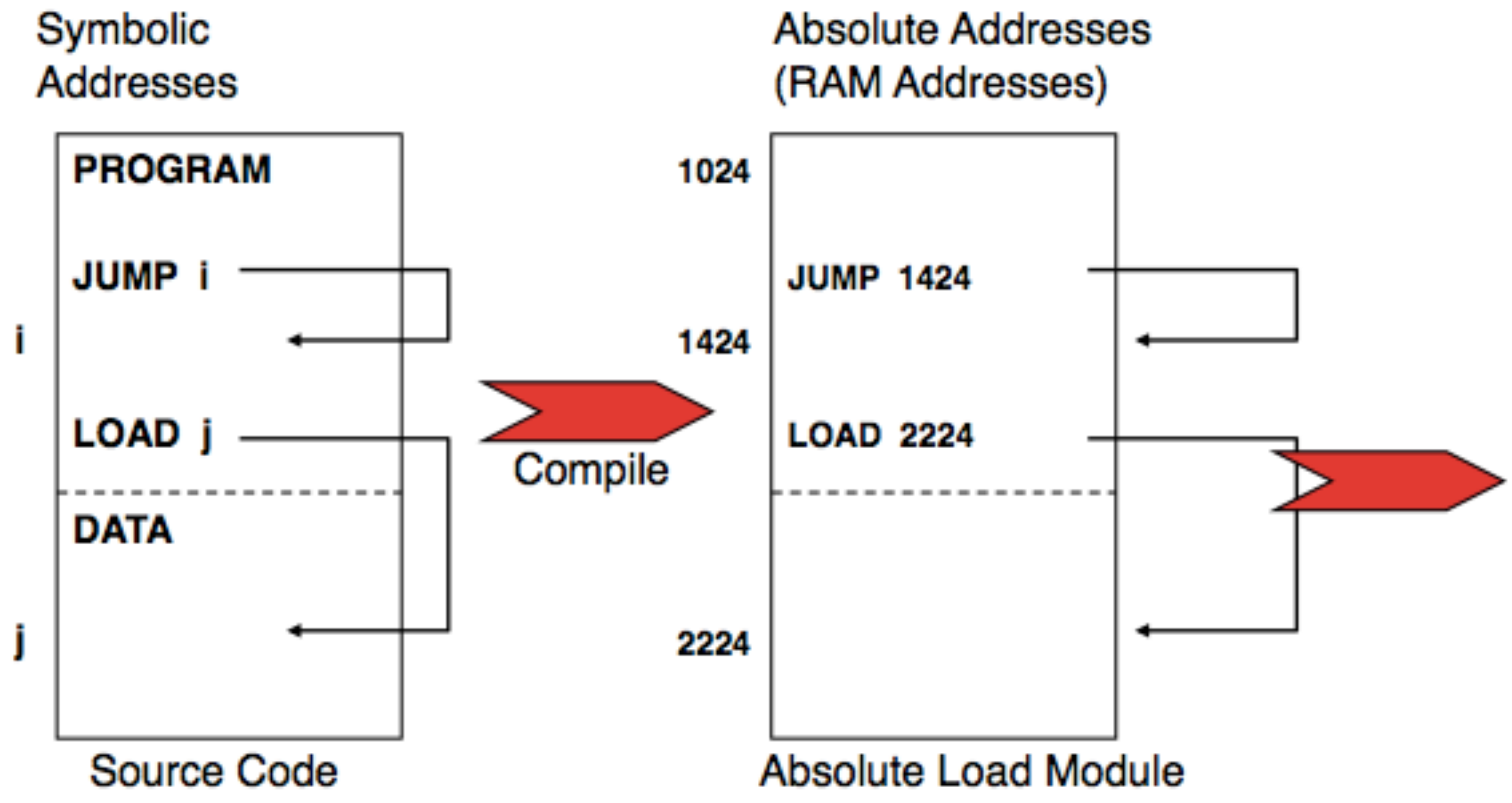
Changing the contents of the base and limit registers is a privileged activity, allowed only to the OS kernel.



Address Binding (cntd)

