a | b = ~(~a & ~b)

a ^ b = (a & ~b) | (~a & b)

A picture containing text, crossword puzzle

Description automatically generatedb = 1 byte, w = 2 bytes, l = 4 bytes, q = 8 bytes

Shape, rectangle

Description automatically generatedShape, rectangle

Description automatically generated

**Table

Description automatically generated with medium confidenceNormalized value** (exp ≠ 000…0 and exp ≠ 111…1)

E = Exp – Bias. Bias =

**Denormalized Value** (exp = 000…0)

Exponent value: E = 1 – Bias (**equispaced**)

**Infinity**: exp = 111…1, frac = 000…0

**NaN**: exp = 111…1, frac ≠ 000…0

Chart

Description automatically generated**x86-64 linux calling convention:**

Integer parameters:

%rdi, %rsi, %rdx, %rcx, %r8 and %r9

Others are stored in stack, pushed in reversed (right-to-left) order

**CF** Carry Flag (for unsigned) **SF** Sign Flag (for signed)

**ZF** Zero Flag **OF** Overflow Flag (for signed)

Implicitly set by arithmetic operations (but **not set by leaq**

**instruction**):

addq Src DestDest (t = a + b)

**CF** set if carry out from most significant bit (unsigned overflow)

**ZF** set if t == 0

**SF** set if t < 0 (as signed)

**OF** set if two’s complement (signed) overflow

**Rules for turning on the carry flag**

1. The carry flag is set if the addition of two numbers causes a carry out of the most significant bits added.

1111 + 0001 = 0000 (carry flag is turned on)

2. The carry (borrow) flag is also set if the subtraction of two numbers requires a borrow into the most significant (leftmost) bits subtracted

0000 - 0001 = 1111 (carry flag is turned on)

**Rules for turning on the overflow flag**

1. If the sum of two numbers with the sign bits off yields a result number with the sign bit on

0100 + 0100 = 1000 (overflow flag is turned on)

2. If the sum of two numbers with the sign bits on yields a result number with the sign bit off

1000 + 1000 = 0000 (overflow flag is turned on)

**Note that different from above (1111 + 0001 = 0000), the result is correct even though CF is set**

unsigned arithmetic -> CF | signed arithmetic -> OF

cmp b, a Computes *b - a* (just like sub). Sets condition codes based on result, but **does not change *b***

test a, b Computes 𝑏 ∧ 𝑎 just like and. Sets condition codes (only SF and ZF) based on result, but **does not change 𝒃**

A picture containing diagram

Description automatically generatedTable

Description automatically generated

**movzbl**: zero-extend, byte -> long. **movslq**: sign-extend, long -> quad. Etc.

**Buffer overflow attacks**

Stack Smashing Attacks: overwrite normal return address. Code Injection Attacks: overwrite normal return address and jump to exploit code

**Measures**

Avoid overflow vulnerabilities: strcpy -> strncpy. Employ system-level protections: randomized stack offsets, nonexecutable code segments. Have compiler use stack canaries

**Return-Oriented Programming Attacks**

Work around stack randomization and marking stack nonexecutable. Does not overcome stack canaries

**Internal Fragmentation**: For a given block, internal fragmentation occurs if payload is smaller than block size

Caused by: Overhead of maintaining heap data structures | Padding for alignment purposes | Explicit policy decisions

Depends only on the pattern of previous requests, easy to measure

Can be reduced by changing our representations of the free list, either through encoding information in unused bits or reducing the size of our free list

nodes.

**External Fragmentation:** Occurs when there is enough aggregate heap memory,but no single free block is large enough

Depends on the pattern of future requests, difficult to measure

Can be decreased by coalescing or using a best-fit algorithm

Diagram

Description automatically generatedGraphical user interface, text, application

Description automatically generated

The main benefit of cache over main memory is that you can access data much quicker: ● Caches are built to be small and easy to access ● They are often on or very close to the CPU chip ● They take in the tens of cycles ● Sizes on the order of kilobytes.

**Local optimizations** work inside a single basic block: Constant folding, strength reduction, dead code elimination, (local) common subexpression elimination, …

**Global optimizations** process the entirecontrol flow graph of afunction: Loop transformations,code motion, (global)CSE, …

**Table

Description automatically generatedText

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated**

Program symbols are either strong or weak: ▪ Strong: procedures and initialized globals ▪ Weak: uninitialized globals

