# CS418—Operating Systems

Lecture 4

Memory Management–recent systems

Textbook: Operating Systems
by William Stallings

## 1. Paging (Paged memory allocation)

- Does a program have to be resided completely and contiguously in the main memory for execution?
- IDEA: Dividing an incoming job into memory blocks (frames) of equal size, which are called **pages**.
- To execute a program, Memory Manager must do the following
  - 1. Decide # of pages in the program.
  - 2. Have enough empty page frames in main memory.
  - 3. Load all the program's pages into them.
- Advantage
  - 1. Memory is certainly used efficiently.
  - 2. No external fragmentation.
  - 3. Almost no internal fragmentation.
- Drawback? *Overhead is increased significantly.* An OS nowadays has to be designed by experts.

- How do we manage paging?
  - 1. Job Table.
  - 2. Page Map Table (for each job).
  - 3. Memory Map Table.

- What if we have a goto statement?
- Offset (displacement) of a line is the factor used to locate that line within the page frame.
- Intuitively, offset represents how far away a line is from the beginning of its page.

- In general, the following is the method to handle a goto statement (or to access any special line).
  - 1. `Using the previous arithmetic computation to compute page # and displacement of the line.`
  - 2. `Look up this job's PMT to find the page frame which contains this page.`
  - 3. `page_frame_address = page_frame_num * page_size`
  - 4. `instruction_address = page_frame_address +`
  - `displacement.`

- Advantage of paging.
  - 1. Job is stored non-contiguously in memory.
  - 2. No external fragmentation.
- Disadvantage of paging.
  - 1. Overhead.
  - 2. Internal fragmentation still exists.
  - 3. Page size too small $\rightarrow$ PMT's have large size.
  - 4. Page size too large $\rightarrow$ internal fragmentation increases.

## 2. Demand Paging

- Demand paging only loads a part of a program into memory for running.
  - 1. Jobs are still decomposed into equally sized pages.
  - 2. Jobs are initially stored in secondary memory.
- Why demand paging is feasible?

- **Demand paging** allows a user to run jobs with less main memory (this is the idea of **virtual memory**: the user would feel that the physical memory is almost infinite, though it is not the case in reality).

- Page Map Table (PMT) needs to be modified.

- How does the computer fetch an instruction?
  - `1.  Start processing instruction`
  - `2.  Generate data address`
  - `3.  Compute page number`
  - `4.  If page is in memory`
  - `     then`
  - `         get data and finish instruction`
  - `         advance to the next instruction`
  - `         return to step 1`
  - `     else`
  - `         generate page interrupt`
  - `         call page interrupt handler`

- **Algorithm: Page Interrupt Handler**
  - 1. If there is no free page frame
  - then
  - select page to be swapped out using
  - a page removal algorithm
  - update job's Page Map Table
  - if content of page had been changed
  - then write page to disk
  - 2. Use page number (step 3 of the previous
  - algorithm to get disk address where
  - page is stored (the File Manager, to
  - be discussed later, uses the page
  - number to get the disk address)
  - 3. Read page into memory
  - 4. Update job's Page Map Table
  - 5. Update Memory Map Table
  - 6. Restart interrupted instruction

- Although demand paging is a solution to inefficient memory utilization, it does not solve all the problems
- **Thrashing**: if a large amount of page swapping is performed, the system efficiency is affected.
- **Page fault**: a failure to find a page in memory.