15-213: Recitation 15

Final Exam Review
Monday April 30th, 2012

Administration

- Congratulations on finishing Proxy Lab!
- Final Exam is on Friday May 11th
  - Time: 5:30pm
  - Location: DH 2210 and DH 2315

Today

- We will review <u>everything!</u>
- Ask questions as we go!

First Half

- Integer and Floating Point Numbers
- X86 Assembly
  - Procedure Calls and Stack Discipline
  - Structures and Memory Layout
- Memory Hierarchy and Caching
- Linking (Not on the Final)

Integer Numbers

Bits: $x = \boxed{b_{n-1} \mid b_{n-2} \mid \ldots \mid b_2 \mid b_1 \mid b_0}$

Signed: $x = -2^{n-1} \cdot b_{n-1} + \sum_{i=0}^{n-2} 2^i \cdot b_i$

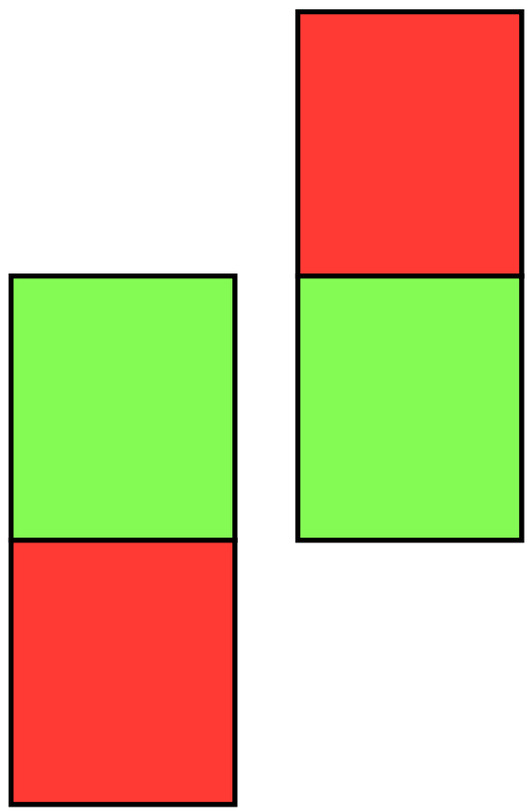
Unsigned: $x = \sum_{i=0}^{n-1} 2^i \cdot b_i$

Little vs. Big Endian

Integer Numbers

$U_{max} = 2^n - 1$

$2^{n-1} - 1 = T_{max}$



Zero

$-2^{n-1} = T_{min}$

Floating Point

- Bits: $x =$ 

- Normalized: $x = -1^s \times 1.\text{frac} \times 2^{\text{exp}-\text{bias}}$
- Denormalized: $x = -1^s \times 0.\text{frac} \times 2^{1-\text{bias}}$
- Infinity and NaN: $\text{exp} = 11\ldots1 \;\; \text{frac} = 0?$

  $\text{bias} = 2^{\text{expbits}-1} - 1$
- Round to Even

X86 Assembly

- Registers:
  - Callee Saved: *%ebx,%esi, %edi*
  - Caller Saved: *%eax, %ecx, %edx*
  - Special: *%eax, %esp, %ebp, %eip*

