操作系统研讨课

蒋德钧

Fall Term 2017-2018

email: jiangdejun@ict.ac.cn
office phone: 62601007

中国科学院大学
University of Chinese Academy of Sciences

Lecture 5 Virtual Memory

2017.11.29

中国科学院大学
University of Chinese Academy of Sciences

Schedule

- Project 4 due
- Project 5 assignment

Project 4 Due

- P4 review
  - We test
    - spawn, kill, and wait
    - the three synchronization primitives
    - IPC with mailbox
  - Please compile your code, running the code on your board, and show the results to TA
  - Answer any questions we may ask

Project 4 Due

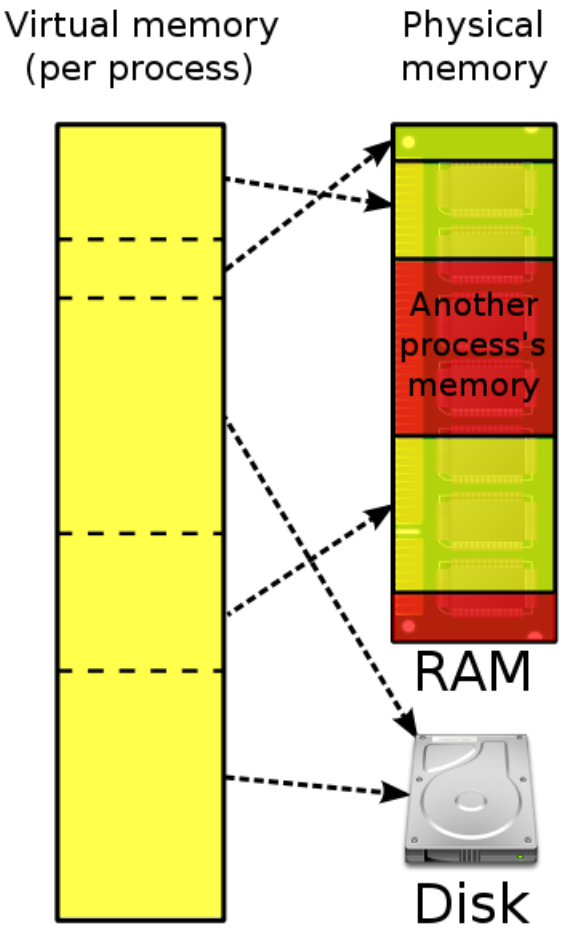
- Requirement for developing (60 points)
  - Implement do_spawn (5)
  - Implement do_kill (5)
  - Implement do_wait (5)
  - Implement the three synchronization primitives (20)
    - Condition variables, Semaphores, Barrier
  - Implement mailbox (25)
  - Bonus (2 points)
    - deal with locks when a task is killed

Project 5 Virtual Memory

- Requirement
  - Implement virtual memory management for user process
    - Setup demand-paged virtual memory for user process
    - Handle TLB miss and flush
    - Handle page fault assuming physical memory is enough
    - Handle pin/unpin pages for stack pages

Project 5 Virtual Memory

- Virtual memory
  - Each process sees a contiguous and linear address space, which is called virtual addresses
  - Virtual addresses are mapped into physical addresses by both HW and SW



[From Wikipedia]

