CENG 3420 Computer Organization & Design

Lecture 15: Virtual Memory

Bei Yu
CSE Department, CUHK
`byu@cse.cuhk.edu.hk`

(Textbook: Chapters 5.7)

2025 Spring

Overview

Introduction

Motivations

Physical memory may not be as large as “possible address space” spanned by a processor, e.g.

- A processor can address 4G bytes with 32-bit address
- But installed main memory may only be 1GB

How if we want to simultaneously run many programs which require a total memory consumption greater than the installed main memory capacity?

Terminology:

- A running program is called a process or a thread
- Operating System (OS) controls the processes

Virtual Memory

- Use main memory as a “cache” for secondary memory
- Each program is compiled into its own virtual address space
- What makes it work? Principle of Locality

Virtual Memory

- Use main memory as a “cache” for secondary memory
- Each program is compiled into its own virtual address space
- What makes it work? Principle of Locality

Why virtual memory?

- During run-time, virtual address is translated to a physical address
- Efficient & safe sharing memory among multiple programs
- Ability to run programs larger than the size of physical memory
- Code relocation: code can be loaded anywhere in main memory

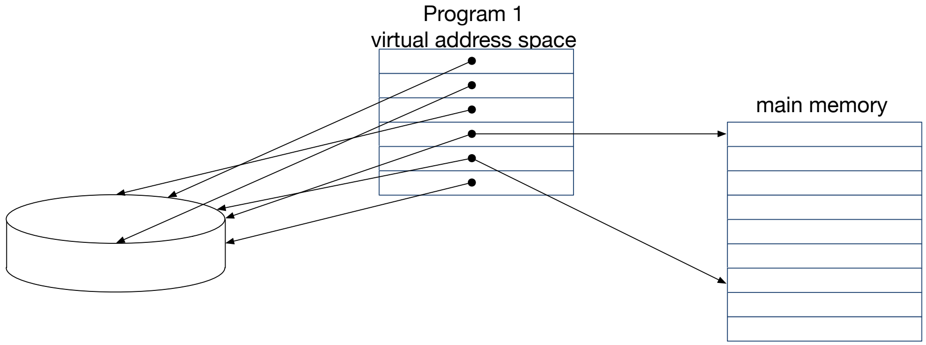
Bottom of the Memory Hierarchy

Consider the following example:

- Suppose we hit the limit of 1GB in the example, and we suddenly need some more memory on the fly.
- We move some main memory chunks to the harddisk, say, 100MB.
- So, we have 100MB of “free” main memory for use.
- What if later on, those instructions / data in the saved 100MB chunk are needed again?
- We have to “free” some other main memory chunks in order to move the instructions / data back from the harddisk.

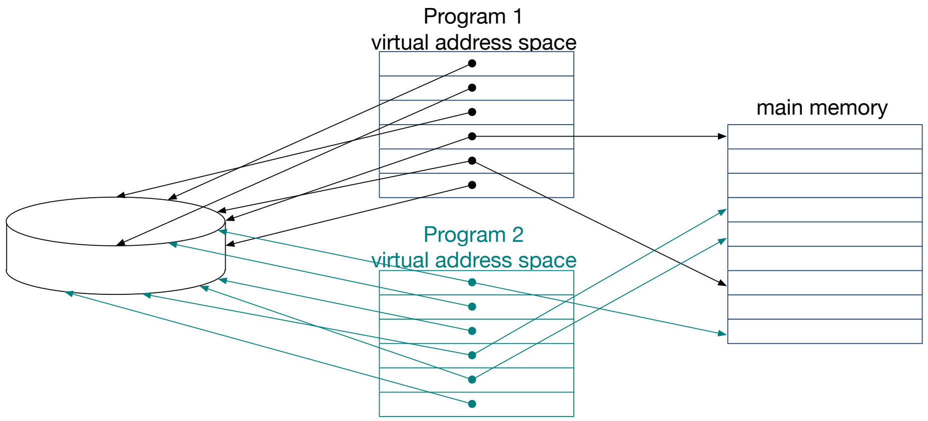
Two Programs Sharing Physical Memory

- A program's address space is divided into pages (fixed size) or segments (variable sizes)



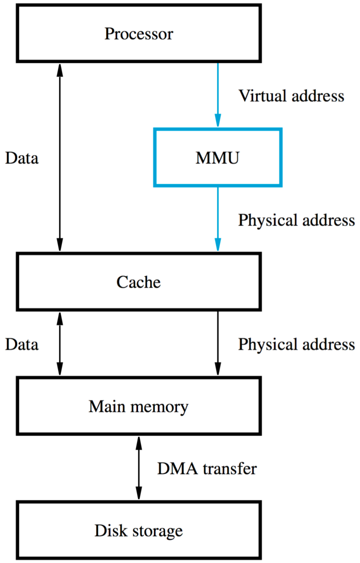
Two Programs Sharing Physical Memory

- A program's address space is divided into pages (fixed size) or segments (variable sizes)



