Main Memory

Hui Chen [a]

[a]CUNY Brooklyn College

March 30, 2023

Outline

Outline

MMU and Address Translation

- Policy. Each process has a separate memory space.
  - Protection. Keep each process isolated.
  - Sharing. Allow memory to be shared between processes.
  - Virtualization. Provide applications with the illusion of "infinite" memory.
- Mechanism. Mediating memory access via a hardware component called the memory management unit (MMU). MMU that translate a logical address to a physical address.
  - Logical address (or virtual address). An address generated by the CPU.
  - Physical address. An address generated by the MMU

Outline

Base-Limit Registers

Introduce the memory management unit (MMU) with a pair of registers within,

- Base register. It holds the smallest legal physical memory address.
- Limit register. It specifies the size of the range.
- The MMU checks whether a (logical) address is valid.

$$\text{Validity} = \begin{cases} \text{True} & 0 \leq \text{Address} - \text{Base Register} < \text{Limit Register} \\ \text{False} & \text{Otherwise} \end{cases} \tag{1}$$

- Logical Address $\equiv$ Physical Address

Base-Limit Registers: Discussion

We can

- (allows *concurrency*) allow processes loaded in memory for concurrent execution, and
- (provides *protection*) have the ability to determine the range of legal addresses.
- (supports *virtualization*???) ...

How, how should we write programs to use this? In another word, does the system permit the *dynamic loading* of programs and how does it impact how we must write a program?

Outline

Relocation Register

Introduce the MMU with a *relocation* register ,

- Relocation register. It holds the smallest legal physical memory address.
- The MMU translates the logical address to the physical address,

$$\text{Physical Address} = \text{Relocation Register} + \text{Logical Address} \quad (2)$$

Relocation Register: Discussion

We can

- (supports *concurrency*) allow processes loaded in memory for concurrent execution, and
- (provides *protection*???) ...
- (supports *virtualization*???) ...

How, how should we write programs to use this? In another word, does the system permit the *dynamic loading* of programs and how does it impact how we must write a program?

Outline

Relocation-Limit Registers

Introduce the MMU with a *relocation* and a limit register ,

- Relocation register. It holds the smallest legal physical memory address.
- Limit register. It specifies the size of the range.
- The MMU checks the validity of the logical address,

$$\text{Validity} = \begin{cases} \text{True} & \text{Logical Address} < \text{Limit Register} \\ \text{False} & \text{Logical Address} \geq \text{Limit Register} \end{cases} \tag{3}$$

- The MMU translates the logical address to the physical address,

$$\text{Physical Address} = \text{Relocation Register} + \text{Logical Address} \tag{4}$$

Relocation-Limit Registers: Discussion

We can

- (support *concurrency*) allow processes loaded in memory for concurrent execution, and
- (provide *protection*) have the ability to determine the range of legal addresses.
- (support *virtualization*???) ...

How, how should we write programs to use this? In another word, does the system permit the *dynamic loading* of programs and how does it impact how we must write a program?

Contiguous Memory Allocation

- The Relocation-Limit Registers scheme is often called the Contiguous Memory Allocation scheme since each process is contained in a *single section of memory* that is *contiguous* to the section containing the next process.
- Simply put, this is the result that the relocation register can hold any valid and continguous values, like 0, 1, 2, ...

Memory Allocation

- Variable Partition Scheme. With the relocaiton-limit registers, the OS assigns processes to variably sized partitions in memory, where each partition may contain exactly one process.
  - General dynamic storage allocation problem.
    - First-fit, best-fit, and worst-fit strategies.
- Problems?
  - Internal fragmentation.
  - External fragmentation.

Outline

Concept of Paging

A memory management scheme that permits a process's physical address space to be non-contiguous.

Basic Paging Scheme

- Physical memory. Breaking physical memory into fixed-sized blocks called *frames*.
- Logical memory. Breaking logical memory into blocks of the same size called *pages*.
- Page size $\equiv$ frame size
- A logical address consists of a page number and a page offset.
- A physical address consists of a frame number and a frame offset.
- Page offset $\equiv$ frame offset
- MMU translates a logical address to a physical address via a page table.