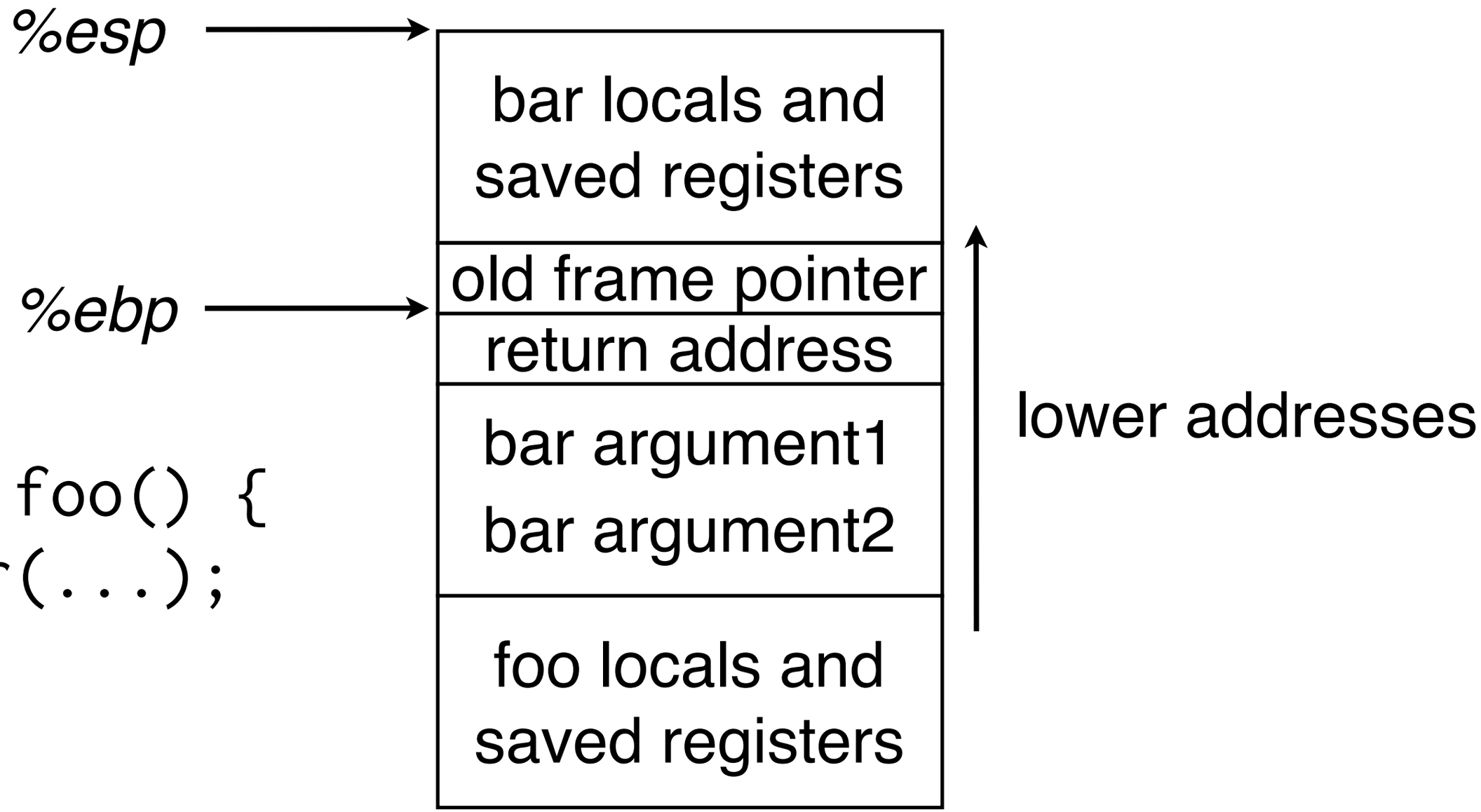
X86 Addressing

- disp(base,offset,scale)
  - base + scale*offset + disp
- *movl (%eax,%edx,4),%ebx*
  - `int x = a[idx];`
- *leal (%eax,%edx,4),%ebx*
  - `int * p = &a[idx];`

Instructions

- Arithmetic Instructions: *add src, dest*
  - *add, sub, imul, ...*
- Bit Manipulation
  - *and, or, xor, sal, sar, shr, ...*
- Condition Codes and Flow control
  - *cmp, test, jmp, je, jne, ...*

IA32 Stack Discipline




```
void foo() {
  bar(...);
}
```

X86-64 Functions

- Frame Pointer is often omitted
- push and pop → sub and mov
- Arguments passed in registers
  - *%rdi, %rsi, %rdx, %rcx*
- The red zone in leaf functions

Structure Layout

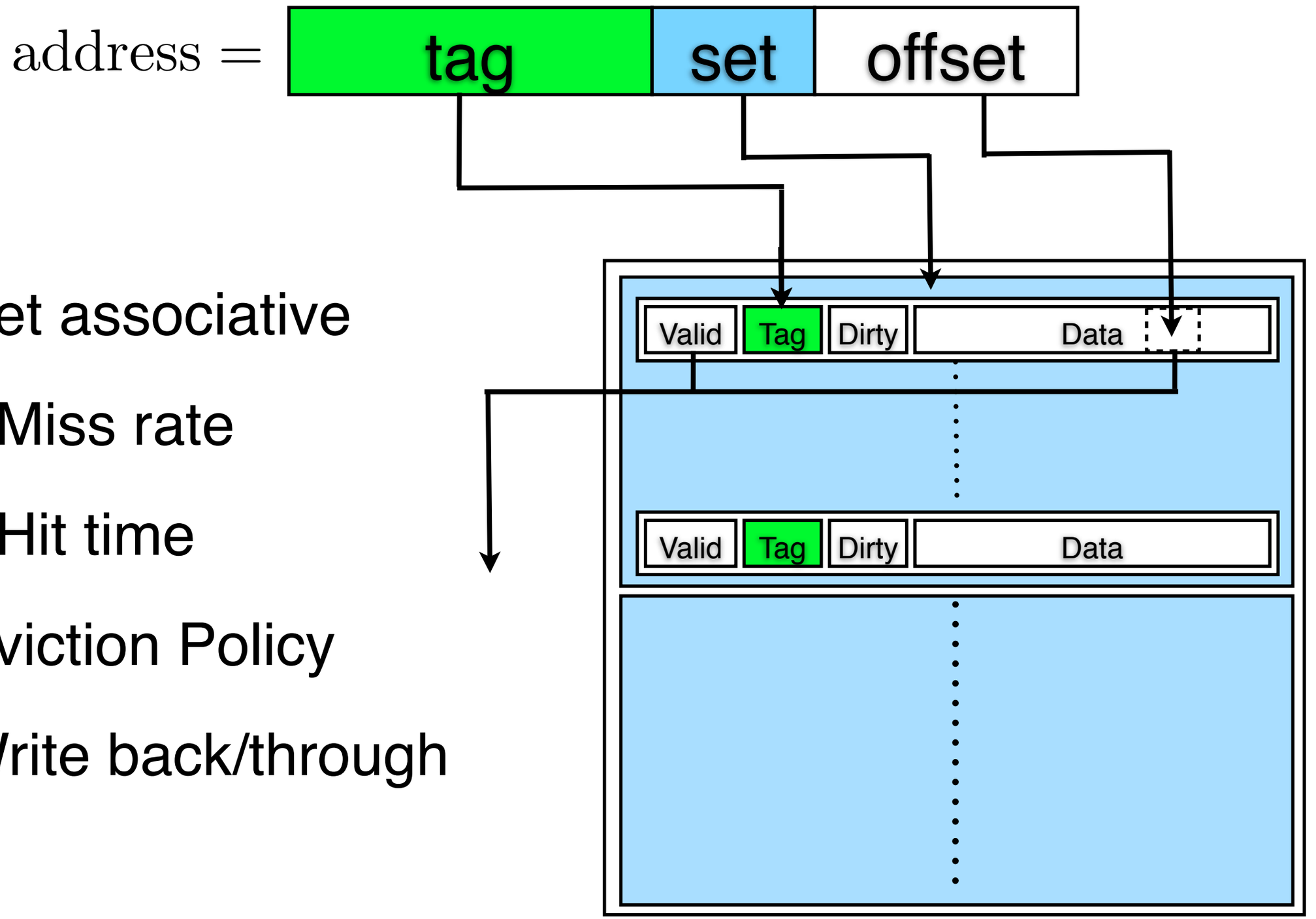
- Elements arranged in order
- Padding inserted to align each element
- Structure aligned to largest primitive
- Alignment rules vary by platform

