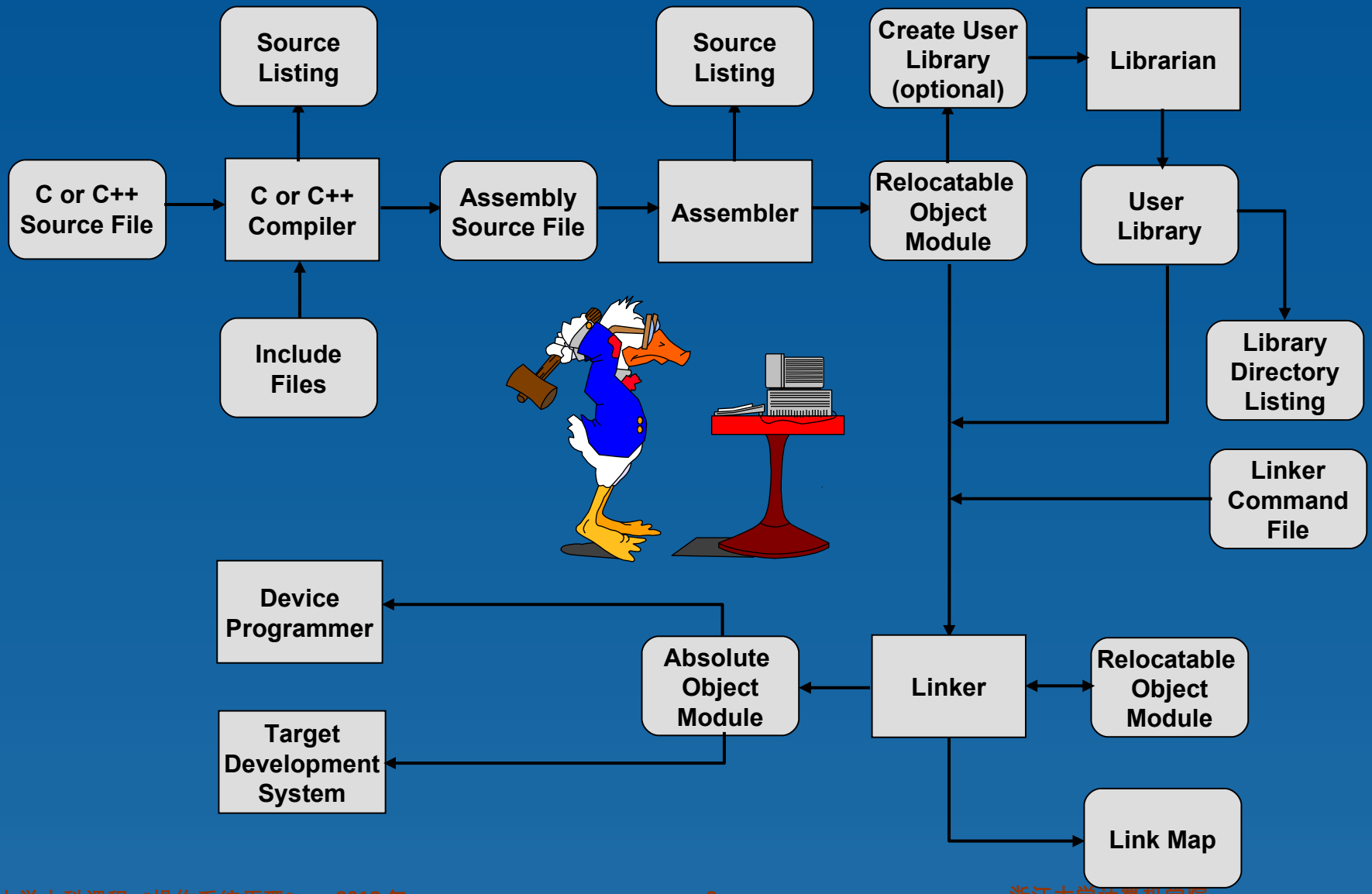
内存管理基本概念

重温一些计算机组成的知识点

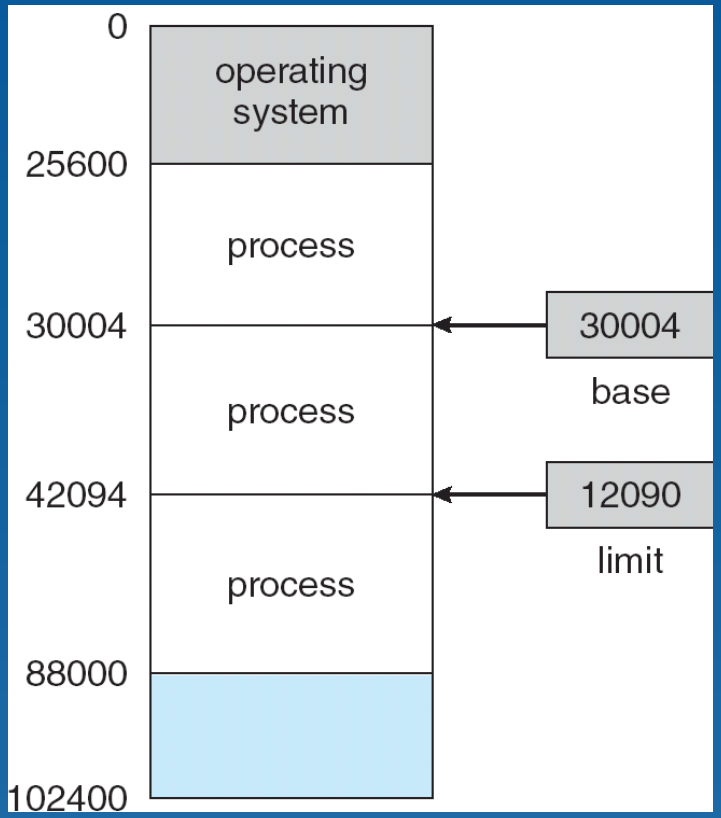
- 程序必须装入内存后，才能（以进程为单位）被 CPU 解释、执行
- CPU 能够直接访问的，只有主存、寄存器
- 访问寄存器需要 1 个 CPU 时钟周期，很快
- 访问主存需要许多时钟周期，或者，需要若干机器周期
- **Cache** 位于主存、寄存器之间

计算机这么处理用户程序



基地址寄存器，界限寄存器

- 基地址寄存器和界限寄存器共同划定逻辑地址空间



汇编器 (Assemblers)

- Assembler 任务
  - 把符号指令翻译成二进制指令
  - 把标号 (labels) 翻译成地址
  - 处理伪指令 (pseudo-ops)
- 区别于编译，它基本上是 one-to-one 的翻译
- 产生的目标文件 (Object file) 添加了
  - Text 段：代码
  - Data 段：初始化了的全程变量
  - BSS 段：未初始化的全程变量

符号表 (Symbol table)

- 利用 program location counter (PLC)
- 顺序扫描汇编程序，同步递增 PLC，使 PLC 总是对应所在位置的地址
- 二进制地址在汇编翻译过程中生成，不是在程序执行时。

符号表的示例

```
0x00        ADD ax,  xx; A90000
0x03        ADD r0,r1,r2
0x07   xx   ADD r3,r4,r5
0x0B        CMP r0,r3
0x0F   yy   SUB r5,r6,r7
```

xx = 0x07
yy = 0x0F

Symbol Table

- 两遍扫描
  - Pass 1: 生成符号表
  - Pass 2: 生成二进制指令

指令和数据的地址绑定 (Binding)

- 指令和数据的地址绑定通常发生在 3 个阶段
  - **编译时（学习汇编）**：如果代码、数据的存放首地址已知，编译阶段即可确定绝对地址。如果首地址变更，则需要重新编译

指令和数据的地址绑定（续）

- **装入时（学习单用户 OS)**：如果代码、数据的存放首地址未知，编译阶段生成可重定位地址，装入时才确定绝对地址
- **执行时（学习 OS 原理）**：如果允许进程在执行时迁移其代码、数据，那么，地址绑定也在执行时进行。需要硬件装置支持其地址映射（e.g.，基地址寄存器和界限寄存器）

