CSCI 2410 Systems Programming

Fall 2026

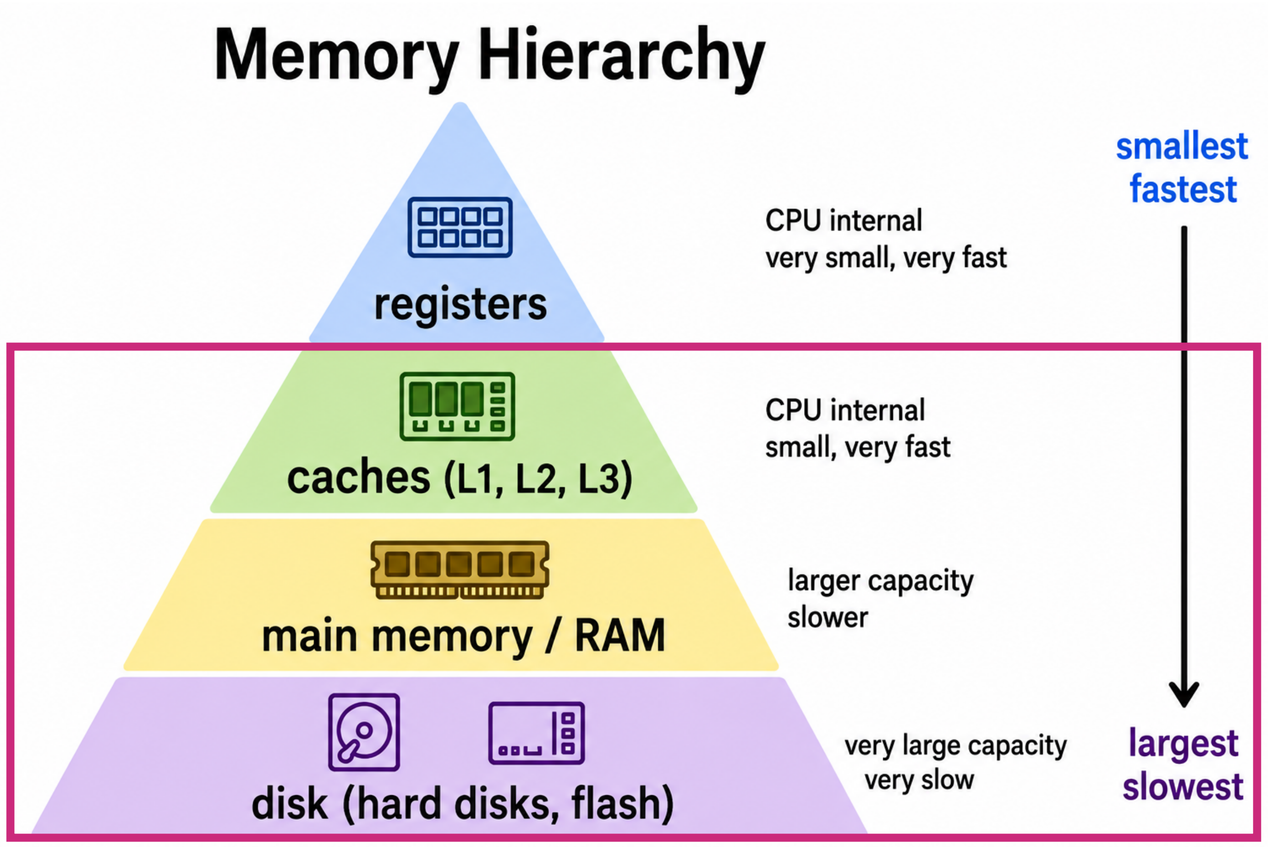
Instructor: Jie Zhou

Department of Computer Science

George Washington University

Slides materials are mostly credited to Sibin Mohan and Gabriel Parmer at GWU.

Memory Hierarchy



Why do we need so many layers?

- Registers are expensive!
- So are caches.
- Also, registers and caches are volatile.
- Speed vs. size: The faster the memory, the more expensive.

