An Overview of Virtual Memory

Azza Abouzied

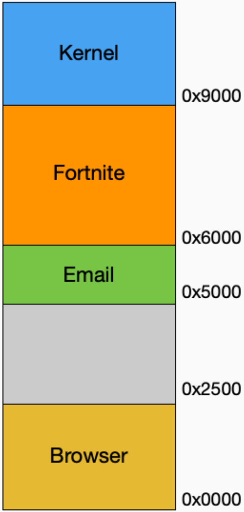
Do you know your latency numbers?

```
Latency Comparison Numbers
--------------------------
L1 cache reference
Branch mispredict
L2 cache reference
Mutex lock/unlock
Main memory reference
L1
Compress 1K bytes with Zippy
Send 1K bytes over 1 Gbps network
Read 4K randomly from SSD*
Read 1 MB sequentially from memory
Round trip within same datacenter
Read 1 MB sequentially from SSD*
memory
Disk seek
datacenter roundtrip
Read 1 MB sequentially from disk
memory, 20X SSD
Send packet CA->Netherlands->CA
```

Do you know your latency numbers?

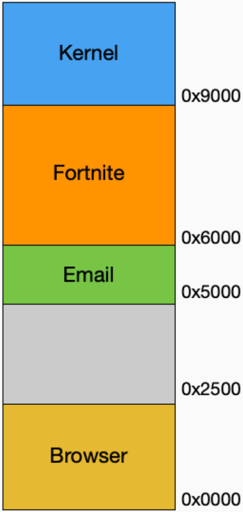
```
Latency Comparison Numbers
--------------------------
L1 cache reference                           0.5 ns
Branch mispredict                            5   ns
L2 cache reference                           7   ns                14x L1
Mutex lock/unlock                           25   ns
Main memory reference                      100   ns                200x
L1
Compress 1K bytes with Zippy             3,000   ns
Send 1K bytes over 1 Gbps network       10,000   ns  0.01ms
Read 4K randomly from SSD*             150,000   ns  0.15ms 300000xL1
Read 1 MB sequentially from memory     250,000   ns    0.25 ms
Round trip within same datacenter      500,000   ns    0.5  ms
Read 1 MB sequentially from SSD*     1,000,000   ns    1    ms   4X
memory
Disk seek                           10,000,000   ns   10    ms   20x
datacenter roundtrip
Read 1 MB sequentially from disk    20,000,000   ns   20    ms   80x
memory, 20X SSD
Send packet CA->Netherlands->CA    150,000,000   ns  150    ms
```

Can we keep everything in memory?



Life would be great if we didn't have to pay 0.15ms to access disk. Let's keep everything in memory!

Problem 1: The Jump Issue

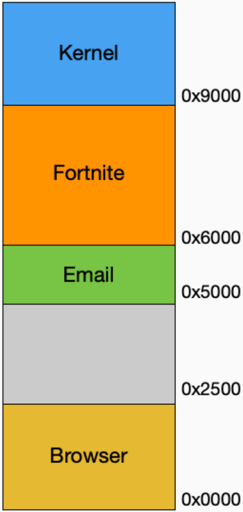


What do the following mean?

Browser: jmp 1000

Fortnight: jmp 1000

Problem 1: The Jump Issue



What do the following mean?

Browser: jmp 1000

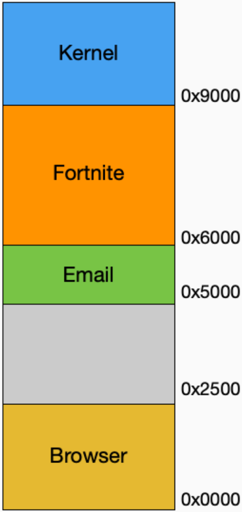
Fortnight: jmp 1000

A work around: static relocation

When loading at adr 0x7000, add 0x7000 to every address in the executable code!

- Slow loading
- Programs need to define what is relocatable and what is not.

Problem 2: The Protection Issue



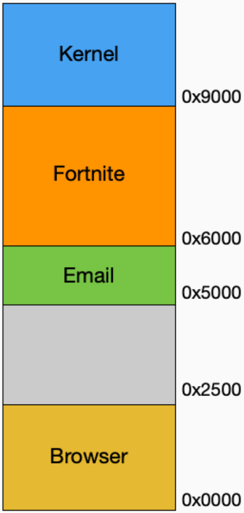
Writes external to my memory

Email's address offsets can range from 0 to 0x1000.

Email writes to address offset 0x1001!

space

Problem 2: The Protection Issue

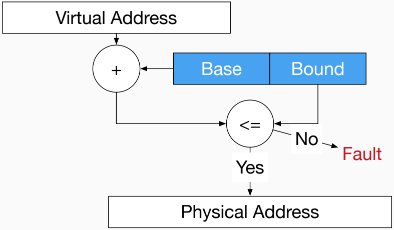


Writes external to my memory

Email's address offsets can range from 0 to 0x1000.

Email writes to address offset 0x1001!

