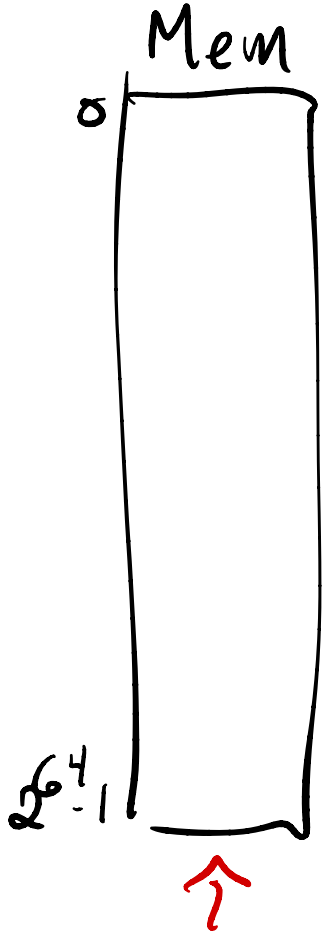
CS 208

F, 6 March 2026

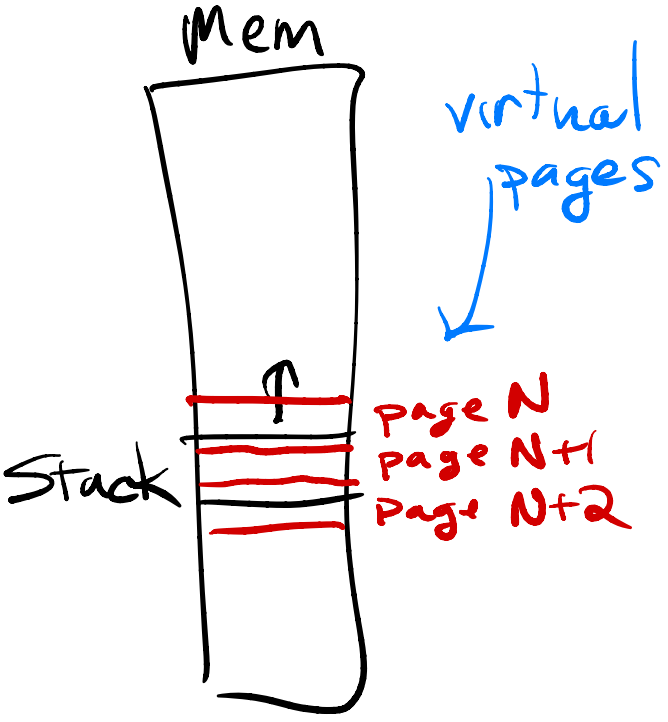


every process gets its own address space

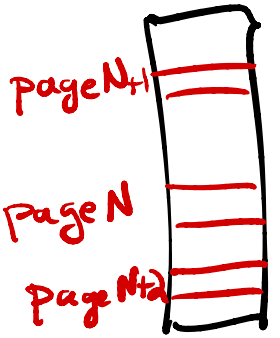
$2^{64} \sim 16$ exabytes

kilo	$\sim 10^3$	$\sim 2^{10}$	$\sim$ kibi
mega	10^6	$\sim 2^{20}$	$\sim$ mebi
giga	10^9	$\sim 2^{30}$	$\sim$ gibi
tera	10^{12}	$\sim 2^{40}$	
peta	10^{15}	$\sim 2^{50}$	
exa	10^{18}	$\sim 2^{60}$	

