# CSCI315 – Operating Systems Design

Department of Computer Science

Bucknell University

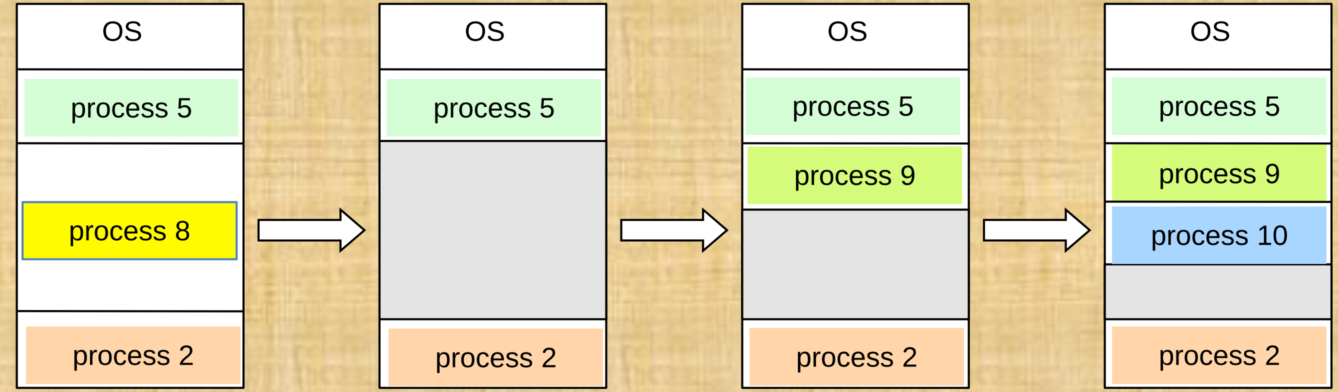
## Memory Management
## Contiguous Allocation and Introduction to Paging

Ch 9.2-9.3

*This set of notes is based on notes from the textbook authors, as well as L. Felipe Perrone, Joshua Stough, and other instructors.*
*Xiannong Meng, Fall 2021.*

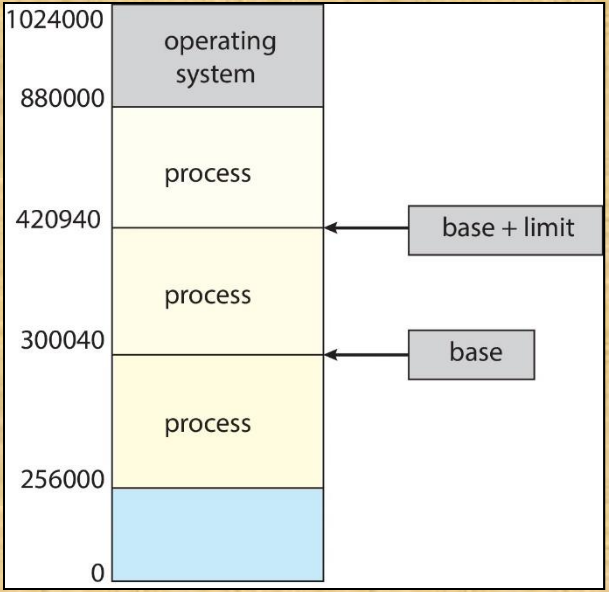
## Contiguous Allocation

- Multiple-partition allocation: Programs are put into various parts of the memory. But each program occupies a **contiguous** part of the memory.