Registers

- Fastest form of memory
- Part of CPUs; what CPU uses for computing
- Each CPU has a set of registers.
- e.g., `a = b + 3;`
  - b is loaded into a register
  - a is computed and the result stored in another register or the same register used by b, if b is not used anymore.

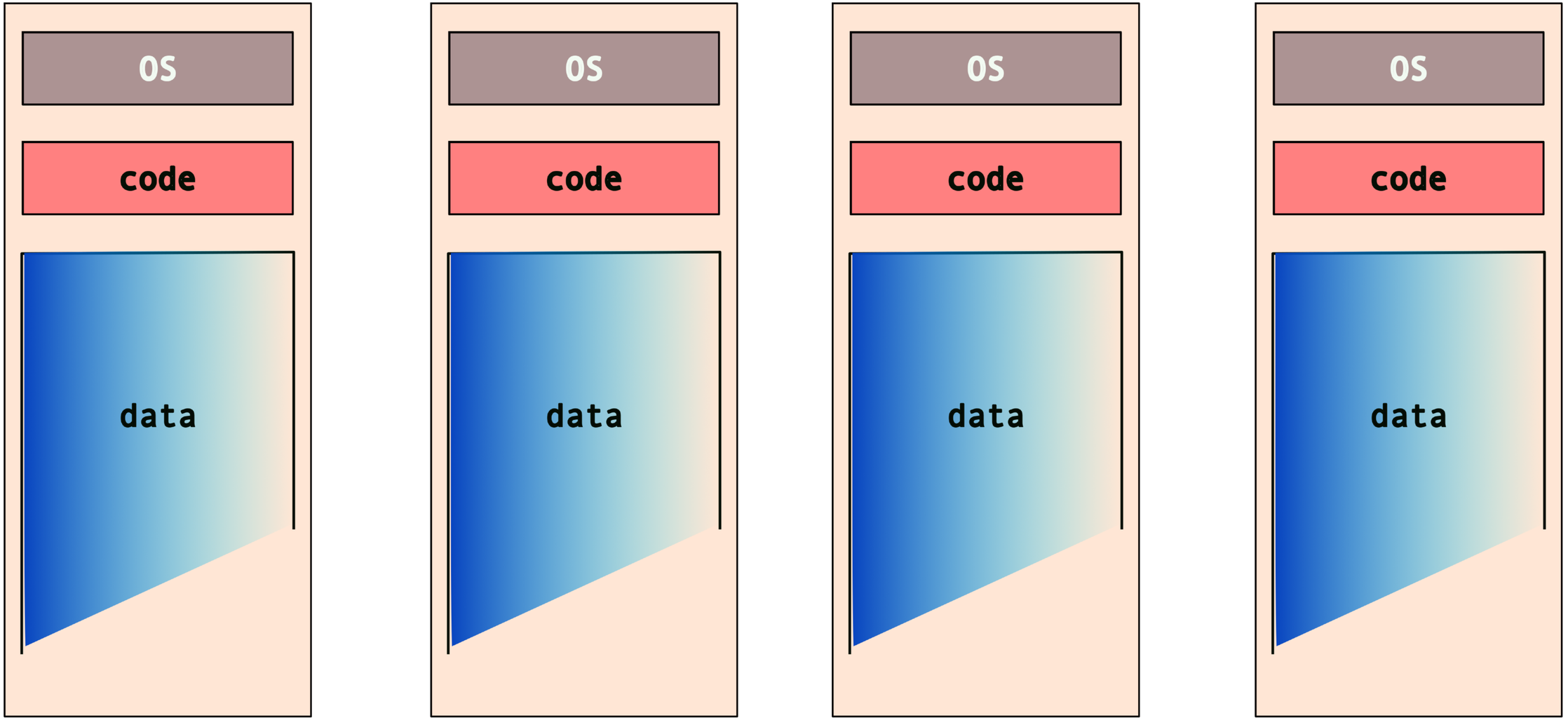
Caches

- Fast, on-chip, volatile memory
- (Largely) Transparent to programs and programmers
- Used by CPU to improve performance
- Typical sizes, hundreds of KBs to hundreds of MBs

