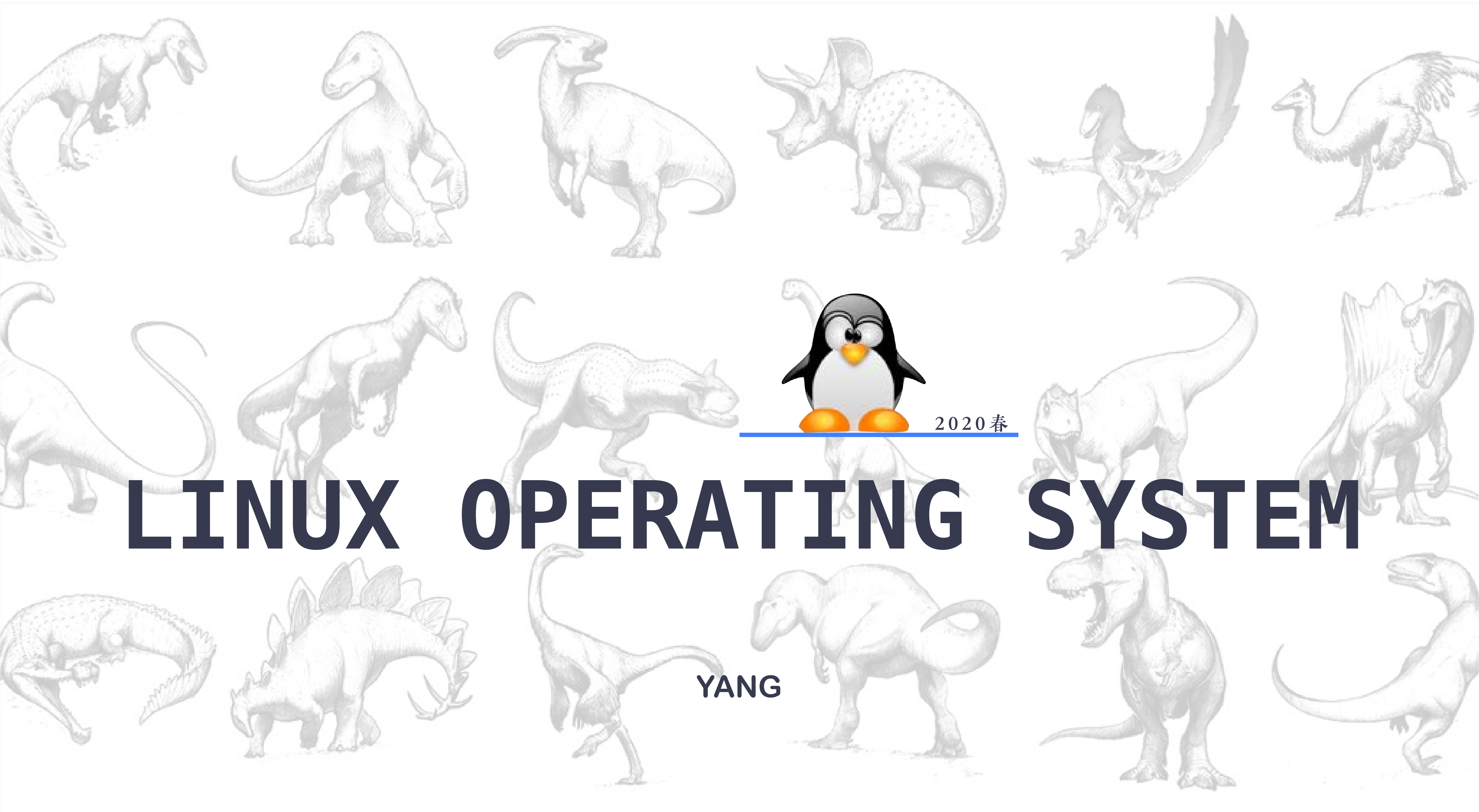


2020春

LINUX OPERATING SYSTEM

YANG

LINUX操作系统（双语）

双语课→课件内容中英混排

Lecture 12

Memory Management

本讲内容

- 内存管理目标
- 逻辑地址和物理地址
- 连续内存分配

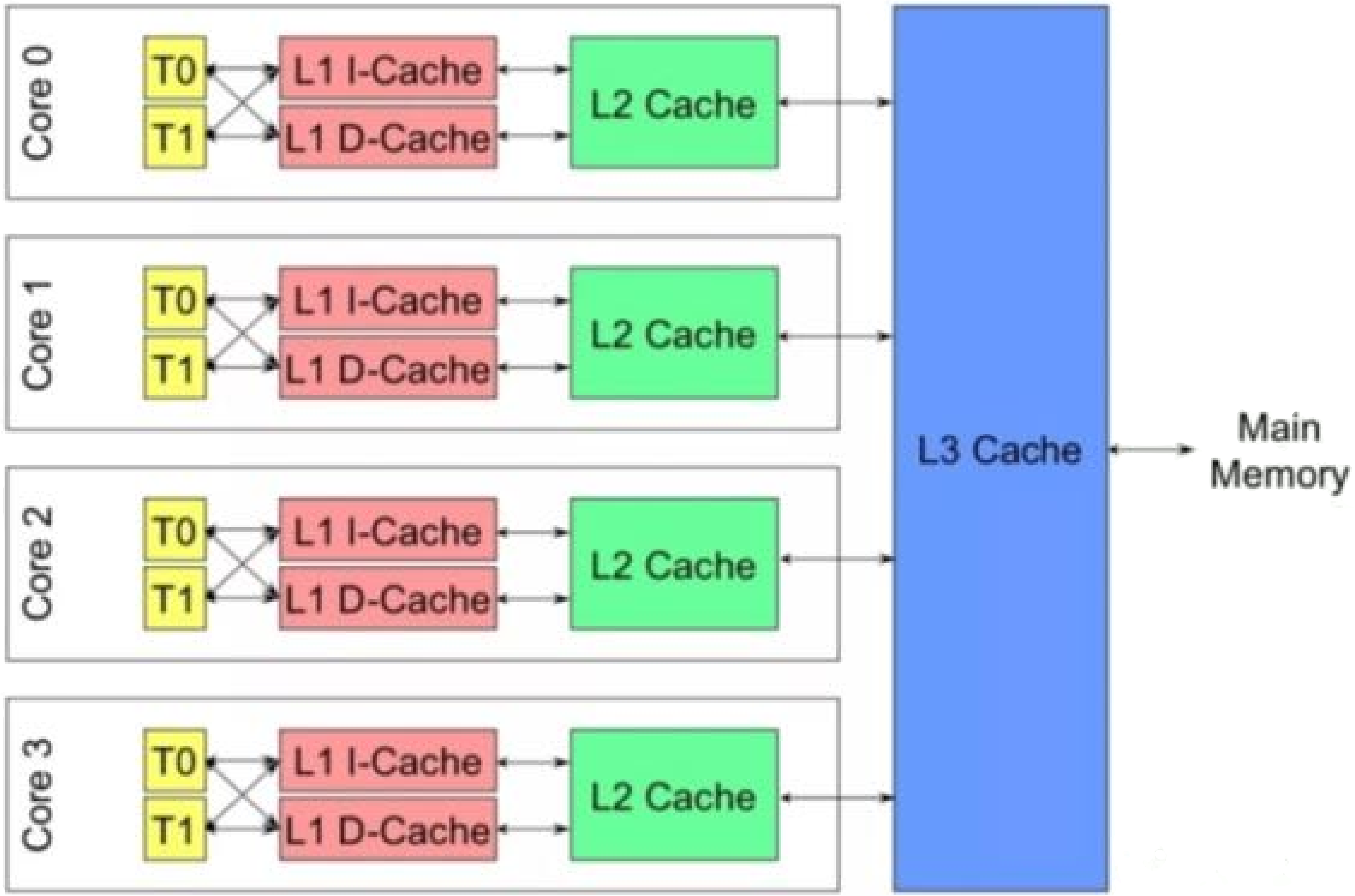
内存管理目标

MAIN MEMORY

- Main memory is central to the operation of a modern computer system.
- Memory consists of a large array of bytes, each with its own address.
- The CPU fetches instructions from memory according to the value of the program counter(PC). These instructions may cause additional loading from and storing to specific memory addresses.
- A typical instruction-execution cycle, for example, first fetches an instruction from memory. The instruction is then decoded and may cause operands to be fetched from memory. After the instruction has been executed on the operands, results may be stored back in memory.

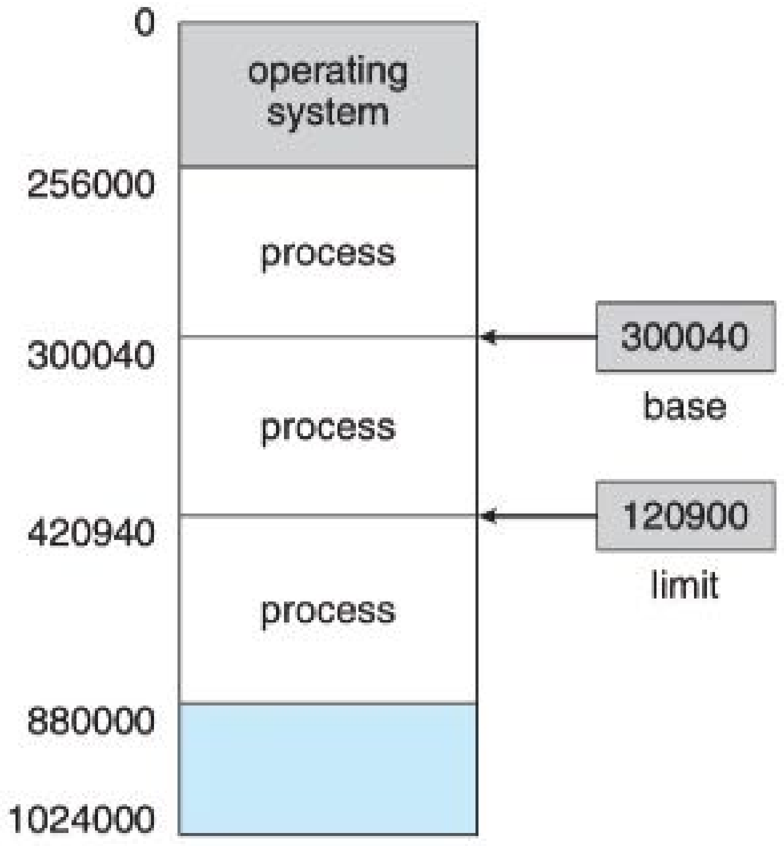
高速缓存CACHE

- 高速缓存是一种存取速度比内存快，但容量比内存小的多的存储器，它可以加快访问物理内存的相对速度。



保护操作系统和用户进程

- 用户进程不可以访问操作系统内存数据，以及用户进程空间之间不能互相影响
  - 通过硬件实现，因为操作系统一般不干预CPU对内存的访问
    - base register：基址寄存器
    - limit register：限长寄存器
  - 上述两个寄存器的值只能被操作系统的特权指令加载