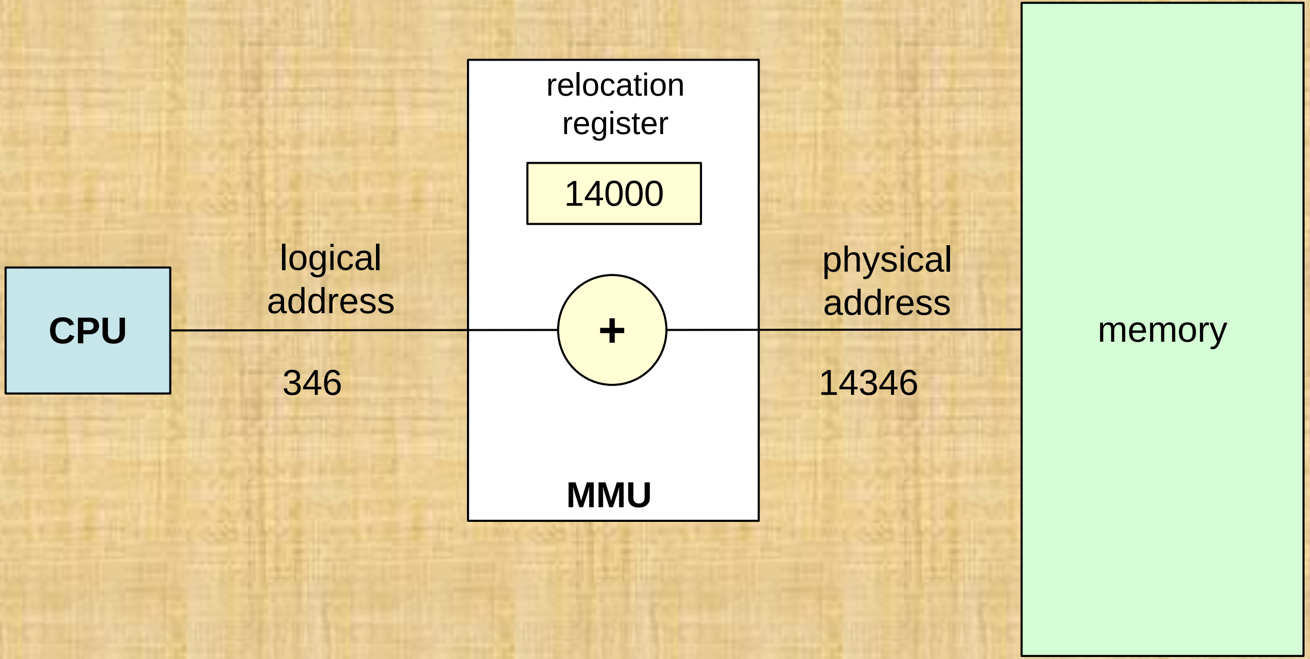


# Protection

- Need to censure that a process can access only those addresses in it address space.
- Provide this protection by using  a pair of **base** and **limit registers** define the logical address space of a process.



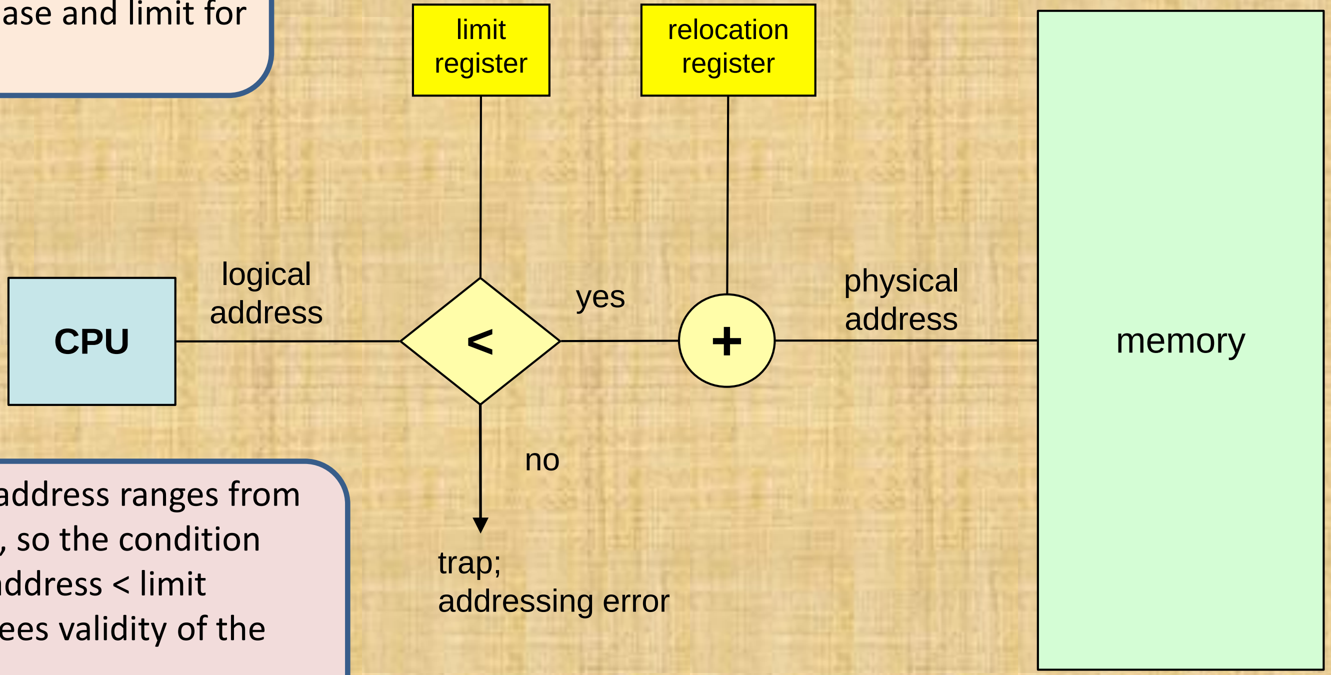
# Dynamic relocation using a relocation register



