

Memory Management

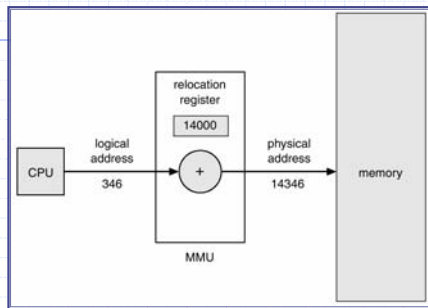
Lectures notes from the text
supplement by Siberschatz and Galvin
Modified by B.Ramamurthy

Binding of Instructions and Data to Memory

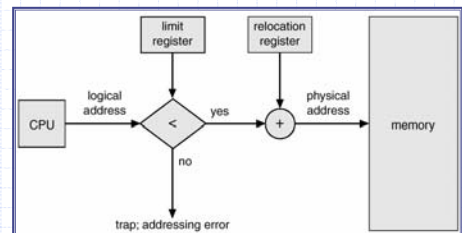
Address binding of instructions and data to memory addresses can happen at three different stages.

- ◆ **Compile time:** If memory location known a priori, absolute code can be generated; must recompile code if starting location changes.
- ◆ **Load time:** Must generate *relocatable* code if memory location is not known at compile time.
- ◆ **Execution time:** Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps (e.g., *base* and *limit registers*).

Dynamic relocation using a relocation register



Hardware Support for Relocation and Limit Registers



Dynamic Linking

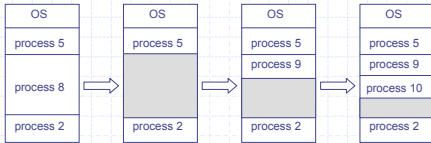
- ◆ Linking postponed until execution time.
- ◆ Small piece of code, *stub*, used to locate the appropriate memory-resident library routine.
- ◆ Stub replaces itself with the address of the routine, and executes the routine.
- ◆ Operating system needed to check if routine is in processes' memory address.
- ◆ Dynamic linking is particularly useful for libraries and remote object access.

Contiguous Allocation

- ◆ Main memory usually into two partitions:
 - Resident operating system, usually held in low memory with interrupt vector.
 - User processes then held in high memory.
- ◆ Single-partition allocation
 - Relocation-register scheme used to protect user processes from each other, and from changing operating-system code and data.
 - Relocation register contains value of smallest physical address; limit register contains range of logical addresses – each logical address must be less than the limit register.

Contiguous Allocation (Cont.)

- Multiple-partition allocation
 - Hole** – block of available memory; holes of various size are scattered throughout memory.
 - When a process arrives, it is allocated memory from a hole large enough to accommodate it.
 - Operating system maintains information about:
 - allocated partitions
 - free partitions (hole)



10/16/2005

7

Dynamic Storage-Allocation Problem

How to satisfy a request of size n from a list of free holes.

- First-fit:** Allocate the *first* hole that is big enough.
- Best-fit:** Allocate the *smallest* hole that is big enough; must search entire list, unless ordered by size. Produces the smallest leftover hole.
- Worst-fit:** Allocate the *largest* hole; must also search entire list. Produces the largest leftover hole.

First-fit and best-fit better than worst-fit in terms of speed and storage utilization.

10/16/2005

8

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.
 - I/O problem
 - Latch job in memory while it is involved in I/O.
 - Do I/O only into OS buffers.

10/16/2005

9

Paging

- Logical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available.
- 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**.
- 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.
- Internal fragmentation.

10/16/2005

10

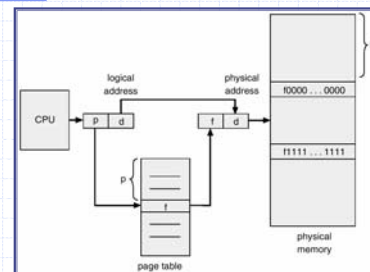
Address Translation Scheme

- 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.

10/16/2005

11

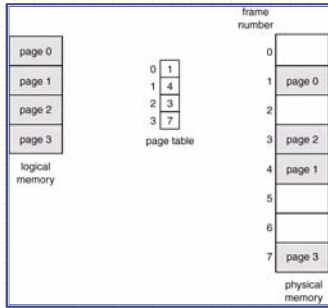
Address Translation Architecture



10/16/2005

12

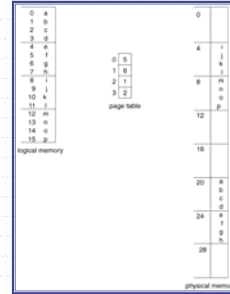
Paging Example



10/16/2005

13

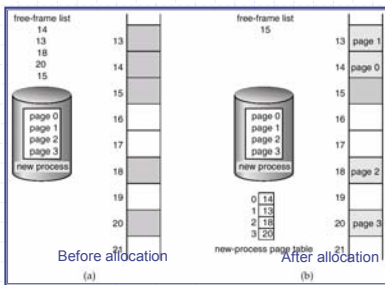
Paging Example



10/16/2005

14

Free Frames



10/16/2005

15

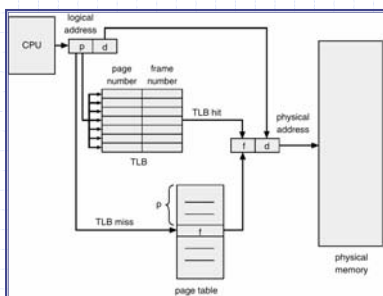
Implementation of Page Table

- ◆ Page table is kept in main memory.
- ◆ *Page-table base register (PTBR)* points to the page table.
- ◆ *Page-table length register (PRLR)* indicates size of the page table.
- ◆ In this scheme every data/instruction access requires two memory accesses. One for the page table and one for the data/instruction.
- ◆ The two memory access problem can be solved by the use of a special fast-lookup hardware cache called *associative memory* or *translation look-aside buffers (TLBs)*

10/16/2005

16

Paging Hardware With TLB



10/16/2005

17

Summary

- ◆ We looked at simple paging scheme.
- ◆ This is the model for memory management you will implement in Project 1.
- ◆ Next class we will look at demand paging.

10/16/2005

18