Introduction to Computer Systems
Lecture 2 – A Tour of Computer Systems

2022 Spring, CSE3030

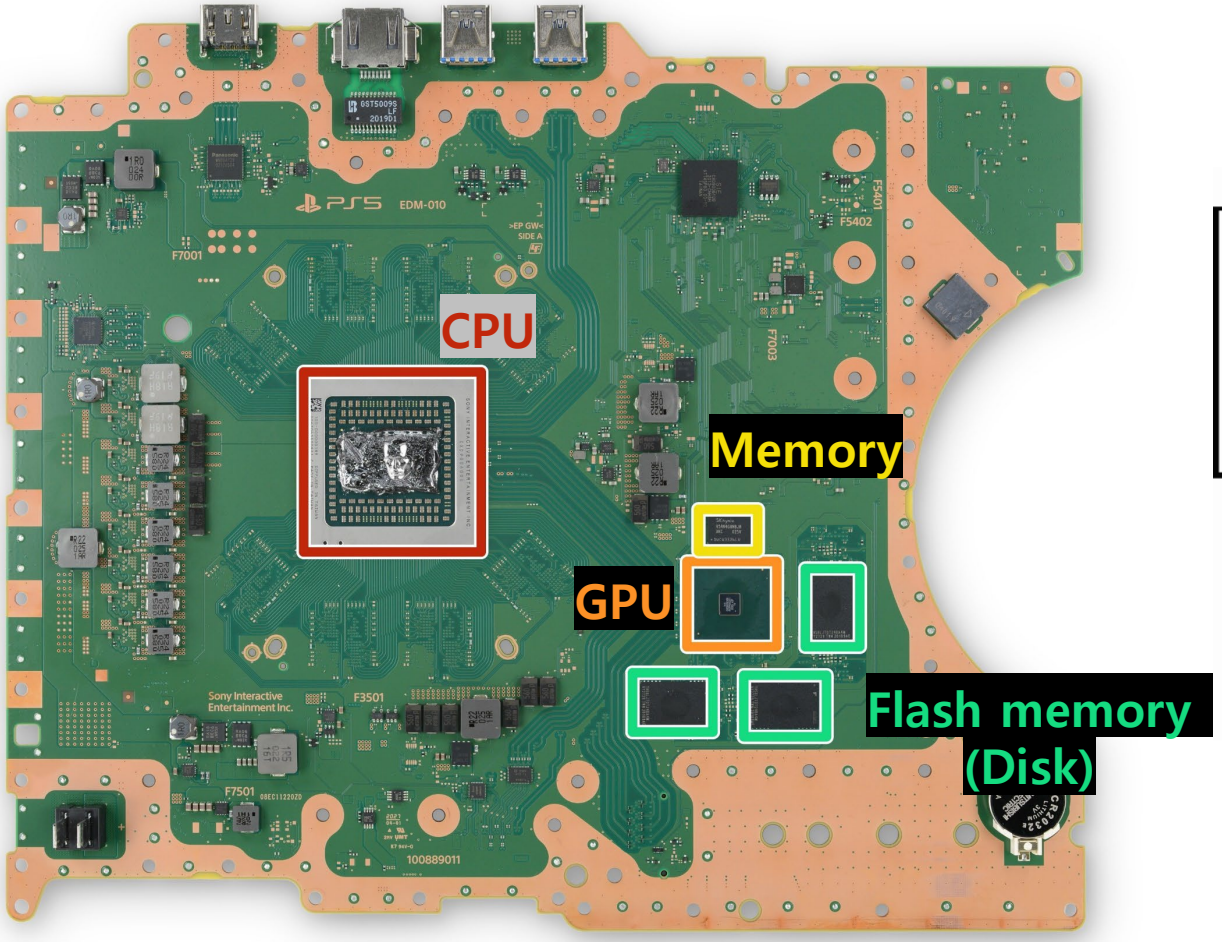
Sogang University

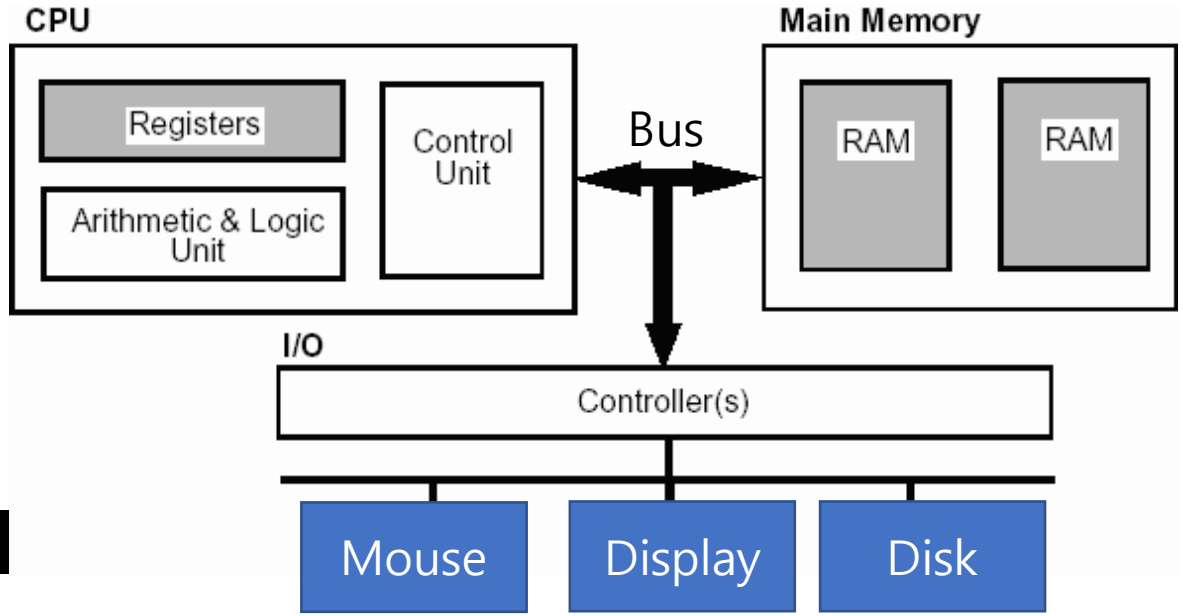
Computer Systems

- Computer systems are used everywhere!

Teardown – PlayStation 5

- Hardware - CPU, GPU, memory, disk and etc.

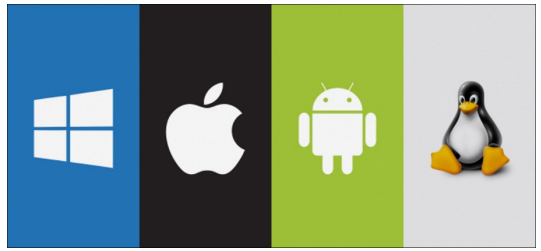




General hardware structure

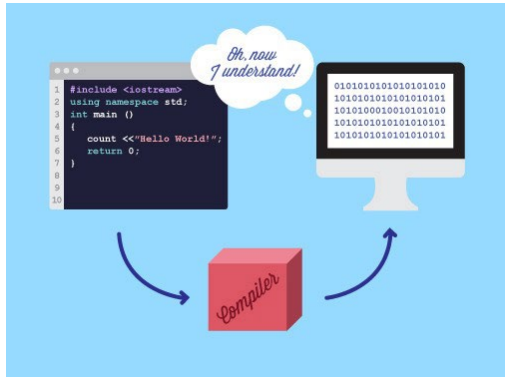
System Software

- A software that is used for functioning of computer hardware or manage to run application software.
  - Operating systems
  - Device drivers
  - Compiler, linker and debugger