Main Memory

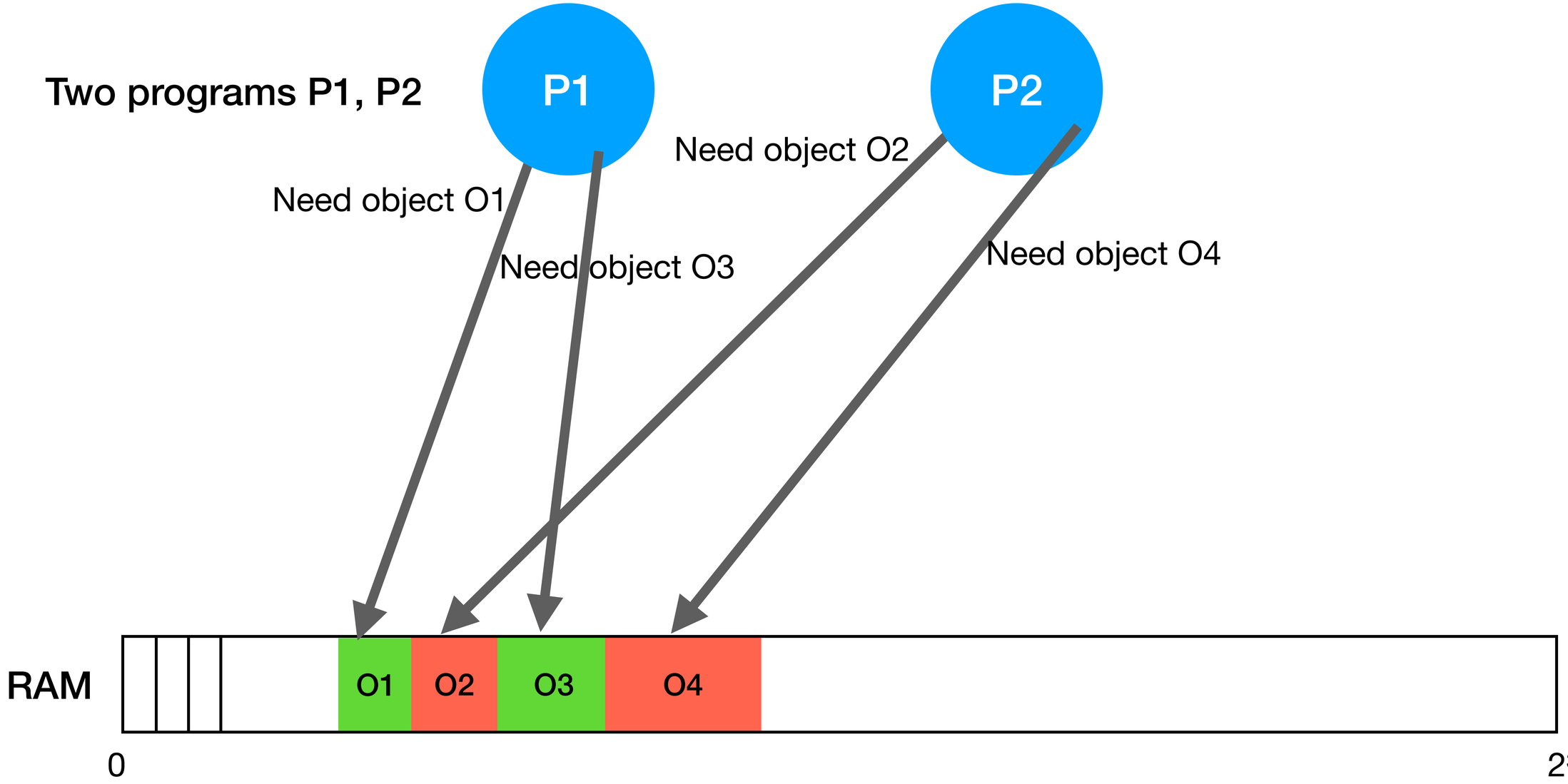
- Random Access Memory (RAM), also known as DRAM
- Typical sizes, GB range
- Limited by “width” of CPU
  - e.g., 32-bit CPU —> 4 GB of memory (generally)
- Memory is divided into many units, and each unit has its *address*.
- In mainstream systems, memory is *byte-addressable.*