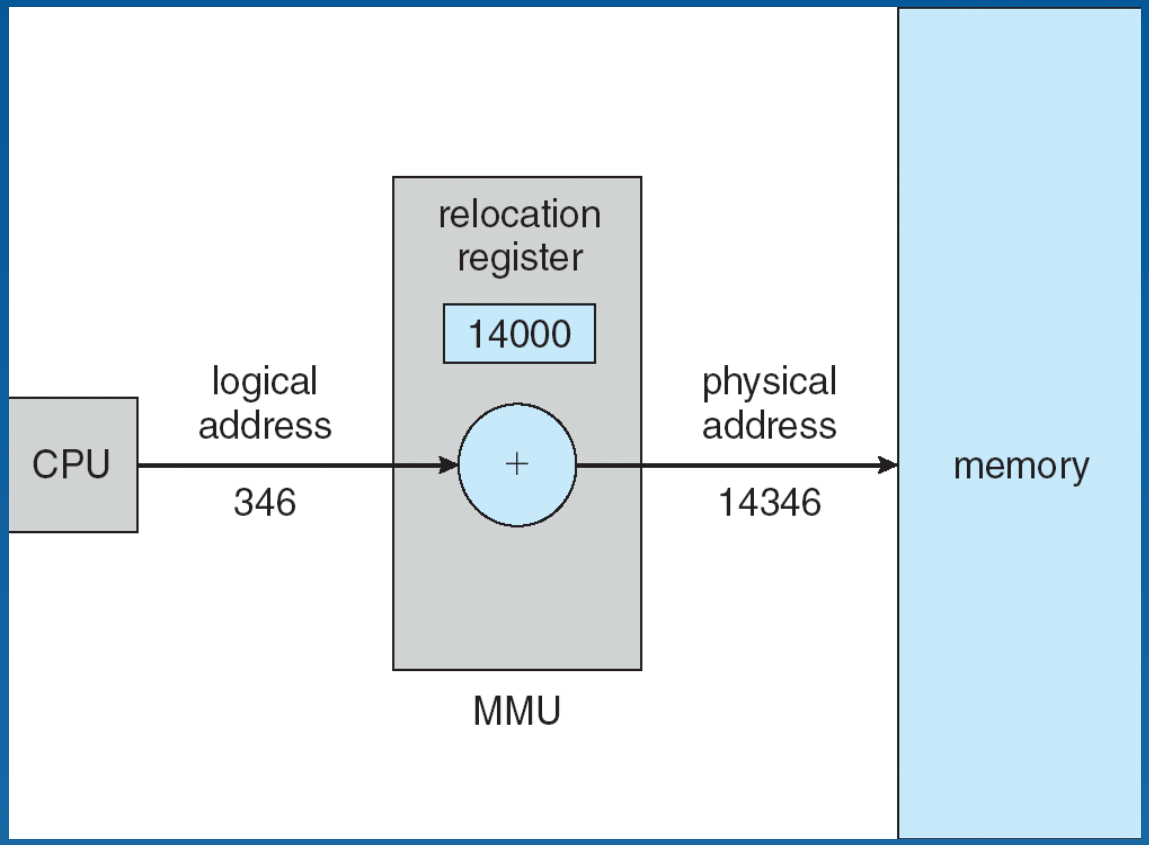
逻辑地址空间 vs. 物理地址空间

- 区别于物理地址空间的各种逻辑地址空间，是 OS 得以管理内存的必要条件
  - **逻辑地址** – generated by the CPU; also referred to as **virtual address**
  - **逻辑地址 – 非物理的各种地址标记**
  - **物理地址** – 内存单元接收到的地址。也就是说，“浮现”在地址总线的地址，以二进制形式表达。
    - 逻辑地址包括符号名，包括编译、汇编、连接、装入操作产生的地址

存储管理单元（MMU)

- 存储管理单元 (Memory-Management Unit ， MMU) 是 CPU 内部的硬件装置，其功能是将虚拟地址 (或逻辑地址) 转换成物理地址
- 例如一种简单的 MMU 策略，在用户进程将逻辑地址送往地址总线前， MMU 把重定位寄存器 (relocation register) 的值，加到这个逻辑地址
- 用户进程只能处理逻辑地址，它无法获取真正的物理地址

基于重定位寄存器的动态重定位 (Dynamic relocation)



动态连接 (Dynamic Linking)

- 进程即将用到的代码段，不被预先连接入程序，只有到真正被调用的时刻才连接
- 需要动态连接库 (Linux 的 .so ， 或者 Windows 的 .dll) 的配合
- 设计一小段代码，称 *stub* 。
- 当真正调用到该段代码时，通过 stub 定位该段代码，或者从外部装入内存
- 流程？