Project 5 Virtual Memory

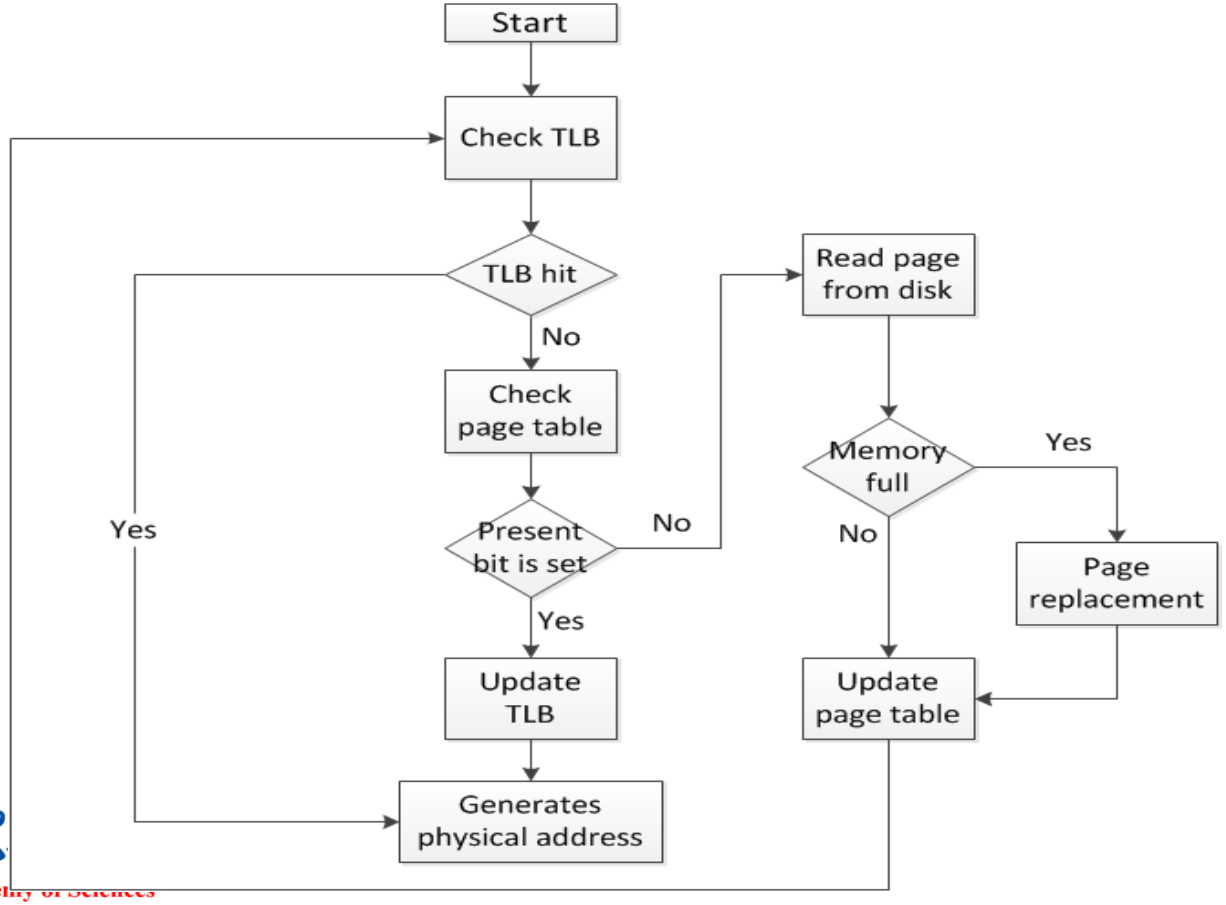
- Virtual memory
  - Virtual address space is divided into pages, which are blocks of contiguous virtual memory addresses
    - e.g. 4KB pages
  - Each page has a virtual address

Project 5 Virtual Memory

- Virtual memory
  - Page tables
    - The data structure to store the mapping between virtual addresses and physical addresses
    - Each mapping is a page table entry
  - MMU and TLB
    - MMU stores a cache of recently used mappings from the page table, which is called translation lookaside buffer (TLB)