内存管理目标

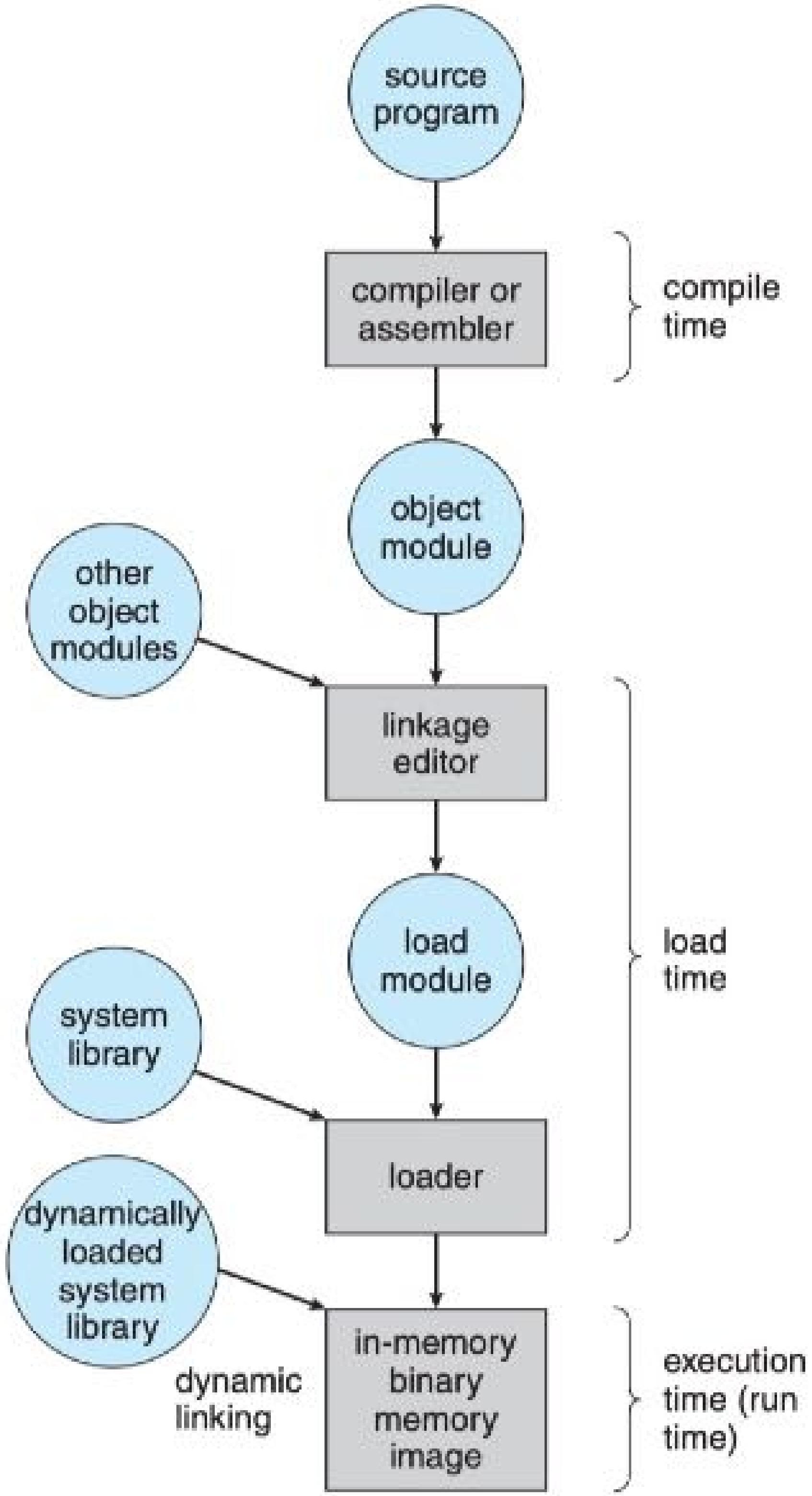
- 存取速度
- 操作正确（分配和回收）
- 保护操作系统
- 保护用户进程
- 地址转换

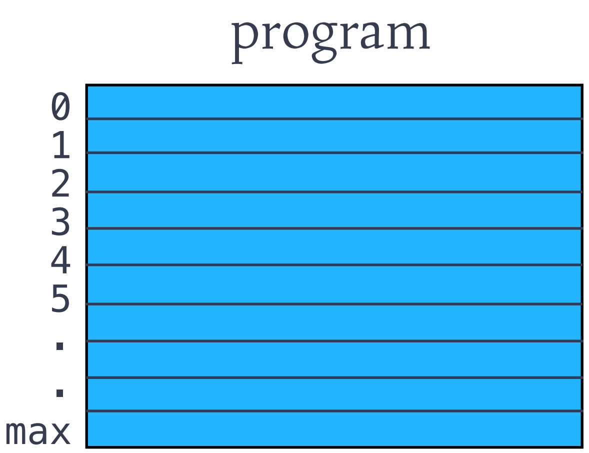
逻辑地址和物理地址

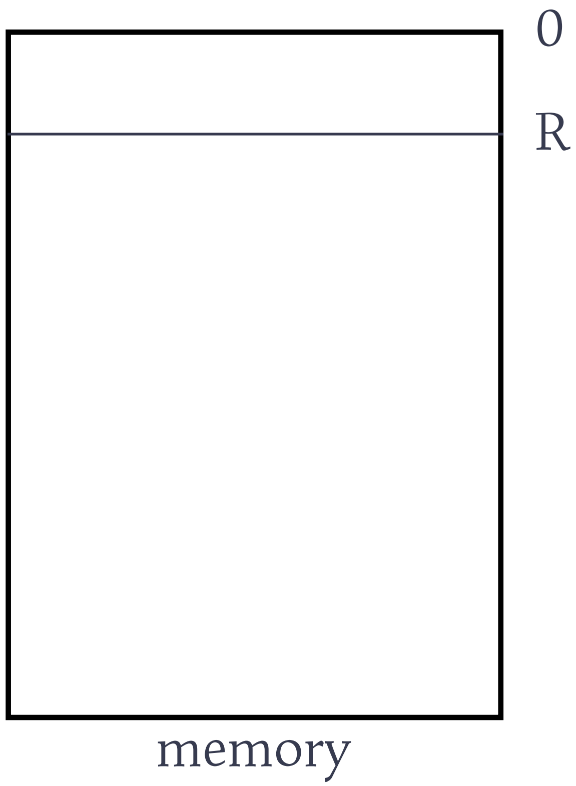
地址空间和地址转换

- 逻辑地址：面向程序的地址，总是从0开始编址，每一条指令的逻辑地址就是与第1条指令之间的相对偏移，因此逻辑地址也叫相对地址或虚拟地址。
- 物理地址：内存单元看到的实际地址，也称为绝对地址。
- 所有逻辑地址的集合称为逻辑地址空间，这些逻辑地址对应的所有物理地址集合称为物理地址空间。
- 地址转换：由逻辑地址转换成物理地址。