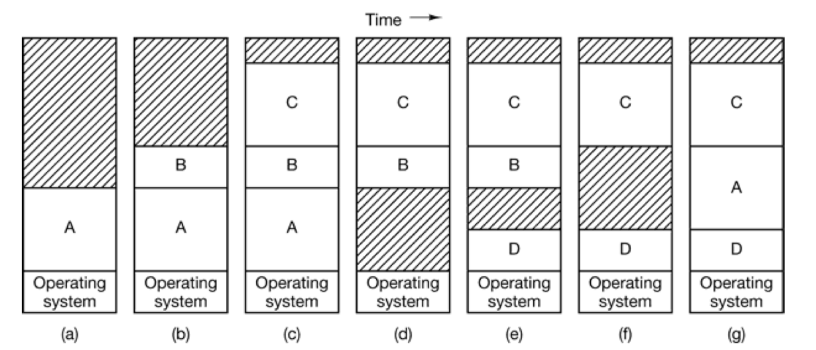
A work around: address space & base+bound



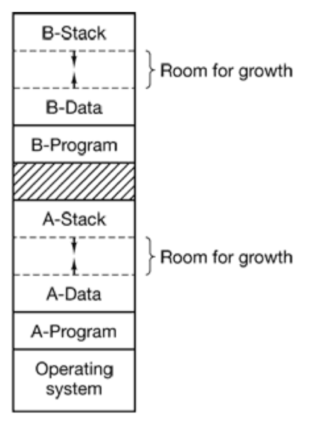
Problem 3: More processes than can fit in memory!

A work around: Swapping

Bring in entire process from disk, run it then put it back on disk



Problem 4: A process grows!



Work arounds?

Give more than needed!

space

Problem 5: What if a process just needs more memory than your entire RAM?

The seeds of virtual memory

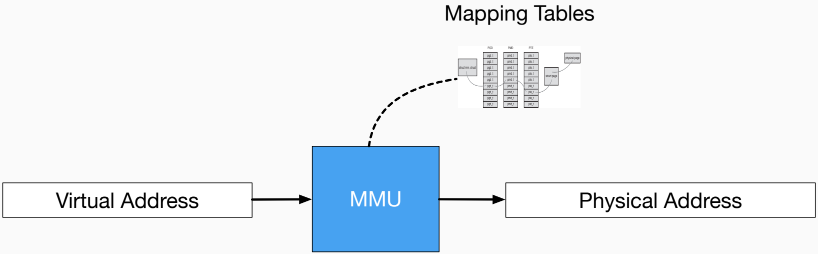
What do you want from your virtual memory system?

1. Simplicity: Processes get a *flat linear address space*. Access addresses from 0 to 2^{32} or 0 to 2^{64} depending on architecture!

2. Flexibility or Deception: Processes need to move in and out of memory with partial parts in memory and other parts on disk. Satisfy processes that require more memory than you have!

3. Efficiency: 80/20 rule. Most of what you need is already on memory. Occasionally, you will go to disk to get the rest!

The seeds of virtual memory



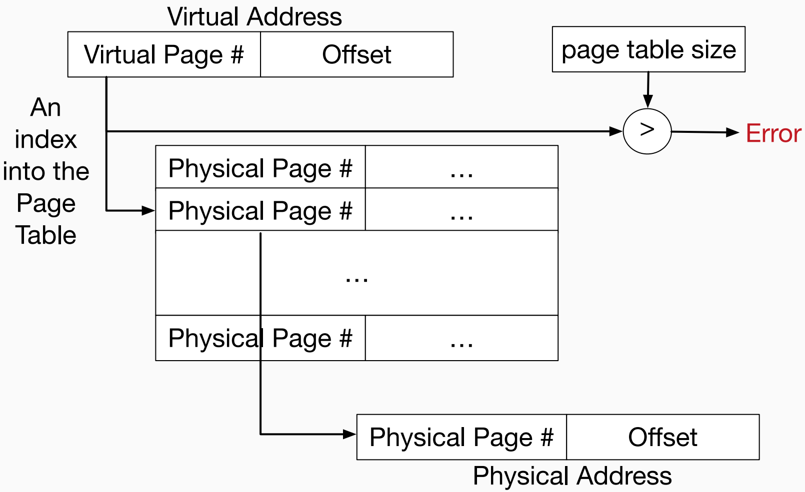
The kernel’s job is set up these mapping tables *for each process*. Hardware handles the mapping table lookup on every virtual address operation.

But what is the right granularity of mapping? a byte-to-byte? a whole segment?

Most operating systems opt for paging. Each virtual address maps to a page address and an offset within it. The mapping tables are called page tables.

How does paging work?

The simplified view of paging



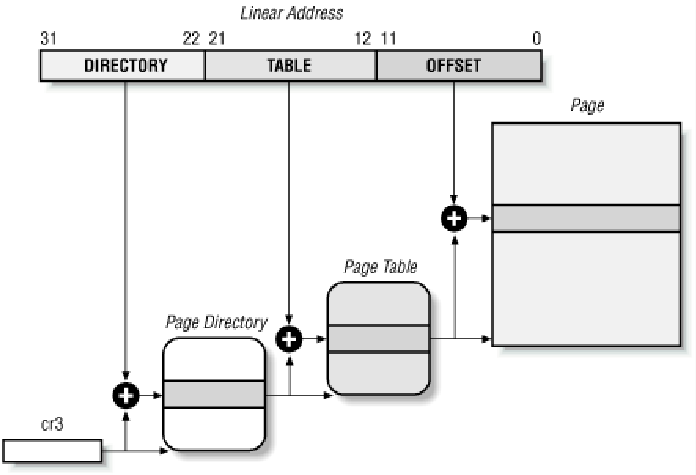
What are the right numbers for?

Page size: On 32-bits, this is typically 4KB. Can also be 8KB on 64-bit architectures.

Bits in the offset: If a page is 4KB we have 4096 unique bytes we should be able to address so $2^{12} = 4096$. 12 bits.

Page table size: If you want to support flat addresses on 32- or 64-bit machines then you need to map every virtual address to a physical address. On 32-bit: $32 - 12 = 20$, the table should have 2^{20} entries. With 2^{52} entries on 64-bit with 4KB pages, you can't fit the page table for a process in memory!

Multi-level paging



How does multi-level paging[1] help keep the size of the page tables small?

[1] Linux uses three-level paging.

Questions?