Project 5 Virtual Memory

- Virtual memory
  - Address translation



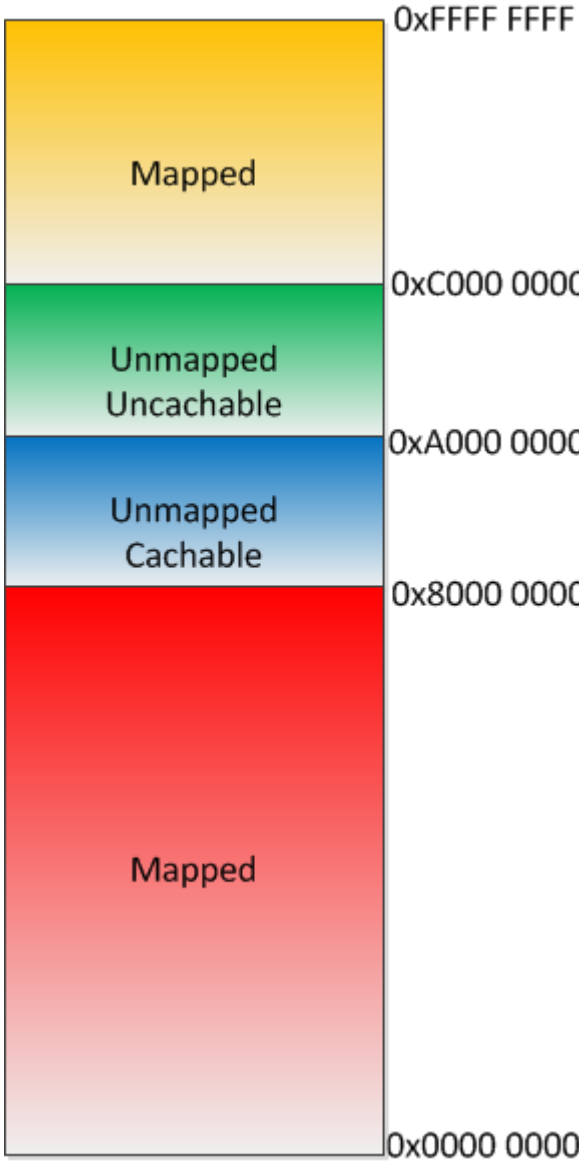
Project 5 Virtual Memory

- Implementing virtual memory
  - MIPS virtual memory layout

Address range	Capacity (GB)	Mapping approach	Cacheable
0xC0000000 -- 0xFFFFFFFF	1	TLB lookup	Yes
0xA0000000 -- 0xBFFFFFFF	0.5	Base + offset	No
0x80000000 -- 0x9FFFFFFF	0.5	Base + offset	Yes
0x00000000 -- 0x7FFFFFFF	2	TLB lookup	Yes

Kernel space

User space



Project 5 Virtual Memory

- Implementing virtual memory – initialization
  - Initialize physical page frames
    - Design data structure for individual page frames
    - An array of *page_map* for all page frames

Project 5 Virtual Memory

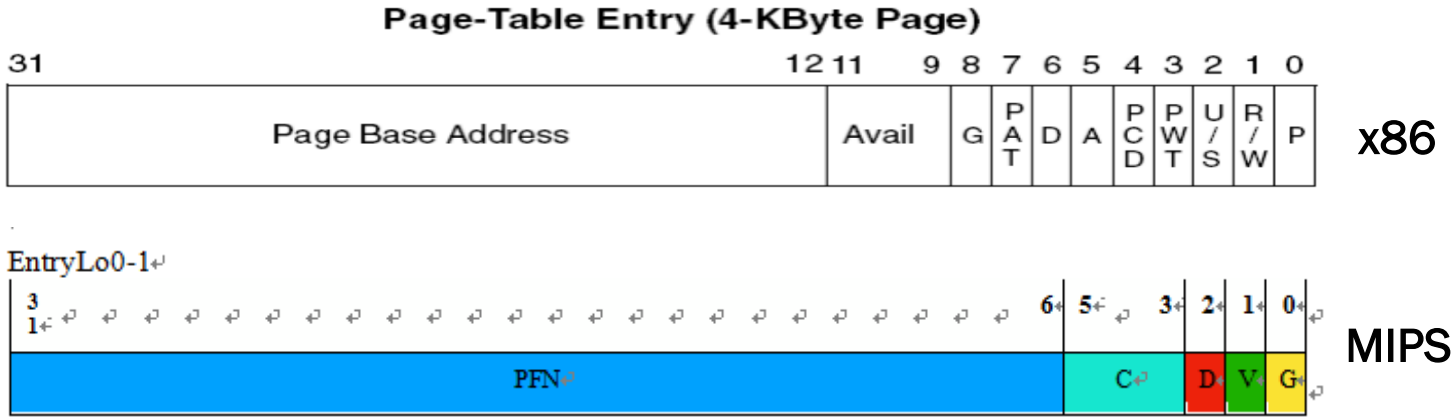
- Implementing virtual memory – initialization
  - Page table setup
    - How many page frames for page table of each use process?
      - Opt.1: Calculate your process image size
      - Opt.2: Set a fixed sized page table