Memory Hierarchy

- SRAM is fast, but expensive
- DRAM is cheaper, smaller, not as fast
- Disk is much cheaper, much slower
  - Seek time is the killer
- Caching: Keep data likely to be accessed in faster storage

L1/L2 Cache

- Set associative
  - Miss rate
  - Hit time
- Eviction Policy
- Write back/through



Second Half

- Signals and Processes
- Concurrent Programming and Threading
- Virtual Memory
- Dynamic Memory Allocation
- Network Programming

Signals

- Notify a process of an event
  - Interrupt, SegFault, Child terminated
- OS handles pending unblocked signals
  - Ignore, terminate, or call handler
  - Interrupts current execution

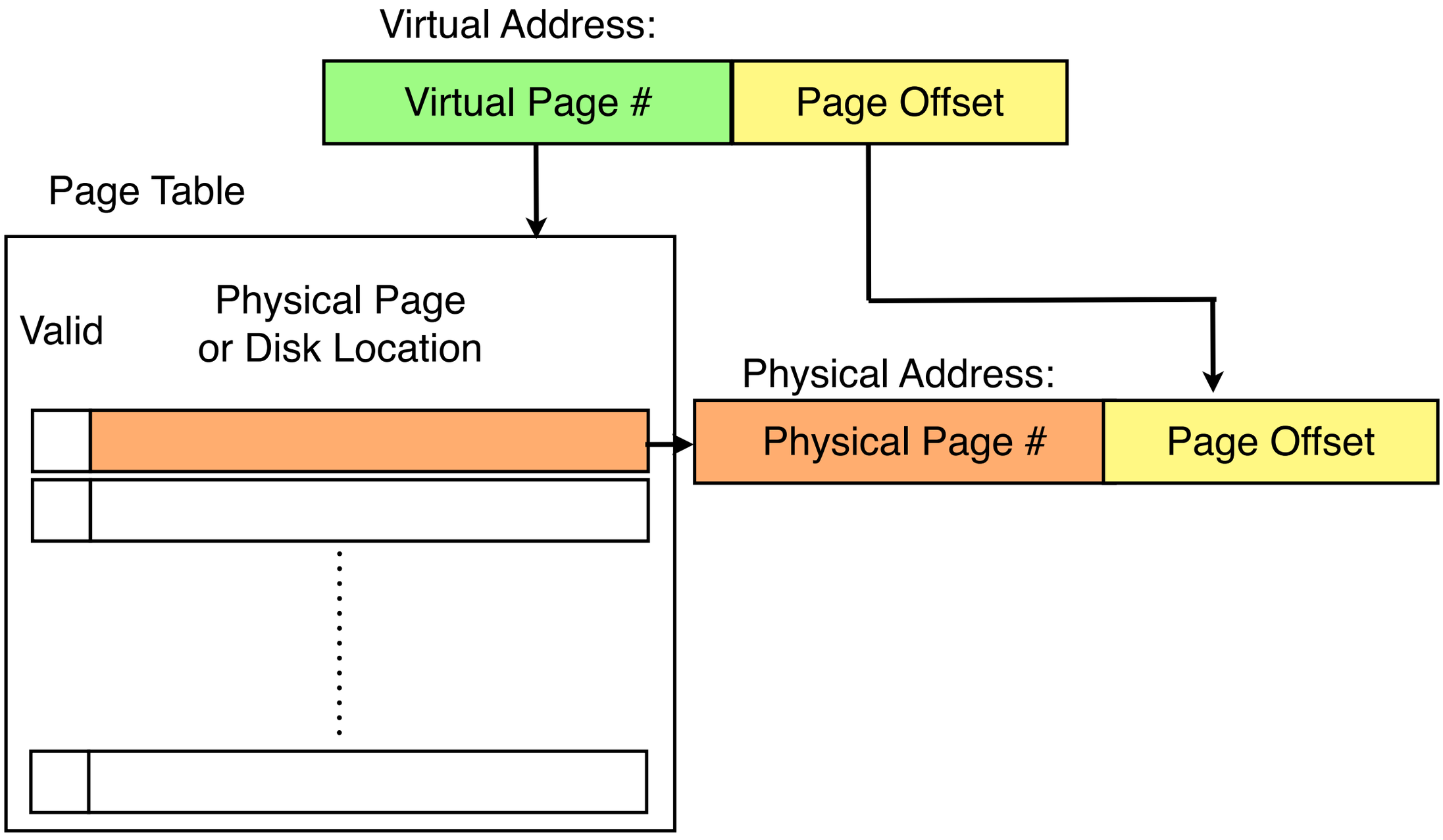
Virtual Memory

- Programs use virtual addresses, but
- RAM accessed with physical addresses
- Processor must map virtual to physical