Virtual Memory

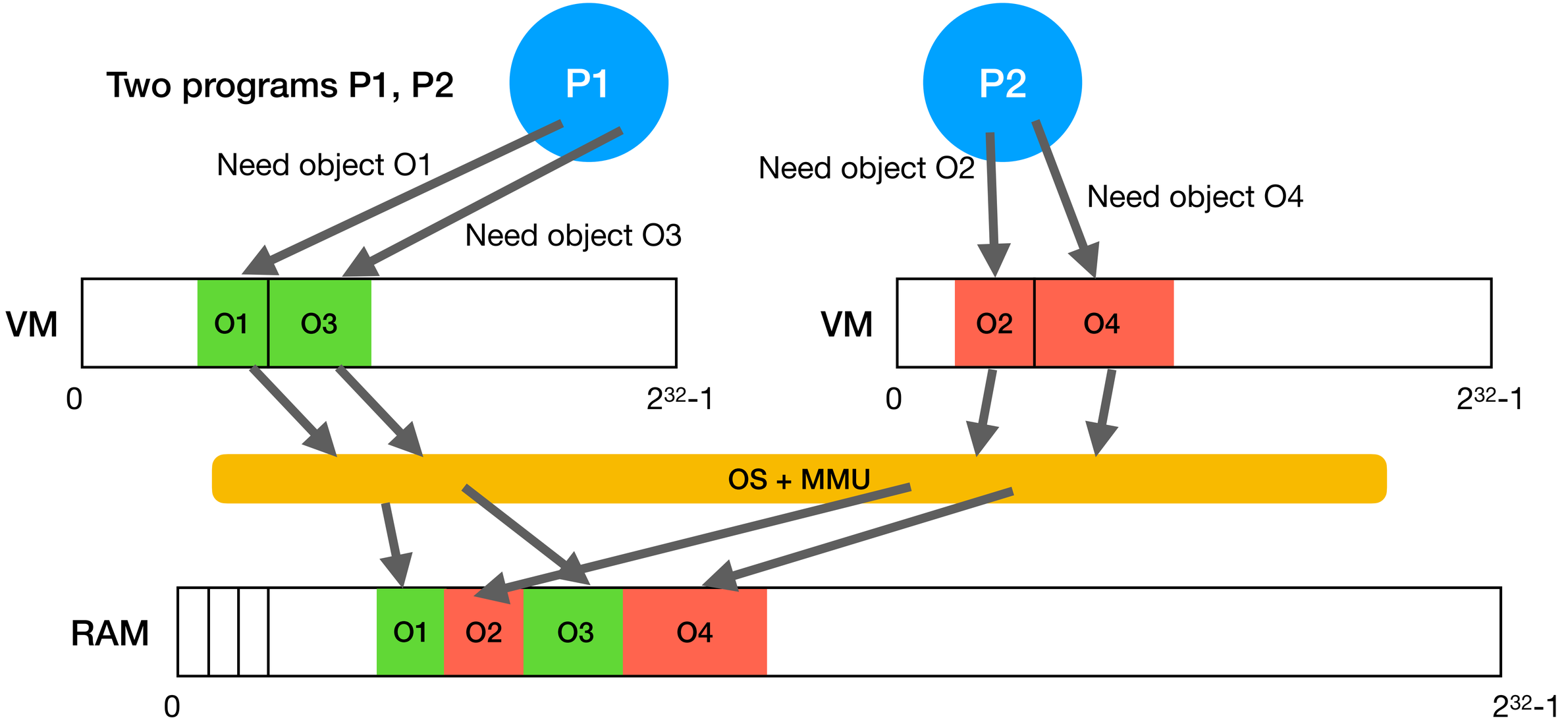


Or, at least the programs “think” so!