Project 5 Virtual Memory

- Implementing virtual memory – initialization
  - Page table setup
    - Allocate page frames for page table itself
      - Where to find the available page frames
      - Physical memory layout
        - » *MEM_START:* 0x0090 8000
        - » *MAX_PHYSICAL_MEMORY:* # pages

Project 5 Virtual Memory

- Implementing virtual memory – initialization
  - Page table setup
    - Initialize page table entries (PTE)
      - Design the structure of PTE

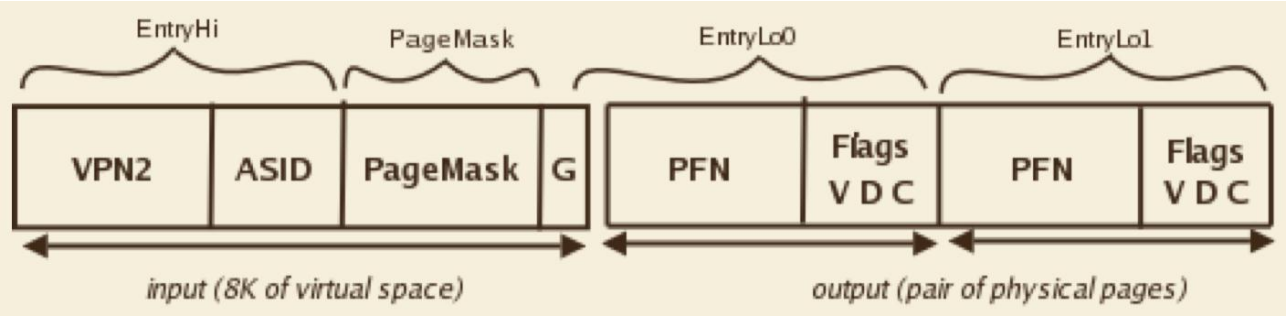


Project 5 Virtual Memory

- Implementing virtual memory – runtime
  - Handling TLB
    - TLB miss
      - Page walk, Reset TLB entry
    - PTE modification (not required in P5)
      - Flush TLB entry

