Memory Hierarchy

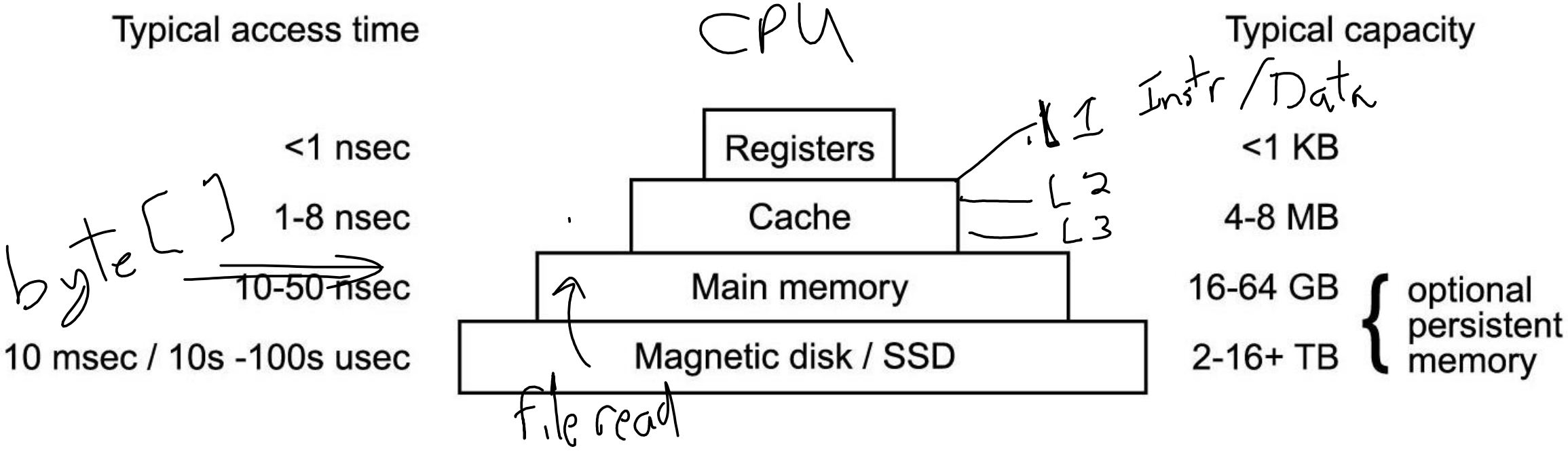


Figure 1.9 A typical memory hierarchy. The numbers are very rough approximations.