Using Main Memory Directly

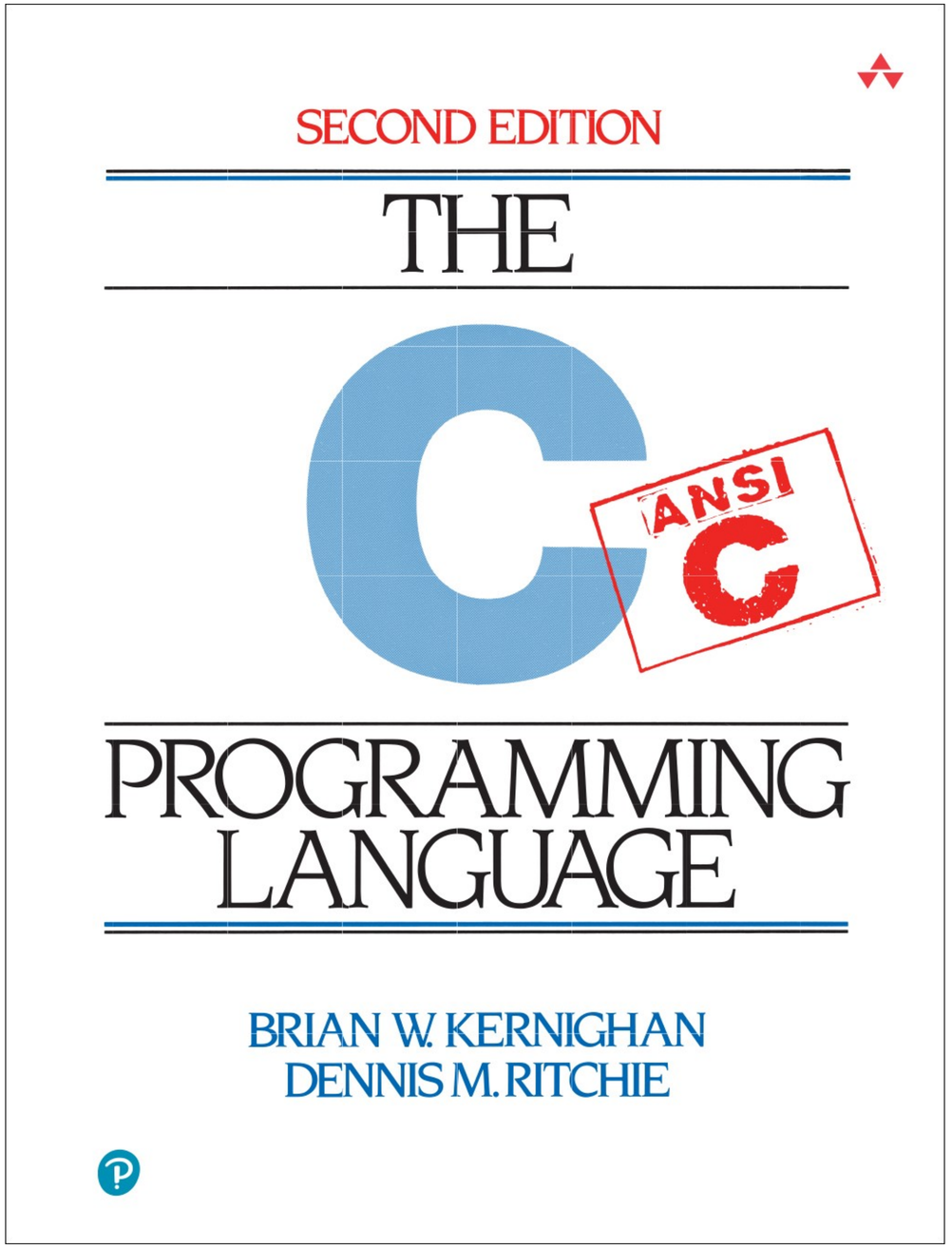


Virtual Memory



Virtual Memory

- Makes each program “believe” it as the *entire* memory it itself!
- Combination of OS + hardware
- Programmers need **not** worry about other programs.
- Easier for OS to manage all programs.
- In this course—and in most of your career as a systems programmer—you will mainly deal with virtual memory.