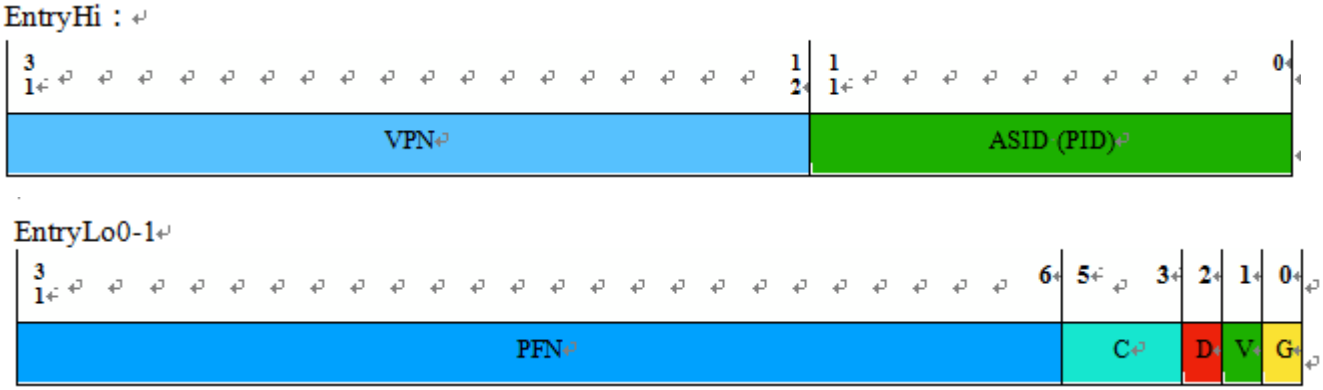
Project 5 Virtual Memory

- Implementing virtual memory – runtime
  - Manipulate MIPS TLB
    - Structure of TLB entry
      - Each entry includes the mapping for two continuous virtual addresses
      - Registers: EntryHi, EntryLo0, EntryLo1



Project 5 Virtual Memory

- Implementing virtual memory – runtime
  - Manipulate MIPS TLB
    - Structure of TLB entry
      - ASID: process ID
      - C: cachable or not, set to 000 in all tasks
      - D: writable or not
      - V: valid or not



Project 5 Virtual Memory

- Implementing virtual memory – runtime
  - Manipulate MIPS TLB
    - Instructions for manipulating TLB entry
      - tlbp: tlb lookup
      - tlbr: read TLB entry at index
      - tlbwi: write tlb entry at index
      - tlbwr: write tlb entry selected by random
      - Setup the corresponding registers and then execute the above instructions
    - *CP0_INDEX* register
      - mfc0: move from Coprocessor 0
      - mtc0: move to Coprocessor 0

Project 5 Virtual Memory

- Implementing virtual memory – runtime
  - Handle page fault
    - Finding the faulting page
    - Allocate a physical page
    - Page in the content from the swap location on the disk
    - Update the page table
    - Flush TLB entry
  - Note that, page fault handler can be interrupted, you need to handle this

Project 5 Virtual Memory

- Implementing virtual memory – runtime
  - Handle page in/out
    - Select one page to page out to disk if there is no free physical pages
    - Which one to select?
      - FIFO vs. memory system efficiency
    - How to read/write pages?
      - Use memory space to emulate file space
      - Define the space in PCB and use bcopy to R/W
    - Note that
      - Pinning pages: page directory, page tables, stack page table
      - How about kernel pages?

Project 5 Virtual Memory

- Step by step
  - Task 1
    - Initialize physical memory
    - Initialize PCBs and setup user-space stacks
    - Setup page tables for user process
      - Initialize page table entries and set all pages valid
      - Note that, in task 1, you need to handle TLB miss by implementing the tlb_refill process
    - Allocate physical pages
    - page mapping functions for address translation

Project 5 Virtual Memory

- Step by step
  - Task 2: Implement on-demand paging
    - Initialize physical memory (the same as task1)
    - Initialize PCBs and setup user-space stacks (the same as task1)
    - Setup page tables
      - Initialize all pages to be invalid for triggering page fault
    - Handle TLB miss
    - Implement page fault handler and page in/out
    - Note that, you need to make sure memory size is not enough such that page replacement occurs

Project 5 Virtual Memory

- Tips on implementing virtual memory
  - Both page directory and page table are a page
  - Change PAGEABLE_PAGES to change the available RAM capacity
  - Read memory.h carefully to get to know the purposes of various functions

Project 5 Virtual Memory

- Requirements for design review (40 points)
  - What is the virtual memory layout in your design?
  - How are virtual addresses translated into physical addresses? When are page faults triggered?
  - How do you track all physical page frames?
  - What is the structure for your page table entry? What are the initialized values for PTEs in tasks 1 and 2 respectively? How many initialized PTEs in both tasks?
  - What is the workflow of your page fault handler?
  - How do you handle TLB miss? When do you need to flush TLB entries?

