Scribe Notes

Computer Systems (CS5600)

10/06/2010
Hour 2 (7.00pm – 8.00pm)

Memory Management

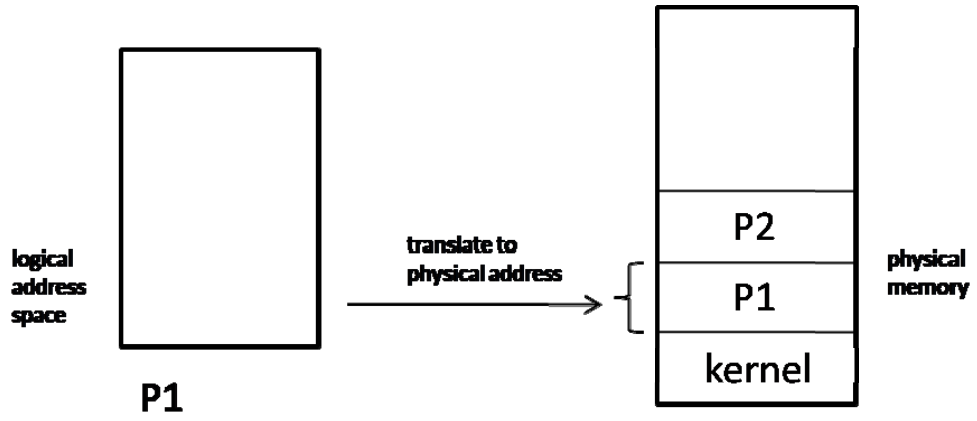
(see Hour 1 notes for the beginning of the topic)

Every process has the same view of memory (a global unit).

Why it is bad? Fragmentation.

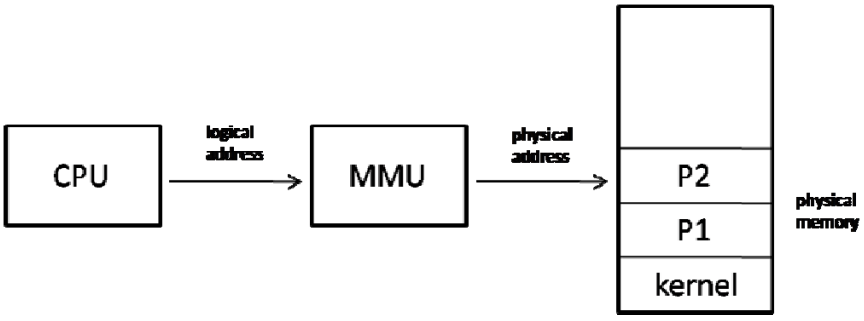
- every program needs to keep track of several regions of memory

What to do? Use logical ('fake') address – memory address that a process uses.



Why it is bad? Translation needs to be done every time; it takes time.
Need hardware support.

Memory Management Unit (MMU)

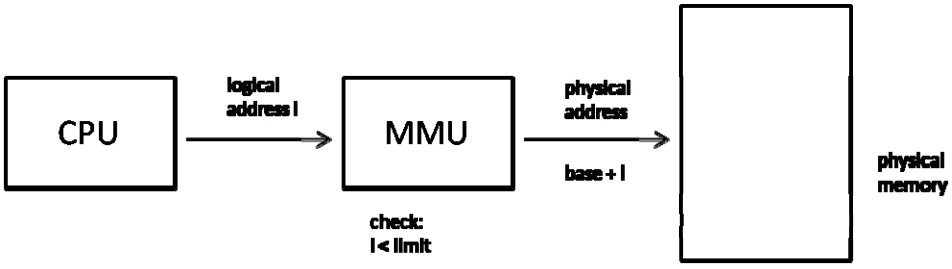


MMU translates from logical memory addressing to physical addressing.

Three MMU Strategies

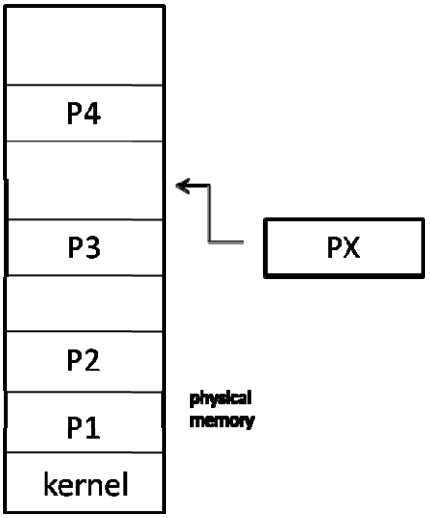
1. Contiguous memory allocation
Each process is contained in a single contiguous section of memory.

There are 2 regions for MMU to map: base and limit.



(The MMU above is the simplest MMU)

How to do that? Choose same, fixed size of memory for every process.
Why it is bad? Limited.
What to do? Choose different size of memory for every process.



A block of available memory is called a hole. On the diagram above, memory contains a set of holes of various sizes, scattered throughout the memory.