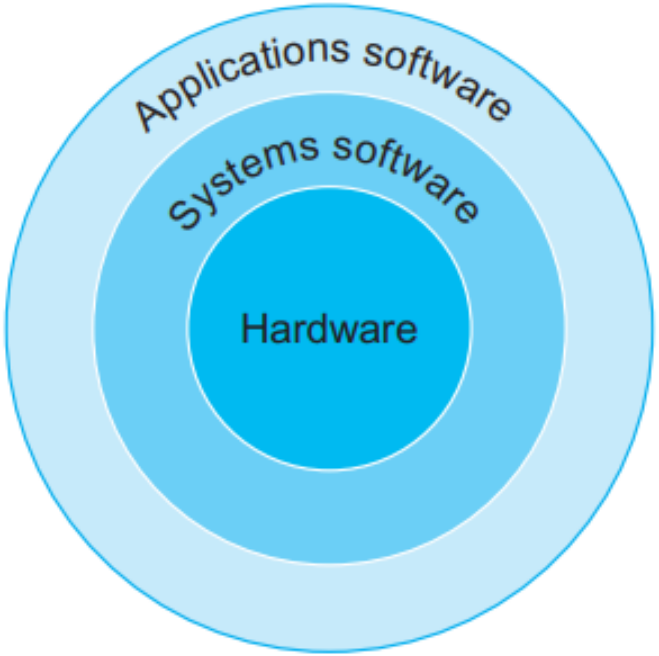
Operating systems

Device drivers



Compiler

System software



Interface between application software and hardware

Computer Systems

- Comprise of Hardware and systems software to run application programs
- Computer systems affect correctness and performance of programmer's programs.
- You will learn…
  - How to avoid strange numerical errors caused by number representation systems
  - How to optimize your C code
  - How procedure call is implemented and to avoid security holes
  - How to write your own Unix shell, own web server, etc.

Journey of Hello.c in Computer Systems

- The hello program

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf ("hello, world₩n");
6}
```

- The ASCII text representation of hello.c

```
#   i   n   c   l   u   d   e   <sp> <   s   t   d   i   o   .   h   >   \n \n i   n   t   <sp>
35 105 110 99 108 117 100 101 32    60 115 116 100 105 111 46 104 62 10 10 105 110 116 32

m   a   i   n   (   )   \n {   \n  <sp> <sp> <sp> <sp> p   r   i   n   t   f   (   "   h   e
109 97 105 110 40 41 10 123 10  32    32    32    32    112 114 105 110 116 102 40 34 104 101

l   l   o   ,   <sp> w   o   r   l   d   \   n   "   )   ;   \n }
108 108 111 44 32    119 111 114 108 100 92 110 34 41 59 10 125
```

- All information is represented as a bunch of bits
  - Disk files, programs, user data, data transferred across network
  - Different context – the same sequence of bytes
    - Integer, floating-point, character string, or machine instruction

ASCII

- American Standard Code for Information Interchange

dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char
0	0	000	NULL	32	20	040	space	64	40	100	@	96	60	140	`
1	1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB	41	29	051	)	73	49	111	I	105	69	151	i
10	a	012	LF	42	2a	052	*	74	4a	112	J	106	6a	152	j
11	b	013	VT	43	2b	053	+	75	4b	113	K	107	6b	153	k
12	c	014	FF	44	2c	054	,	76	4c	114	L	108	6c	154	l
13	d	015	CR	45	2d	055	-	77	4d	115	M	109	6d	155	m
14	e	016	SO	46	2e	056	.	78	4e	116	N	110	6e	156	n
15	f	017	SI	47	2f	057	/	79	4f	117	O	111	6f	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1a	032	SUB	58	3a	072	:	90	5a	132	Z	122	7a	172	z
27	1b	033	ESC	59	3b	073	;	91	5b	133	[	123	7b	173	{
28	1c	034	FS	60	3c	074	<	92	5c	134	\	124	7c	174	|
29	1d	035	GS	61	3d	075	=	93	5d	135	]	125	7d	175	}
30	1e	036	RS	62	3e	076	>	94	5e	136	^	126	7e	176	~
31	1f	037	US	63	3f	077	?	95	5f	137	_	127	7f	177	DEL

www.alpharithms.com

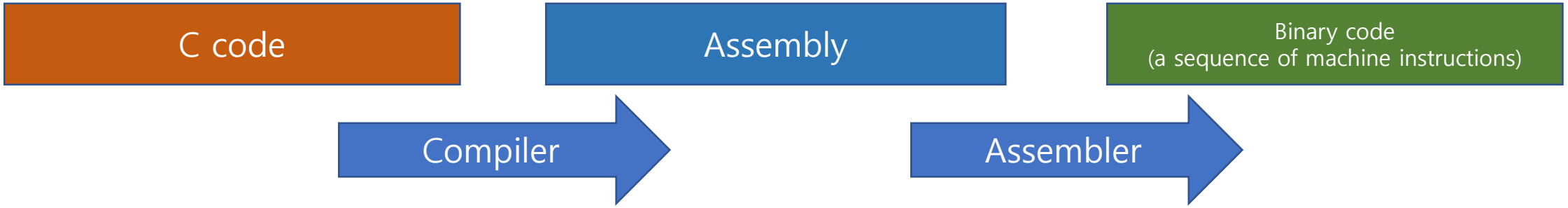
Compilation System

- Translate a high-level C program into the binary code that is read and operated by the processor.

```c
#include <stdio.h>

void main(){

        printf("hello world!\n");