Project 5 Virtual Memory

- Requirements of developing (60 points)
  - Implement virtual memory for user-space process with TLB miss but without page fault(25)
  - Implement virtual memory for user-space process with TLB miss, page fault, and page replacement (35)

Project 5 Virtual Memory

- Bonus (4 points)
  - Task 1 (2 points): implement efficient page replacement
    - Limit the size of available physical memory, and you may need to increase the process image size by changing the test cases
    - Implement the page replacement when the required memory size exceeds the physical memory size
    - Choose an algorithm for the page replacement except FIFO

Project 5 Virtual Memory

- Bonus (4 points)
  - Task 2 (2 points): implement two-level page table
    - Increase the process image size by changing the test cases in order to have more PTEs to fit into a two-level page table
    - Implement the two-level page table for your own designed test cases

Project 5 Virtual Memory

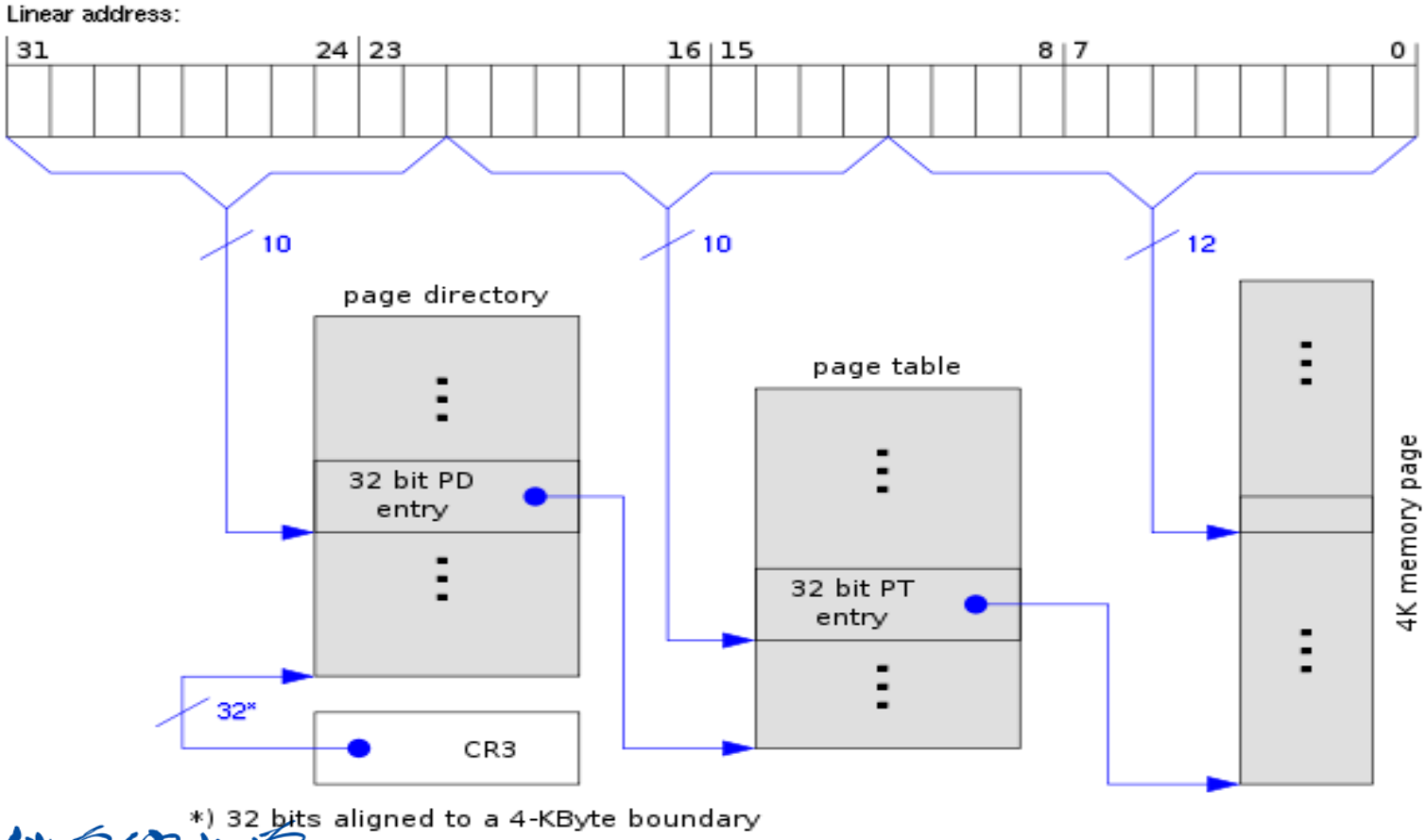
- Two-level page table
  - Assuming i386 architecture
  - Linear address
    - Indirectly refers to a physical address

