Implementing the MMU

Let’s think of this in the simulator ...
- introduce a class to simulate the MMU hardware

```java
class MMU extends MainMemory {
  byte [] physicalMemory;
  AddressSpace currentAddressSpace;

  void setAddressSpace (AddressSpace* as) {
    byte readByte (int va) {
      int pa = currentAddressSpace.translate (va);
      return physicalMemory.read (pa);
    }
    void writeByte (int va, byte value) {
      int pa = currentAddressSpace.translate (va);
      physicalMemory.write (pa, value);
    }
  }
}
```

- currentAddressSpace is a hardware register
- the address space performs virtual-to-physical address translation

Implementing Address Translation

```java
class MMU extends MainMemory {
  byte [] physicalMemory;
  AddressSpace currentAddressSpace;

  void setAddressSpace (AddressSpace* as) {
    byte readByte (int va) {
      int pa = currentAddressSpace.translate (va);
      return physicalMemory.read (pa);
    }
    throw new IllegalAddressException (va);
  }
}
```

- Problem: allow segments to expand?
- Solution: let’s look at a couple of alternatives ...

But, Memory Use Not Known Statically

- An address space is
  - a set of segments
- A segment is
  - a single, variable-size, non-expandable chunk of physical memory
    - named by its base virtual address, physical address and length

Implementation in Simulator

```java
class AddressSpace ( Segment segment[] ) {
  int translate (int va) {
    for (int i=0; i<segments.length; i++) {
      if (offset >= segment[i].bounds[0] && offset <= segment[i].bounds[1]) {
        return segment[i].basePA + offset;
      }
    }
    throw new IllegalAddressException (va);
  }
}
```

- Problem: segments are not expandable; their size is static
  - some data structures should be expanded dynamically
  - make them contiguous

Expand Segments by Adding Segments

- What we know
  - segments should be non-expandable
- Idea
  - instead of expanding a segment
    - make a new one that is adjacent virtually, but not physically

```java
virtual addresses m ... n-1
allocate a new segment
virtual addresses n ... p-1
```

- Problem: oh no! another problem! what is it? why does it occur?

Eliminating External Fragmentation

- The problem with what we are doing is
  - allocating variable size segments leads to external fragmentation of memory
  - this is an inherent problem with variable-size allocation
- What about fixed sized allocation
  - could we make every segment the same size?
    - this eliminates external fragmentation
  - but, if we make segments too big, we’ll get internal fragmentation
    - so, they need to be fairly small and so we’ll have lots of them

```java
virtual addresses m ... n-1
allocate a new segment
```

- Problem: there is only one memory shared by all programs ...

Multiple Concurrent Program Executions

- So far we have
  - a single program
  - multiple threads
- Allowing threads from different program executions
  - we often have more than one thing we want to do at once(s)
    - threads spend a lot of time blocked, allowing other threads to run
    - but, often there aren’t enough threads in one program to fill up the gaps
- What is a program execution
  - an instance of a program running with its own state stored in memory
  - compiler-assigned addresses for all static memory state (globals, code etc.)
  - security and failse semantics suggest memory isolation for each execution
- But, we have a problem
  - there is only one memory shared by all programs ...

Virtual Address Space

- an abstraction of the physical address space of main (i.e., physical) memory
- programs access memory using virtual addresses
- hardware translates virtual address to physical memory addresses
- Process
  - a program execution with a private virtual address space
  - associated with authenticated user for access control & resource accounting
  - running a program with 1 or more threads
- MMU
  - memory management unit
  - the hardware that translates virtual address to physical address
  - performs this translation on every memory access by program

Base and Bounds

- An address space is
  - a single, variable-size, non-expandable chunk of physical memory
    - named by its base physical address and its length

As a class in the simulator

```java
class AddressSpace {
  int baseVA, basePA, bounds;
  void translate (int va) {
    int offset = va - baseVA;
    if (offset < 0 || offset > bounds) {
      throw new IllegalAddressException (va);
    }
    return basePA + offset;
  }
}
```