}
```

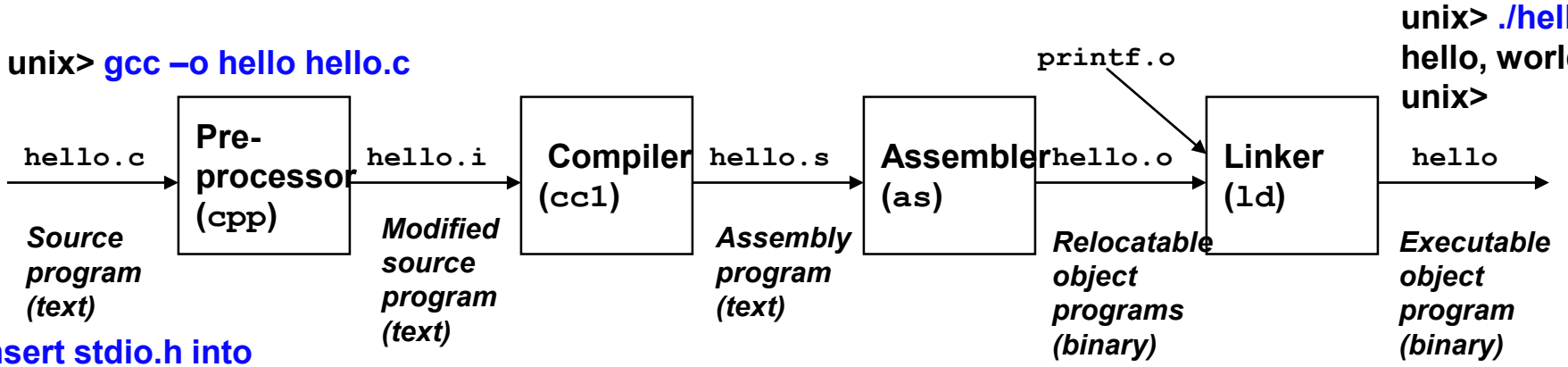
```
main:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        leaq    .LC0(%rip), %rdi
        call    puts@PLT
        nop
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
```

Memory address	Values (Heximal)	Corresponding Instructions
1149:	f3 0f 1e fa	endbr64
114d:	55	push %rbp
114e:	48 89 e5	mov %rsp,%rbp
1151:	48 8d 3d ac 0e 00 00	lea 0xeac(%rip),%rdi
1158:	e8 f3 fe ff ff	callq 1050 <puts@plt>
115d:	90	nop
115e:	5d	pop %rbp
115f:	c3	retq



Compilation Overview

[1] unix> gcc –o hello hello.c



unix> ./hello
hello, world
unix>

[2] *insert stdio.h into program text, hello.i

unix> gcc –E -o hello.i hello.c
unix> vi hello.i

[3] unix> gcc –S hello.c
unix> vi hello.s

[4] unix> as –o hello hello.s

[5] *merge hello.o with printf.o

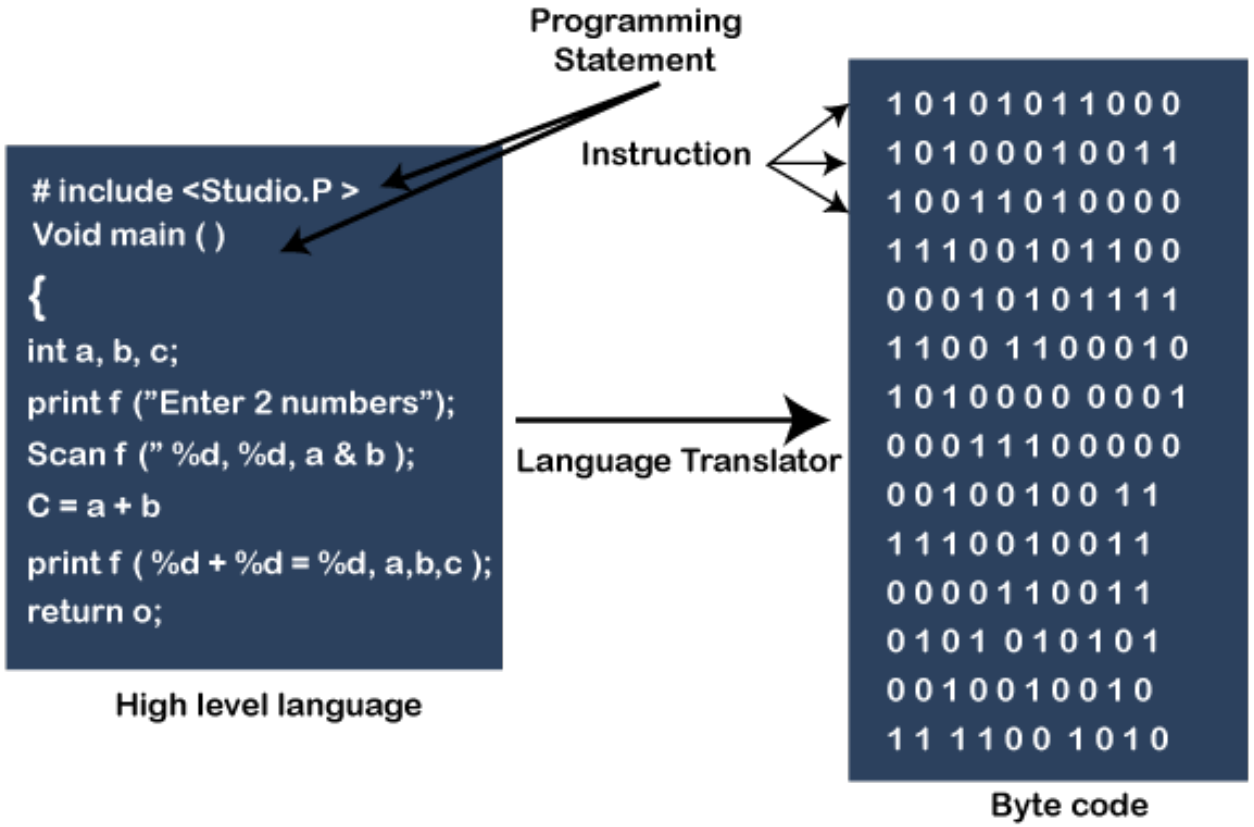
[6] try gdb

unix> gcc –g –o hello hello.c
unix> gdb hello
gdb > r
Starting program:
hello, world
gdb > disassemble main