obvious fiction

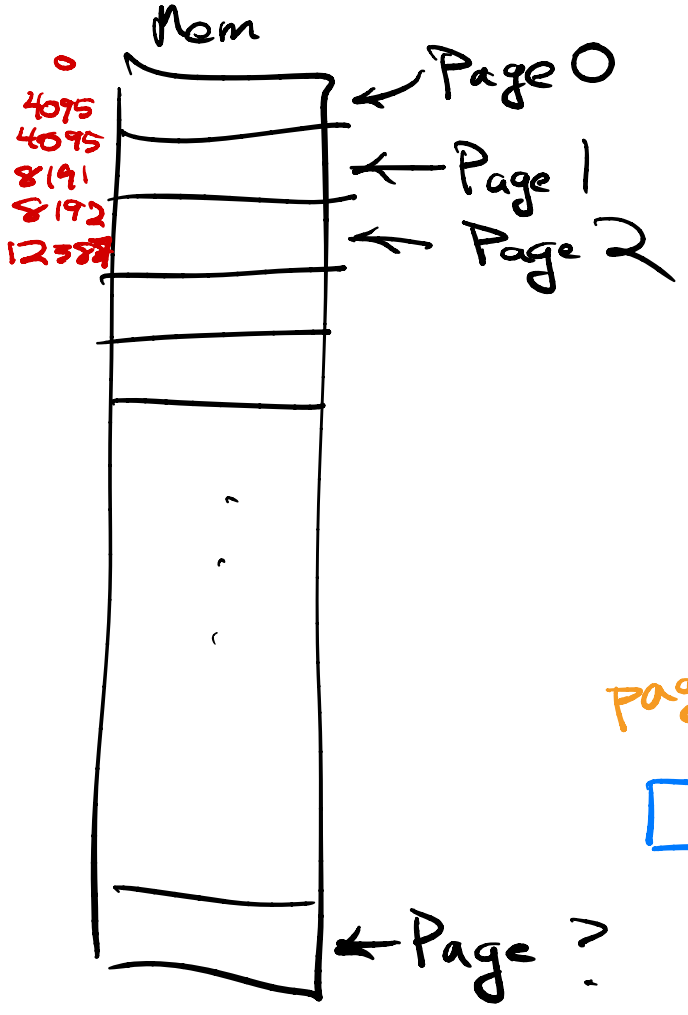


process PID = 20



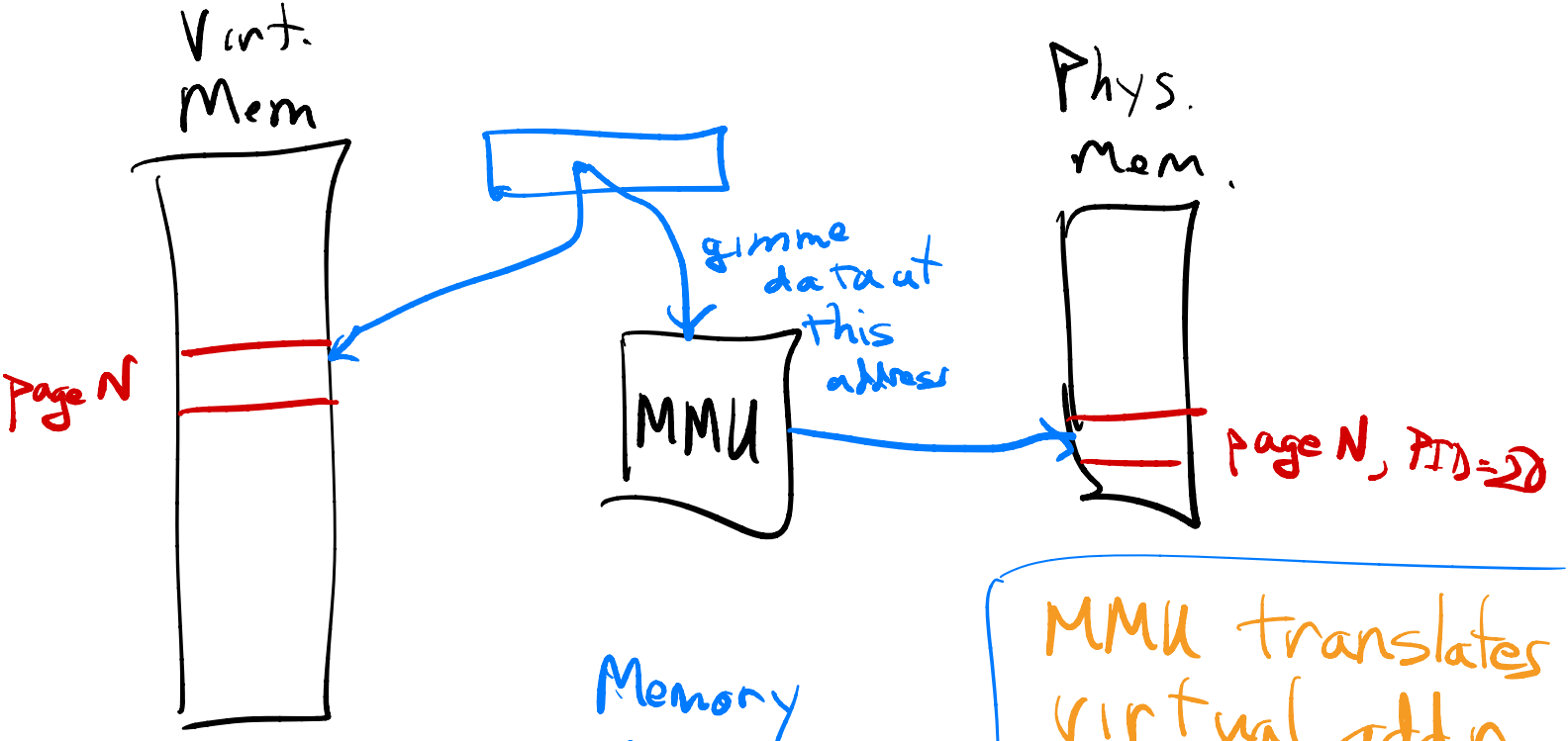
physical mem.