**Benefits of virtual memory**

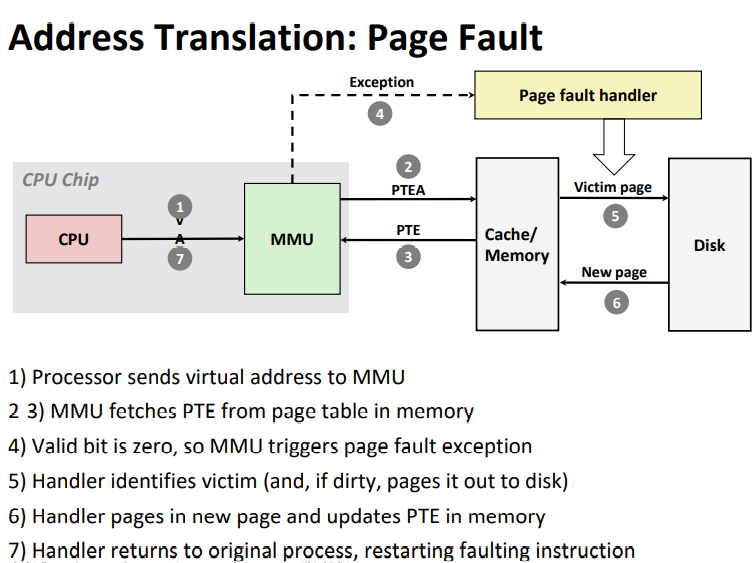
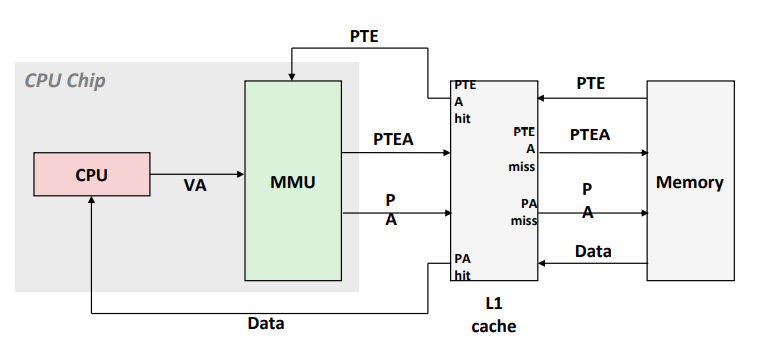
Uses main memory efficiently: Use DRAM as a cache for parts of a virtual address space

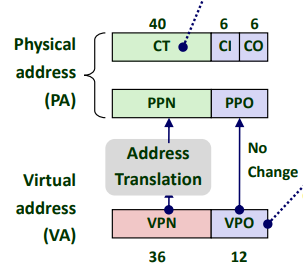
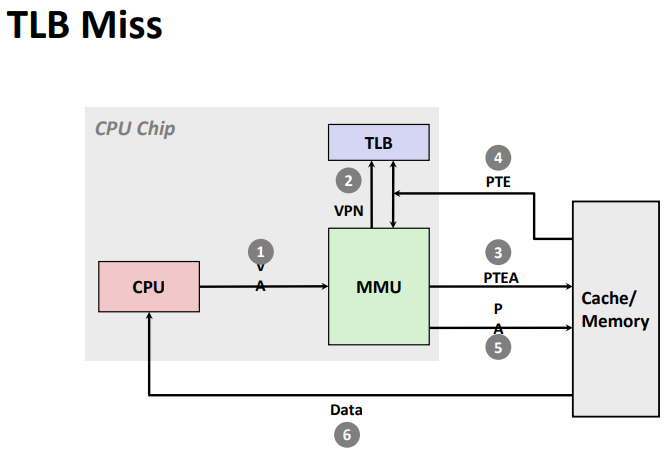
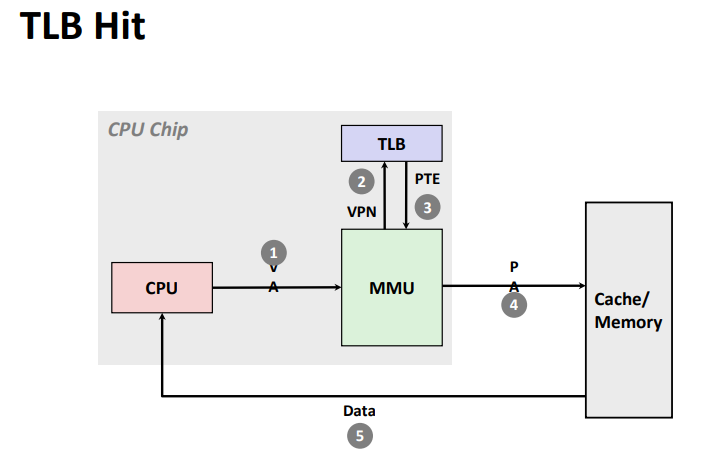
Simplifies memory management: Each process gets the same uniform linear address space