Multi-core CPU

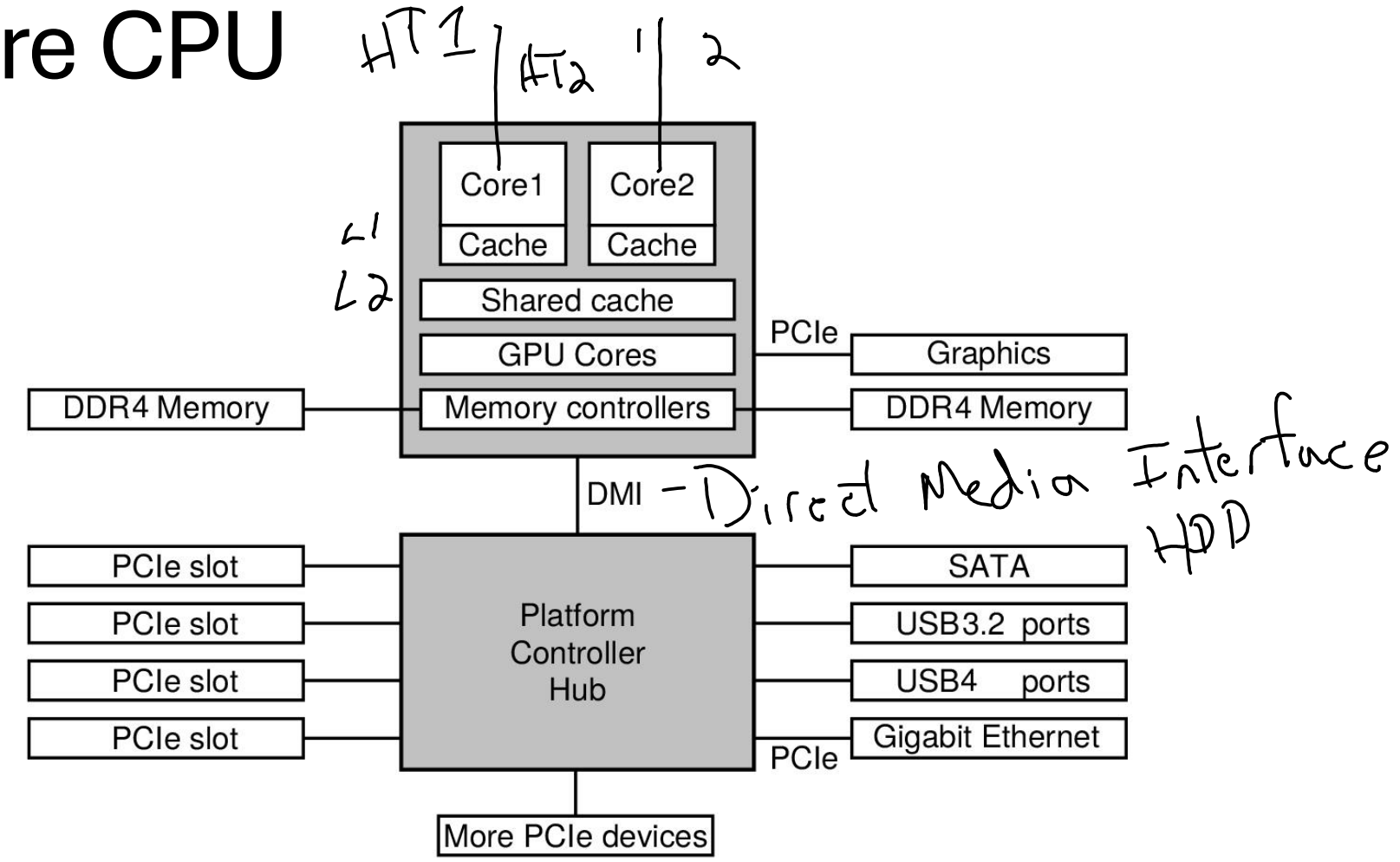


Figure 1.12 The structure of a large x86 system: many buses (e.g., cache, memory, PCIe, USB, SATA, and DMI)

CPU Operation

PC: program counter (register)

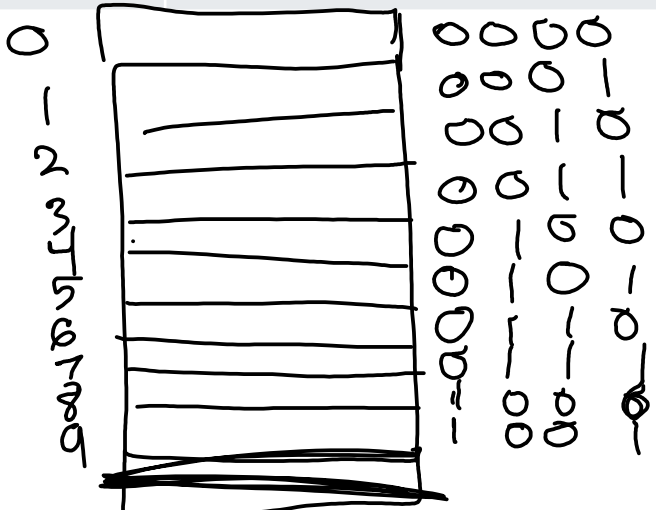
1. Fetch: get the next instruction from memory.
2. Decode: determine the operands.
3. Execute: run the instruction.
4. Write back: store the solution.