```
Dump of assembler code for function main:
0x0000000000400498 <main+0>:    push   %rbp
0x0000000000400499 <main+1>:    mov    %rsp,%rbp
0x000000000040049c <main+4>:    mov    $0x400598,%edi
0x00000000004004a1 <main+9>:    callq  0x4003c0 <puts@plt>
0x00000000004004a6 <main+14>:   leaveq
0x00000000004004a7 <main+15>:   retq
End of assembler dump.
```

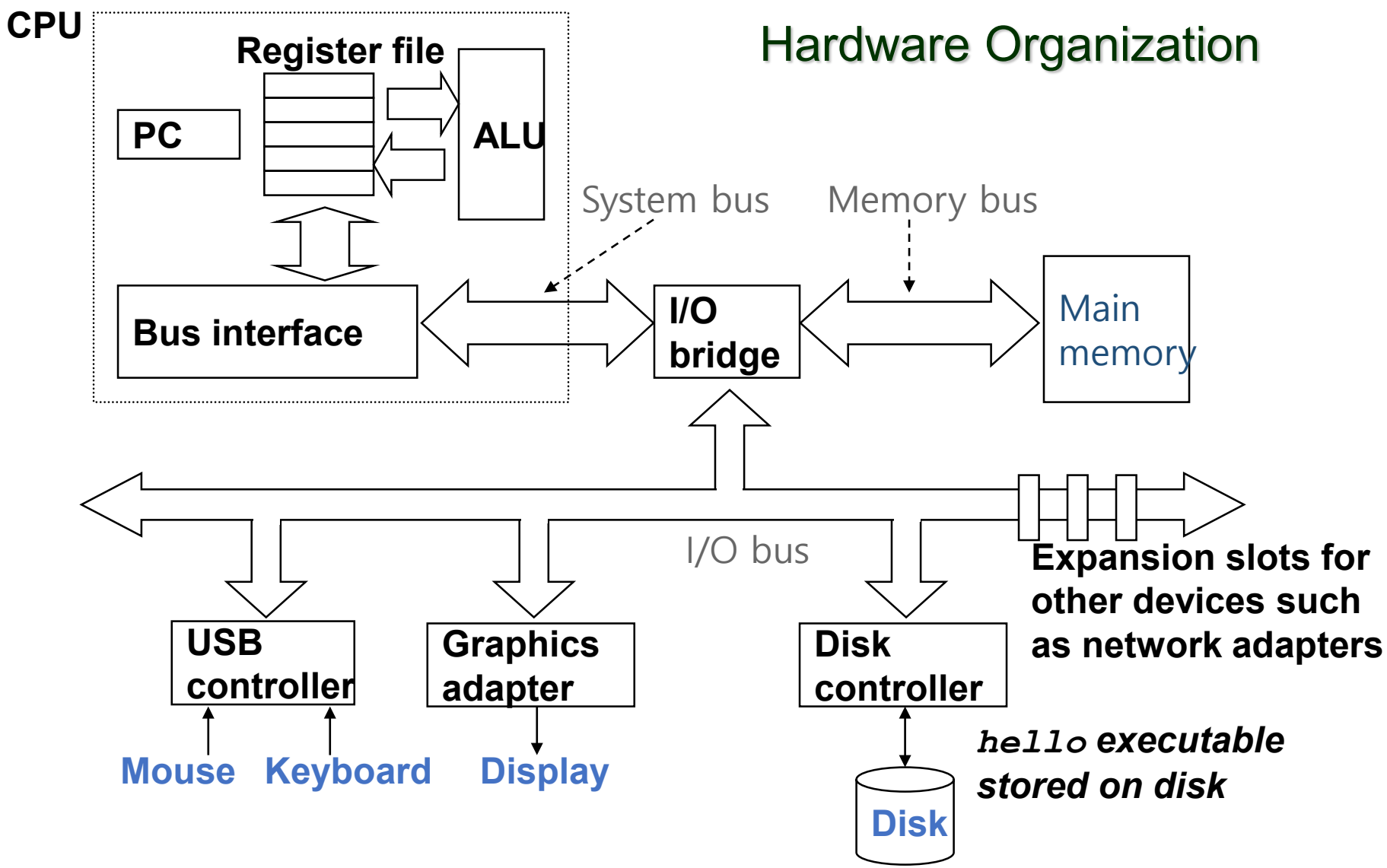
Executable Object Program

- A program that is ready to be loaded into memory and executed by the computer system.



https://www.tutorialandexample.com/machine-instructions

Hardware Organization of a System



Hardware Organization of a System

- **Bus** – a collection of electrical wires that carry bytes of information back and forth between the components.