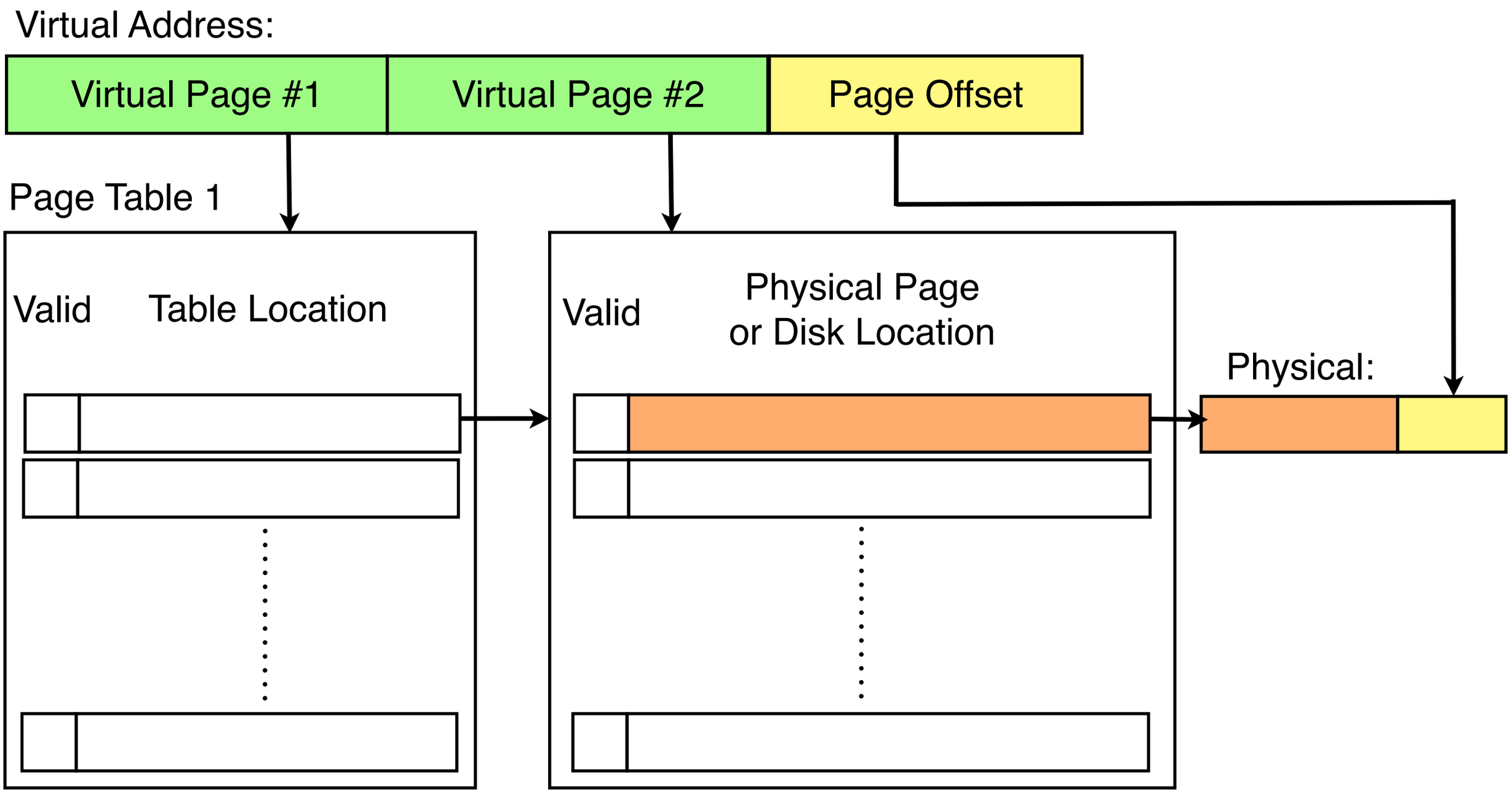
Virtual Memory

- Why?
  - Use disk as a cache for RAM
  - Process isolation
  - Share resources between processes