This is an example of how to convert a logical address to a physical one.

# Hardware Support for Relocation and Limit Registers

CPU must check every memory access generated in user mode to be sure it is between base and limit for that user.



Logical address ranges from 0 to n-1, so the condition logical address < limit guarantees validity of the address

## Dynamic Loading

- Routine (program parts) is not loaded until it is called.
- Better memory-space utilization; unused routine is never loaded.
- Useful when large amounts of code are needed to handle infrequently occurring cases.
- No special support from the operating system is required; implemented through program (compiler, loader) design.

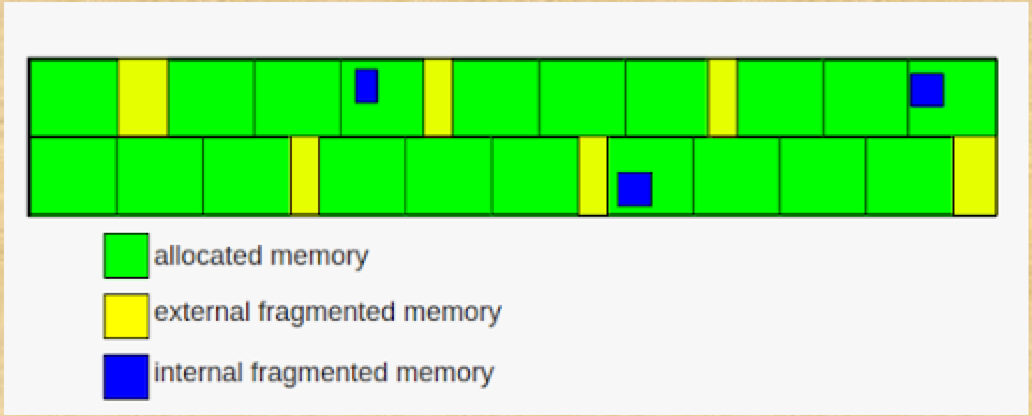
## Dynamic Linking

- Linking postponed until execution time.
- Small piece of code, *stub*, used to locate the appropriate memory-resident library routine.
- Stub replaces itself with the address of the routine, and executes the routine.
- Operating system needed to check if routine is in process's memory address space.
- Dynamic linking is particularly useful for libraries.
  - In Windows, they are *.dll files (e.g., C:\Program Files (x86)\Windows Defender\)
  - In Linux, they are *.so.* files (e.g., /usr/lib)

# 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
- First fit analysis reveals that given *N* blocks allocated, another 0.5 *N* blocks lost to fragmentation (a total of 1.5N blocks)
  - **50-percent rule:** “1/3 may be unusable”

# External and Internal Fragmentation



https://images.app.goo.gl/ghC55ZDKDBXLFHMH6

http://www.differencebetween.net/technology/difference-between-internal-fragmentation-and-external-fragmentation/