- **I/O devices** – the system's connection to the external world
  - Ex) display, mouse, disk, and network adapter.
- **Main memory** – a temporary storage device that holds both a program and the data it manipulates while the processor is executing the program
- **Processor or central processing unit (CPU)** – an engine that interprets or executes program instructions stored in main memory.
  - Register file: small storage
  - Arithmetic/logic unit: computing unit
  - Program counter: line indicator

Running “hello” Program

1. Get a command executing “hello” program in the shell program.

2. The operating system loads “hello” executable program stored in the disk to the memory and start running the program.

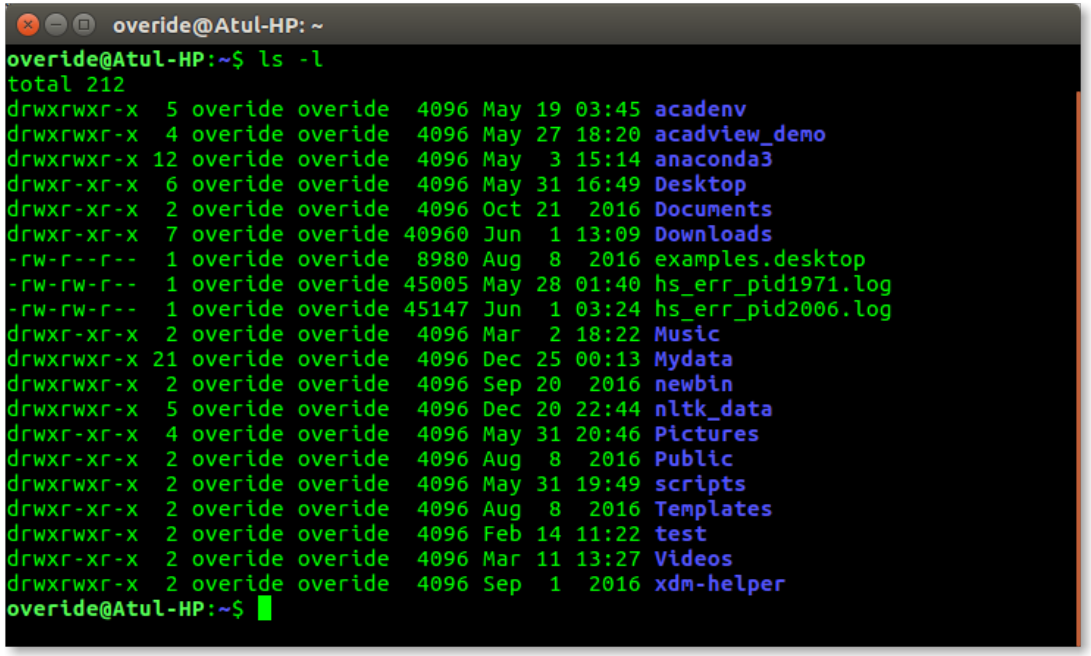
3. A processor executes the instructions of loaded “hello” program.

4. After termination, switch back to the shell program.

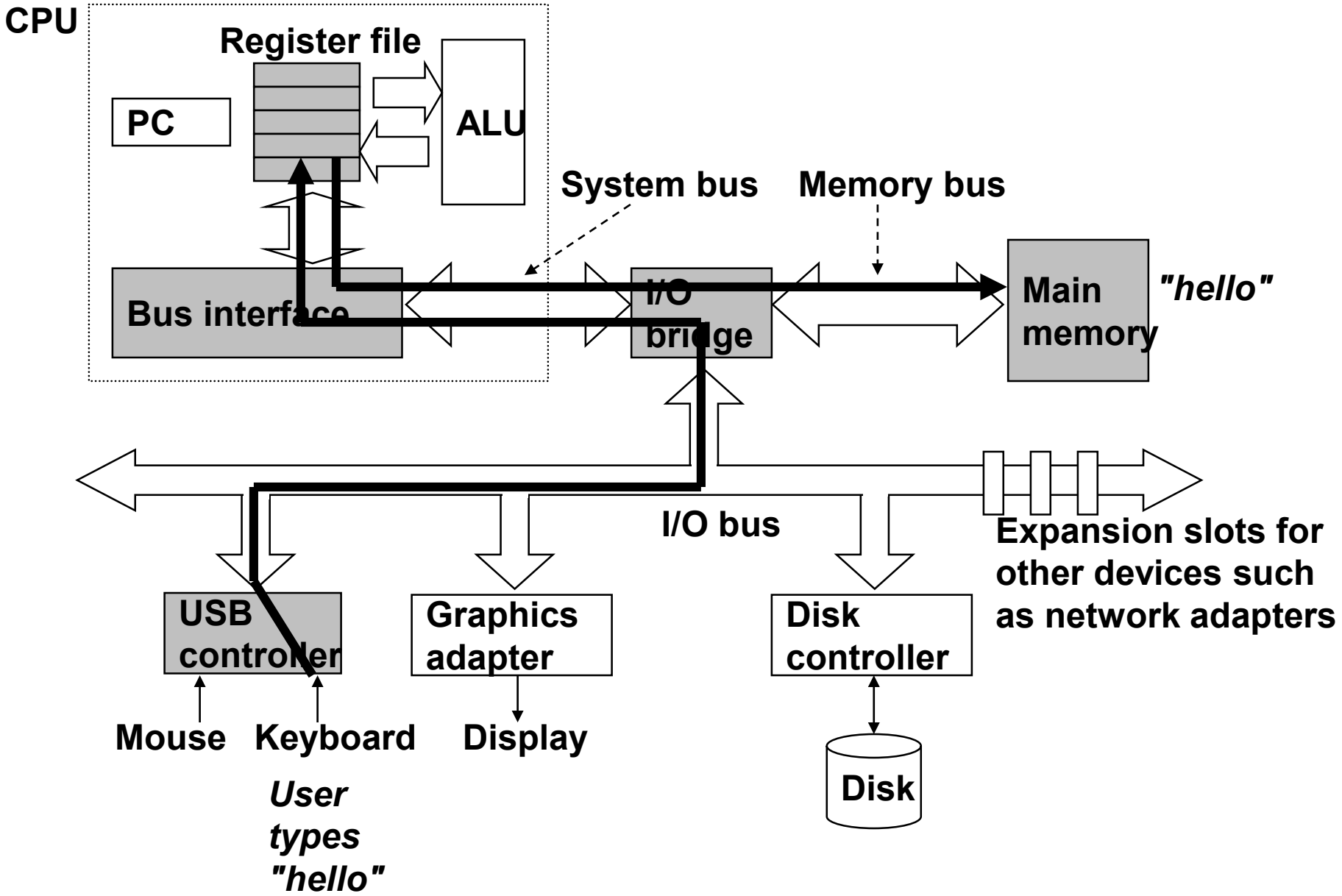
Shell program

- Text-based user interface

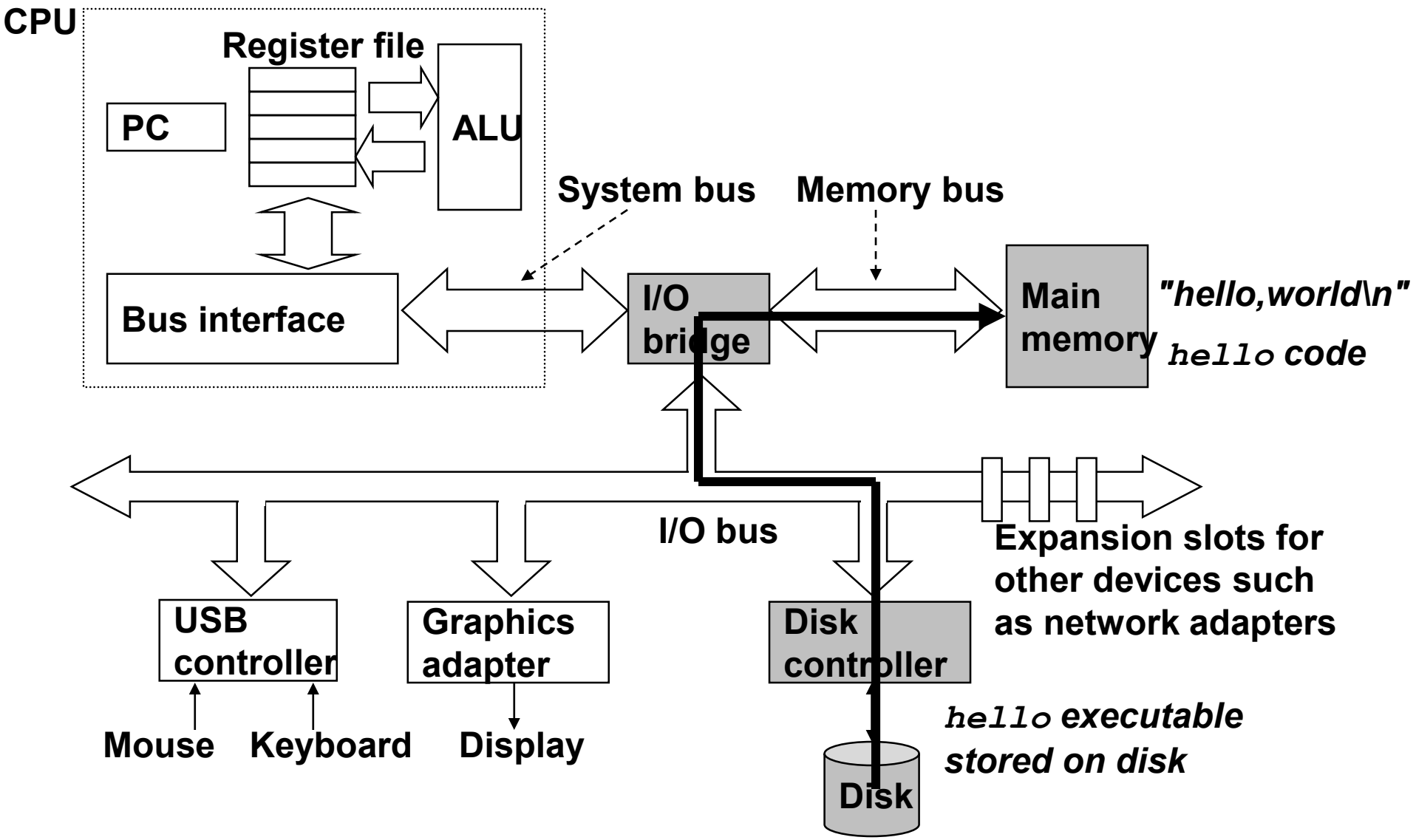
- A command-line-based interpreter
  - run the command directly
  - exploring file systems
  - deleting and creating files
  - navigating the path
  - running the program
  - printing texts



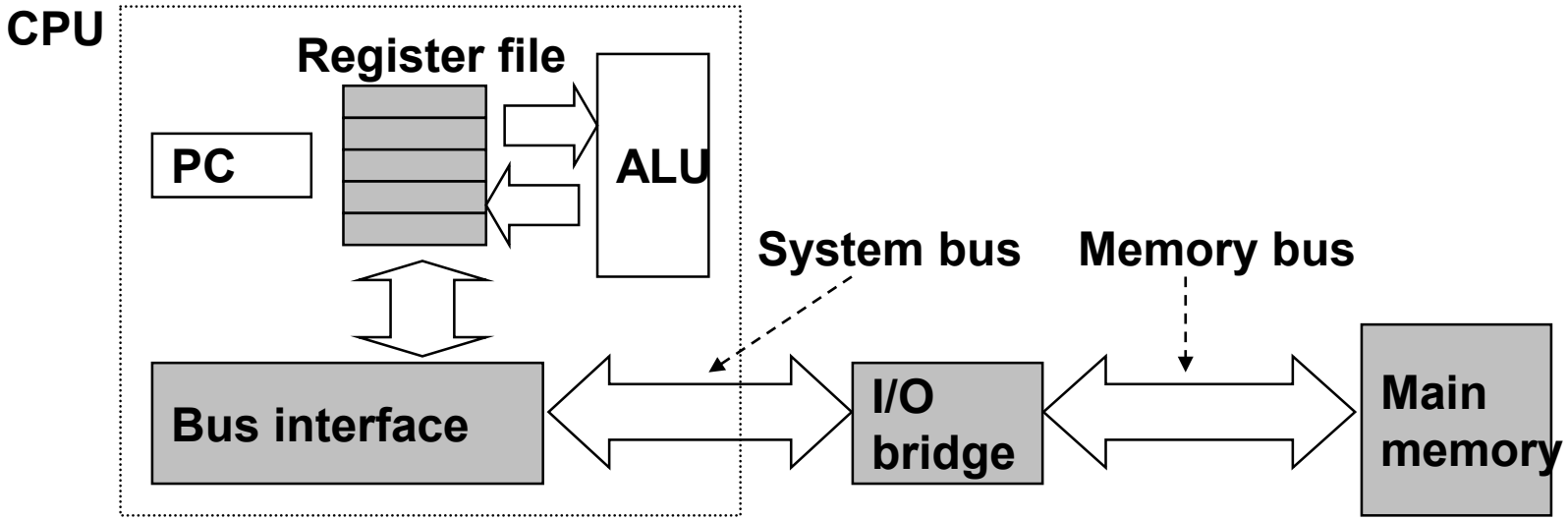
Reading hello command from keyboard



Loading executable from disk into memory



Running “hello” program



2. Execute & increase Program Counter (PC)

1. Load an instruction at PC

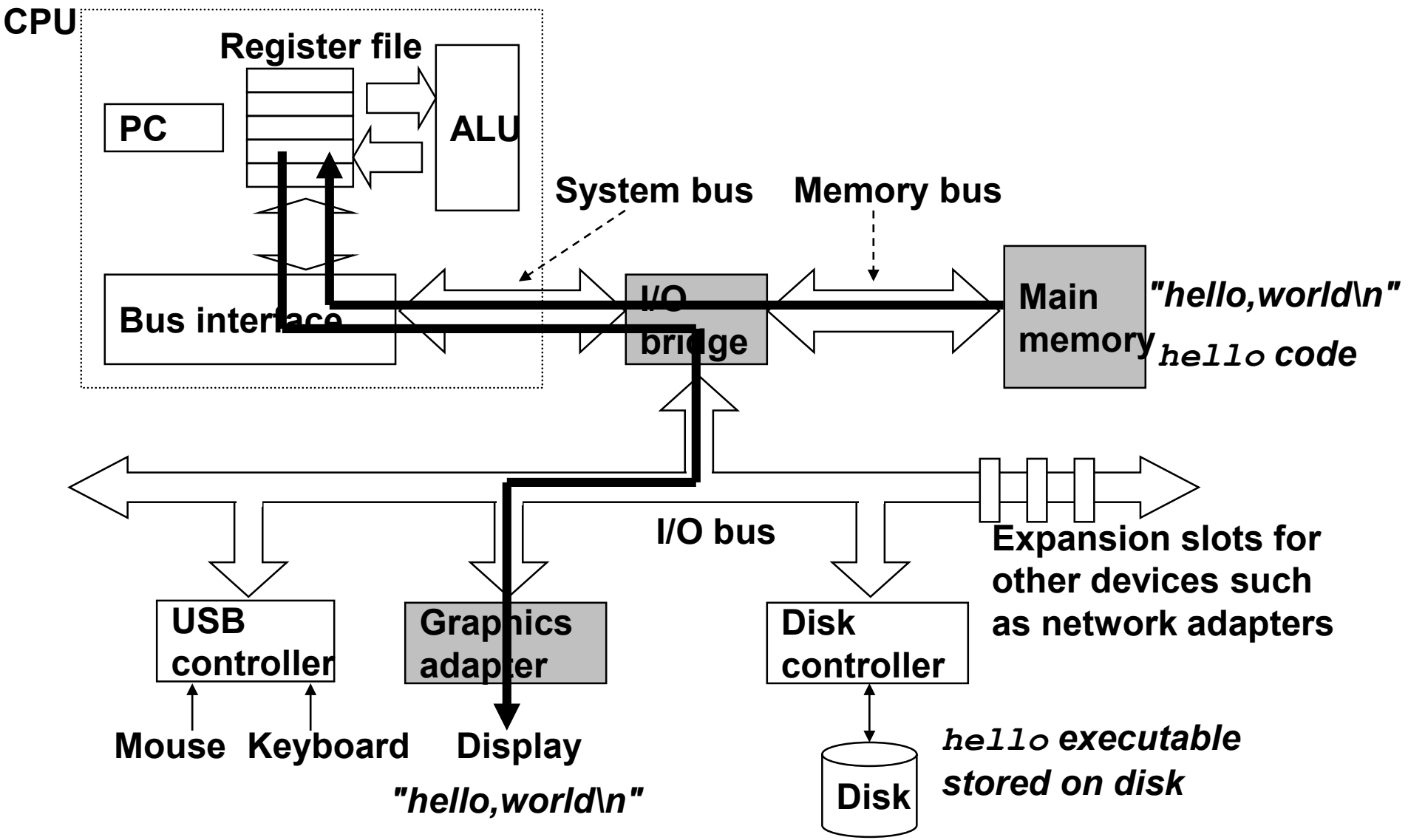
Memory