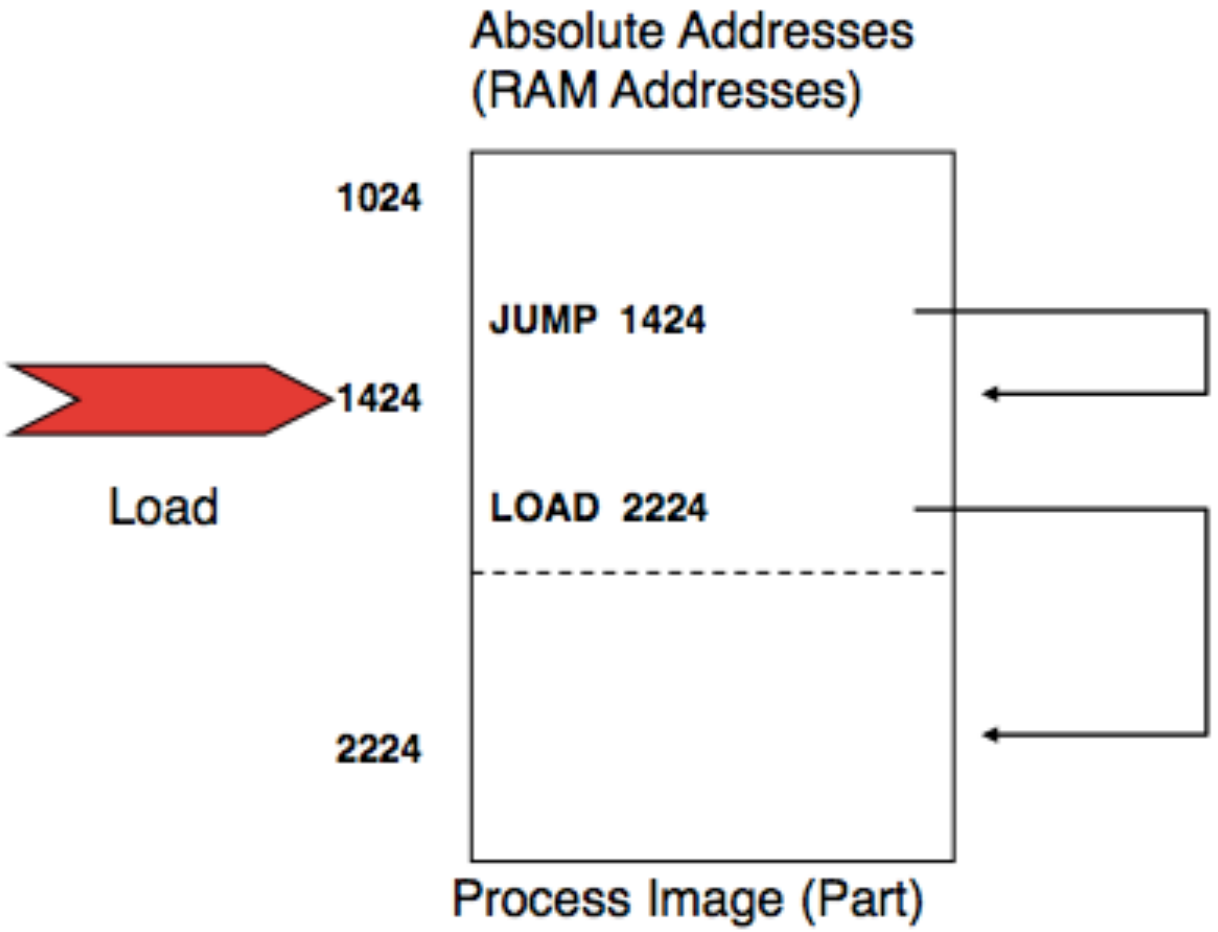
- Classically, the binding of program addresses can be done at any step along the way:
- **Compile time**: The compiler translates symbolic addresses to absolute addresses. If you know at compile time where the process will reside in memory, then absolute addresses can be generated (Static).
- For example, if we 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.

Binding at Compile Time



The CPU generates the absolute addresses

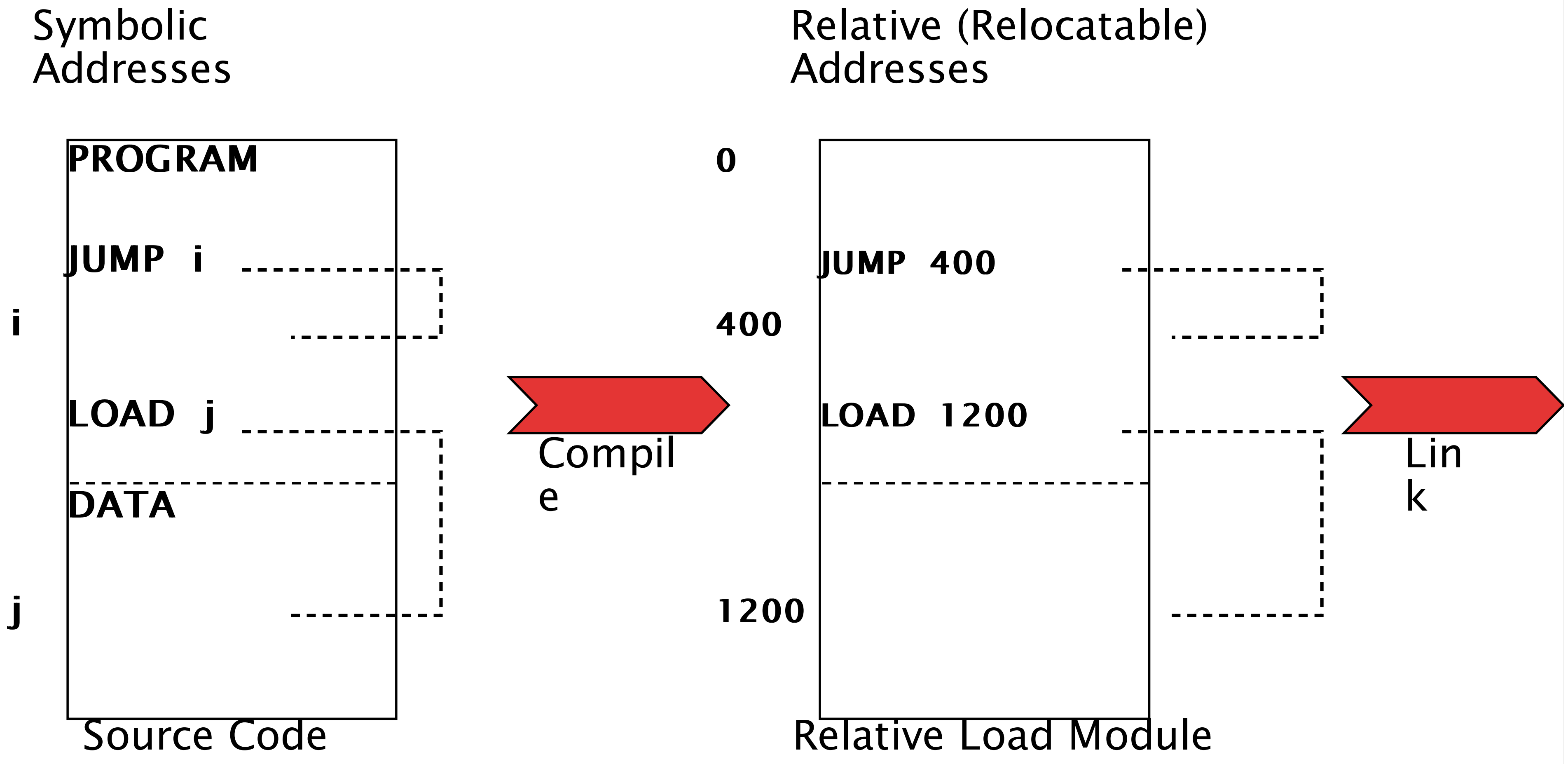
Binding at Compile Time (Cont.)