## Fragmentation (Cont.)

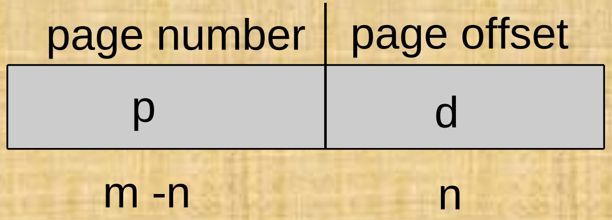
- Reduce external fragmentation by **compaction**
  - Shuffle memory contents to place all free memory together in one large block
  - Compaction is possible *only* if relocation is dynamic, and is done at execution time
  - I/O problem
    - Latch job in memory while it is involved in I/O
    - Do I/O only into OS buffers
- Now consider that backing store (e.g., hard disks) has the same fragmentation problems

## Paging

- Physical  address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available
  - Avoids external fragmentation
- Divide physical memory into fixed-sized blocks called **frames**
  - Size is power of 2, between 512 bytes and 16 Mbytes
- Divide logical memory into blocks of same size called **pages**
- Keep track of all free frames
- To run a program of size ***N*** pages, need to find ***N*** free frames and load program
- Set up a **page table** to translate logical to physical addresses

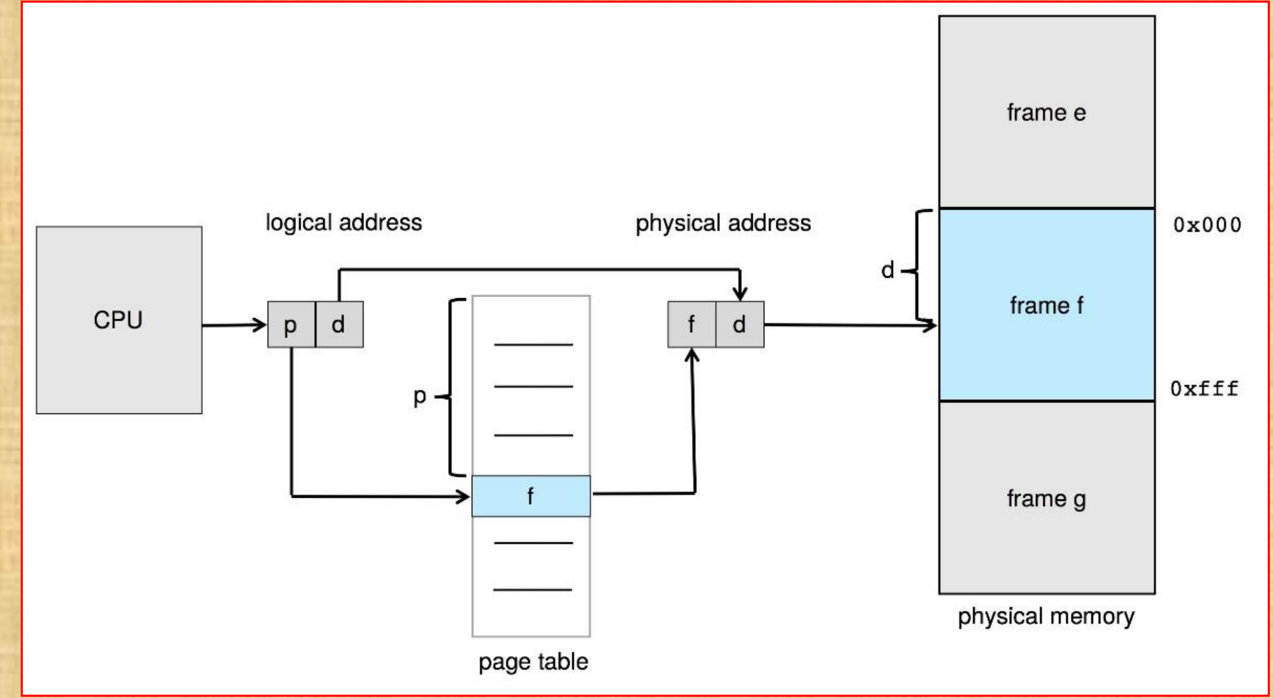
## Address Translation Scheme

- 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

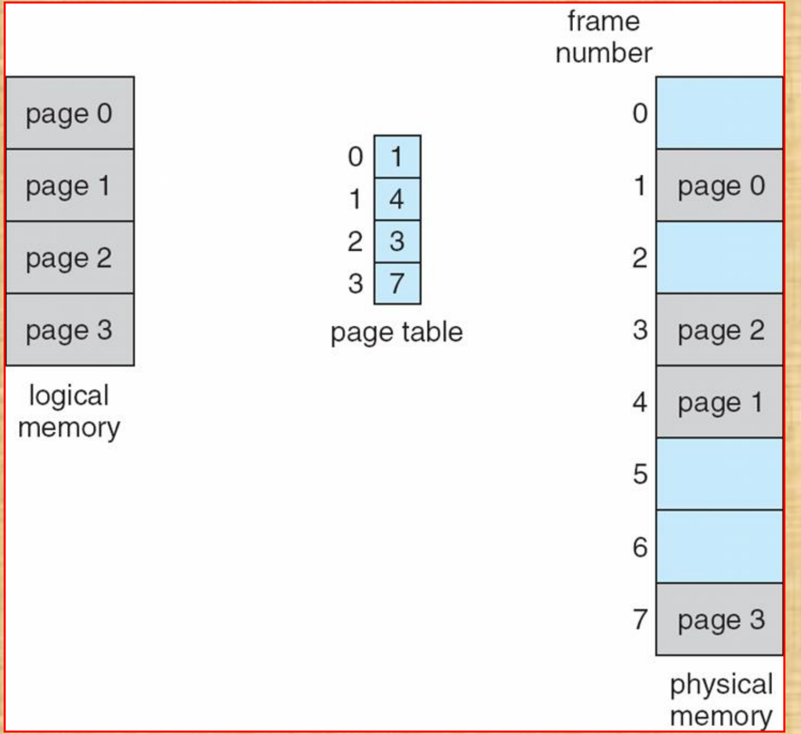


  - For given logical address space $2^m$ and page size $2^n$ assuming the address has **m** bits.

## Paging Hardware

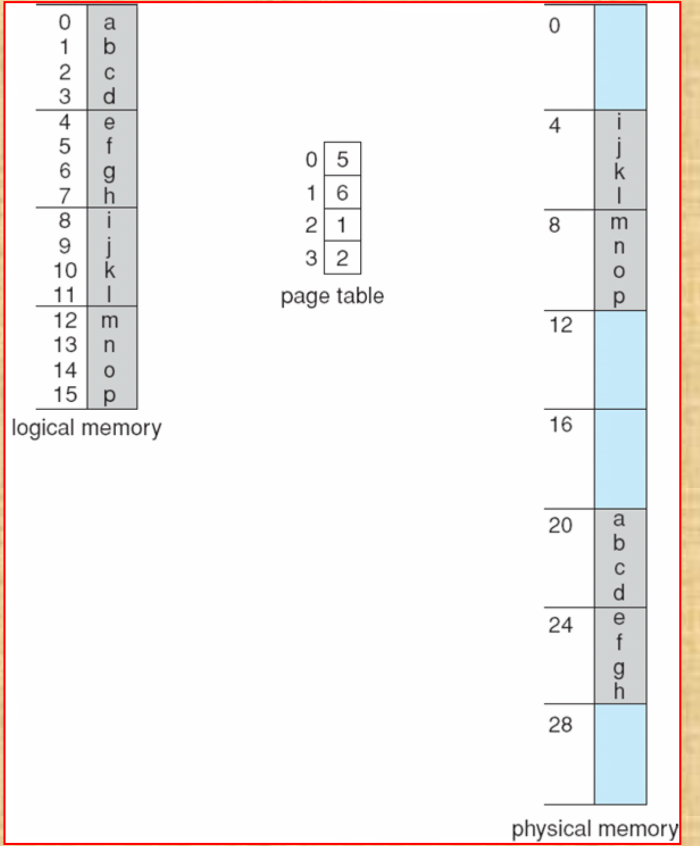


## Paging Model of Logical and Physical Memory



# Paging Example

- Logical address: n = 2 and m = 4. Using a page size of 4 bytes and a physical memory of 32 bytes (8 pages)



