OPERATING SYSTEMS

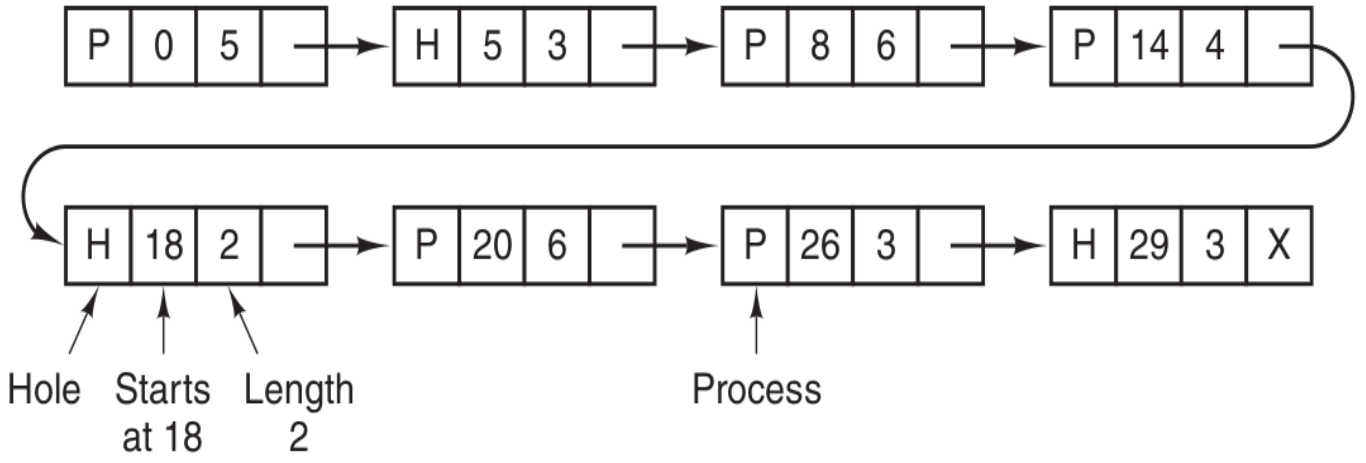
by
Marwa Yusuf

Lecture 13
Thursday 17-12-2020

Chapter 3 (3.2 to 3.3.3)
Memory Management

Allocation Algorithms

- **First Fit**
- **Next Fit**
  - Slightly worse.
- **Best Fit**
  - Slower.
  - More waste.
- **Worst Fit**
  - Not that good.



- Ex: A space of 2 is needed. Then another 2.