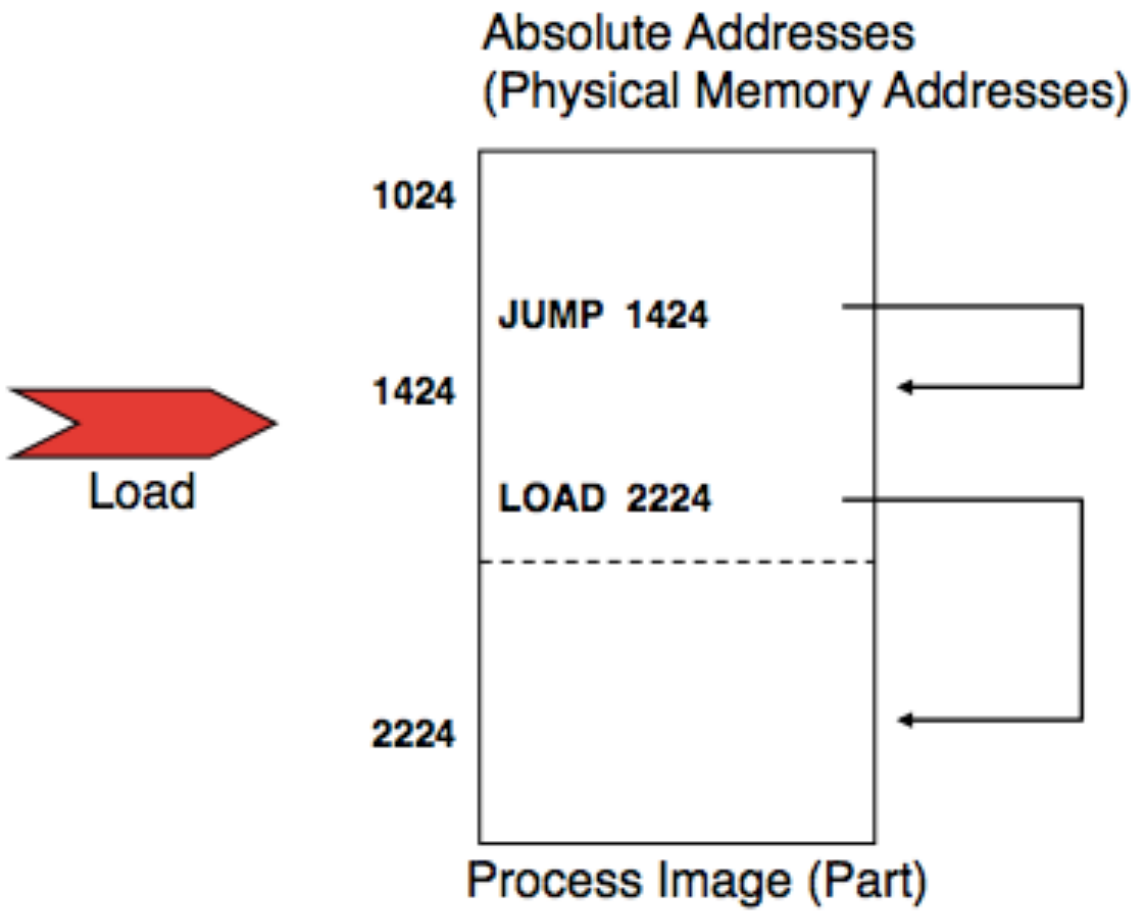
Address Binding (cntd)

- **Load time**: If it is not known at compile time where the process will reside in memory, then the compiler must generate **relocatable code**. In this case, final binding is delayed until load time.
- It references addresses relative to the start of the program.
- If the starting address changes in RAM, then the program must be reloaded but not recompiled.

Binding at Loader Time



Binding at Loader Time (Cont.)