Project 5 Virtual Memory

- Two-level page table
  - Assuming i386 architecture
  - Linear address
    - Indirectly refers to a physical address

Project 5 Virtual Memory

- Two-level page table



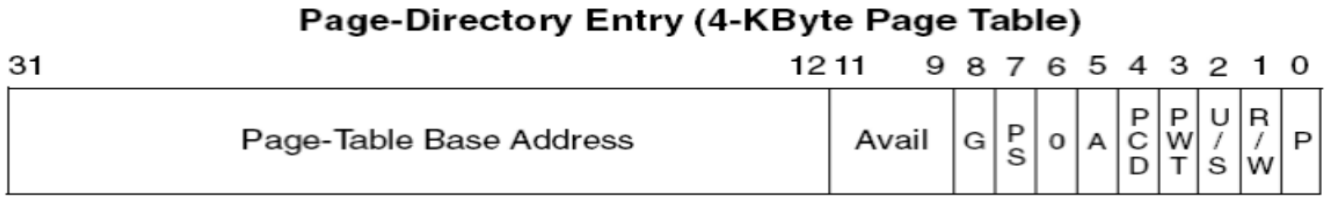
[From Wikipedia]

Project 5 Virtual Memory

- Two-level page table
  - Page directory (Level 1)
    - bits 22 to 31 index an entry in the page directory
  - Page table (Level 2)
    - bits 12 to 21 index an entry in the page table
  - Page offset
    - bits 0 to 11 address a byte within the page

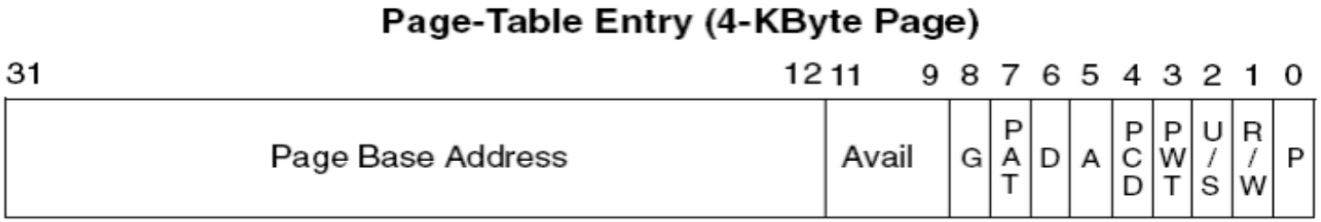
Project 5 Virtual Memory

- Two-level page table
  - Page directory entry



P: Present
R/W: Read/Write
U/S: User/Supervisor
D: Dirty

  - Page table entry



[From Wikipedia]

Project 5 Virtual Memory

- P5 schedule
  - P5 design review: 6th Dec.
  - P5 due: 20th Dec.
  - 13th Dec.: No class