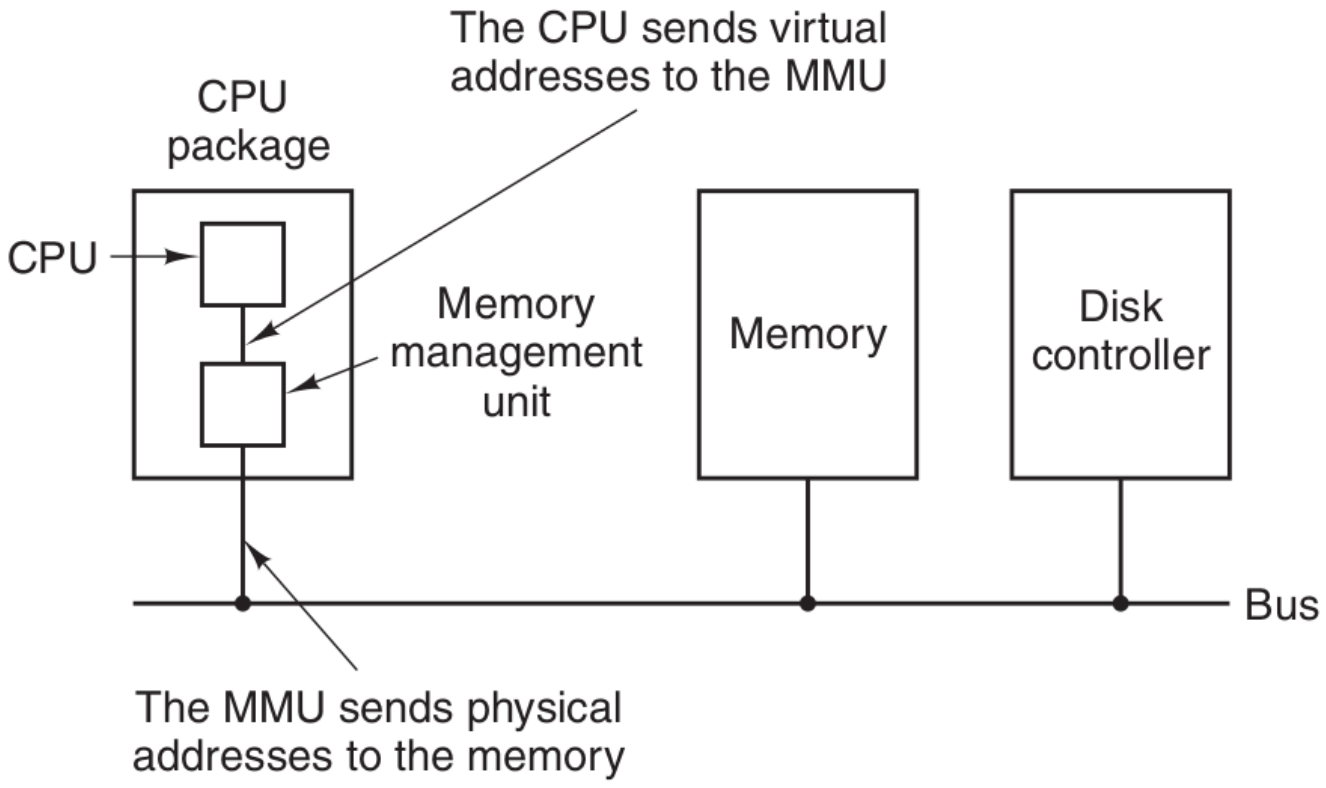
Allocation Algorithms Optimizations

- Keeping separate lists for processes and holes → enhances allocation time and hurts and complicates deallocation.
- If holes list sorted by size → **best fit** and **first fit** are equal, **next fit** is pointless.
- Keep holes list in the holes themselves. Instead of 3 words and a bit for each node, 2 words (size and next).
- **Quick Fit**:
  - Separate lists for common sizes.
  - A table where each entry as a pointer to a list of a specific size (4, 8, 12).
  - 21 KB can be put into 20 list or in odd sizes list.
  - Finding a hole is really fast.
  - Merging deallocated holes is really complicated.
- With no merging, quick severe fragmantation.

Virtual Memory

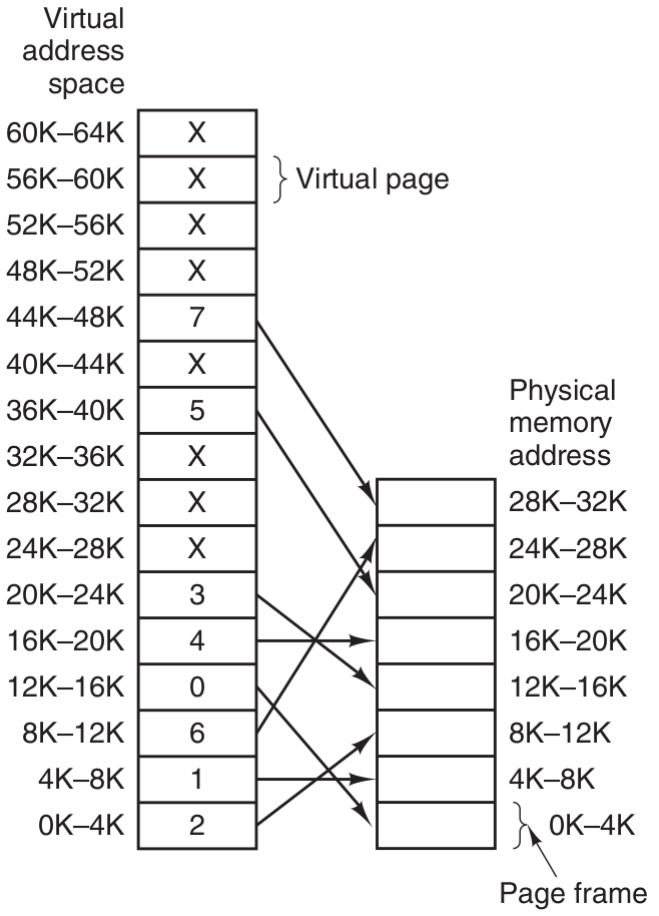
- Constant need for more memory (time-sharing, multimedia … etc.) for one process and for mutiple processes.
- Swapping is slow (seconds to swap 1 GB out and seconds to swap in).
- Large programs problem is old (scientific applications).
  - In 1960s, **overlays** were used, managed by **overlay manager**.
  - OS swaps them, but the programmer defines them.
    - Time consuming, boring and error prone.
- **Virtual memory** was the solution.
  - A program is divided into small chunks called pages.
  - Only needed pages reside in memory, while others stay on disk.
  - A generalization of base and limit with smaller memory chunks.

