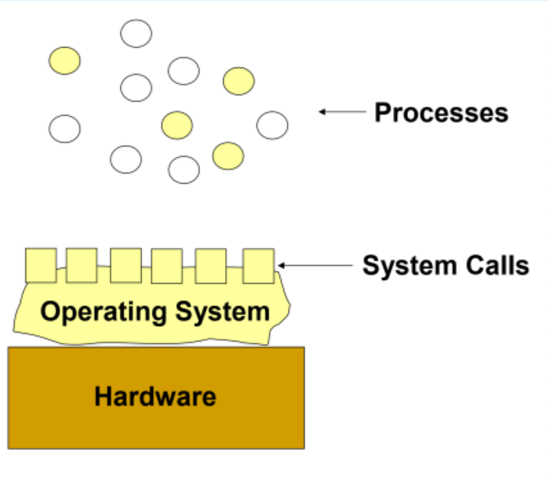
System Calls

- Recall Relationship btw processes, hardware and OS



- system calls → function like entities
  → interface into the hardware system
  ↳ ex fork()

- OS Code Must Have Special Privileges
  → yellow parts of the above diagram must be capable of doing privileged operations on the hardware
  → ∴ What we saw in the ARM instruction set is only a subset of the processor capabilities
  ↳ there must be some additional must be included for very privileged programs such as the operating system

- ∴ A complication rises in terms of who can make system calls
  ↳ programs that you and I write should not be allowed to write privileged instructions
  ↳ This has to be addressed by the hardware.
  ↳ We need to understand what happens when a system takes place
  ↳ not same as a function call.

Mechanics of System Calls

- Processor hardware is designed to operate in at least two modes of operation

↳ Ordinary or user mode

↳ Privileged or system mode

- How does hardware know if you are in user mode or privileged mode?
  ↳ special registers

- If I write a program that makes a system call, then in order to execute this system call, my program has to change mode from user mode to system mode.
  ↳ How can this change of mode take place?
    ↳ This must be done explicitly with the execution of an instruction

- A system call is entered using a special machine instruction that switches processor mode from user to system before transferring control

Process Lifetime

- The time between the creation and termination of a process.

$$\underset{\text{exit()}}{\underset{\downarrow}{t_{term}}} - \underset{\text{fork()}}{\underset{\downarrow}{t_{creation}}}$$

- At any given point in time, a running process is executing either in user mode or in system mode

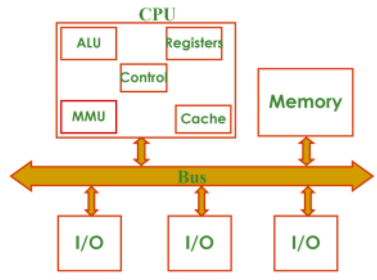
- Therefore if we look at the lifetime of a process as a time interval some component will be spent executing in user mode and some component will be spent on system mode.

- As a result, we can find CPU time spent in user mode vs. system mode.
  ↳ there are commands that will give you this information

↳ there are commands that will give you this information

Operating System

- Entirely software, not hardware but the OS may have affected the way the hardware is designed.
- The software manages the resources of a computer system.
- What do we mean by resources? There are 4 types of resources
  - → CPU time
  - → Main memory
  - → I/O devices
  - → Software resources

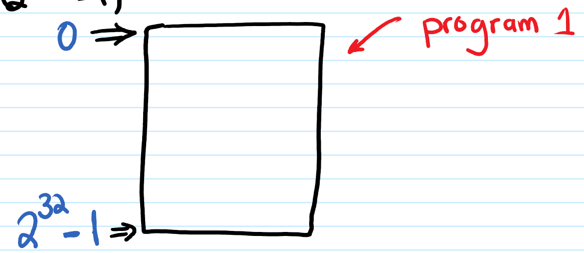


Main Memory Management

- How does the operating system manages the main memory

Address Translation

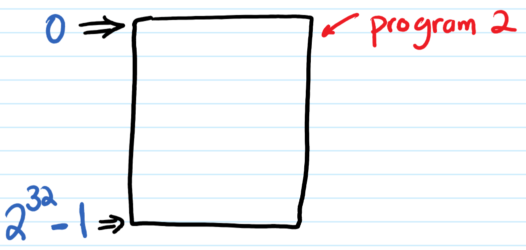
- On any particular computer system, there could be many programs in execution
- Each of these programs is compiled and assumed to use a virtual addresses from $0 - (2^{32}-1)$

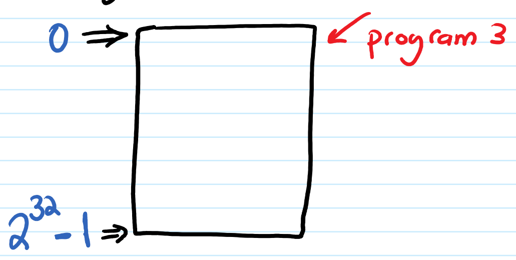


virtual memory

32 bits

- There could be many such programs executing at the same time





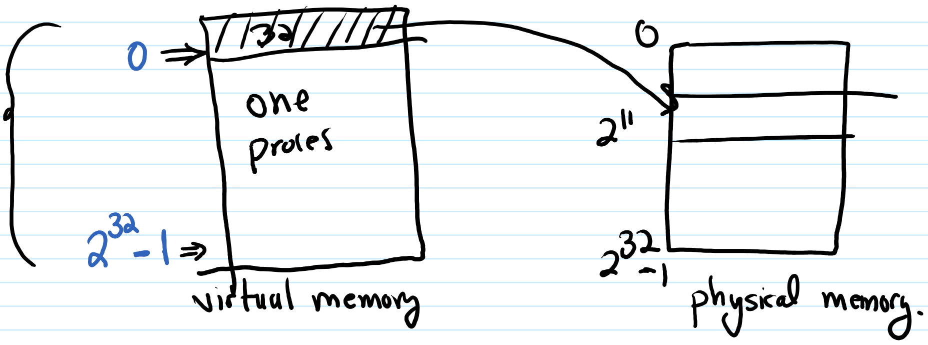
- Therefore, programs can affect each other and there is a need to protect one program from another

  ↳ This is done through address translation

- The idea is that addresses $0-(2^{32}-1)$ are assumed to be specific to a process or program and prior to being sent to main memory during program execution, the address will be translated to an actual memory address

- This translation is done by a piece of hardware called MMU

- To translate a virtual address to the corresponding physical address, a table of translation information is needed.



- This raises the issue of the address translation table size...

Address Translation Table Size

- Assume that we have on entry in the table per each 32 bit address

- Therefore the size of this table would be

$$\underline{\overset{4GBytes}{2^{32}}} \times \underline{\overset{4Bytes}{32b}} = 16GBytes \Rightarrow \underline{16\,GB}$$

number of entries in the table

each entry is 32 bit in size.

each word = 4Byte

mistake in previous version of notes

- If there is an entry for each word address

$$2^{32}/\underset{4}{\cancel{2}} \times 32b = \overset{4}{\cancel{8}}GB$$

- If there is an entry for each 256B unit address $16GB/256 =$ 64MB

- Therefore, the size of the address translation table can be reduced by not managing translations on byte basis but at some larger granularity.