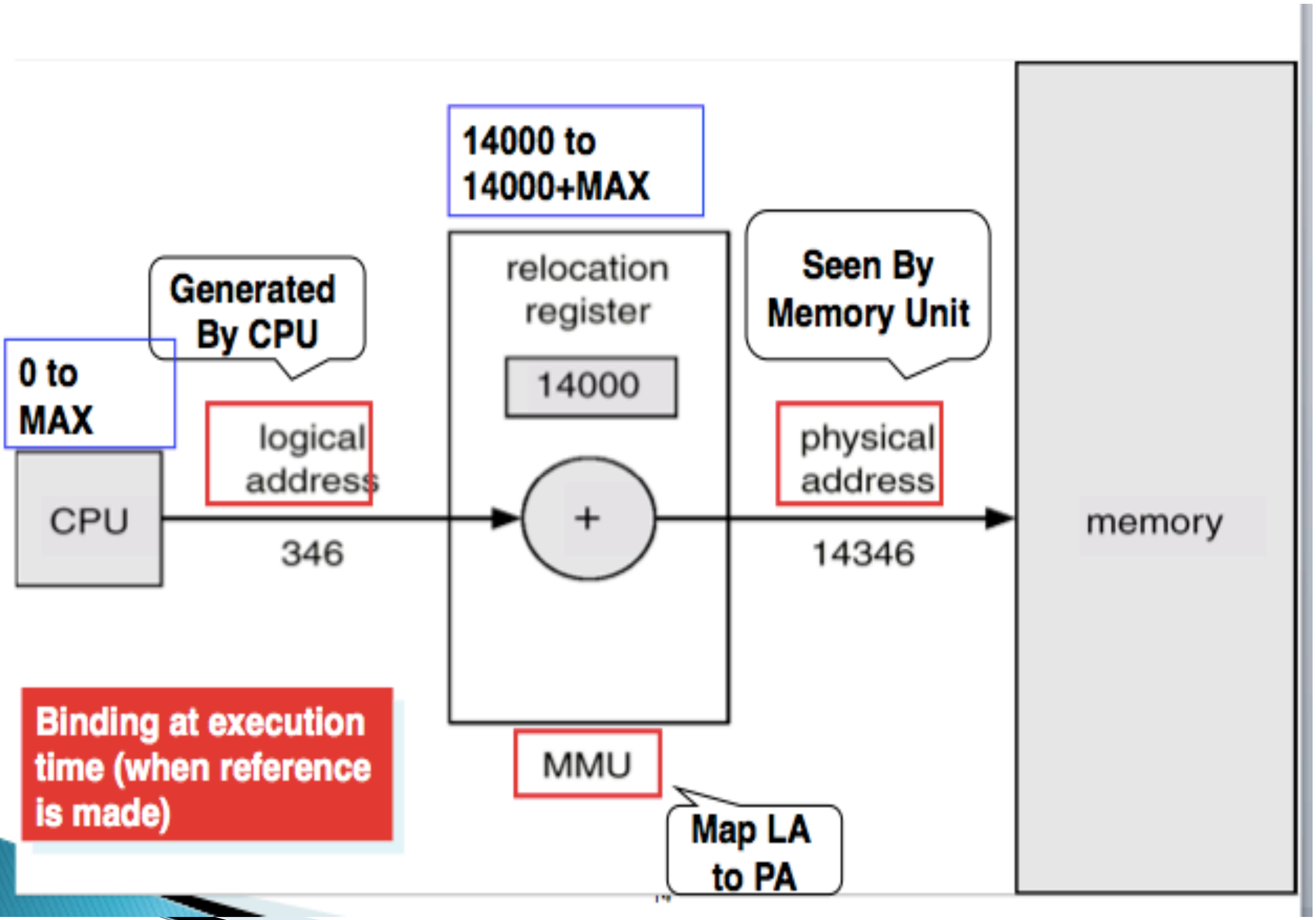


The CPU generates the absolute addresses

Address Binding (cntd)

- **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.
- Special hardware must be available for this scheme to work. Most general-purpose operating systems use this method.

Dynamic Relocation Using a Relocation Register



Logical vs. Physical Address Space

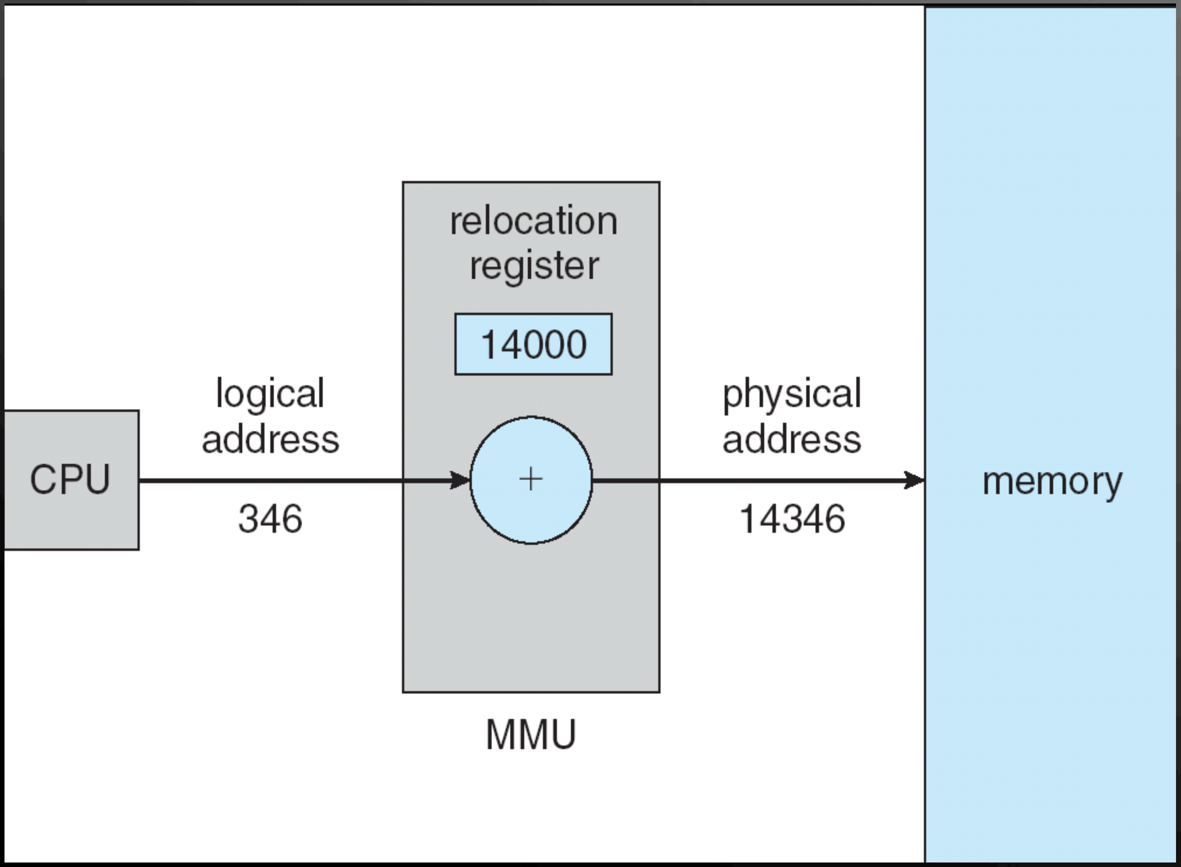
- **Logical address** – generated by the CPU; also referred to as *virtual address*
- **Physical address** – address seen by the memory unit. Actual addresses of RAM

- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme

Memory-Management Unit (MMU)

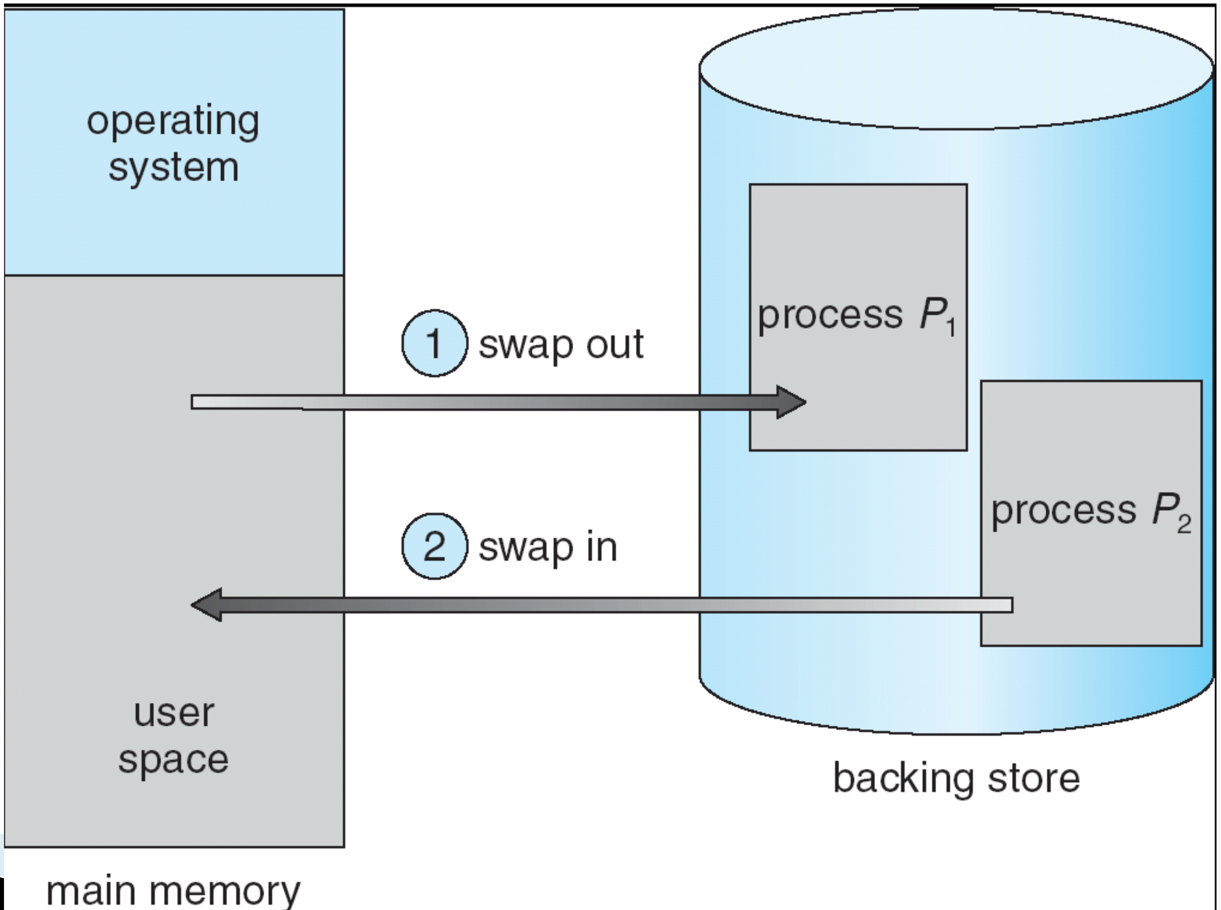
- Hardware device that maps virtual to physical address
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory
- The user program deals with *logical* addresses; it never sees the *real* physical addresses

Dynamic relocation using a relocation register



Swapping

- A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution.
- **Backing store** – fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images
- **Roll out, roll in** – swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed



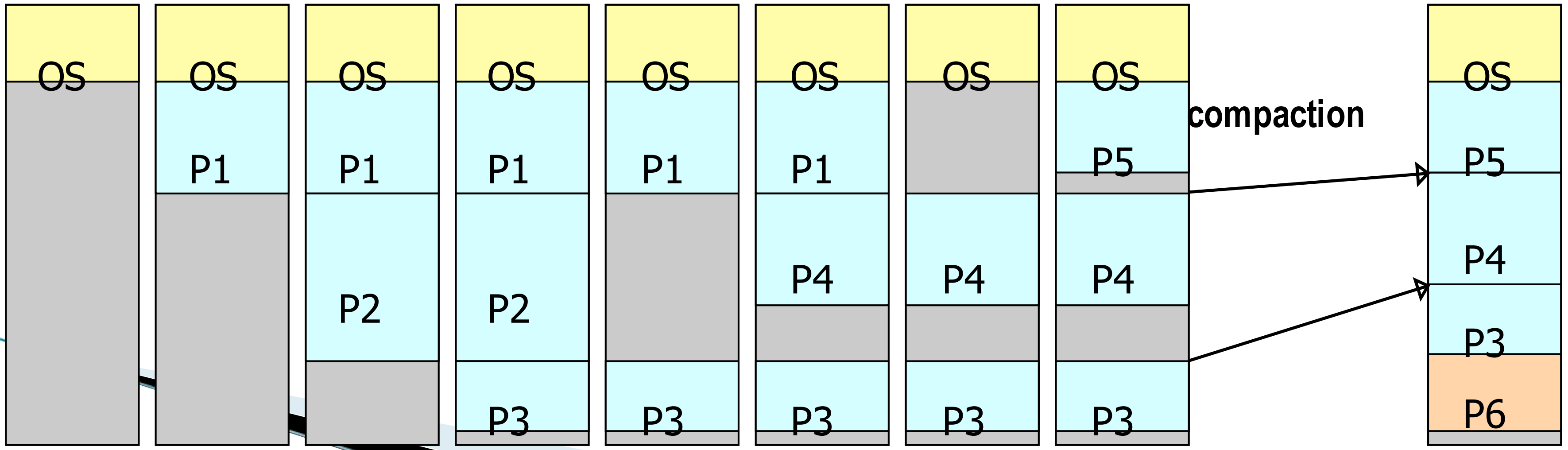
Swapping (cntd)

Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped.

- A modification of swapping is used in many versions of UNIX. Swapping is normally disabled but will start if many processes are running and are using a threshold amount of memory.
- Swapping is again halted when the load on the system is reduced.

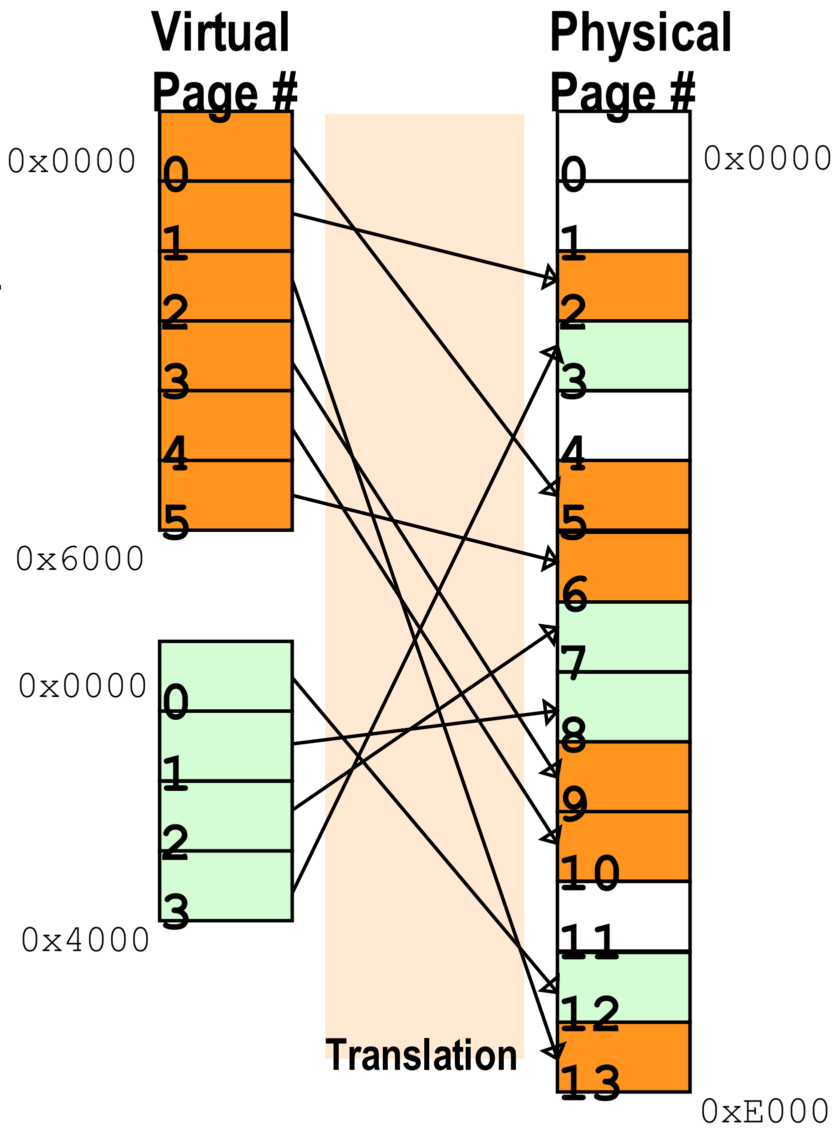
Fragmentation

- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used
- Reduce external fragmentation by **compaction**
  - Shuffle memory contents to place all free memory together in one large block
  - Compaction is possible *only* if relocation is dynamic, and is done at execution time



