Project 5: Virtual Memory

COS 318

Fall 2013

Project 5 Schedule

- Design Review
  - Monday, Nov 25
  - 10-min time slots from 10am to 8pm
- Due date: Wed Dec 4, 11:55pm

General Suggestions

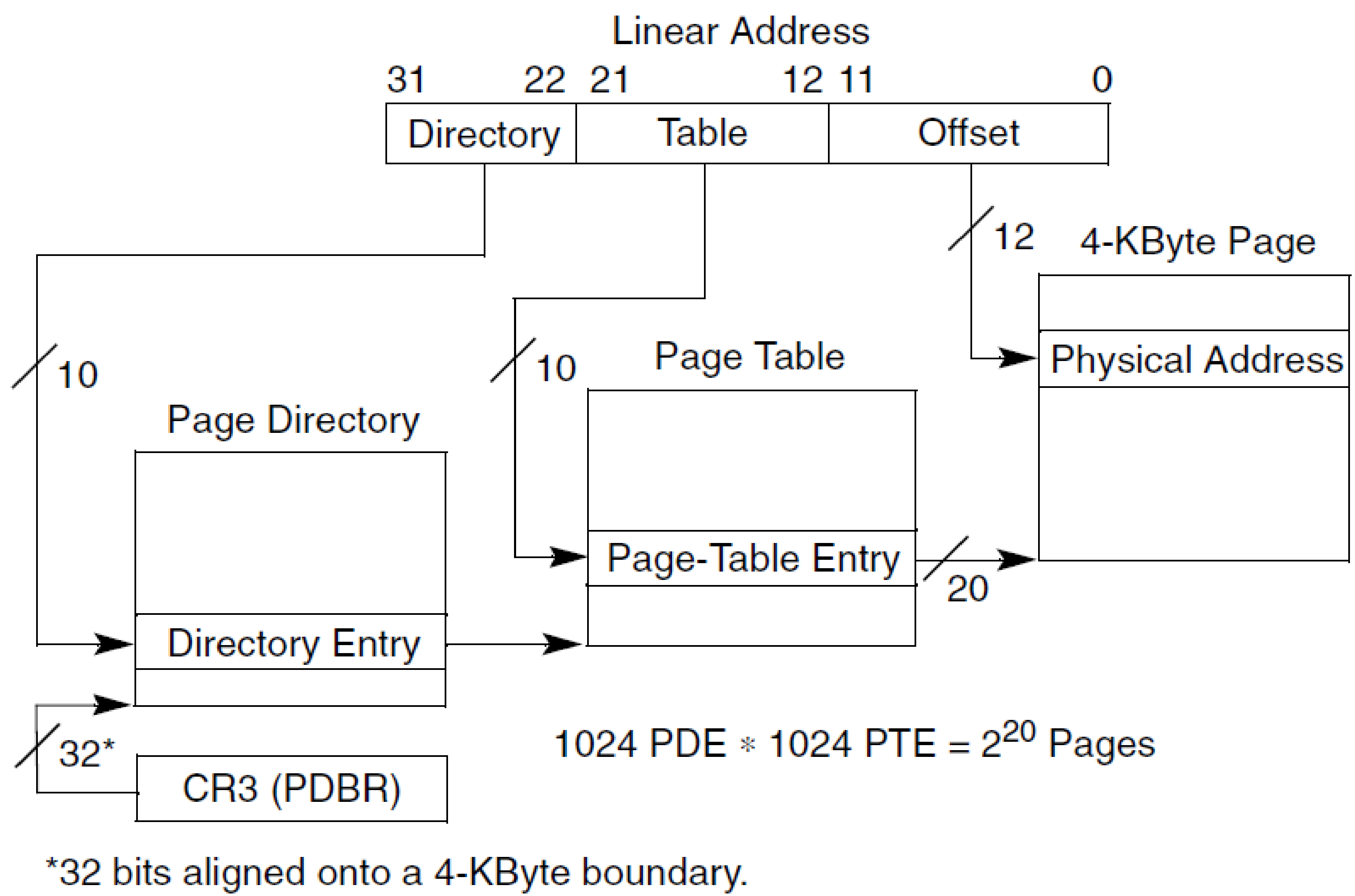
- Project is <u>not</u> divided into phases.
- Follow the rough checklist in the project 5 specs.
- Get familiar with the 2-level page table description of i386.
- Read section 3.7.1 and 4.2 of the Intel manual.
- Look at new PCB structure in ***kernel.h***.
- As always, start as early as you can, and get as much done as possible by the design review.

Project 5 Overview

- Implement page allocation and eviction policy.
- Initialize the memory layout (kernel pages).
- Set up each process' memory.
- Swap pages in/out of disk → demand paging.
- Page fault handler.
- Relevant files: ***memory.h*** and ***memory.c***
- No assembly programming ☺
- Extra credit: Better eviction policy.