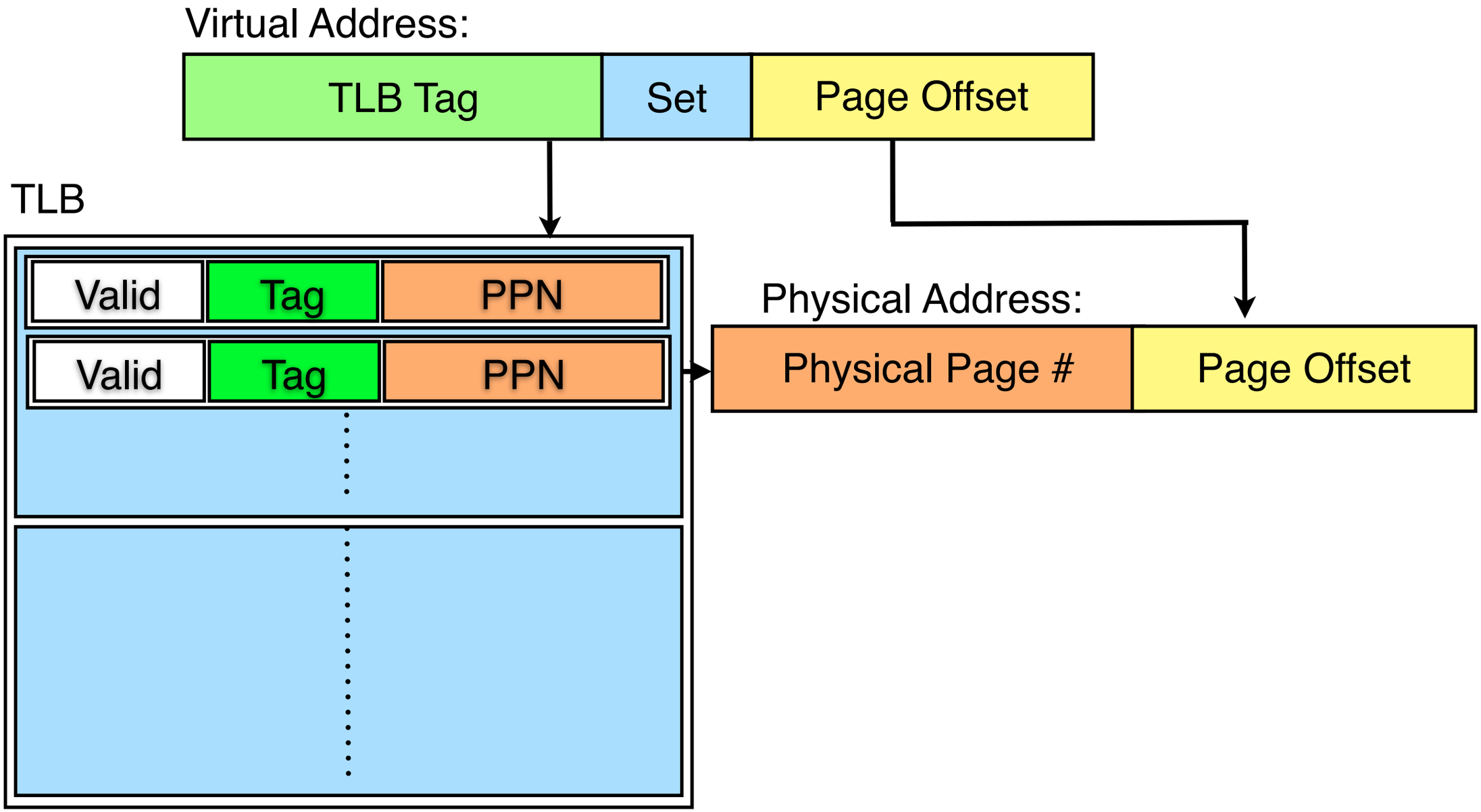
Page Table


Virtual Address:
Virtual Page #
Page Offset
Page Table
Valid
Physical Page
or Disk Location
Physical Address:
Physical Page #
Page Offset