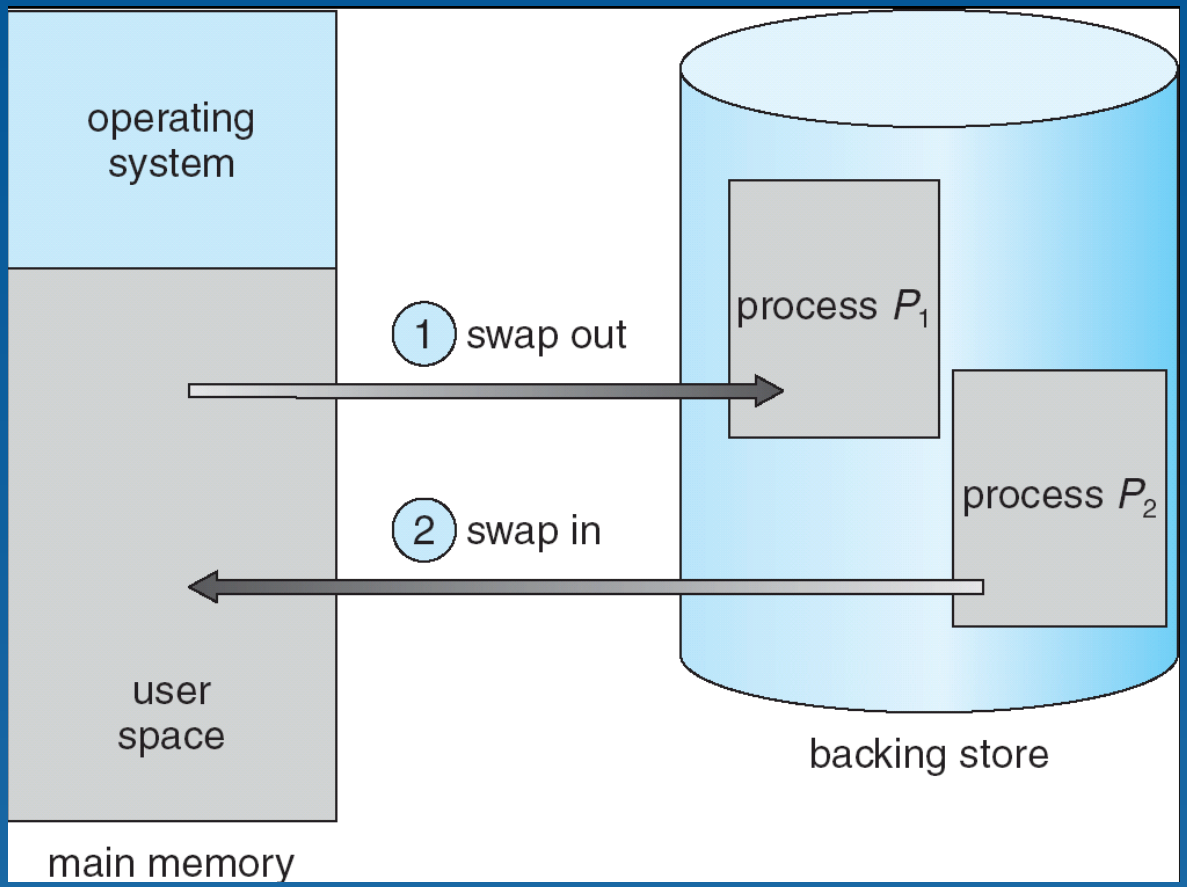
动态装入 (Dynamic Loading)

- 进程即将用到的子程序，不被预先装入，只有到真正被调用的时刻才装入内存
- 这样，进程本次运行中没有调用的子程序，就不会被装入内存
- 更有效地利用了内存空间
- 不需要操作系统特别的支持。例如 SOA 。只要 ？

交换 (Swapping)

- 进程映像暂时传输至后备存储空间保存 (换出， swap out) ， 需要 (执行) 时再装入内存 (换入， swap in) 。称为“交换”
- 后备存储空间
  - 快速;
  - 大容量;
  - 直接访问机制 (direct access)
- 交换操作结合 CPU 调度算法，使得及时换出低优先级的进程，让高优先级的进程装入、执行

图示：交换过程


operating system
process P_1
1 swap out
process P_2
2 swap in
user space
backing store
main memory

交换（续）

- 大部分交换时间用于传输。传输时间与交换数据量成正比
- 交换的思想或者变种，频频见诸于 UNIX 、 Linux 、 Windows 等
- 系统只要保障就绪队列里的就绪进程全部驻留内存。其它进程映像可以被换出

存储管理基本思想

- y = f(x)

- 存储管理算法的评估和比较， p309
  - 硬件支持， Hardware Support
  - 性能， Performance
  - 碎片， Fragmentation
  - 重定位， Relocation
  - 交换， Swapping
  - 内存共享， Memory sharing
  - 内存保护， Memory protection


End