SECOND EDITION
THE
C
ANSI
C
PROGRAMMING
LANGUAGE
BRIAN W. KERNIGHAN
DENNIS M. RITCHIE

The C Programming Language

- Invented in 1972 by Dennis Ritchie
- Building blocks for our modern digital world.
  - OS, web servers, databases, systems libraries, …

What are the fundamental elements of a programming language?

What Makes Up a Programming Language?

- Data types
  - int, char, boolean, etc.
- Variables
- Operators and expressions
  - arithmetic, assignment, comparison, etc.
- Functions
  - parameters and return
- Control flow
  - conditional/unconditional branch, loop, call/return

You already know C—to a great extent.

C Constructs (Familiar)

Construct	Syntax
conditionals	`if` and `else`
loops	`while {}`, `do {} while`, `for`
basic types	`int`, `float`, `double`, `char`
aggregate	arrays, struct(ure),…
functions	`ret_val fn_name (arg_list) {}`

What is Different From Java?

- No object-oriented programming. C is *procedural*.
- No function overloading
- No classes
- But C has `struct` instead, which is similar but also different.
- Compiled not interpreted
- No public/private
  - By default, everything is public.
- Pointers!!
  - Source of superpower but also root causes for disastrous bugs.

Compound Types

- Consist of basic types of data
- Continuous memory: each data is laid out next to each other.

Type	Usage
`struct`	related variables (like class, but no functions)
`union`	same, but *shared* memory
array	continuous sequence of items of the same

`enum`

- Enumeration, essentially named integer constants
- Much less rich than Java's enum

No boolean

- Mimic it using `int` values
  - `0 -> false`
  - any non-zero value `-> true`

Header Files (.h)

- Declares and define elements shared across C source files.
  - e.g., functions, data types, constants, etc.
- Included in C source files and other header files
- Libraries usually only expose header files.
  - both standard C lib and 3rd-party libs

Including Header Files

- Tells the current src what it can use defined somewhere else
- Similar to `import` in Java
  - but must *explicitly* include everything needed

Method	What is included
`#include <stdio.h>`	Common libraries
`#include “my_header.h”`	User header files

Including Header Files

- Tells the current src what it can use defined somewhere else
- Similar to `import` in Java
  - but must *explicitly* include everything needed

Method	Location of Header Files
`#include <…>`	Installed in the system, e.g., `/usr/include/*` in Linux; can also be anywhere and specified during compilation, e.g., “`-I/my_home/libs/lib_crypto/include`”
`#include “…”`	path in the current project

- We can ask the compiler to list the default search directories.
  - `gcc -E -v`

Some Common C Libraries

Library	Function
`stdio.h`	standard input/output, e.g. `printf()`
`stdlib.h`	common utilities, e.g. memory allocations
`string.h`	string manipulation, `strcpy()`
`math.h`	math utility functions, `sqrt()`
`unistd.h`	POSIX system interfaces, e.g. read/write

Unix Manual (man) Pages

- Compilation of Unix (and C) knowledge
- In terminal, type “man <topic>”, e.g., “man printf”
- Organized into sections

#	Contents
1	genera commands, e.g., `ls`
2	system calls, e.g., `read/write`
3	library functions, e.g., `printf`
4	special files, e.g., `/dev/null`
5	file formats, e.g., `passwd`
6	games & screensavers
7	miscellaneous, e.g., `socket`
8	system administration, e.g., `useradd`

Unix Manual (man) Pages

- Compilation of Unix (and C) knowledge
- In terminal, type “man <topic>”, e.g., “man printf”
- Organized into sections

#	Contents
1	**genera commands**, e.g., `ls`
2	**system calls**, e.g., `read/write`
3	**library functions**, e.g., `printf`
4	special files, e.g., `/dev/null`
5	file formats, e.g., `passwd`
6	games & screensavers
7	miscellaneous, e.g., `socket`
8	system administration, e.g., `useradd`