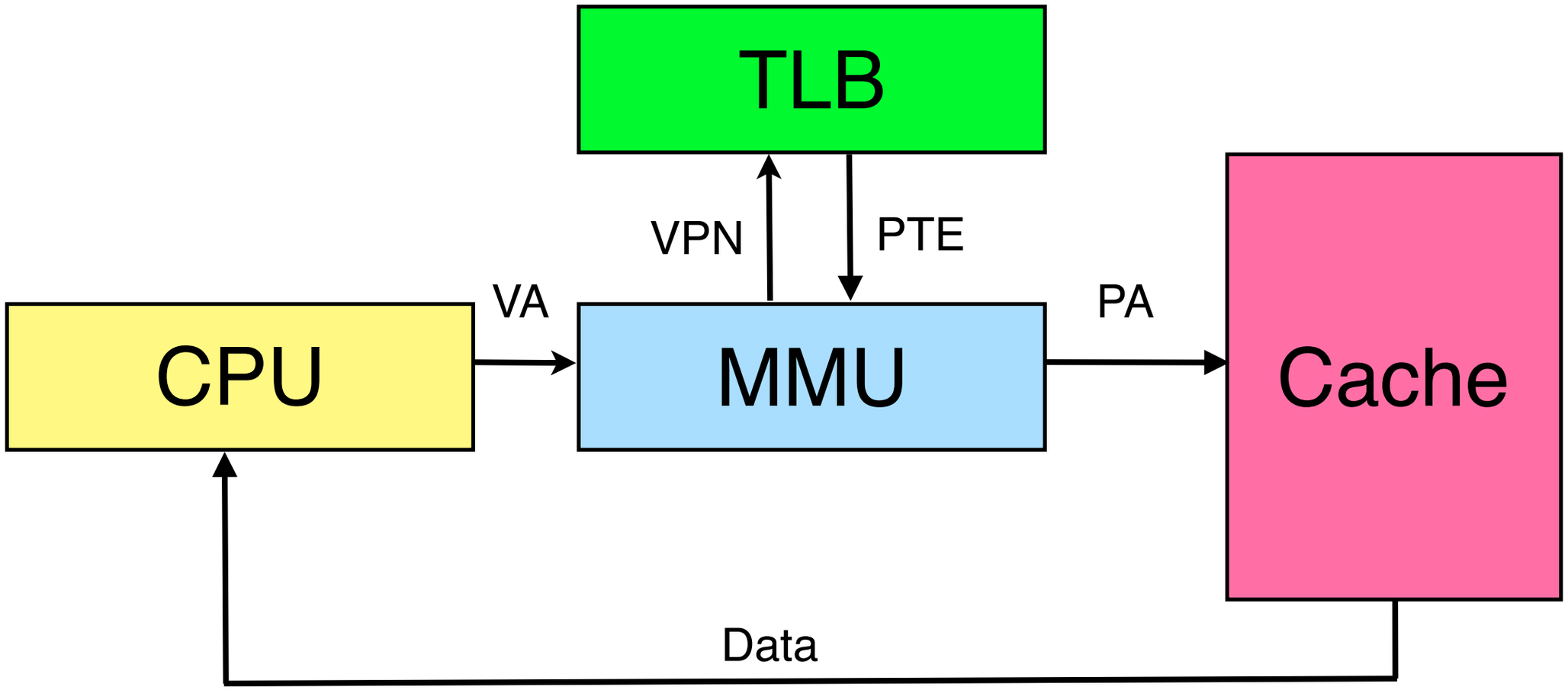
k-Level Page Table



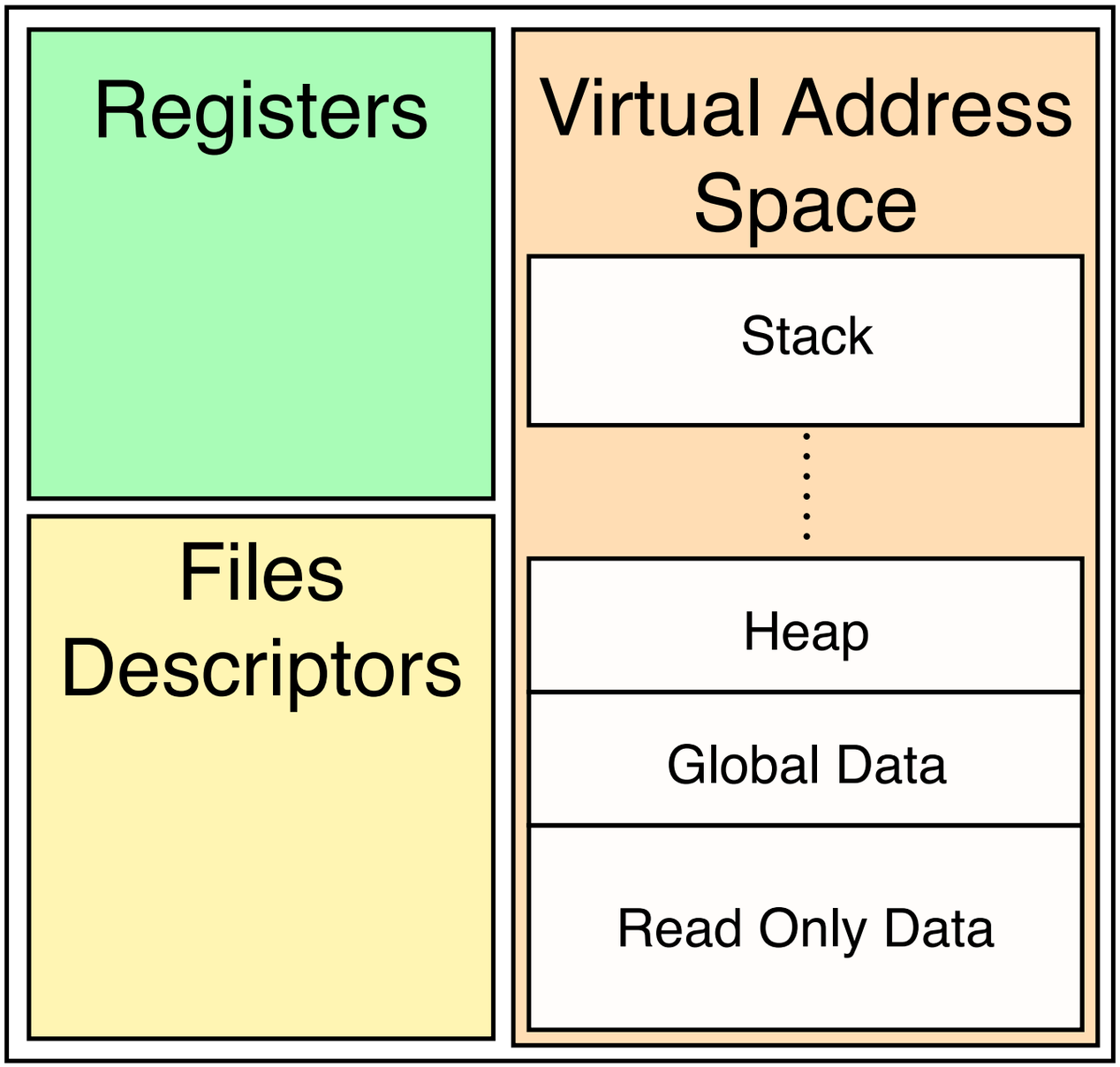
TLB


Virtual Address:
TLB Tag
Set
Page Offset
TLB
Valid
Tag
PPN
Valid
Tag
PPN
Physical Address:
Physical Page #
Page Offset