Paging


The CPU sends virtual addresses to the MMU
CPU package
CPU
Memory management unit
Memory
Disk controller
Bus
The MMU sends physical addresses to the memory

Paging (cont.)

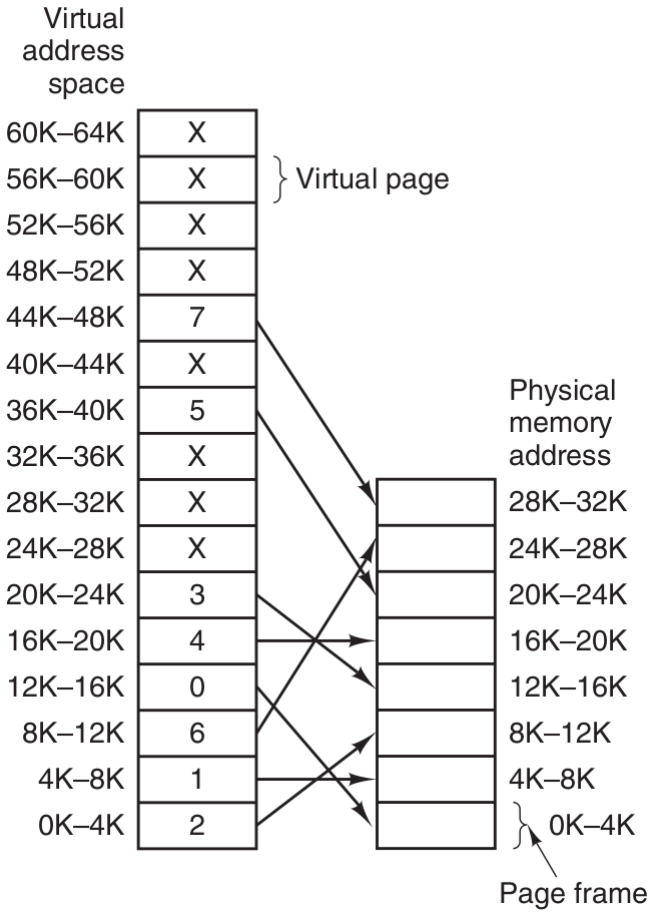
- **Virtual addresses** of 16 bits are generated. The **virtual address space is** from 0 to 64KB-1.
- Memory itself is 32KB.
- The whole program is kept in disk.
- **Virtual address space** consists of **pages**, while physical memory consists of **page frames**, both of the same size (here =4KB).
- We have here 16 pages and 8 page frames.
- Transfers are in whole pages.
- A mix of page sizes can be used in some systems (x86-64 has 4KB for users, 2 MB, 1GB for kernel)



Paging (cont.)