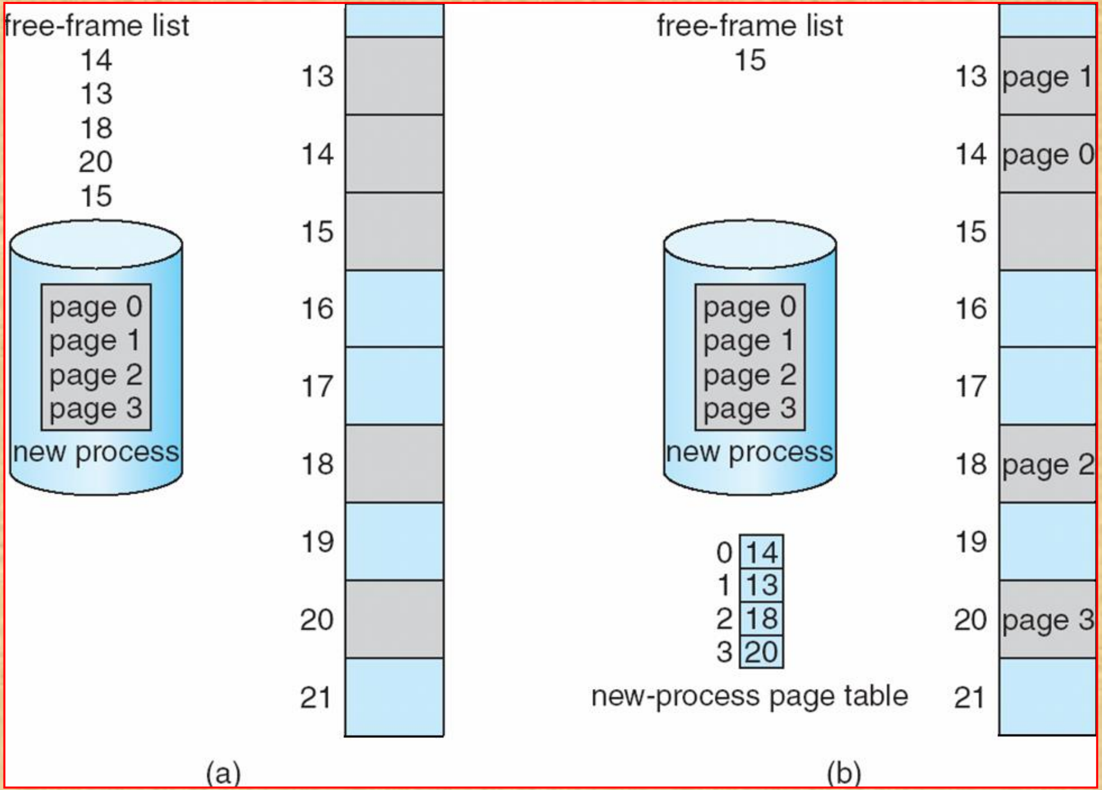
## Paging – Example of internal fragmentation

- Page size = 2,048 bytes
- Process size = 72,766 bytes
- 35 pages + 1,086 bytes
- Internal fragmentation of 2,048 - 1,086 = 962 bytes
- Worst case fragmentation = 1 frame – 1 byte
- On average fragmentation = 1 / 2 frame size
- So small frame sizes desirable?
- But each page table entry takes memory to track

## Page Size Settings

- Page sizes growing over time with larger memory capacity
  - Solaris supports two page sizes – 8 KB and 4 MB
- Our current Linux page size is 4 K by default. We can use larger size by setting the system configuration file
  - Run pagesize.c http://www.eg.bucknell.edu/~cs315/F2021/meng/code/memory/pagesize.c
  - How to use larger page size? https://linuxgazette.net/155/krishnakumar.html

## Free Frames



Before allocation

After allocation