```
1149:       f3 0f 1e fa             endbr64
114d:       55                      push   %rbp
114e:       48 89 e5                mov    %rsp,%rbp
1151:       48 8d 3d ac 0e 00 00    lea    0xeac(%rip),%rdi
1158:       e8 f3 fe ff ff          callq  1050 <puts@plt>
115d:       90                      nop
115e:       5d                      pop    %rbp
115f:       c3                      retq
```

PC

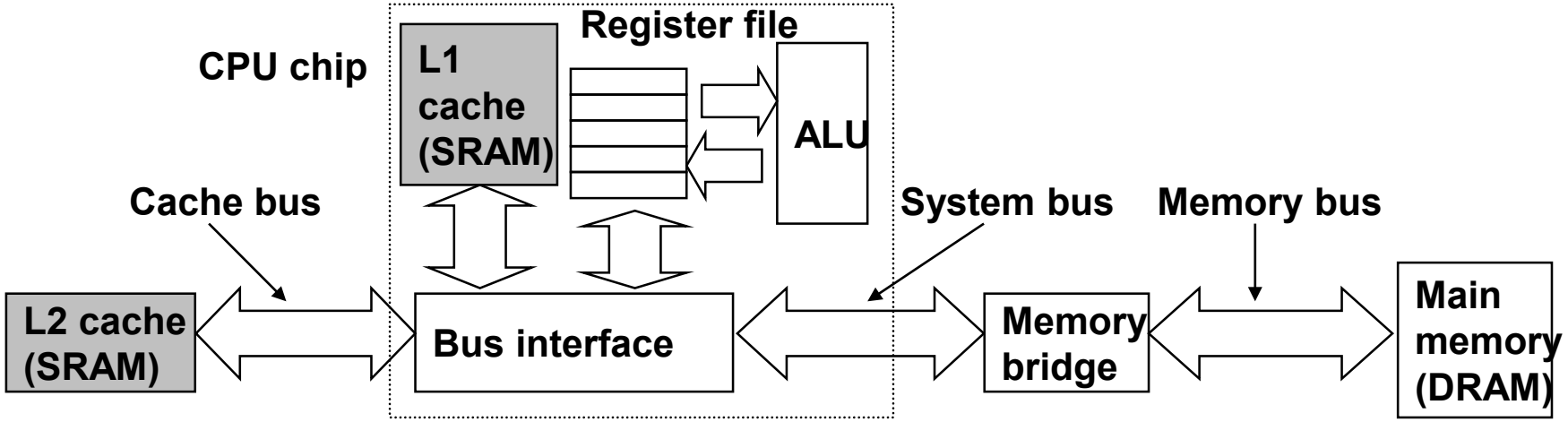
DATA

Writing “hello, world” from memory to display



Caches matters

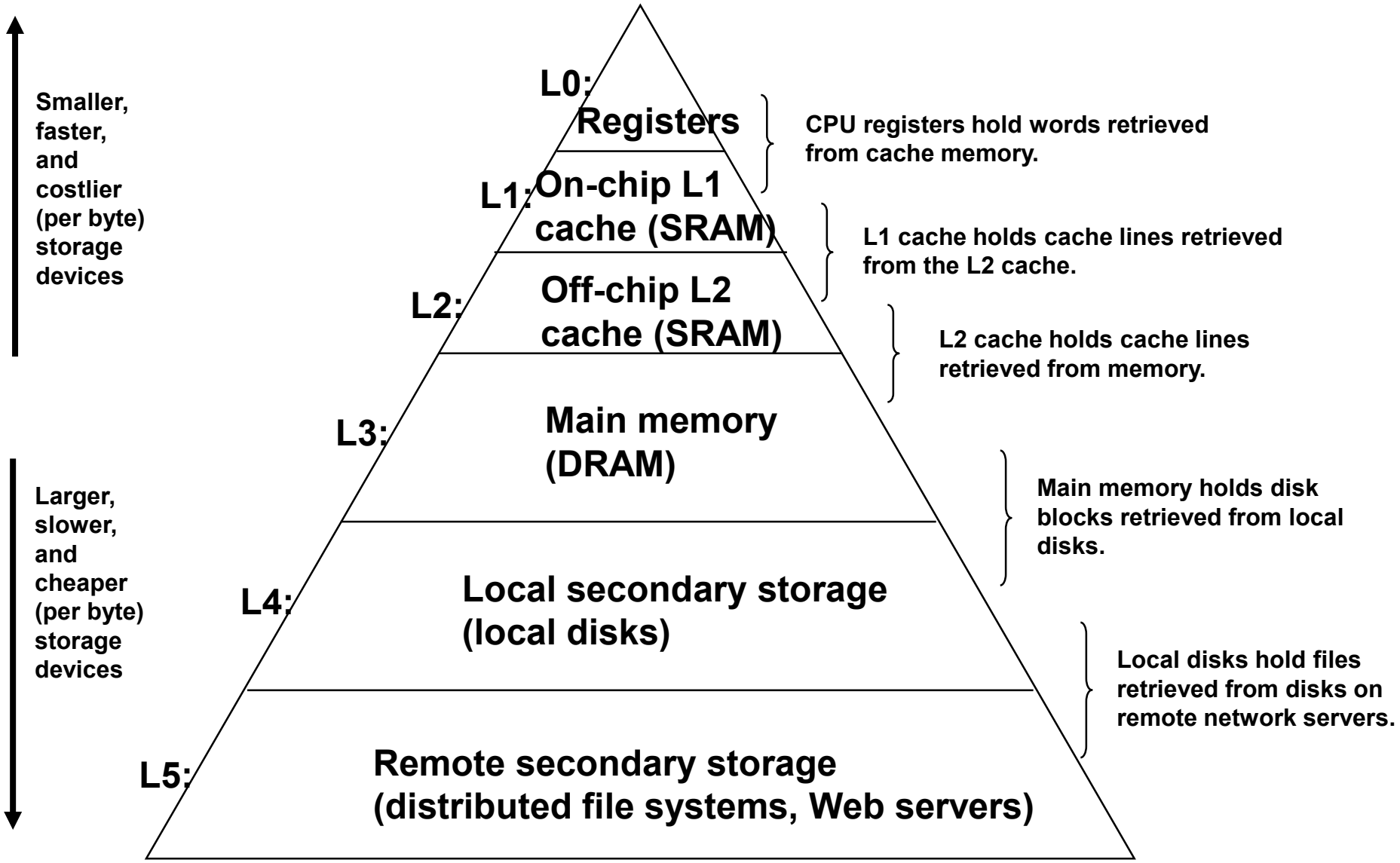
- Processor memory gap increases
  - a few hundreds bytes of register file / millions of bytes in main memory (but 100 times faster than main memory)



Cache memories

- Programmer can exploit caches to improve the performance of their programs

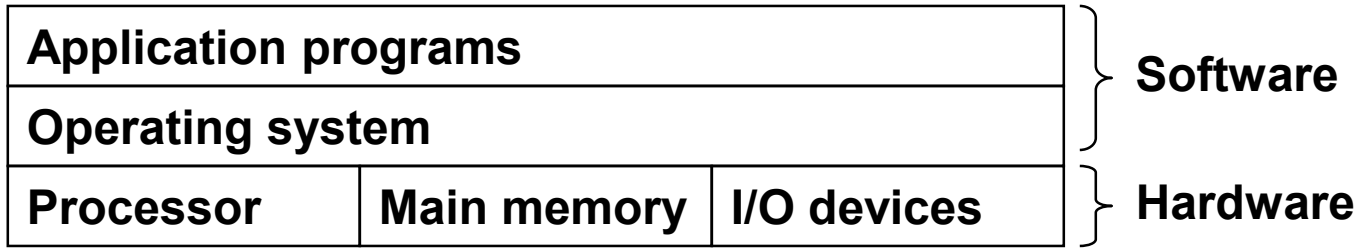
Memory hierarchy



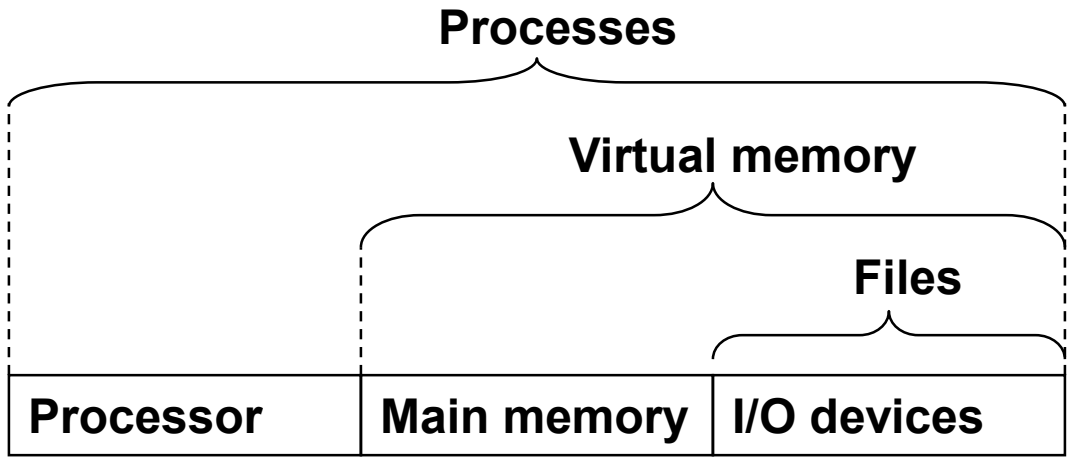
The operating system manages the hardware

Who accesses keyboard, display, disk?

- Operating system : protection and hardware interface



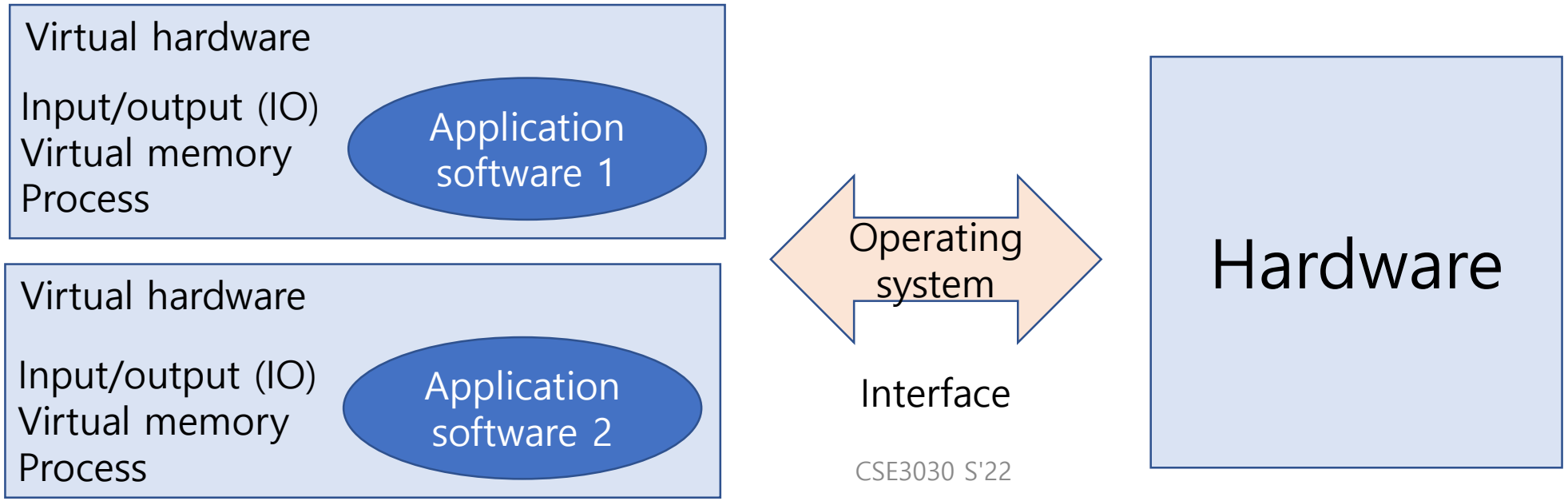
Layered view of a computer system



Abstractions provided by an operating system

Abstractions?