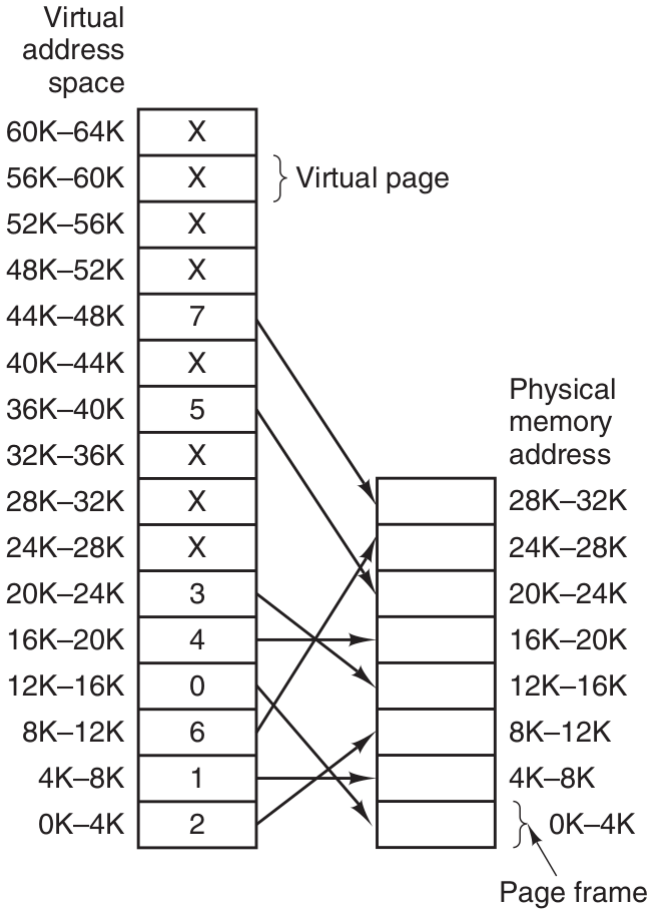
- 0K-4K → from 0 to 4095

  4K-8K → from 4096 to 8191
- **MOV REG,0**
  - Virtual address 0 (in page 0 mapped to frame 2) is sent to MMU, which translates it to 8192, puts the address on memory bus. Memory is not aware.
- **MOV REG,8192 → MOV REG,24567**
- Address 20500 → 12308
- **Present/absent bit** is used to identify present/absent pages.



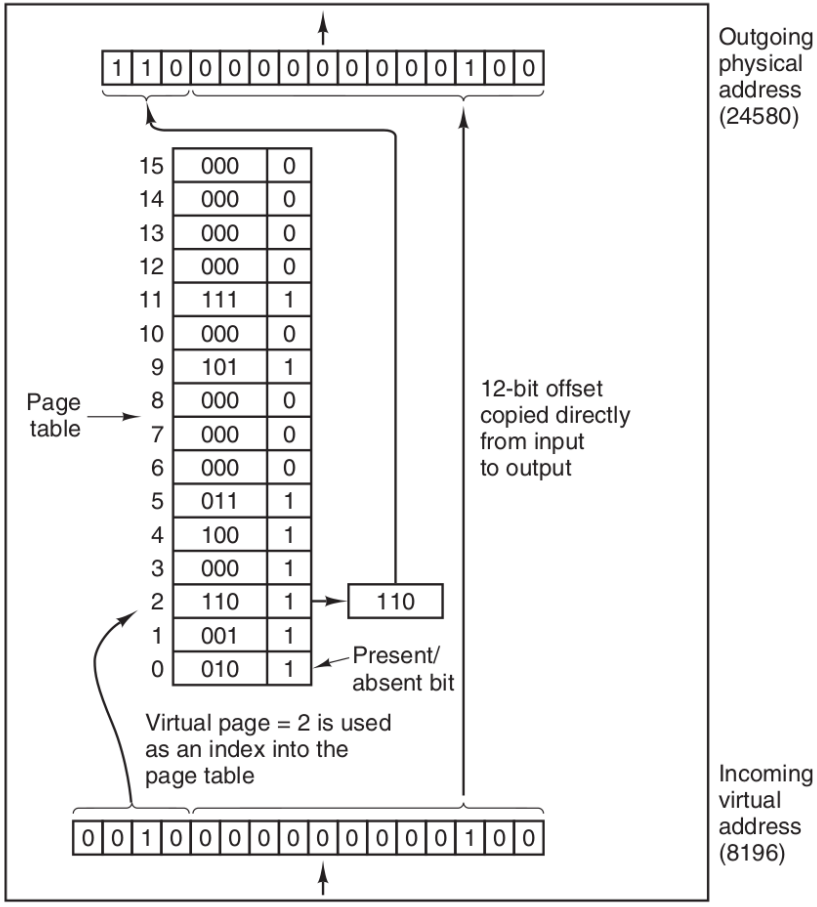
Paging (cont.)