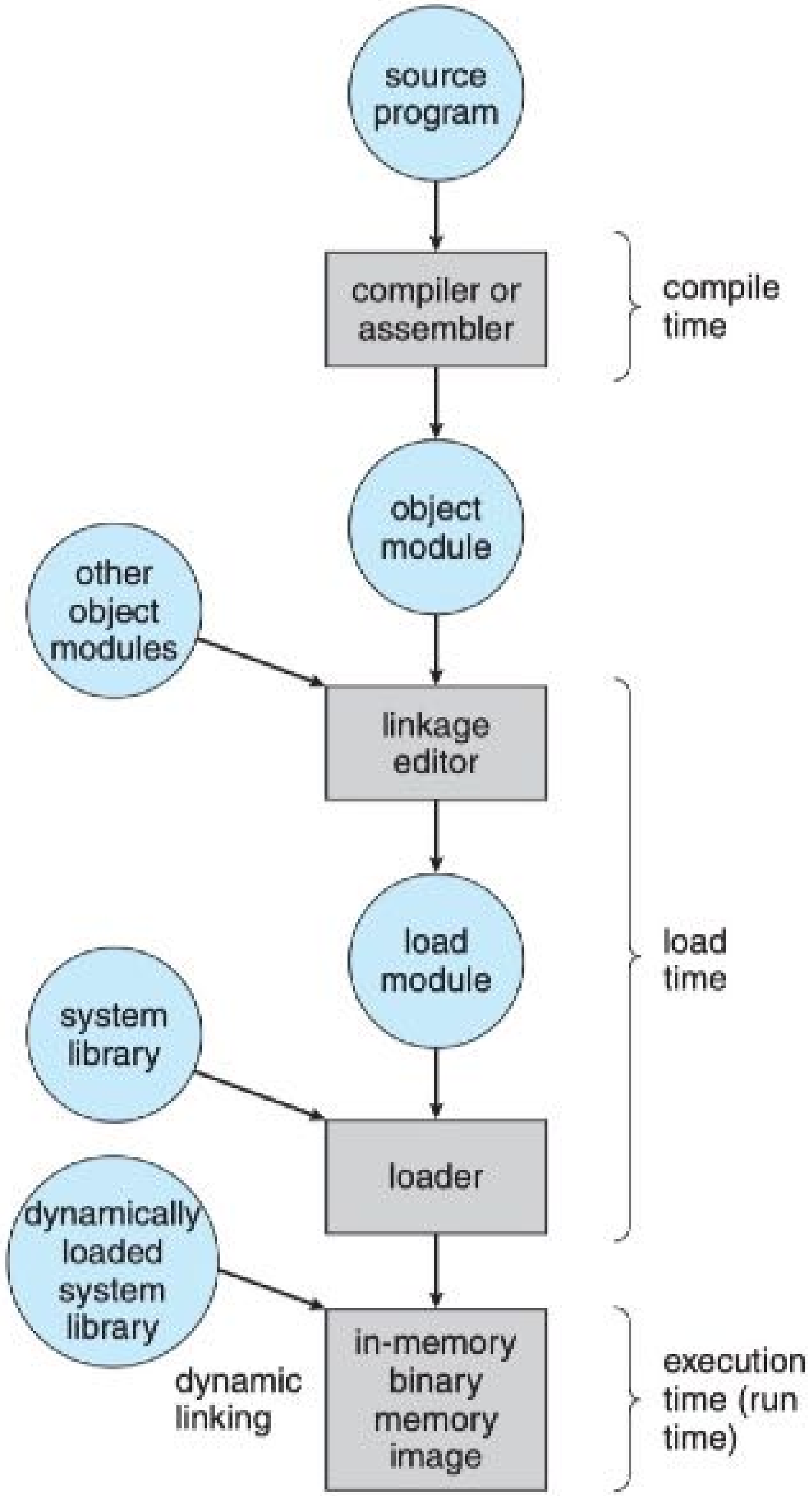
地址转换时机


source program
compiler or assembler
compile time
object module
other object modules
linkage editor
load module
load time
system library
loader
dynamically loaded system library
dynamic linking
in-memory binary memory image
execution time (run time)

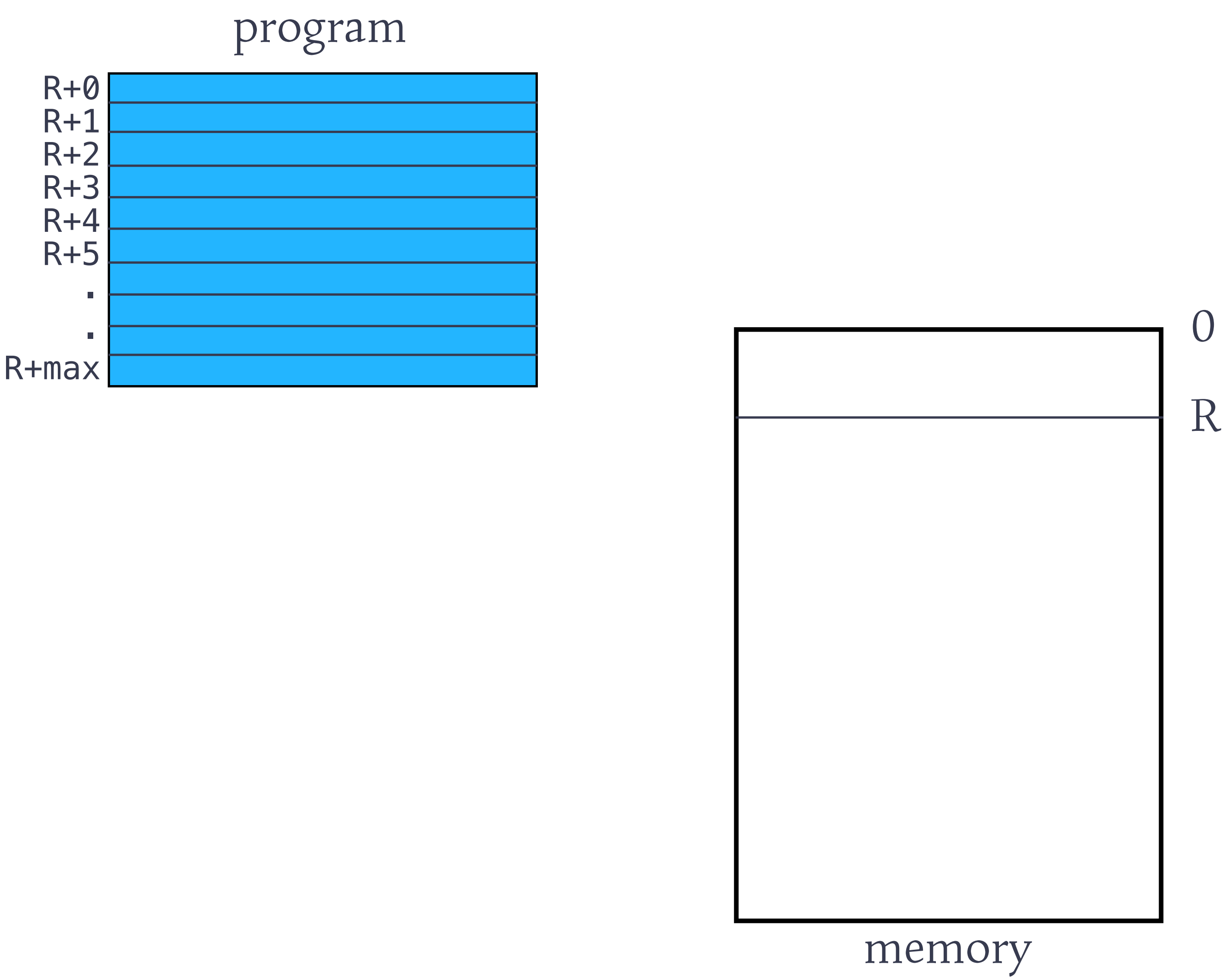

program
0
1
2
3
4
5
.
.
max


0
R
memory

地址转换时机-编译时

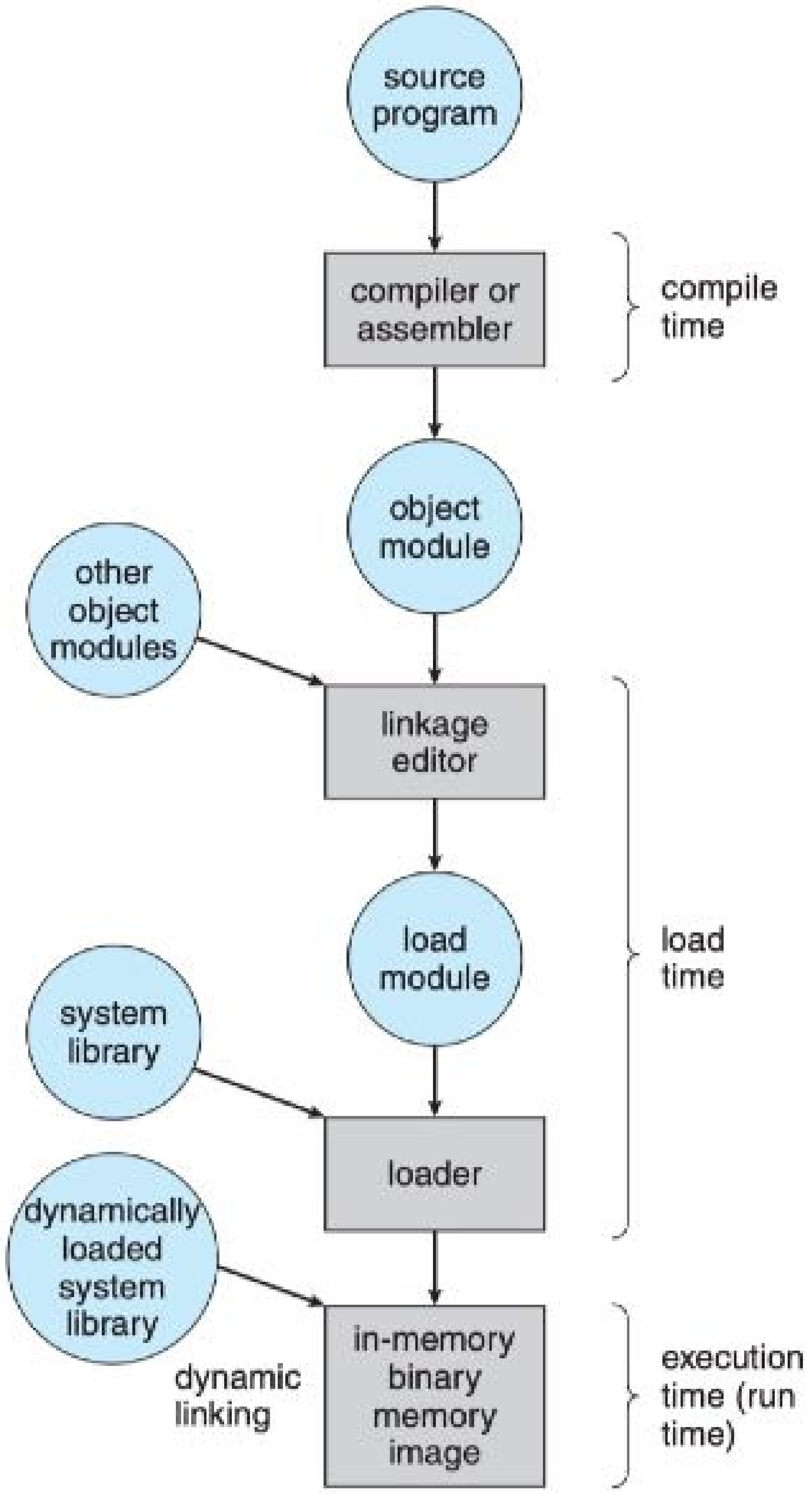


source program
compiler or assembler
compile time
object module
other object modules
linkage editor
load module
load time
system library
loader
dynamically loaded system library
dynamic linking
in-memory binary memory image
execution time (run time)