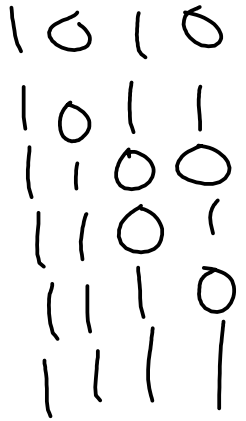
register

```
li      $r2,    12              # $r2 = 12
lw      $r3,    x               # $r3 = x
add     $r1,    $r2,    $r3     # $r1 = $r2 + $r3
```

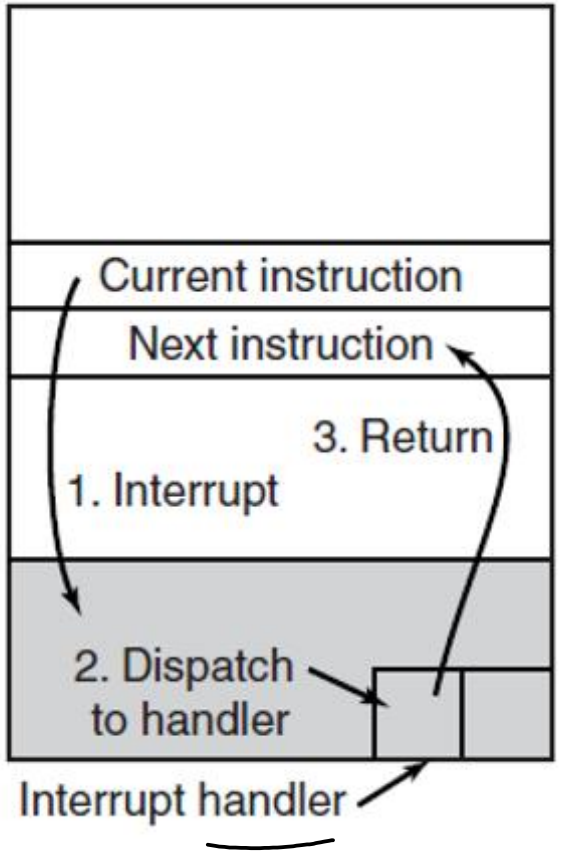
Units

Prefix	Base 10	Value	Base 2	Value
Kilo	10^3	1,000	2^{10}	1,024
Mega	10^6	1,000,000	2^{20}	1,048,576
Giga	10^9	1,000,000,000	2^{30}	1,073,741,824
Tera	10^{12}	1,000,000,000,000	2^{40}	1,099,511,627,776
Peta	10^{15}	1,000,000,000,000,000	2^{50}	1,125,899,906,842,624
Exa	10^{18}	1,000,000,000,000,000,000	2^{60}	1,152,921,504,606,846,976





Interrupts (I/O)



O.S. Kernel

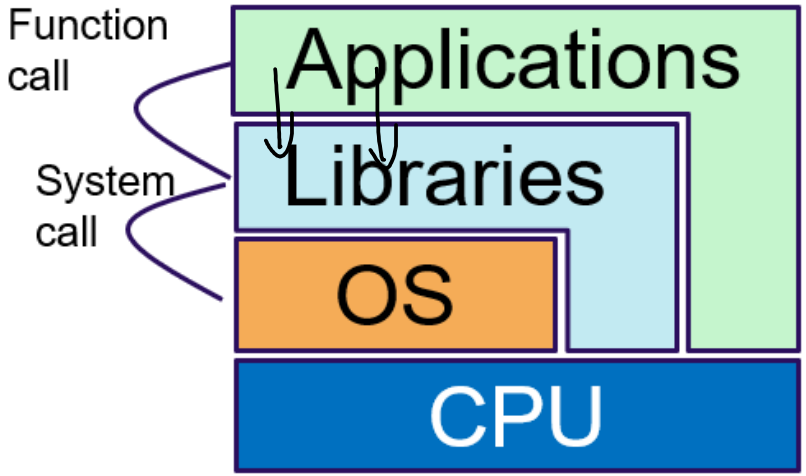
How interrupts work

in.nextInt()
- blocks
- I/O finishes
- runs your program

Key strokes

Figure 1.11 (b) Interrupt processing involves taking the interrupt, running the interrupt handler, and returning to the user program.