All Together



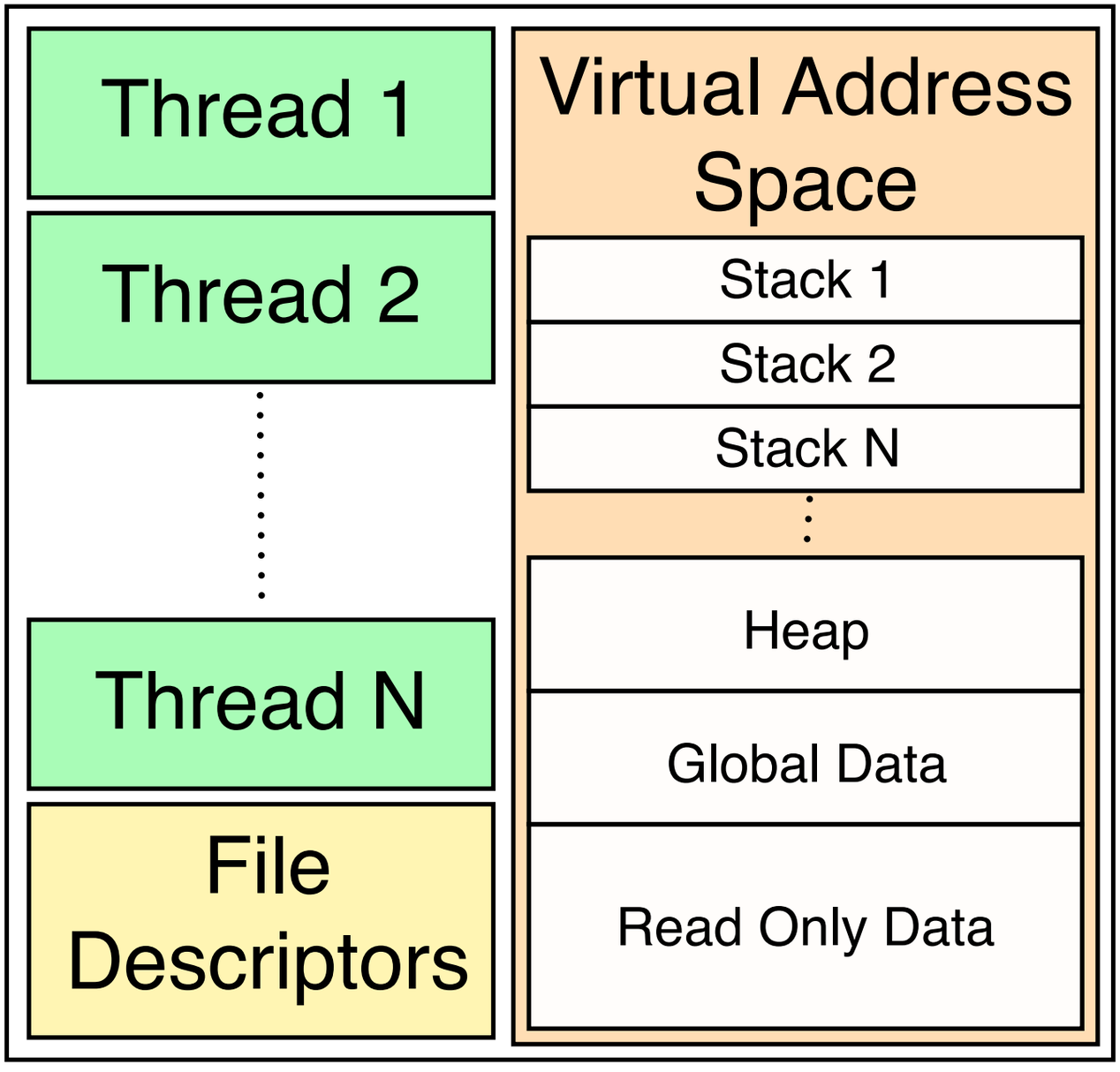
A Process


Registers
Files Descriptors
Virtual Address Space
Stack
Heap
Global Data
Read Only Data

Processes

- `fork` clones a process
- `exit` terminates current process
- `exec` “replaces” current process
- `wait` reaps a child process

Threading


Thread 1
Thread 2
Thread N
File Descriptors
Virtual Address Space
Stack 1
Stack 2
Stack N
Heap
Global Data
Read Only Data

Pthreads

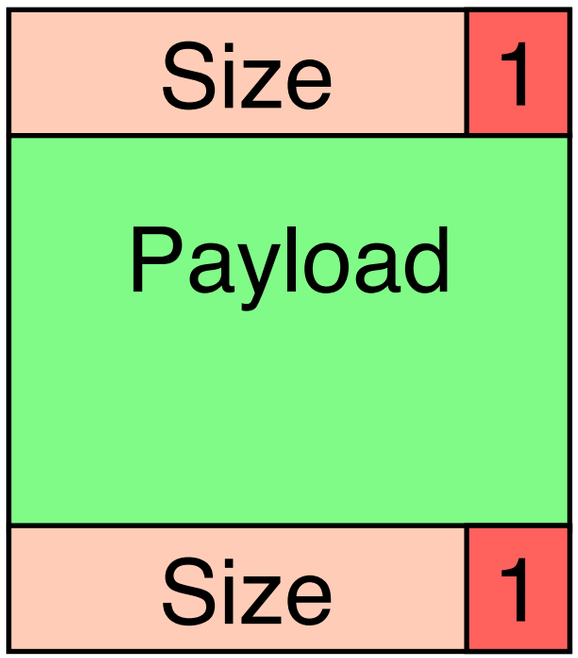
- `pthread_create` creates a new thread
- `pthread_exit` terminates current thread
- `pthread_join` reaps a thread
- Creating a thread is faster than creating a process