When processes are loaded and removed from memory, free memory space is broken into little pieces. External fragmentation exists when there is enough total memory space to satisfy a request, but the available spaces are not contiguous, which means that storage is fragmented into a large number of small holes.

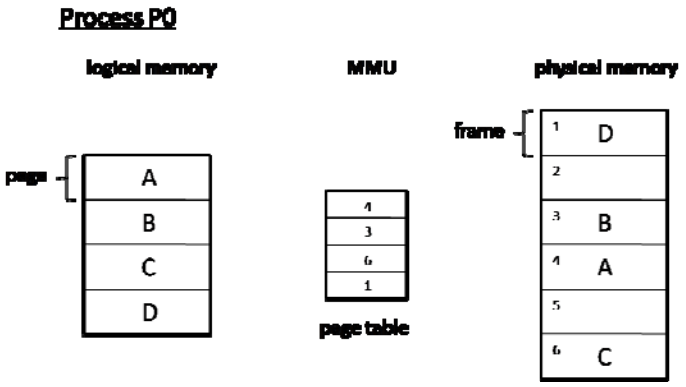
What to do if a new process is to be put into the physical memory, but there is no more space (no big enough hole)? Reorganize and find size from holes (defragment).
Why it is bad? Expensive.

The memory could be broken into fixed-sized blocks and allocated in units based on block size. With this approach, the memory allocated to a process may be slightly larger than the requested memory. The difference between these two numbers is called internal fragmentation – unused memory that is internal to a partition.

2. Paging

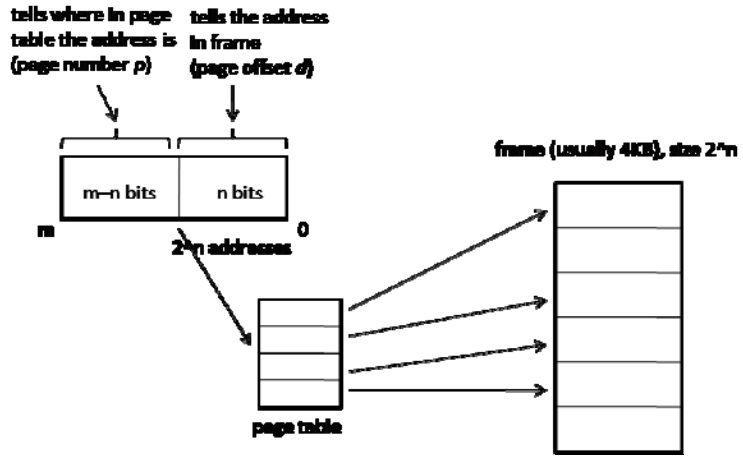
Paging is a memory-management scheme that permits the physical address space of a process to be noncontiguous.
For paging, we need to have more hardware support, more advanced MMU.

Small, fixed-sized chunks of physical memory are called frames. They are defined by hardware and are usually of size powers of 2. Small, fixed sized chunks of logical memory are called pages.

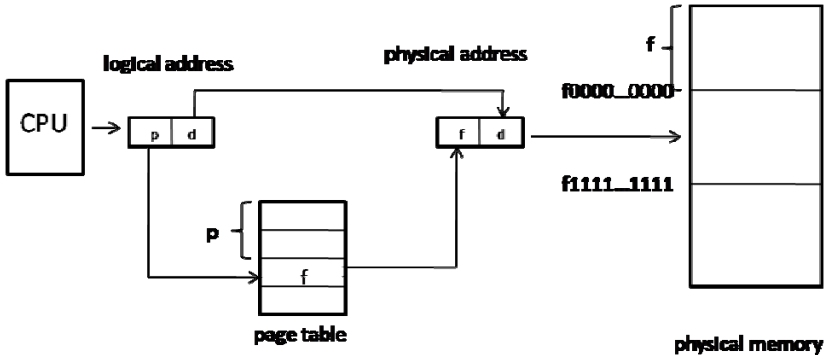


Page table maps pages to frames. Every process has a page table.

How to implement page tabling?



In simple, page table is just an array of addresses. Page number is globally unique.



A page is same size as frame.

Page table base register (PTBR) points to the beginning of the page table.

To keep track of frames, there is a frame table. In simple, it is just an array A of length of number of frames. Each position i in array A (A[i]) tells whether a frame is allocated to some process or not (and if yes, to which process it is allocated).

The page table is usually in a Translation Lookaside Buffer (TLB), which is a small, fast-lookup hardware cache. It is associative, high-speed memory. Each TLB consists of 2 parts: a key (or tag) and a value. When the associative memory is presented with an item, the item is compared with all keys simultaneously. If the item is found, the corresponding value field is returned.

To provide correctness, we can:

1) flush the TLB
2) use Address Space Identifier (ASID). An ASID uniquely identifies each process and is used to provide address-space protection for that process.
   <asid, page number, frame number>
   ASID ≅ process id
   for example, <0, 0, f_i>, <2, 0, f_j>