Example

man 3 printf

```
PRINTF(3)                                                  Linux Programmer's Manual

NAME
       printf, fprintf, dprintf, sprintf, snprintf, vprintf, vfprintf, vdprintf, vsprintf, vsnprintf - formatted output conversion

SYNOPSIS
       #include <stdio.h>

       int printf(const char *format, ...);
       int fprintf(FILE *stream, const char *format, ...);
       int dprintf(int fd, const char *format, ...);
       int sprintf(char *str, const char *format, ...);
       int snprintf(char *str, size_t size, const char *format, ...);

       #include <stdarg.h>

       int vprintf(const char *format, va_list ap);
       int vfprintf(FILE *stream, const char *format, va_list ap);
       int vdprintf(int fd, const char *format, va_list ap);
       int vsprintf(char *str, const char *format, va_list ap);
       int vsnprintf(char *str, size_t size, const char *format, va_list ap);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       snprintf(), vsnprintf():
           _XOPEN_SOURCE >= 500 || _ISOC99_SOURCE ||
```

Basic Types

Type	Description/Size in C	Size in Java
`char`	smallest type, typically one byte[1]	one byte
`short int / short`	typically two bytes[2]	two bytes
`int`	typically four bytes[3]	four bytes
`long int / long`	larger int, typically four bytes[4]	eight bytes
`float`	floating point, typically four bytes	four bytes
`double`	double precision, typically eight bytes	eight bytes

1. The C standard requires at least 8 bits.
2. short: at least $-(2^{15} - 1)$—$(2^{15} - 1)$, so at least 16 bits or 2 bytes
3. int: Also $-(2^{15} - 1)$—$(2^{15} - 1)$, must be larger or equal to short; typically 4 bytes
4. Floating point numbers are not defined by the C standard.

Code Samples

```c
#include <stdio.h>

int main()
{
    return 0;
}
```

- Every C program requires a `main()` function, where the execution starts and ends.
- Here `main` takes no arguments, but it can also take arguments.

Code Samples

```c
#include <stdio.h>

int main()
{
    char c = 'a';
    int i;
    float f;
    double d;

    i = 100;
    f = 1.0;
    d = 12398723897.2332;

    printf( "Memory sizes of variables...\n\n \
            size of char: %lu \
            size of int: %lu \
            size of float: %lu \
            size of double: %lu\n",
            sizeof(c), sizeof(int), sizeof(f), sizeof(d) );

    printf( "\n" ); // adding an extra line for nice printing at the end
    return 0;
}
```

Compound Types

- Consist of basic types of data
- Continuous memory: each data is laid out next to each other.

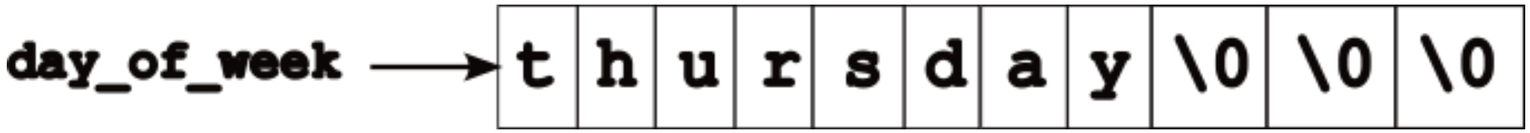
Type	Description	Size
`struct`	collection of different values	sum of all fields + padding
`union`	one of a set of values	size of the largest field
array	continuous sequence of items of the same type	sum of all items

`enum`

- Enumeration, essentially named integer constants
- Much less rich than Java's enum
- Size: typically size of `int`

Strings

- An array of characters that are null terminated, i.e., \0



Type Modifiers

- `unsigned`: variables that cannot be negative
- `signed`: signed variable. char, int, long all default to signed.
- `static`: this variable/function are only accessible inside its source file.
- `const`: an immutable value

Format Strings

- %d: int
- %ld: long int
- %u: unsigned int
- %c: char
- %x: unsigned int printed as hexadecimal
- %s: prints out a string
- %p: prints out a pointer value