2-Level Page Table (i386)

- See section 3.7.1 in Intel Manual (p. 84-85)



Directory Entries

- See section 4.2 in Intel Manual (p. 106-107)

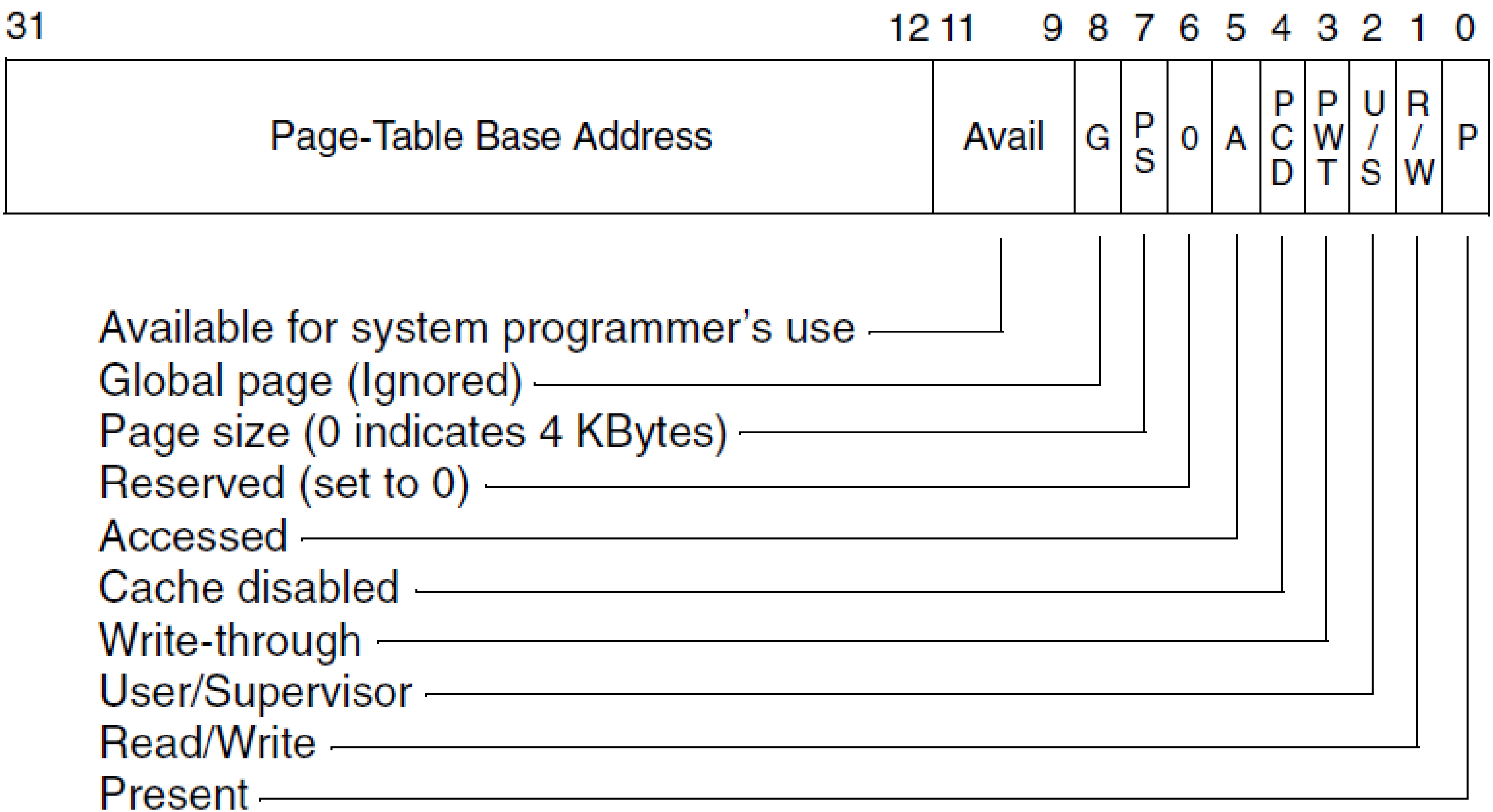
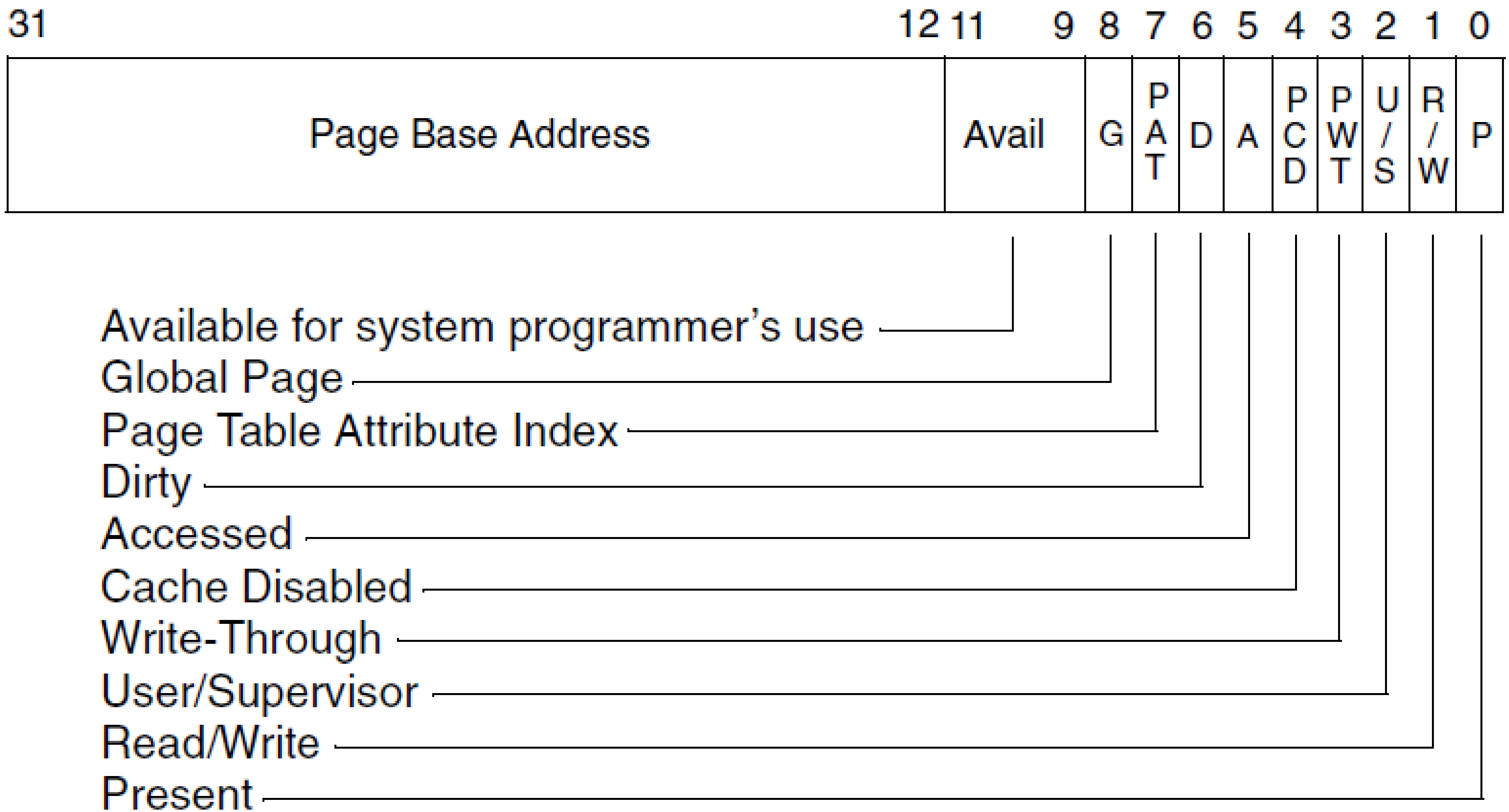