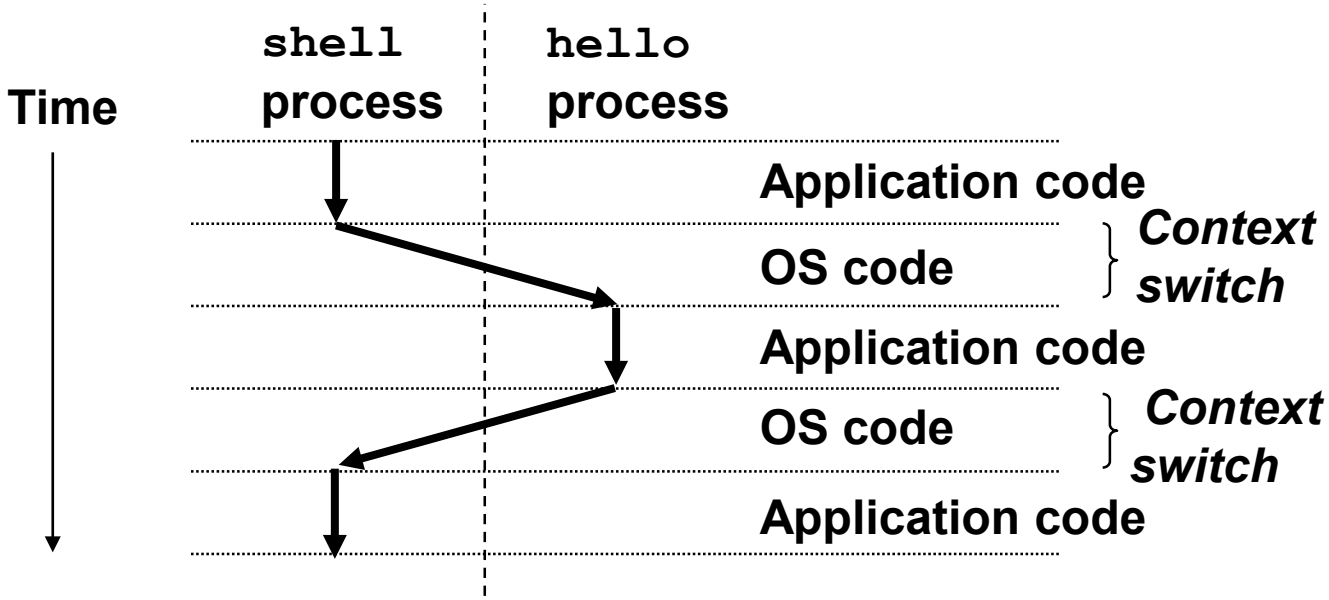
- The **operating system** provides the abstracted/imaginary hardware environment to **application software** in order to remove hardware details and decouple hardware dependency from application software.
- Application software uses those imaginary system features (virtual memory, I/O devices) and the operating system interpret those features to the real hardware features



The operating system manages the hardware

Process (later Thread)

- operating system's abstraction for a running program
- multiple processes can run concurrently on the same system
- each process appears to have exclusive use of the hardware



Process Context Switching

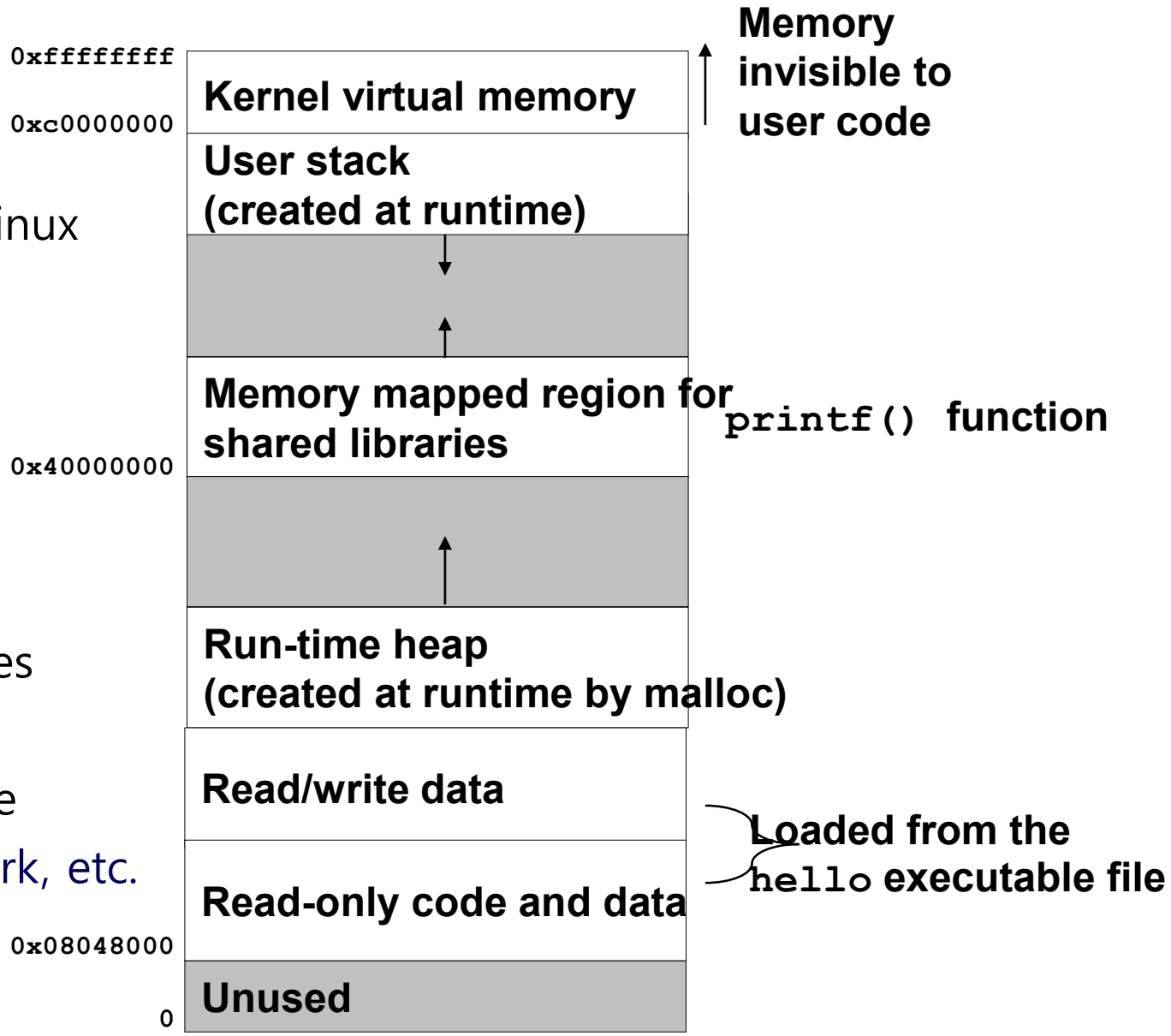
The operating system manages the hardware

Virtual memory

- Process virtual address space in Linux

Files

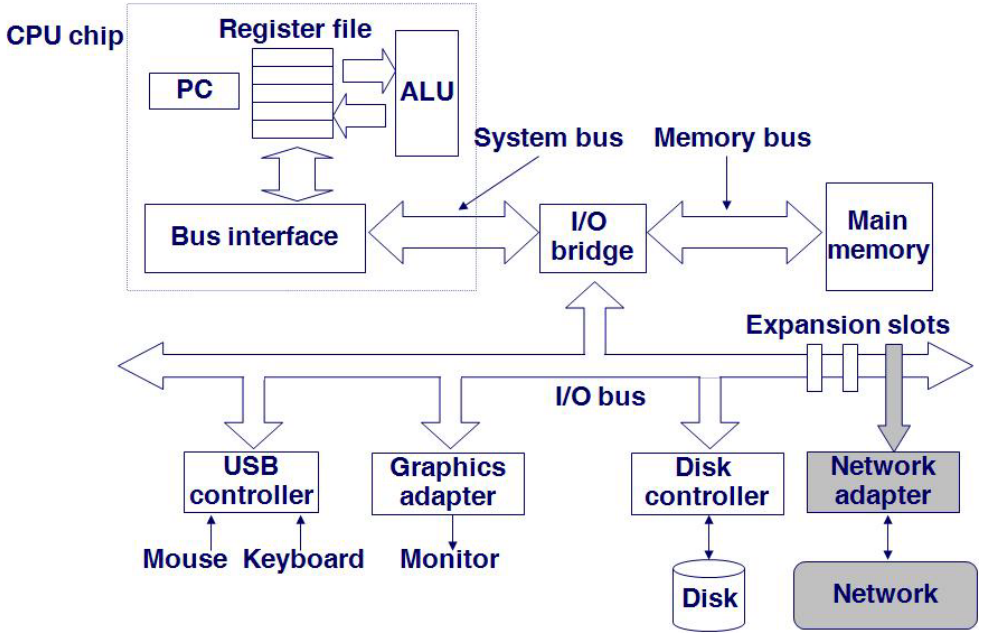
- sequence of bytes
- every I/O device is modeled as a file
- keyboard, network, etc.
- read/write



Systems communicate with other systems using net works

Network applications

- Email, instant messaging, WWW, FTP, telnet, etc.



A network is another I/O device

Run hello in remote machine