Semaphores

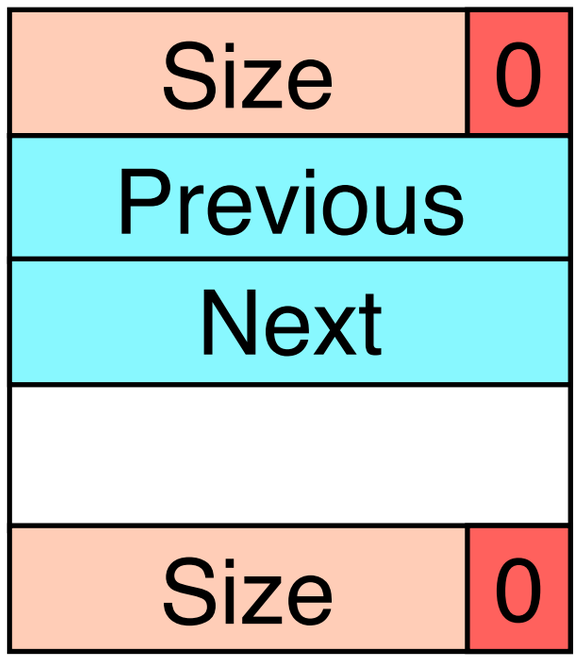
- Post: `[ ++sem; ]`
- Wait: `[ while(sem == 0); --sem; ]`
- Synchronize access to shared variables
- Watch out for deadlocks

Dynamic Memory

Allocated Block



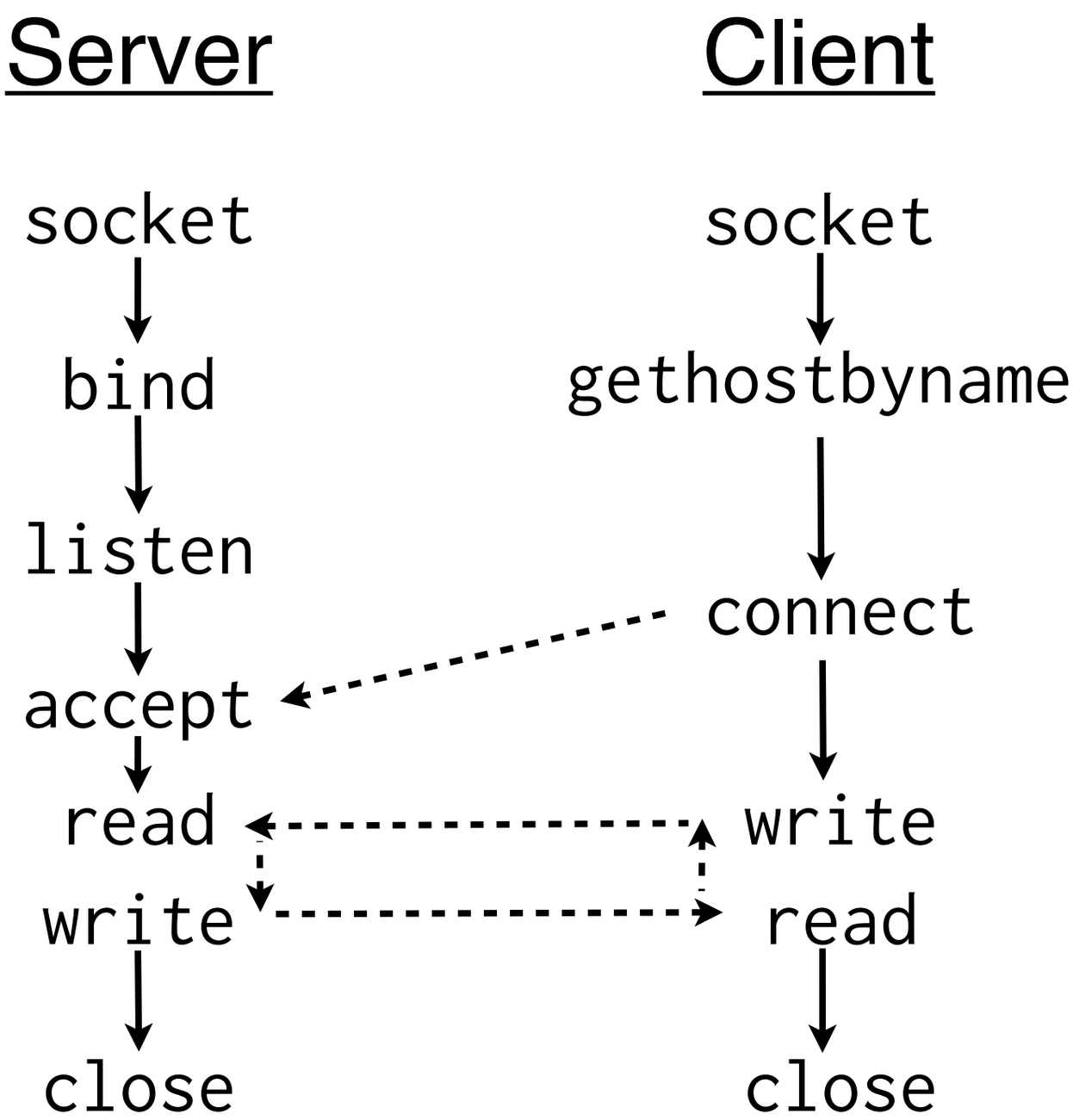
Freed Block



Dynamic Memory

- Implicit list with header and footer
  - Enables coalescing of adjacent blocks
- Explicit free list using payload
  - Placement policy? Best, first, next?
  - Insertion policy?
- Segregated lists, alignment, splitting, garbage collection...

TCP/IP Flow Control



Networking

- TCP sockets act like files
  - use read and write
  - reliable and ordered
- UDP sockets
  - unreliable and unordered

Web Services

- Proxies: You are experts at this now

15-213: Final Exam Review

Questions?