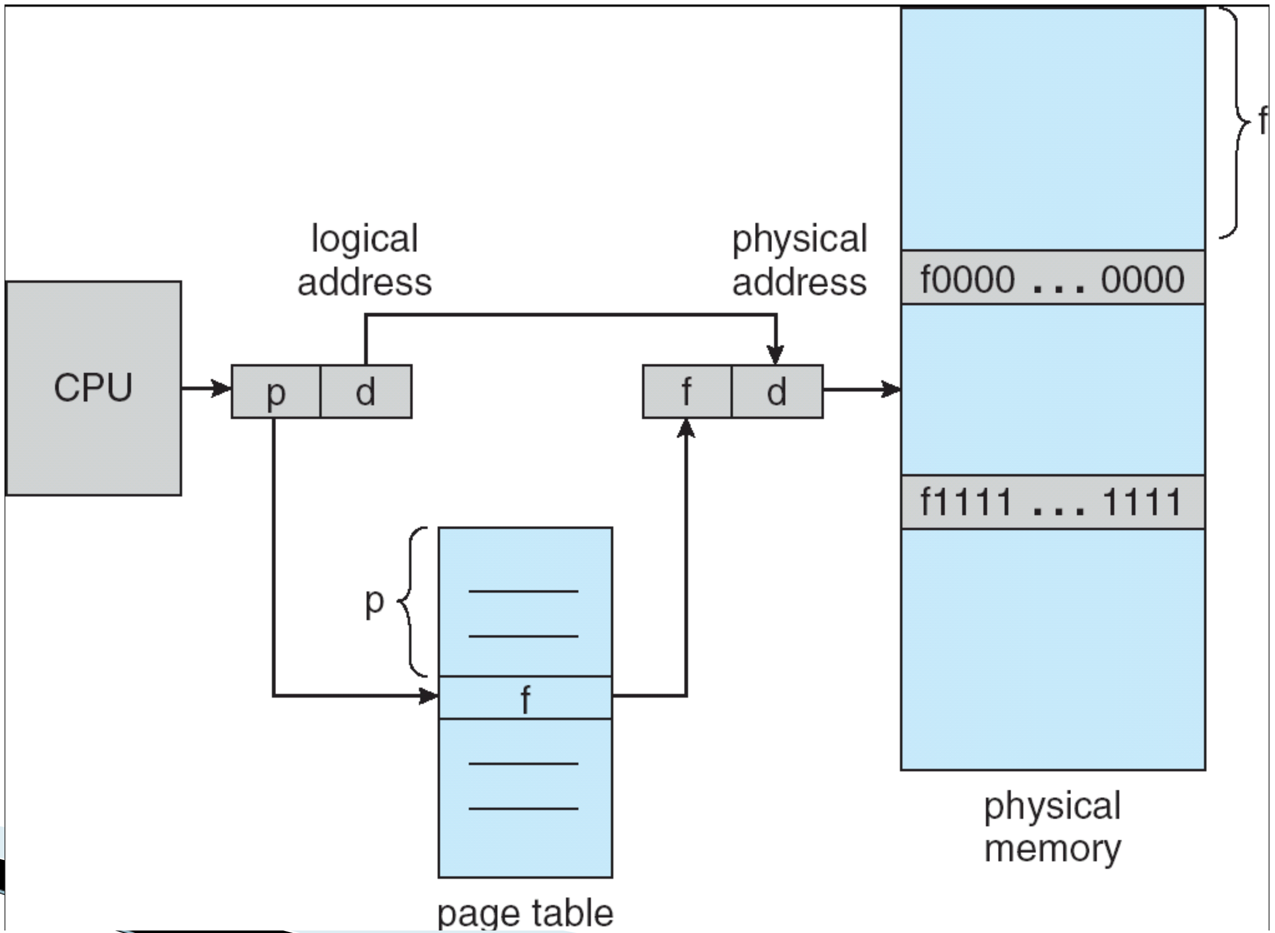
Paging

- Divide physical memory into fixed-sized blocks called **frames** (size is power of 2, between 512 bytes and 8192 bytes)
- Divide logical memory into blocks of same size called **pages**.
- **Any virtual page can be located at any physical page**
- Translation box converts from virtual pages to physical pages