Querying Page Size

On Debian Linux,

```
$ getconf PAGESIZE
$ man 2 getpagesize
```

Examples and Exercises

Let's examine a few examples and do a few exercises ...

Fragmentation

- Does paging has internal fragmentation?
- Does paging has external fragmentation?

Protection and Sharing?

Recall

- Policy. Each process has a separate memory space.
  - Protection. Keep each process isolated.
  - Sharing. Allow memory to be shared between processes.
  - Virtualization. Provide applications with the illusion of "infinite" memory.

Protection

- Associating protection bits with each frame
  - Read-write, read-only, executable?
  - Valid and invalid?
  - Page table size? (page-table length register (PTLR))?
- Storing these bits in page table

Shared Pages

Paging allows us to realize shared library and shared memory efficiently.

Outline

Access Latency of Basic Paging Scheme

How much time does it take to access the memory?

- Let's compare the Variable Partition scheme and the Basic Paging scheme.
- Constraints.
  - For most contemporary CPUs, page tables are large, and can only be kept in main memory.
  - page tables are located via a page-table base register (PTBR)
  - page table size is in page-table length register (PTLR)

Translation Look-Aside Buffer

- To reduce access latency of the paging scheme, introduce cache memory called the Translation Look-Aside Buffer (TLB)
- Realized via an associative, high-speed memory.
  - When the associative memory is presented with an item, the item is compared with all keys simultaneously.
  - A TLB lookup in modern hardware is part of the instruction pipeline, essentially adding no performance penalty.

Querying TLB

On Debian Linux,

```
$ sudo apt-get install cpuid
$ cpuid | grep -i tlb
```

Access Latency with TLB

Let's do a few exercises to analyze access latency with TLB?

TLB Replacement Algorithm

What happens when a process accesses a page that isn't cached in TLB and TLB is full?

- TLB replacement policy (Generally, LRU; cf. virtual memory)

How big is a page table?

- Page tables can be very large if implemented in straight-forward fashion.
- Let's do some exercise ...

Hierarchical Page Tables

To avoid allocating a page table contiguously in main memory, we page the page table.

2-Level Paging

Let's consider 32-bit logical address space, and a logical address will have the format like the following,

page number		page offset
p_1	p_2	d
10	10	12

There are two level of pages tables

- Outer page table indexed by p_1.
- Inner page table indexed by p_2.

1. How does TLB look like?
2. What if we use 2-level paging for 64-bit logical address spaces?

3-Level Paging?

Let's consider 64-bit logical address space. We would structure a logical address as follows,

page number			page offset
p_1	p_2	p_3	d
32	10	10	12

How big is the 1st outer page table? How about the following?

page number					page offset
p_1	p_2	p_3	p_4	p_5	d
12	10	10	10	10	12

1. How does TLB look like?
2. What is the access latency when there is a TLB miss?

Hashed Page Tables

Use a hash table (sometimes called a map)

- Key. $H(pagenumber)$
- Value. A linked list of $(pagenumber, framenumber)$
  - Why do we need a list instead of a single value of $(pagenumber, framenumber)$?

1. How does TLB look like?
2. What is the access latency when there is a TLB miss?

Inverted Page Table

Observation. A computer system typically has small amount of physical memory when compared to logical address space.

- Rather than each process having a page table and keeping track of all possible logical pages, track all physical pages
  - Conceptually in a page table, an entry is for a page
  - In an inverted page table, an entry is for a frame (thus inverted)
- One page table entry for each real page (or frame) of memory. Each consists of
  - process identification information, and
  - frame information.
- Decreases memory needed to store each page table, but increases time needed to search the table when a page reference occurs
  - Use hash table to limit the search to one — or at most a few — page-table entries
  - TLB can accelerate access