- Problems
  - there is no guarantee there will be room to expand a segment
  - memory management unit
  - the hardware that translates virtual address to physical address

But, There May Be No Room to Expand

- Issue
  - the address space of a program execution is divided into regions
    - for example: code, globals, heap, shared-libraries and stack
  - there are large gaps of unused address space between these regions

- Problem
  - a single base-and-bounds mapping from virtual to physical addresses
  - means that gaps in virtual address space will waste physical memory
    - this is the Internal Fragmentation problem

- Solution
  - introduce a class to simulate the MMU hardware

But, Address Space Use May Be Sparse

- Issue
  - the available memory space is not where we want it (i.e., adjacent to segment)
    - memory isolation for each execution
    - multiple concurrent program executions
    - external fragmentation is resolved by moving things to consolidate free space

- Problem
  - moving is possible, but expensive
  - to move a segment, all of its data must be copied
  - segments are large and memory copying is expensive

Paging

- Key Idea
  - Address Space is divided into set of fixed-size segments called pages
  - number pages in virtual address order
  - page number = virtual address / page size
- Page Table
  - indexed by virtual page number (vnp)
  - stores valid flag, because some segment numbers may be unused
  - more information needed to locate a segment, when segments are fixed size

Readings for Next Two Lectures

- Text
  - Physical and Virtual Addressing - Address Spaces, Page Tables - Page Faults
    - 2nd edition: 9.1-9.2, 9.3.2-9.3.4
    - 1st edition: 10.1-10.2, 10.3.2-10.3.4
- Multiple Concurrent Program Executions
  - a program execution with a private virtual address space
  - associated with authenticated user for access control & resource accounting
  - running a program with 1 or more threads
- Virtual Memory
  - an abstraction of the physical address space of main (i.e., physical) memory
  - hardware translates virtual address to physical memory addresses
  - Process
    - a program execution with a private virtual address space
    - associated with authenticated user for access control & resource accounting
    - running a program with 1 or more threads
- MMU
  - memory management unit
  - the hardware that translates virtual address to physical address
  - performs this translation on every memory access by program

Segmentation

- An address space is
  - a set of segments
- A segment is
  - a single, variable-size, non-expandable chunk of physical memory
    - named by its base virtual address, physical address and length

Implementation in Simulator

```java
class AddressSpace ( Segment segment[] ) {
  int translate (int va) {
    for (int i=0; i<segments.length; i++) {
      int pa = segment[i].basePA + offset;
        return pa;
      }
    throw new IllegalAddressException (va);
  }
}
```

- Problem
  - there are large gaps of unused address space between these regions
  - external fragmentation is resolved by moving things to consolidate free space

- Solution
  - introduce a class to simulate the MMU hardware

But, Moving Segments is Expensive

- Issue
  - if there is space in memory to store expanding segment, but not where it is
  - could move expanding segment or other segments to make room

- Problem
  - external fragmentation is resolved by moving things to consolidate free space

- Solution
  - introduce a class to simulate the MMU hardware

Translation with Many Segments

- What is wrong with this approach if there are many segments?
  - a segment can only expand to fill space between it and the next segment

- Problem
  - there is no guarantee there will be room to expand a segment
  - the available memory space is not where we want it (i.e., adjacent to segment)

- Solution
  - introduce a class to simulate the MMU hardware
**New terminology**

- **page**: a small, fixed-sized (4-KB) segment
- **page table**: virtual-to-physical translation table
- **pte**: page table entry
- **pfn**: physical page frame number
- **offset**: byte offset of address from beginning of page

**Translation using a Page Table**

```java
class PageTableEntry {
    boolean isValid;
    int pfn;
}
```

**The simulator code**

```java
int offset = va % PAGE_SIZE;
int vpn = va / PAGE_SIZE;
int pfn = pte[vpn].pfn * PAGE_SIZE + offset;
throw new IllegalAddressException (va);
```