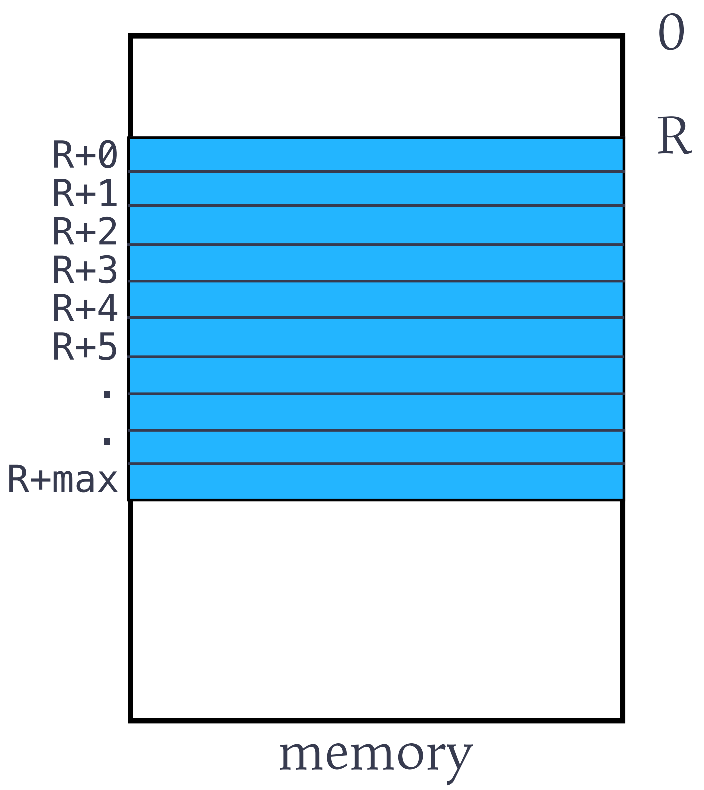


program
R+0
R+1
R+2
R+3
R+4
R+5
.
.
R+max
0
R
memory

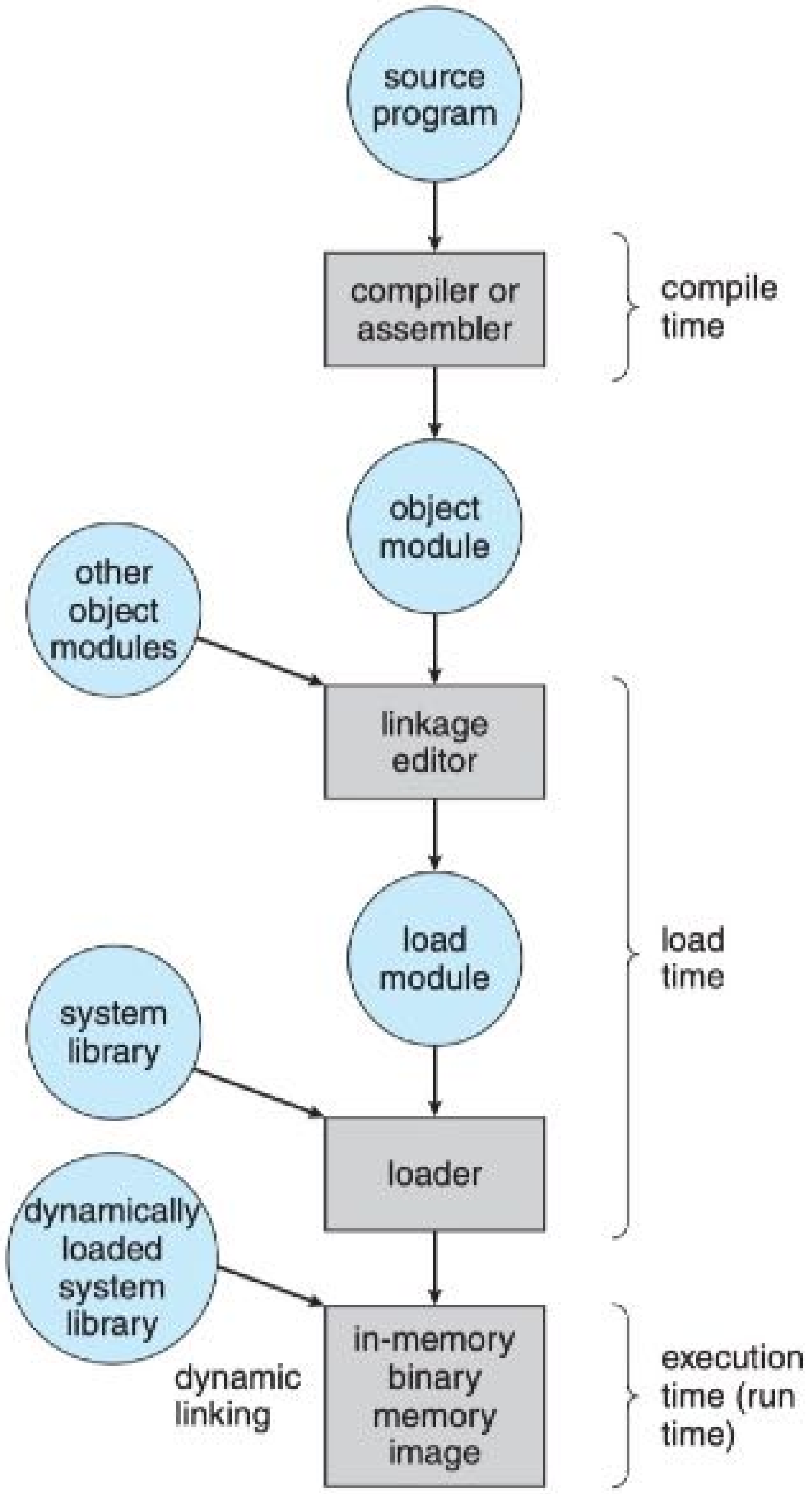
地址转换时机-编译时

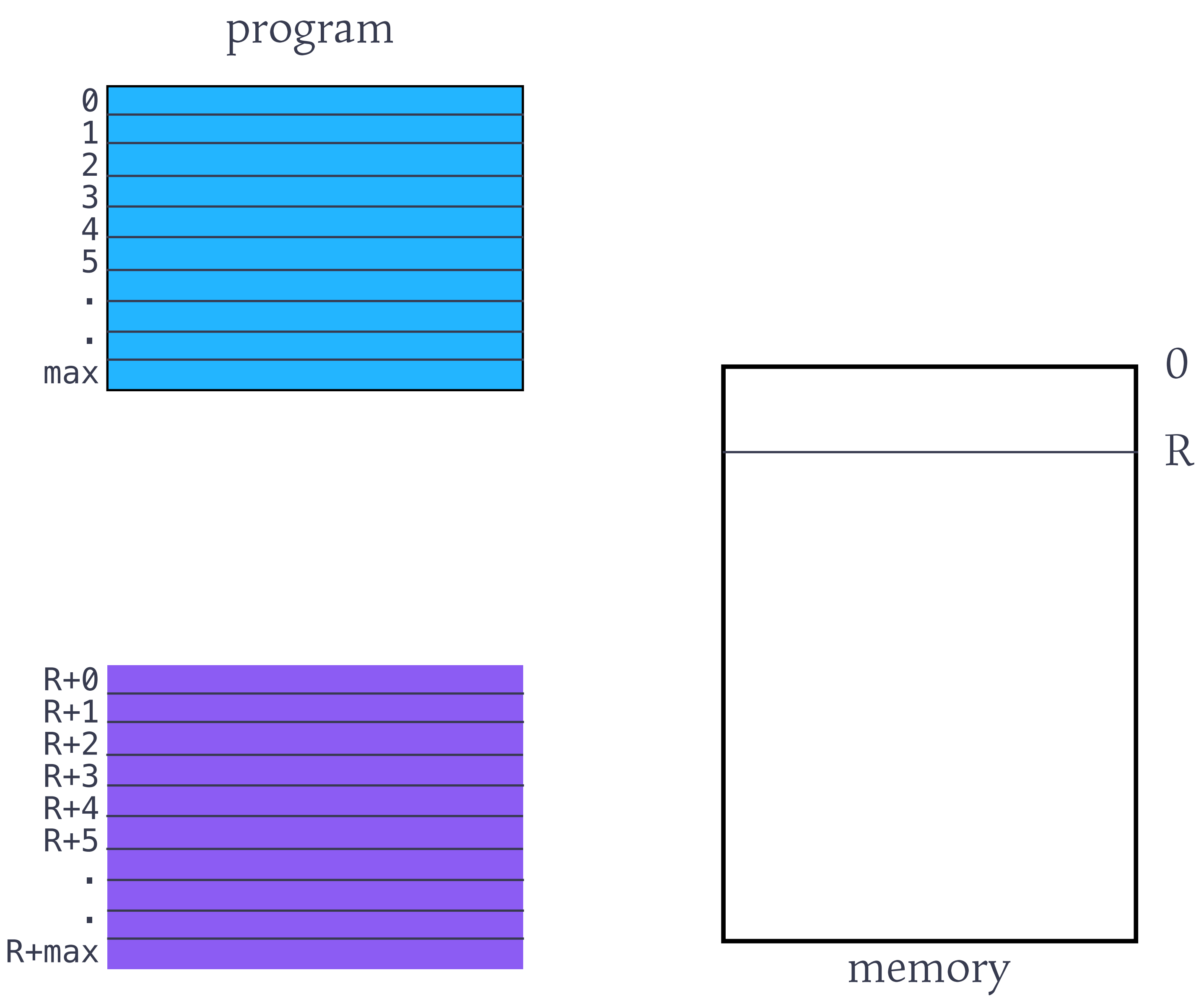


program

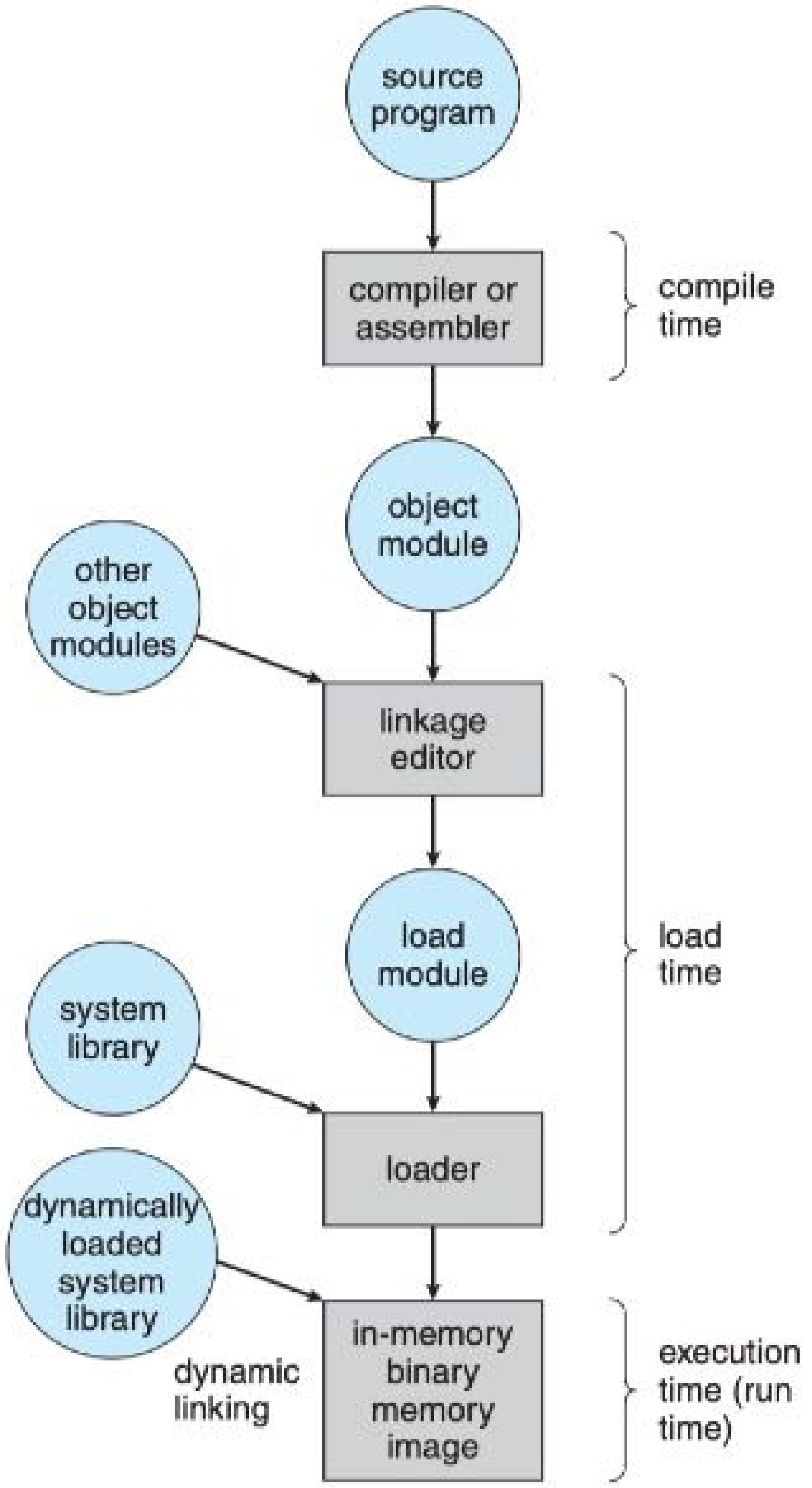


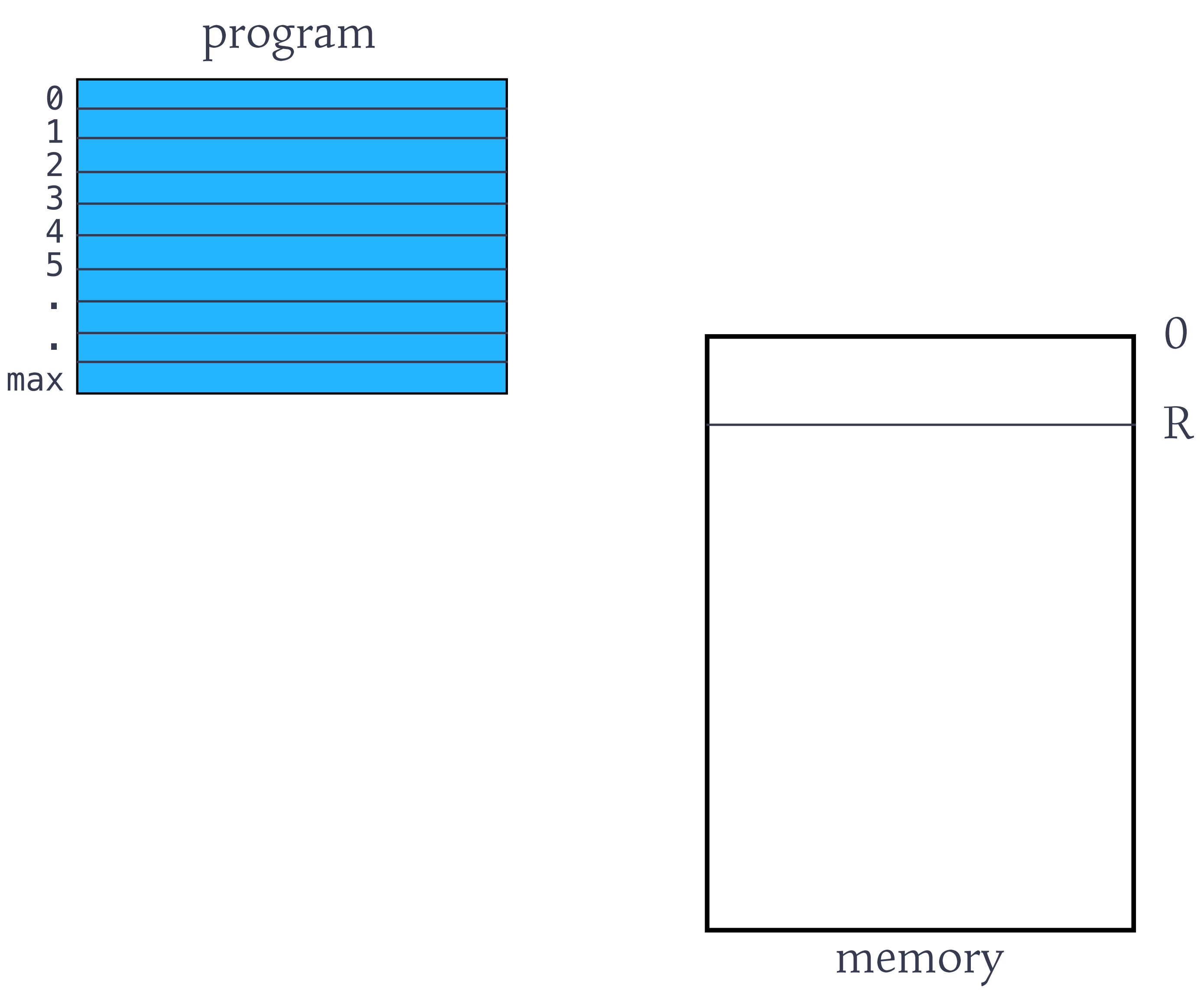
地址转换时机-加载时


source program
compiler or assembler
compile time
object module
other object modules
linkage editor
load module
load time
system library
loader
dynamically loaded system library
dynamic linking
in-memory binary memory image
execution time (run time)

