Resumen Adress Translation - SistOp

Lautaro Bachmann

Contents

Mechanism: Address Translation

Introduction

address translation

With address translation, the hardware transforms each memory access changing the **virtual** address provided by the instruction to a **physical** address where the desired information is actually located.

the hardware alone cannot virtualize memory,

The OS must get involved at key points to set up the hardware so that the correct translations take place; it must thus **manage memory,** keeping track of which locations are free and which are in use, and intervening to maintain control over how memory is used.

create a beautiful illusion:

The goal is to create the illusion that the program has its own private memory, where its own code and data reside.

Assumptions

- the user's address space must be placed **contiguously** in physical memory.
- the size of the address space is **less than the size of physical memory.**
- each address space is exactly the **same size.**

Dynamic (Hardware-based) Relocation

base and bounds;

the technique is also referred to as **dynamic relocation**;

we'll need two hardware registers within each CPU: the base register, and the bounds register

In this setup,

each program is written and compiled as if it is loaded at address zero.

when a program starts running,

the OS decides where in physical memory it should be loaded and sets the **base register** to that value.

when the process is running.

When any memory reference is generated by the process, it is translated by the processor in the following manner:

physical address = virtual address + base

Each memory reference

generated by the process is a virtual address;

address translation;

the hardware takes a virtual address the process thinks it is referencing and transforms it into a physical address which is where the data actually resides.

the bounds register

is there to help with protection. the processor will first check that the memory reference is **within bounds** to make sure it is legal;

If a process generates a virtual address that is greater than the bounds, or one that is negative, the CPU will raise an exception,

memory management unit (MMU);

Sometimes people call the part of the processor that helps with address translation the **memory management unit (MMU)**;

bound registers can be defined two ways.

In one way
it holds the size of the address space, and thus the hardware checks the virtual address against it first before adding the base.

In the second way,
it holds the physical address of the end of the address space, and thus the hardware first adds the base and then makes sure the address is within bounds.

Hardware Support: A Summary

Hardware Requirements

- **Privileged mode:** Needed to prevent user-mode processes from executing privileged operations
- **Base/bounds registers:** Need pair of registers per CPU to support address translation and bounds checks

- **Ability to translate virtual addresses and check if within bounds:** Circuitry to do translations and check limits; in this case, quite simple
- **Privileged instruction(s) to update base/bounds:** OS must be able to set these values before letting a user program run
- **Privileged instruction(s) to register exception handlers:** OS must be able to tell hardware what code to run if exception occurs
- **Ability to raise exceptions:** When processes try to access privileged instructions or out-of-bounds memory

Operating System Issues

OS Requirements

- **Memory management:** Need to allocate memory for new processes; Reclaim memory from terminated processes; Generally manage memory via free list
- **Base/bounds management:** Must set base/bounds properly upon context switch
- **Exception handling:** Code to run when exceptions arise; likely action is to terminate offending process

Summary

address translation.

With address translation, the OS can control each and every memory access from a process, ensuring the accesses stay within the bounds of the address space.

hardware support,

helps to perform the translation quickly for each access, turning virtual addresses into physical ones

All of this is performed in a way the process has no idea its memory references are being translated,

base and bounds charasteristics

Efficiency

Base-and-bounds virtualization is quite efficient,

protection;

the OS and hardware combine to ensure no process can generate memory references outside its own address space.

Space inefficiencies.

internal fragmentation, the space inside the allocated unit is not all used (i.e., is fragmented) and thus wasted.