Table Entries

- See section 4.2 in Intel Manual (p. 106-107)



Entry Flags

- See section 4.2 in Intel Manual (p. 106-107)
- P: Page/Page table loaded?
- U/S: User access? 0 → no user access
- R/W: User read/write? 0 → user read-only
- A: Accessed? set on swap-in
- D: Dirty page? use at swap-out

Page Allocation and Eviction

- Define a page map data structure to track all pages and their metadata (in ***memory.h***).
- If there is a free page, simply use it.
- Otherwise, you need to swap a page out.
- Recall that you can ***pin*** pages → can't evict these pages!
- Simple eviction policy: e.g. FIFO

Initialize Kernel Memory

- Allocate N_KERNEL_PTS (page tables)
- For each page table, allocate pages until you reach MAX_PHYSICAL_MEMORY.
- Important: physical address = virtual address.
- Make sure to set correct flags!
- Give the user permission to use the screen.

Setting up Process Memory

- Processes keep track of 4 types of pages:
  - Page directory
  - Page tables
  - Stack page table
  - Stack pages
- PROCESS_START (vaddr of code + data)
  - Use one page table and allocate all pages.
  - Needs pcb->swap_size memory.
- PROCESS_STACK (vaddr of stack top)
  - Allocate N_PROCESS_STACK_PAGES.

Swapping pages in and out

- USB disk image for swap storage.
- Swap in for allocation, swap out for eviction.
- Assume that processes do not change size.
- Processes use whichever location they were originally loaded from (pcb->swap_loc).
- Use ***usb/scsi.h*** for read and write functions.
- Keep in mind: When do you need to flush the TLB?

Handling Page Faults

- Get a free page from the page allocator.
- Swap in the page.
- Update the page table entry to the page's address and set the present flag.

Some more tips...

- One page table is enough for a process' code and data memory space.
- Some functions (especially page fault handler) can be interrupted!
  - Use a synchronization primitive.
- Some pages don't need to be swapped out.
  - Kernel pages, process page directory, page tables, stack page tables and stack pages.
  - With respect to grading!