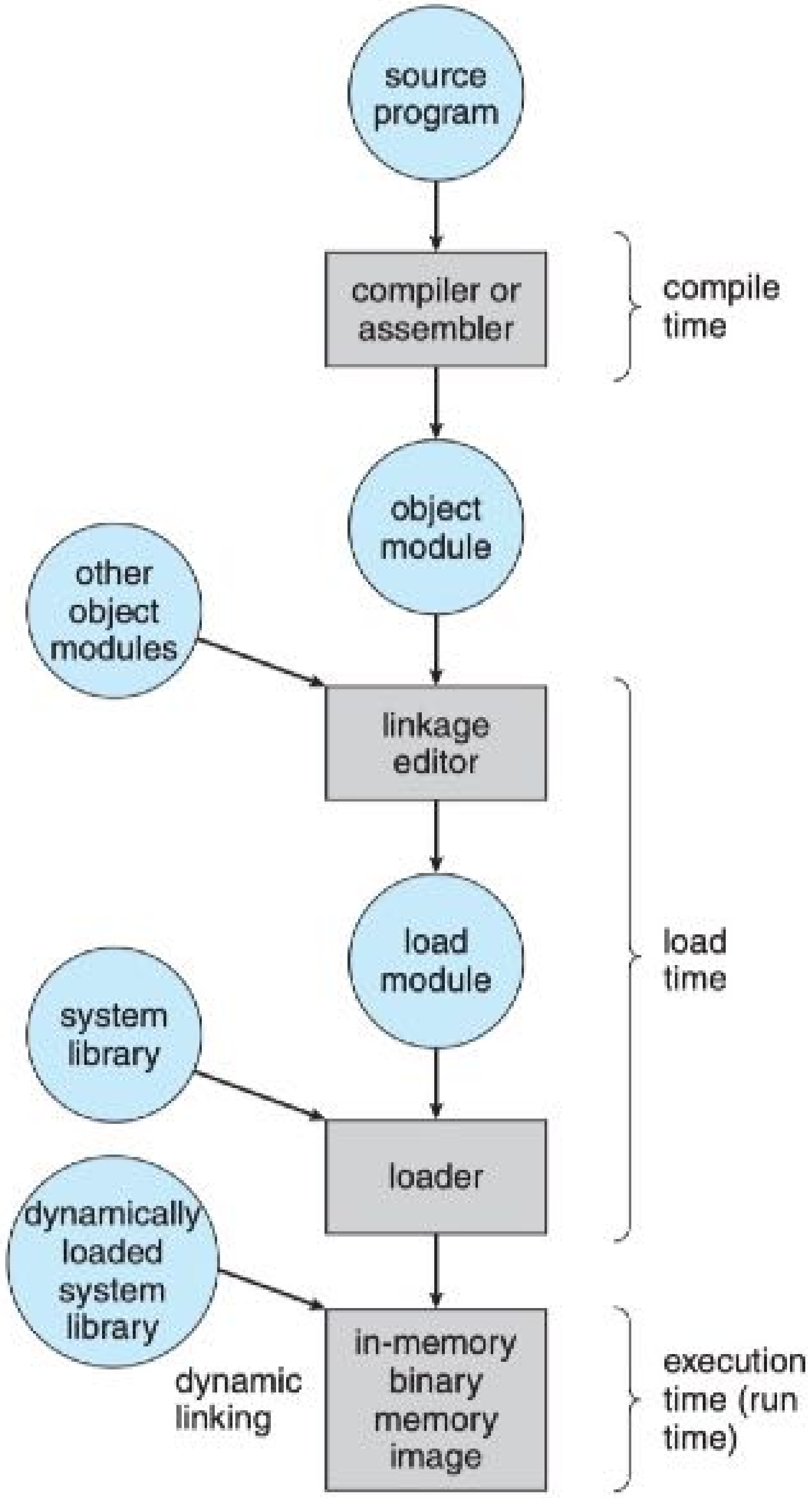

program
0
1
2
3
4
5
.
.
max
R+0
R+1
R+2
R+3
R+4
R+5
.
.
R+max
0
R
memory

地址转换时机-运行时

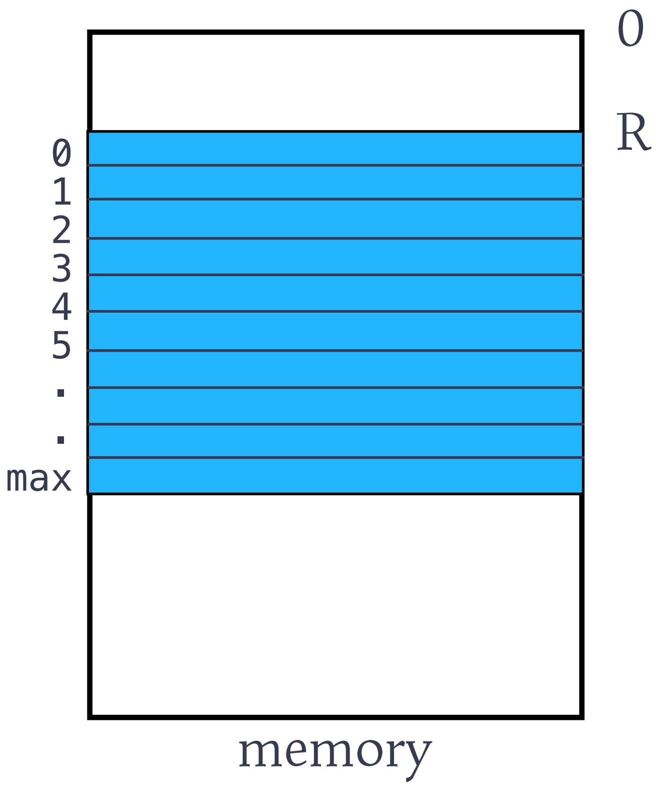

source program
compiler or assembler
compile time
object module
other object modules
linkage editor
load module
load time
system library
loader
dynamically loaded system library
dynamic linking
in-memory binary memory image
execution time (run time)


program
0
1
2
3
4
5
.
.
max
0
R
memory

地址转换时机-运行时

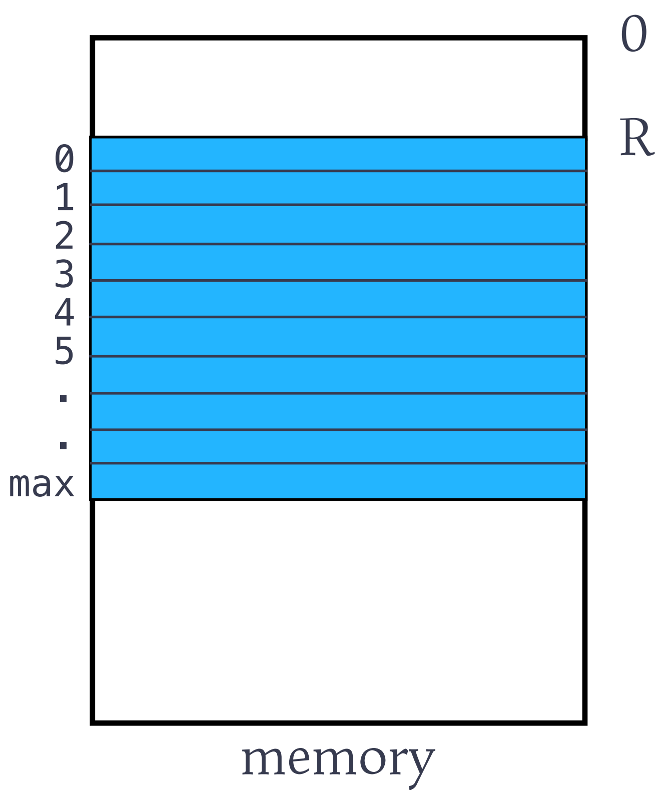


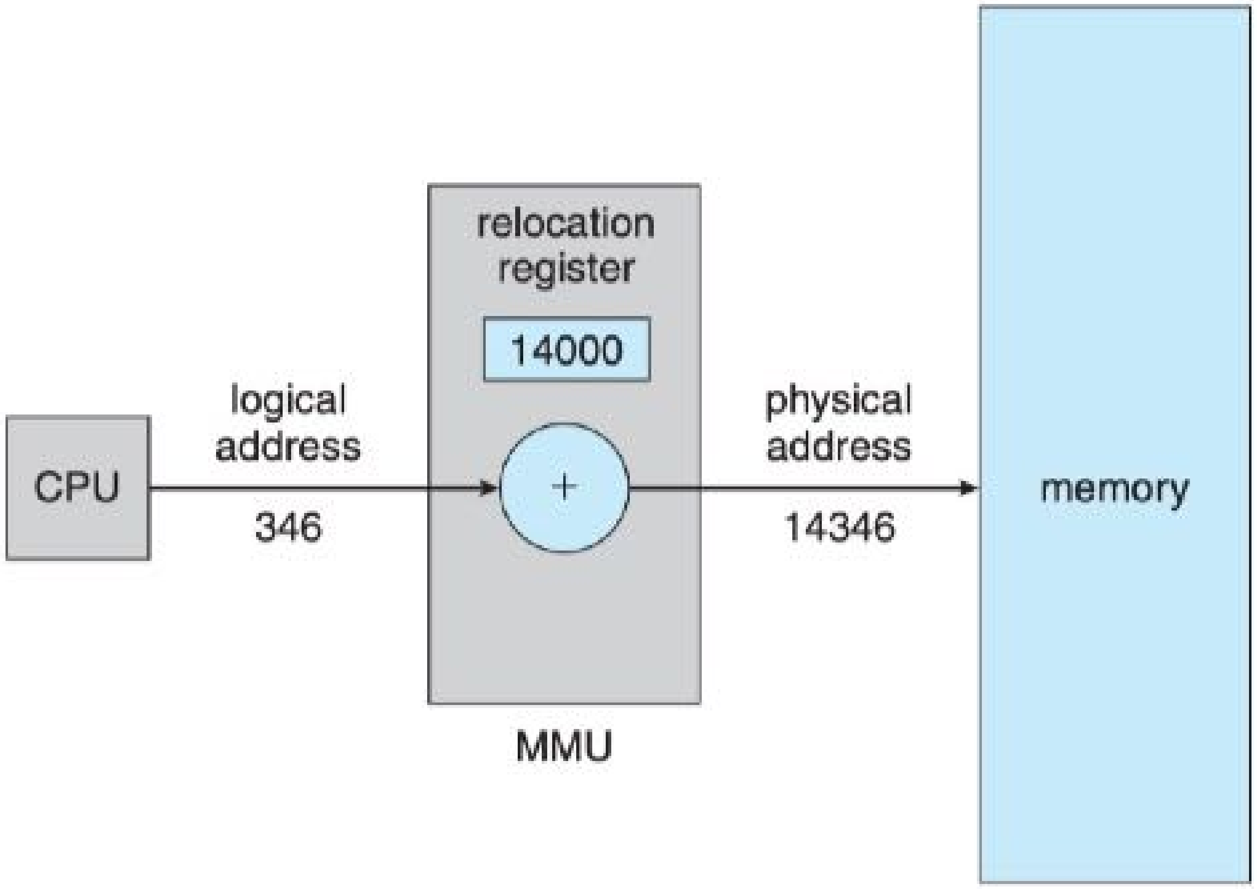
program



内存管理单元MMU

- Memory-Management Unit完成逻辑地址到物理地址运行时的转换工作。
  - 重定位寄存器（relocation register）或基址寄存器





连续内存分配

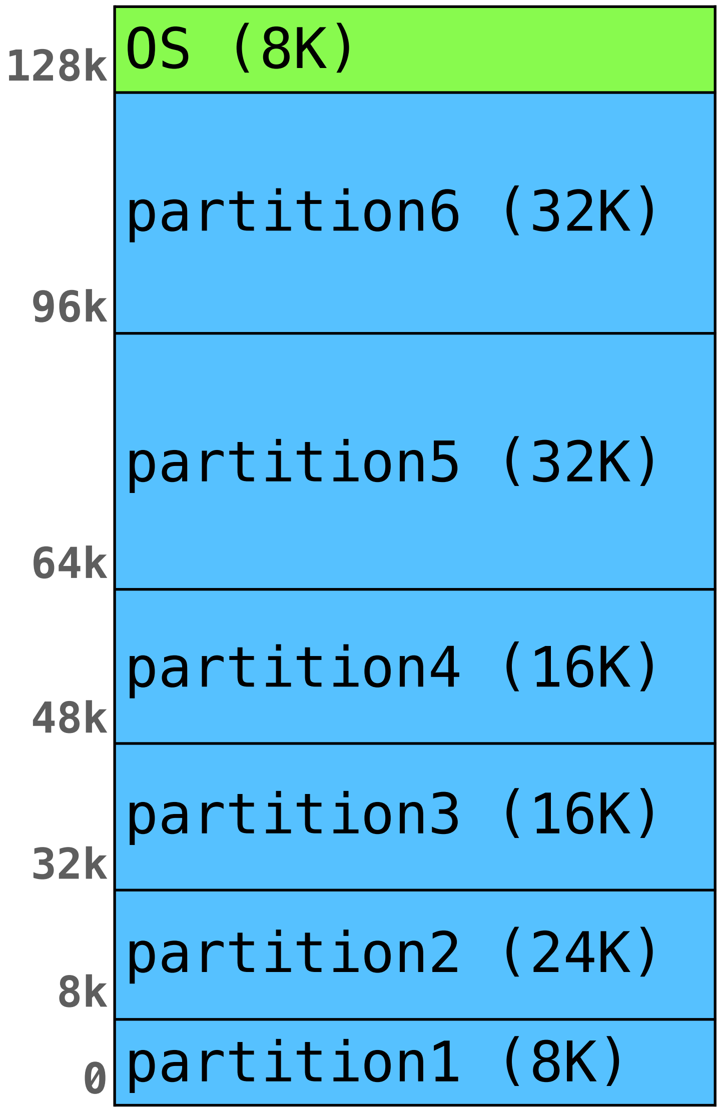
CONTIGUOUS MEMORY ALLOCATION

- In contiguous memory allocation, each process is contained in a single section of memory that is contiguous to the section containing the next process.
    - Memory allocation
    - Memory recycle
    - Memory protection

FIXED-SIZED PARTITION

- Memory is divided to several fixed-sized partitions. Each partition may contain exactly one process.