Virtual Memory Organization

- Part of process(es) are stored temporarily on harddisk and brought into main memory as needed
- This is done automatically by the OS, application program does not need to be aware of the existence of virtual memory (VM)
- Memory management unit (MMU) translates virtual addresses to physical addresses



Virtual Memory

Address Translation

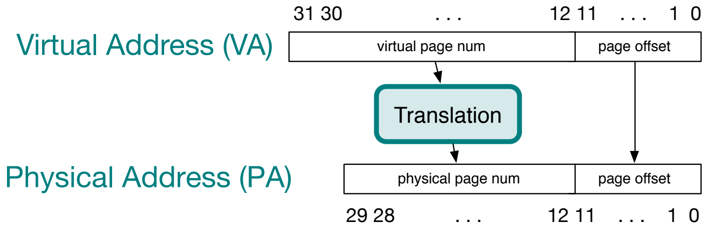
- Memory divided into pages of size ranging from 2KB to 16KB
    - Page too small: too much time spent getting pages from disk
    - Page too large: a large portion of the page may not be used
    - This is similar to cache block size issue (discussed earlier)
- For harddisk, it takes a considerable amount of time to locate a data on the disk but once located, the data can be transferred at a rate of several MB per second.
- If pages are too large, it is possible that a substantial portion of a page is not used but it will occupy valuable space in the main memory.

Address Translation

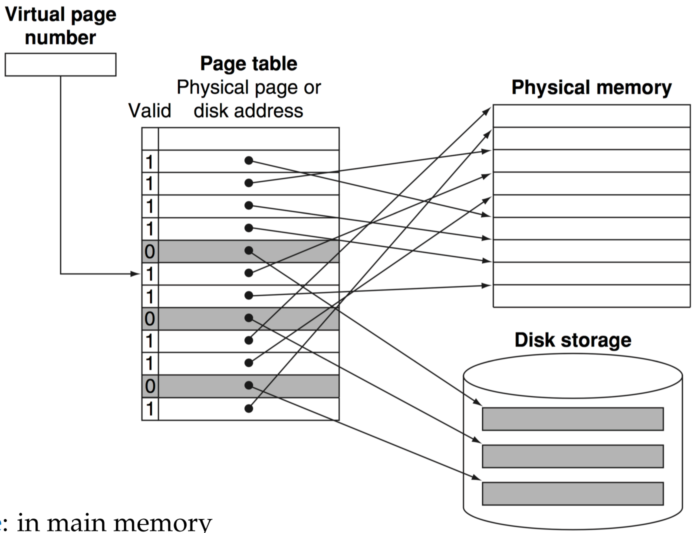
- An area in the main memory that can hold one page is called a page frame.
- Processor generates virtual addresses
  - MS (high order) bits are the virtual page number
  - LS (low order) bits are the offset
- Information about where each page is stored is maintained in a data structure in the main memory called the **page table**
  - Starting address of the page table is stored in a page table base register
  - Address in physical memory is obtained by indexing the virtual page number from the page table base register

Address Translation

- Virtual address $\rightarrow$ physical address by combination of HW/SW
- Each memory request needs first an address translation
- Page Fault: a virtual memory miss



Address Translation Mechanisms

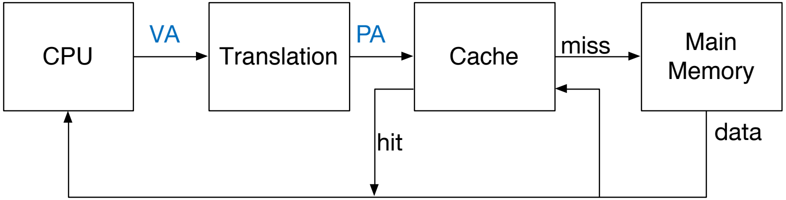


- Page Table: in main memory
- Process: page table + program counter + registers

Virtual Addressing with a Cache

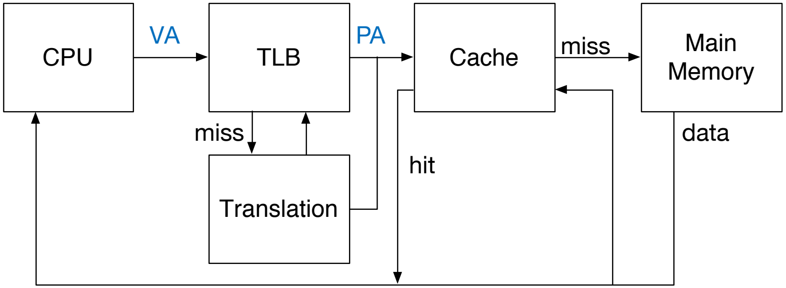
Disadvantage of virtual addressing:

- One extra memory access to translate a VA to a PA
- memory (cache) access very expensive...