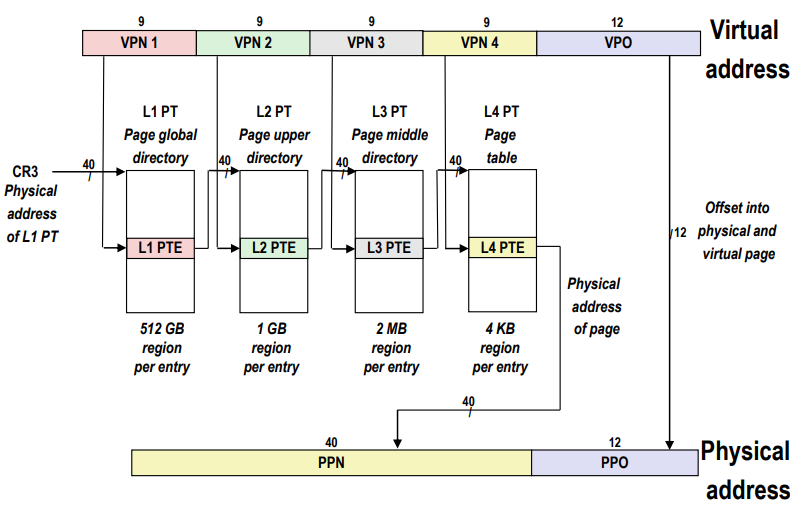
Isolates address spaces: One process can’t interfere with another s memory, User program cannot access privileged kernel information and code

Virtual memory keeps address spaces separate by ensuring that each process has its own page table that maps virtual addresses to physical addresses. This is done inside the operating system. Therefore, multiple processes can access the same virtual address simultaneously--this is possible because the virtual address within each process would map to a different physical page in physical memory.

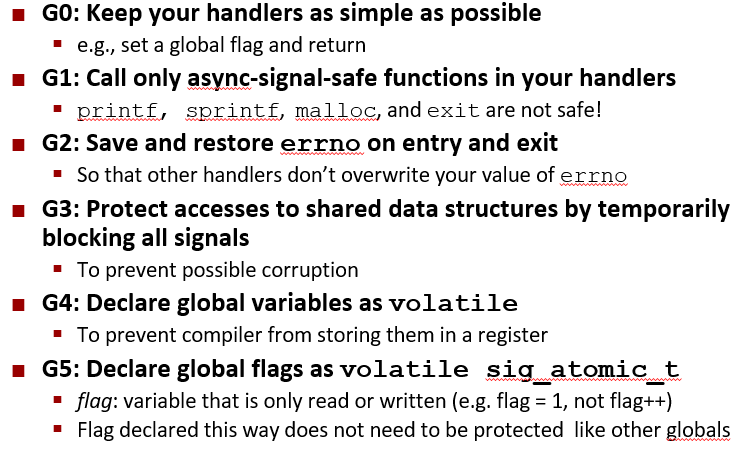
The OS can share information between address spaces by mapping virtual pages in each of the page tables to the same physical page of memory. This is most useful for code libraries--if multiple processes use the same code library, any process that needs the library can map





Diagram

Description automatically generated

Diagram

Description automatically generated

**Classical problem classes of concurrent programs: Races**: outcome depends on arbitrary scheduling decisions elsewhere in the system | **Deadlock**: improper resource allocation prevents forward progress | **Livelock/Starvation/Fairness**: external events and/or system scheduling decisions can prevent sub task progress

Graphical user interface, text, application

Description automatically generatedText

Description automatically generatedText

Description automatically generatedGraphical user interface, text

Description automatically generated

Text

Description automatically generatedGraphical user interface, text, application

Description automatically generatedTable

Description automatically generated

**Internet protocol** software running on each host and router. Protocol is a set of rules that governs how hosts and routers should cooperate when they transfer data from network to network. Smooths out the differences between the different networks.

1. Provides a naming scheme: An internet protocol defines a uniform format for host addresses. Each host (and router) is assigned at least one of these internet addresses that uniquely identifies it. 2. Provides a delivery mechanism: An internet protocol defines a standard transfer unit (packet). Packet consists of header and payload. Header: contains info such as packet size, source and destination addresses. Payload: contains data bits sent from source host.

Text

Description automatically generatedTable

Description automatically generated

The stacked architecture of internet protocol consists of many protocols that interact with the protocols just above and below the layer of the given protocol. Each layer of a specific network model may be responsible for a different function of the network. Each layer will pass information up and down to the next subsequent layer as data is processed.

Advantages: **Interoperability** and allows for so many protocols supported by the current internet | **Portability**: Layered networking protocols are much easier to port from one system or architecture to another | **Compartmentalization of Functionality**: The compartmentalization or layering of processes, procedures and communications functions gives developers the freedom to concentrate on a specific layer or specific functions within that layer’s realm of responsibility without the need for great concern or modification of any other layer.

Disadvantage: Overhead increased due to headers of each layer