- MOV REG,32780 → **Page fault** trap to OS.
  - OS picks a little-used frame,
  - Writes it back to disk (if needed),
  - Fetches the requested page into this frame,
  - Updates the map (the evacuated and the newly fetched)
  - And restarts the instruction.



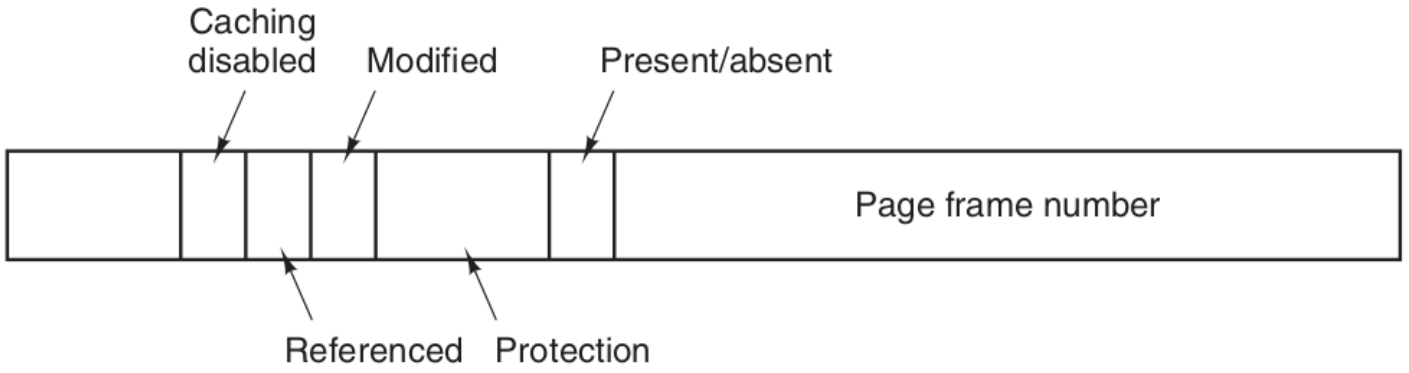
MMU (why powers of 2?)

- 8196 (0010000000000100)
- Split into:
  - 4 bit page number (we have 16 pages) used to index into the page table.
  - 12 bit offset (page size = 4KB)
- The 4 bit page number is replaced by 3 bit page frame number (we have 8 frames).
- The resulting address is put into output register, then to memory bus.
- Page table is like a function, input: page number, output: frame number.



Page Tables Entries

- Layout differs from machine to machine, but the info are similar.
- 32 bits is a common size.
- Protection bits for permissions:
  - 1 bit read only or read/write
  - 3 bits one for each (read, write, execute)
- Modified: sometimes called dirty bit.
- Referenced: used for replacement decision.
- Caching disabled: important for pages mapped to device registers.
- Info about disk is managed by OS, not by hardware.



Paging Implementation

- 2 Issues:
  - Mapping speed
  - Page table size (if virtual address space is large)
- Mapping Speed
  - One, two or maybe more memory access per instruction.
  - If instruction takes 1nsec, mapping should be under 0.2 nsec.
- Page table size
  - With 32 bit virtual address space, and 4KB pages → 1 million pages.
  - Can you think about 64bit address space?!!
  - Each process has its table!
- Single page table of array fast hardware registers.
  - No memory reference for mapping.
  - Loaded with context switch → slows down switching dramatically.
  - Expensive specially with large tables.
- Entire table in memory with 1 register pointing to its start.
  - At switching, only change register.
  - Memory references for mapping → slow.

Translation Lookaside Buffers

- An instruction accessing memory slows execution by half → not acceptable.
- Noticed: a lot of accesses to a few of pages.
- **TLB (Translation Lookaside Buffer)** or **associative memory**

Translation Lookaside Buffers (cont.)

- Usually in MMU.
- Small number of entries (8 here, not more than 256)
- Like page table with extra page number.
- Ex: **19, 20, 21**: loop code, **129, 130**: data, **140**: indexing, **860, 861**: stack.
- At page lookup, MMU looks in TLB first.
  - If hit: check permissions, and load from there.
  - If miss: ordinary page table + bring entry into TLB, copy modified bit of the evicted entry into page table.

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Software TLB Management

- Skipped