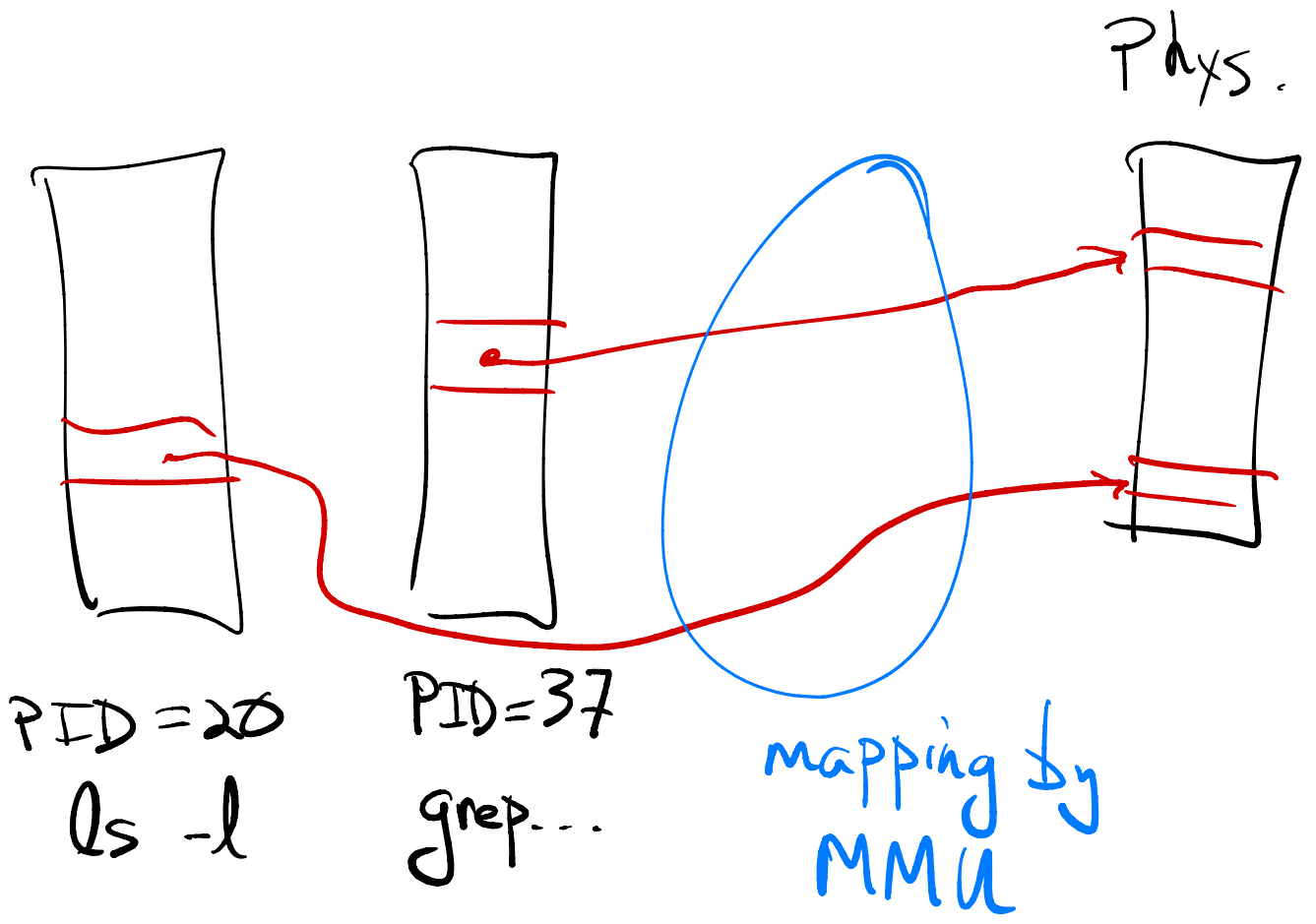
page frames
or
physical pages



Typical page size

$2^{12} = 4096$ bytes


Virt. Mem
Page N
gimme data at this address
MMU
Phys. mem.
Page N, PID=20
PID = 20
Memory Management Unit.
MMU translates virtual addr into physical address


Phys.
PID=20
ls -l
PID=37
grep...
mapping by
MMU

Want: map (PID, page#) → (phys. page #)

PID	V. Page#	Ph. Page #
26	3000	2200
36		
20		

Too big

Alternatives to full mapping table

- Hierarchy of tables
  (Works because "ls -l" doesn't need many pages)
- Hash tables

.
.
.

Why bother w/ this complexity?

- Want code to run on any machine w/ right instruction set (x86-64 or ARM64 etc.)
- Makes compiling way easier
- Can share pages read-only between processes
- My process can't read your process's memory

printf
printf
printf
PID 20
PID 38