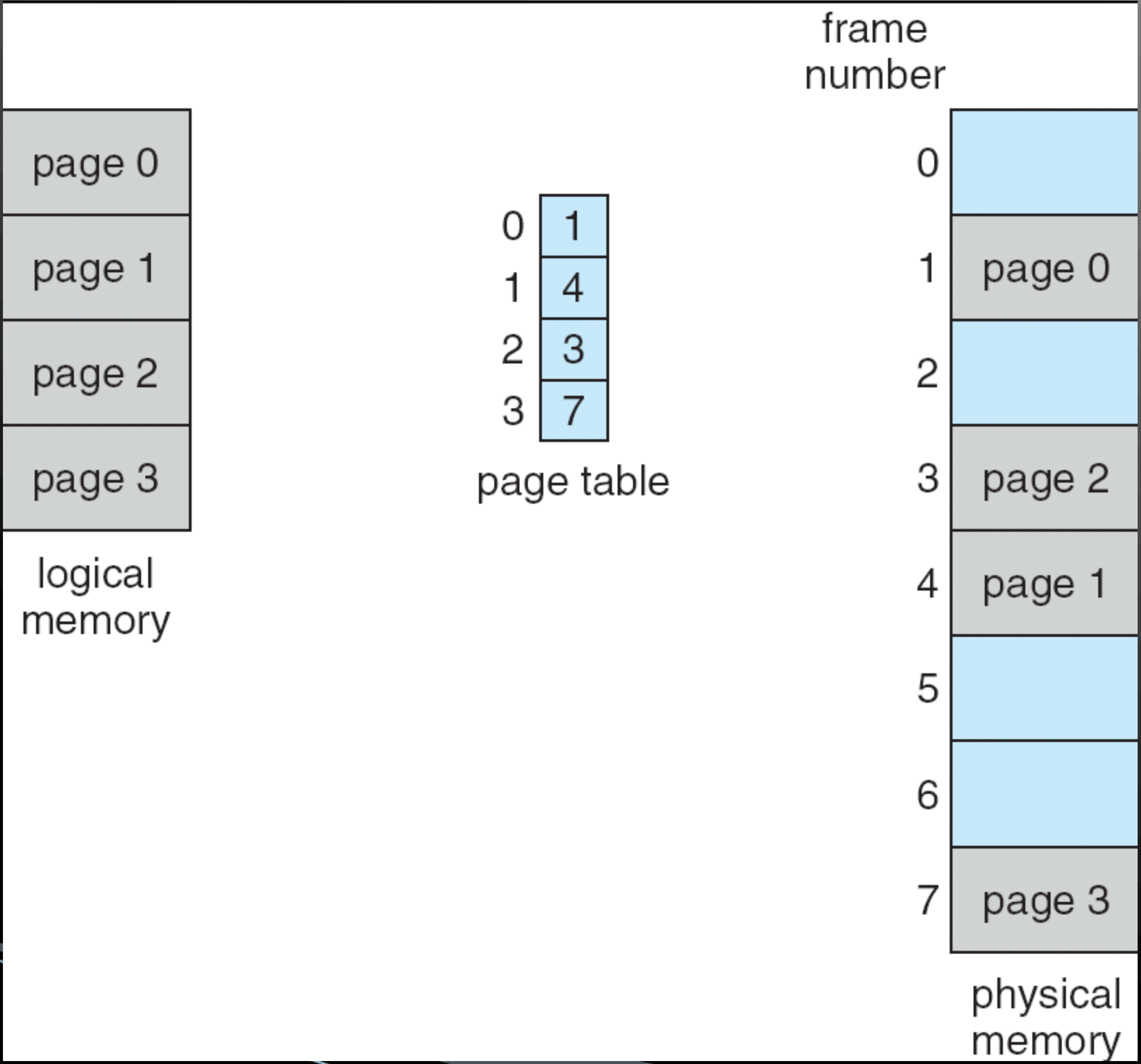


Address Translation Scheme

- Address generated by CPU is divided into:
  - ***Page number (p)*** – used as an index into a *page table* which contains base address of each page in physical memory
  - ***Page offset (d)*** – combined with base address to define the physical memory address that is sent to the memory unit

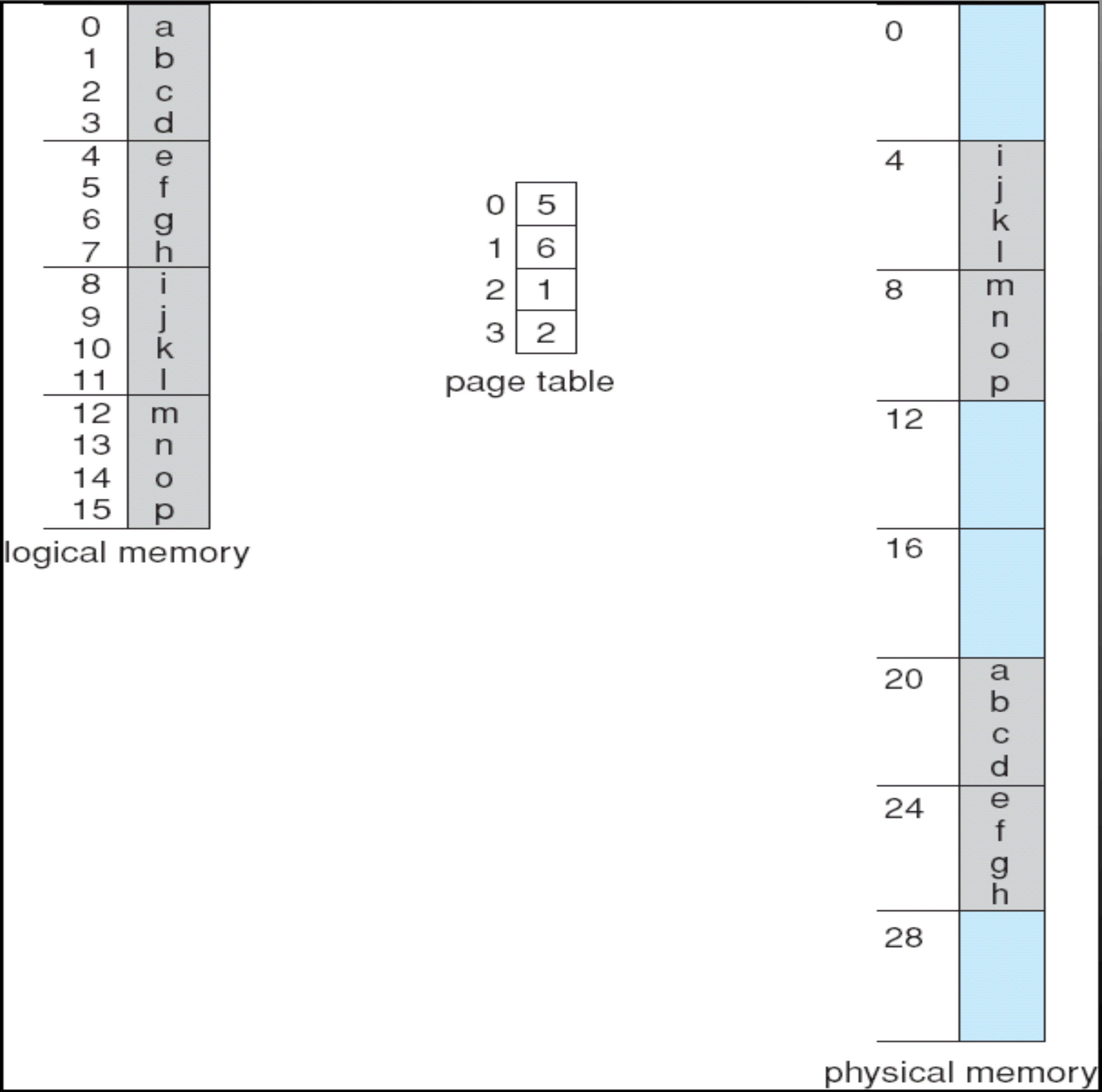


Paging Example



Page Size= 4bytes

Physical Memory= 32bytes



Paging

- In paging we have **no external fragmentation**. Any free frame can be allocated to a process that needs it. However, we may have some internal fragmentation?

- Today, pages typically are between 4 KB and 8 KB in size, and some systems support even larger page sizes.

- Important aspect of paging is the clear separation between the user's view of memory and the actual physical memory.
- The user program views memory as one single space, containing only this one program. In fact, the user program is scattered throughout physical memory, which also holds other programs.