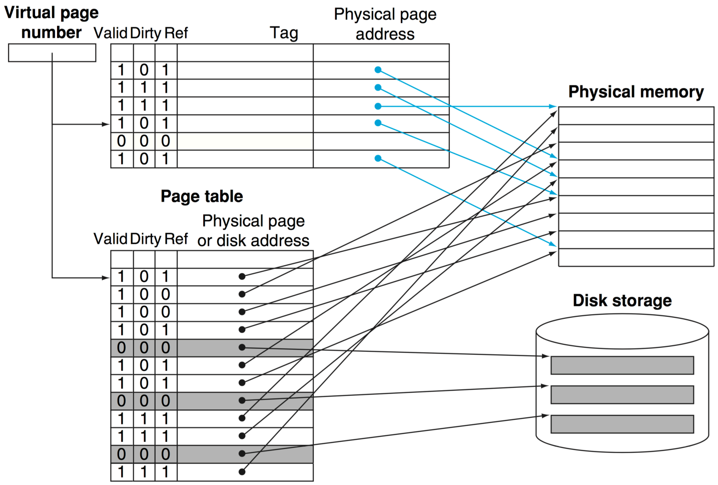


Translation Look-aside Buffer (TLB)

- A small cache: keeps track of recently used address mappings
- Avoid page table lookup



Translation Look-aside Buffer (TLB)



- Dirty bit:
- Ref bit:

More about TLB

Organization:

- Just like any other cache, can be fully associative, set associative, or direct mapped.

Access time:

- Faster than cache: due to smaller size
- Typically not more than 512 entries even on high end machines

A TLB miss:

- If the page is in main memory: miss can be handled; load translation info from page table to TLB
- If the page is NOT in main memory: page fault

TLB Event Combinations

- TLB / Cache miss: page / block not in "cache"
- Page Table miss: page NOT in memory

TLB	Page Table	Cache	Possible? Under what circumstances?
Hit	Hit	Hit	
Hit	Hit	Miss	
Miss	Hit	Hit	
Miss	Hit	Miss	
Miss	Miss	Miss	
Hit	Miss	Miss / Hit	
Miss	Miss	Hit	

TLB Event Combinations

- TLB / Cache miss: page / block not in "cache"
- Page Table miss: page NOT in memory

TLB	Page Table	Cache	Possible? Under what circumstances?
Hit	Hit	Hit	Yes – what we want!
Hit	Hit	Miss	Yes – although page table is not checked if TLB hits
Miss	Hit	Hit	Yes – TLB miss, PA in page table
Miss	Hit	Miss	Yes – TLB miss, PA in page table but data not in cache
Miss	Miss	Miss	Yes – page fault
Hit	Miss	Miss / Hit	
Miss	Miss	Hit	

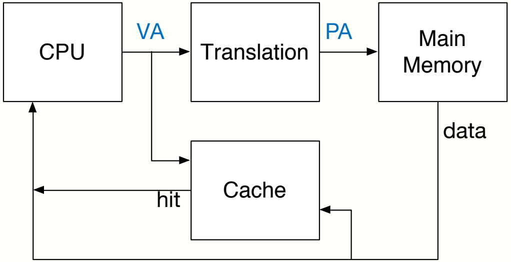
TLB Event Combinations

- TLB / Cache miss: page / block not in “cache”
- Page Table miss: page NOT in memory

TLB	Page Table	Cache	Possible? Under what circumstances?
Hit	Hit	Hit	Yes – what we want!
Hit	Hit	Miss	Yes – although page table is not checked if TLB hits
Miss	Hit	Hit	Yes – TLB miss, PA in page table
Miss	Hit	Miss	Yes – TLB miss, PA in page table but data not in cache
Miss	Miss	Miss	Yes – page fault
Hit	Miss	Miss / Hit	Impossible – TLB translation not possible if page is not in memory
Miss	Miss	Hit	Impossible – data not allowd in cache if page is not in memory

QUESTION: Why Not a Virtually Addressed Cache?

- Access Cache using virtual address (VA)
- Only address translation when cache misses



Answer: