Virtual Memory

Chapter 8 of [OS4e],
Chapters 9 and 10 of [OSC]:
- Virtual-Memory Paging
- Page Table Organisation
- OS Policies

Virtual-Memory Paging

... is a memory-management scheme that
- allows a process to execute with some pages in main memory (resident set), rest in secondary memory
  - recall: logical address = (page number, offset)
  - virtual memory (i.e., logical address space of process) can be larger than real memory
- helps improve processor utilisation
  - resident-set size versus degree of multiprogramming

Virtual-memory segmentation is similar
**page fault**
- interrupt generated by reference to page outside resident set
  - OS swaps pages between main and secondary memory (*paging*)
  - recall: locality of reference
  - resident-set size versus page-fault frequency

**thrashing**
- process spends more time paging than executing

---

**Page Table Organisation**

Each page-table entry includes extra bits
- *present*
- *modify*
- *lock*

*Translation look-aside buffer (TLB)*
- small hardware cache consisting of page-table entries
- either each TLB entry must include process id or TLB must be flushed on process switch
How large is a page table?

**Example**

- 32-bit logical addresses
- page size 4K bytes ($2^{12}$)
  - size of offset field = 12 bits
  - size of page number field = 32 - 12 = 20 bits
  - 1 million entries ($2^{20}$)
- entry occupies 4 bytes
  - 4M bytes page table (per process)

but this is unacceptably high...
- **two-level paging**
  - e.g., Pentium
  - page number = (p1, p2)
  - *page directory* converts p1 to frame number, where page table is located
  - page table converts p2 to frame number, where page is located
  
  N.B. page directory and tables each fit in page

- **inverted page table**
  - e.g., PowerPC, UltraSPARC
  - one entry per frame
  - uses *hash function* to speed up search for entry that matches page number for given process

---

**Figure 8.6 Inverted Page Table Structure**
OS Policies

**Fetch policy**
- *demand paging*
  - swap in only required page on fault
- *prefetching*
  - additional pages swapped in

N.B. In steady state, most frames are occupied
- swapping in one page usually means swapping out another

**Resident Set Management**
Size of resident set depends upon ...
- number of frames allocated to process
  - *fixed*
  - *variable*

Frame into which page is loaded following a page fault depends upon ...
- scope of page replacement
  - *local* (in frame already allocated)
  - *global* (anywhere)

N.B. Fixed-allocation implies local replacement
**working set**
- set of pages “recently” referenced
  - i.e., within given window of current time
  - e.g., if pages 24, 15, 18, 23 referenced, then working set with window size 2 contains 18, 23

With variable-allocation and local-scope policies, need strategy for allocation
- periodically, swap pages with aim of maintaining working set in main memory
  - recall: middle-term scheduling
  - if insufficient frames for working sets of all processes, suspend one or more processes

**Replacement policy**
Which particular page (within local/global scope) should be replaced in order to produce fewest number of page faults?
- **optimal**
  - page with longest time until next reference
  - unimplementable, but used to judge effectiveness of other policies
- **least-recently-used**
  - page least likely to be referenced in near future
  - does nearly as well as optimal policy
  - expensive to implement
- **first-in, first-out**
  - page may no longer be in use
  - performs relatively poorly
  - allocated frames form cyclic array
  - cheap to implement

- **clock (second-chance)**
  - modifies FIFO policy to consider usage information
  - each frame has use bit, assigned 1 on reference
  - optionally, scan for frame with both modify and use bits equal to 0
  - scan for frame with use bit equal to 0, assigning 0 to any frames by-passed
Belady’s anomaly

- for some page-replacement algorithms, page-fault rate may increase as number of allocated frames increases
  
  e.g., FIFO on 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5 results in 10 faults if four frames, but only 9 if three frames
  
  – neither optimal nor LRU replacement suffer from this anomaly