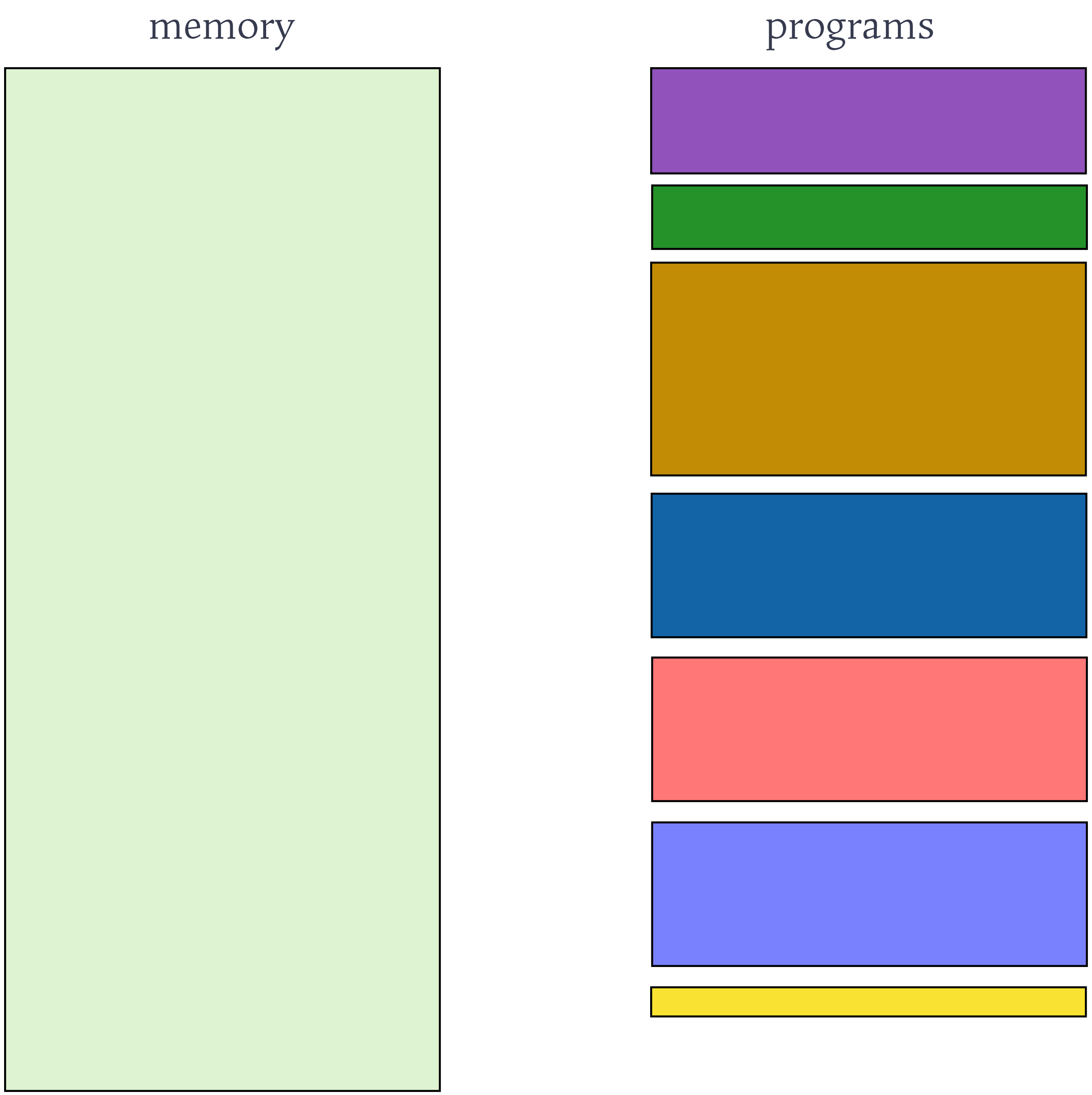
no.	base address	length	occupied
1	0K	8K	0
2	8K	24K	0
3	32K	16K	0
4	48K	16K	0
5	64K	32K	0
6	96K	32K	0



VARIABLE-PARTITION

- In the variable-partition scheme, the operating system keeps two tables indicating which parts of memory are available and which are occupied.
- Initially, all memory is available for user processes and is considered one large block of available memory, a hole.
- Eventually, as you will see, memory contains a set of holes of various sizes.

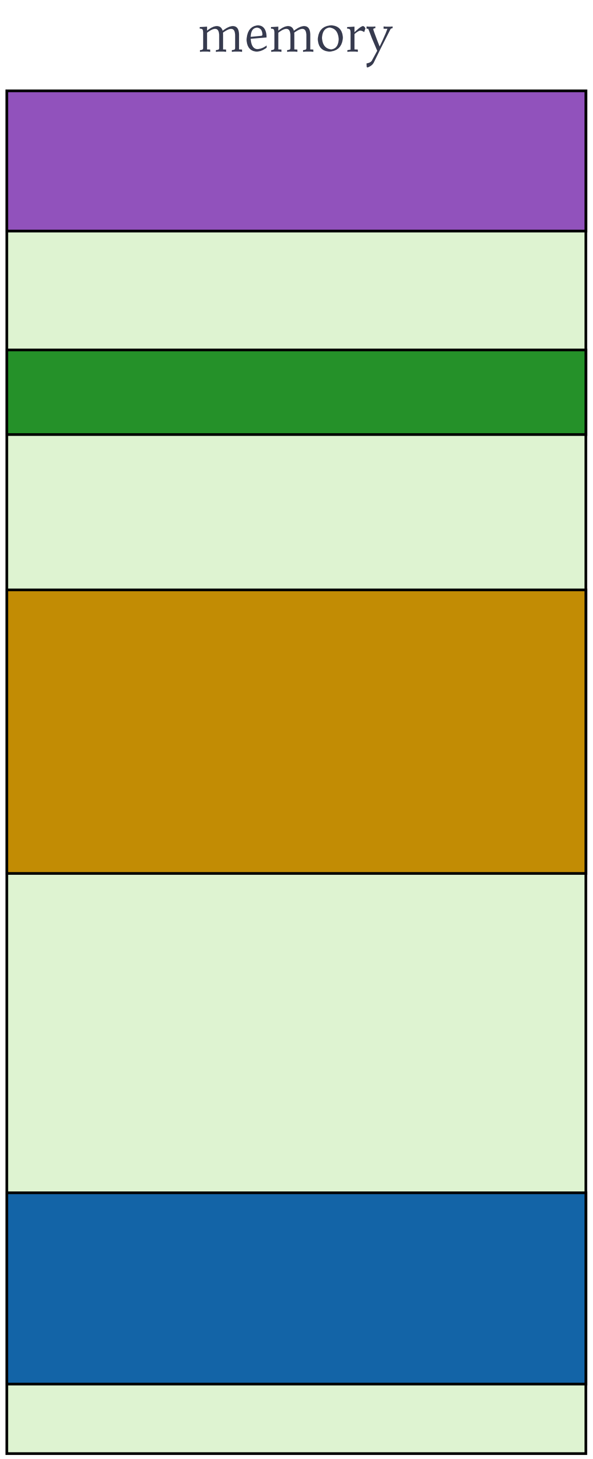
HOLES



memory

programs

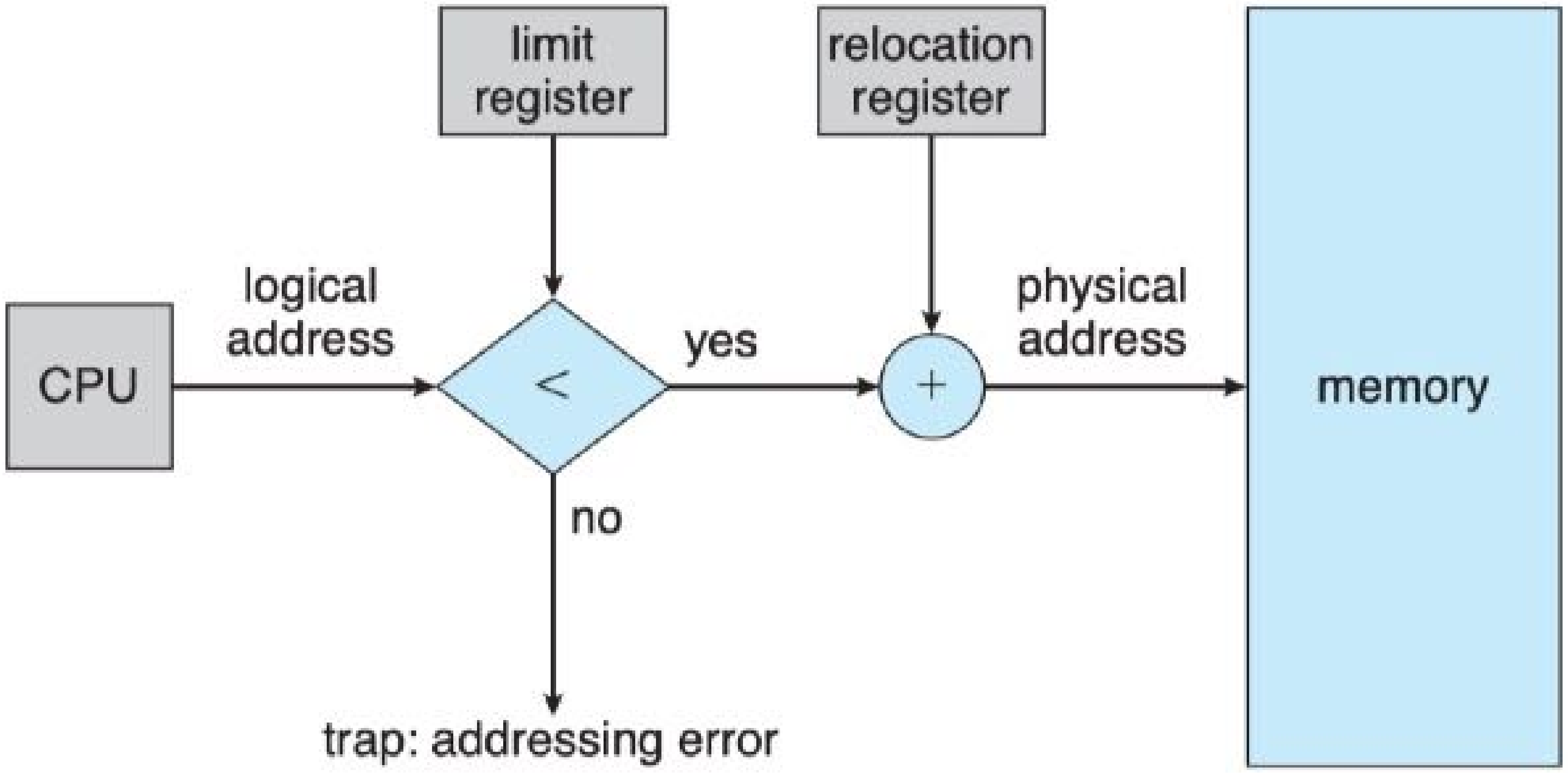
动态存储分配问题



- 首次适应
  - 分配首个足够大的孔，效率最高
- 最佳适应
  - 分配最小的足够大的孔，浪费最小
- 最坏适应
  - 分配最大的孔，产生的剩余孔更可能被再利用

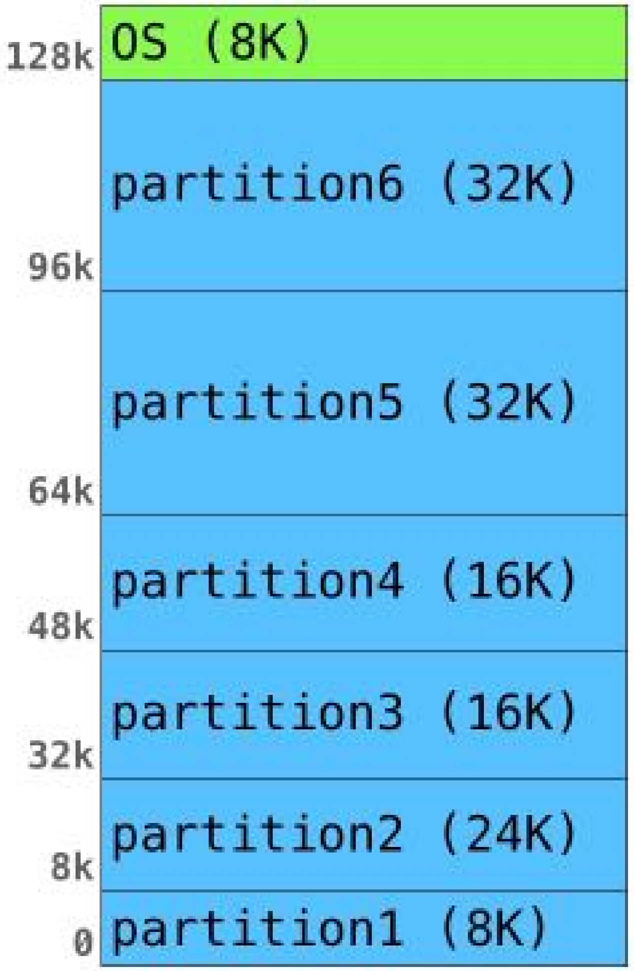
地址转换与保护

- 两种连续分配方案的地址转换方式是相似的：
  - 物理地址 = 基址 + 逻辑地址
- 地址保护策略：与限长limit进行比较



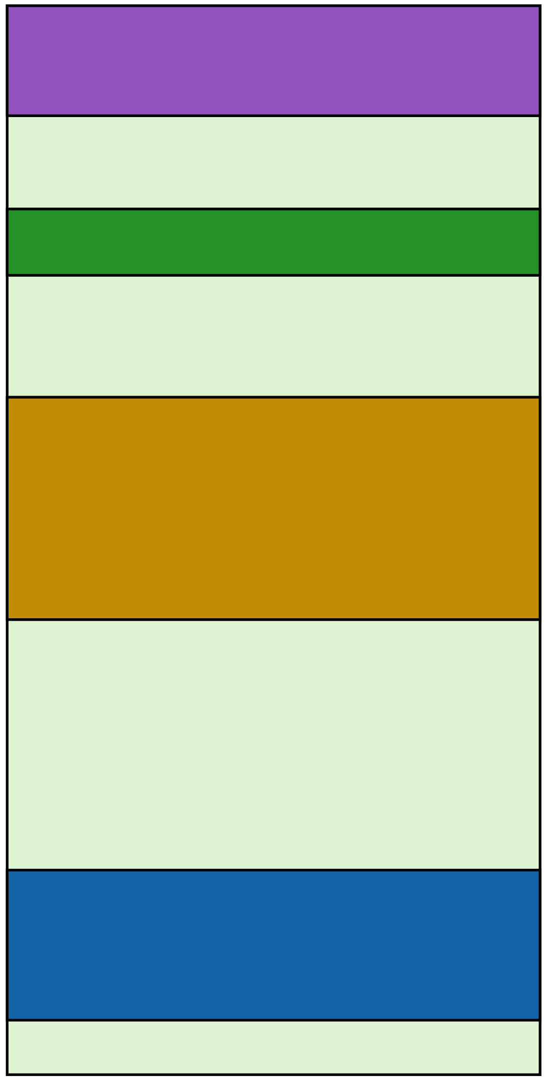
碎片

- Fragmentation: some little pieces of memory hardly to be used.
  - internal fragmentation
  - external fragmentation
    - compaction



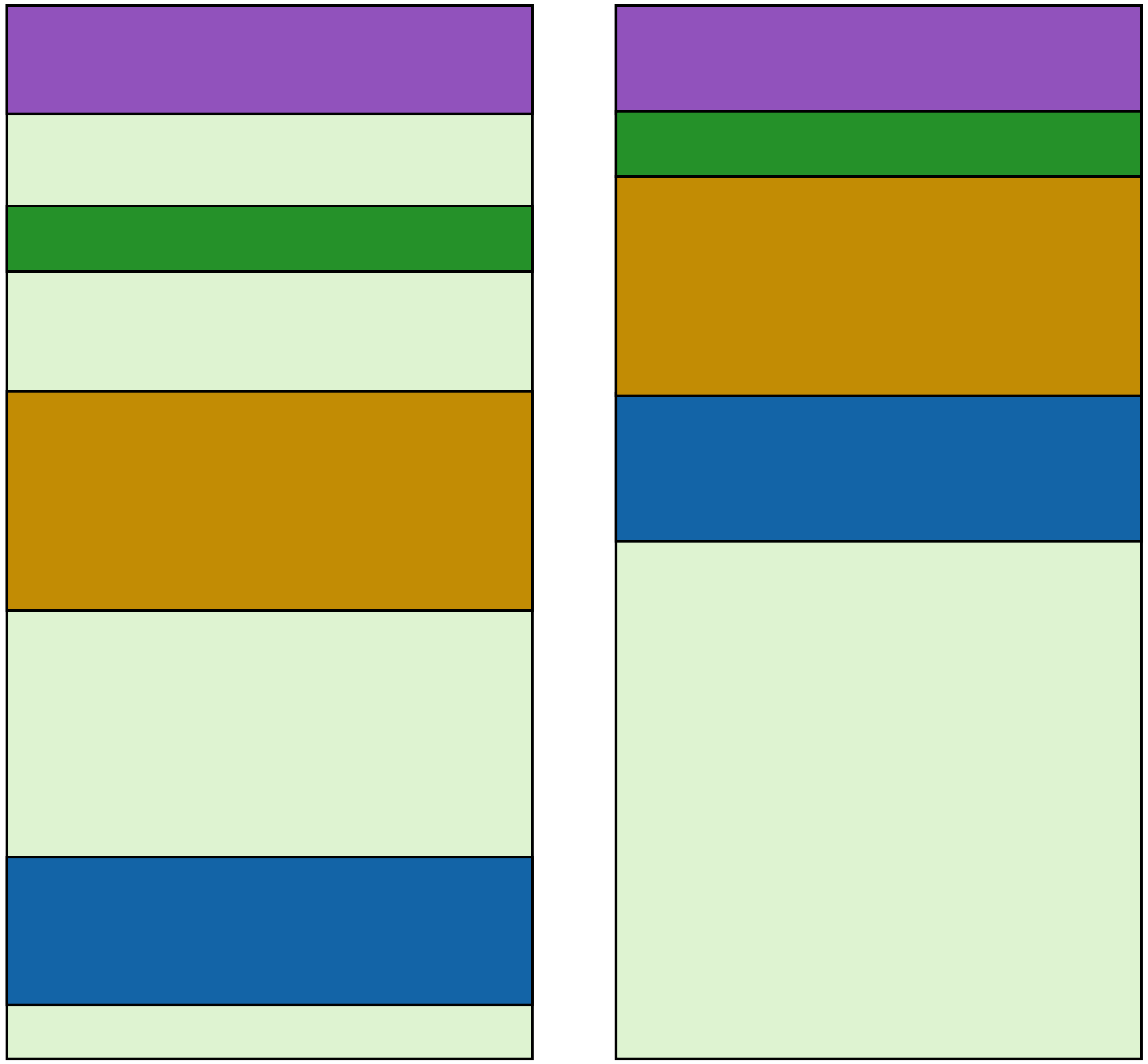
碎片

- Fragmentation: some little pieces of memory hardly to be used.
    - internal fragmentation
    - external fragmentation
        - compaction



碎片

- Fragmentation: some little pieces of memory hardly to be used.
  - internal fragmentation
  - external fragmentation
    - compaction
      - static relocation
      - cost



下期预告

- 下次直播时间：3月18日 上午9:30
- 课程内容
  - Lecture 13 Segmentation and Paging
- Q&A

Lecture 12

The End