System Calls



System Calls

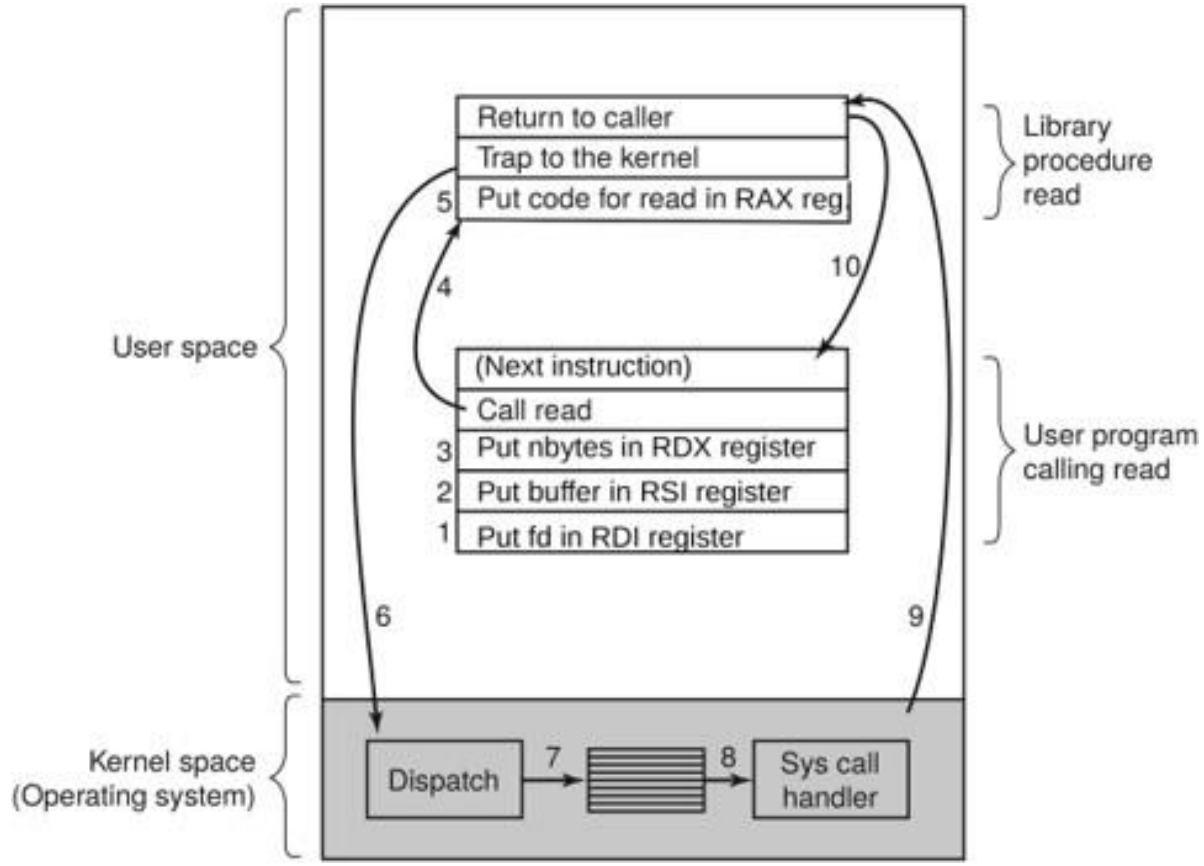


Figure 1.17 The 10 steps in making the system call read(fd, buffer, nbytes).

Command Line Interface

- cat, more, less, grep, wc, ls, ps
- Getting help: apropos, man, -h, --help
- Redirects and pipes:
  - Overwrite: >, 2>, 2>&1, &>
  - Append: >>
  - Input: <
  - Pipe: |