What is wait up to? wait(6, status)

1. Puts calling process to sleep until one of its children terminates
2. When child terminates, child's exit status gets put here

+ data structures representing the child get deallocated
parent:

fork
if (parent)
exit
else child does its thing
exits

if (child becomes a zombie)
JEFF GUESSING

RAM responds in hundreds of nanoseconds ($10^{-9}$)

```
mov 0x8(%rsp), %rbx
```

RAM < registers

Registers respond in single-digit nano-seconds

SSD — maybe 10x slower than RAM

Hard disk — 10,000 times slower

— Slowwww, Cheap

2nd fastest
2nd most expensive

Fastest
Most expensive per bit
mov 0x8(%rsp), %rbx

RAM

Regs

small fast expensive

big more cheap
mov 0x8(%rip), %rdx
mov 0x8(%rip), %rax

medium fast
digger
a little cheaper
child in race.c

movl $0x2, %edi

If child stops running here, the parent gets to run before the child.

cmp $0x2, %edi

je

If child

before the child
Compiler thinks your process has this much virtual memory:

$$2^{34} \text{ GB}$$

Physical memory:

8 GB
The diagram illustrates the interactions between the memory management unit (MMU), the CPU, and physical memory. The diagram includes the following notes:

- The CPU is processing a move operation: `mov 0x08(%rsp), %rbx`.
- The address `0x5555208` is referenced.
- Process #73 wants to access the memory address `0x5555208`.
- The MMU is managing memory and requests a block.
- The CPU issues a request: "Gimme the block #X."
\[ 2^{10} \sim 10^3 \text{ (thousand)} \]
\[ 2^{20} \sim 10^6 \text{ (million)} \]
\[ 2^{30} \sim 10^7 \text{ (billion)} \]
\[ 2^{60} \sim 10^{18} \text{ (quintillion)} \]