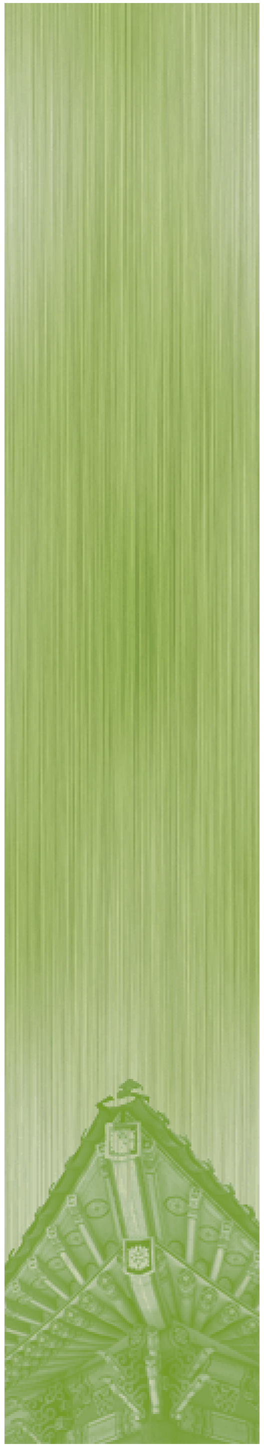


[Chap.1] A Tour on Computer Systems

Young Ik Eom (yieom@skku.edu, 031-290-7120)
Distributing Computing Laboratory
Sungkyunkwan University
http://dclab.skku.ac.kr

Goal

- **Understand what happens and why**
    - When you run hello program on your system

```
#include <stdio.h>

int main()
{
   printf(“hello, world\n”);
   return 0;
}
```

Source File

Source file

```
#include <stdio.h>
 directive
int main()
{
   printf(“hello, world\n”);
   return 0;
}
```

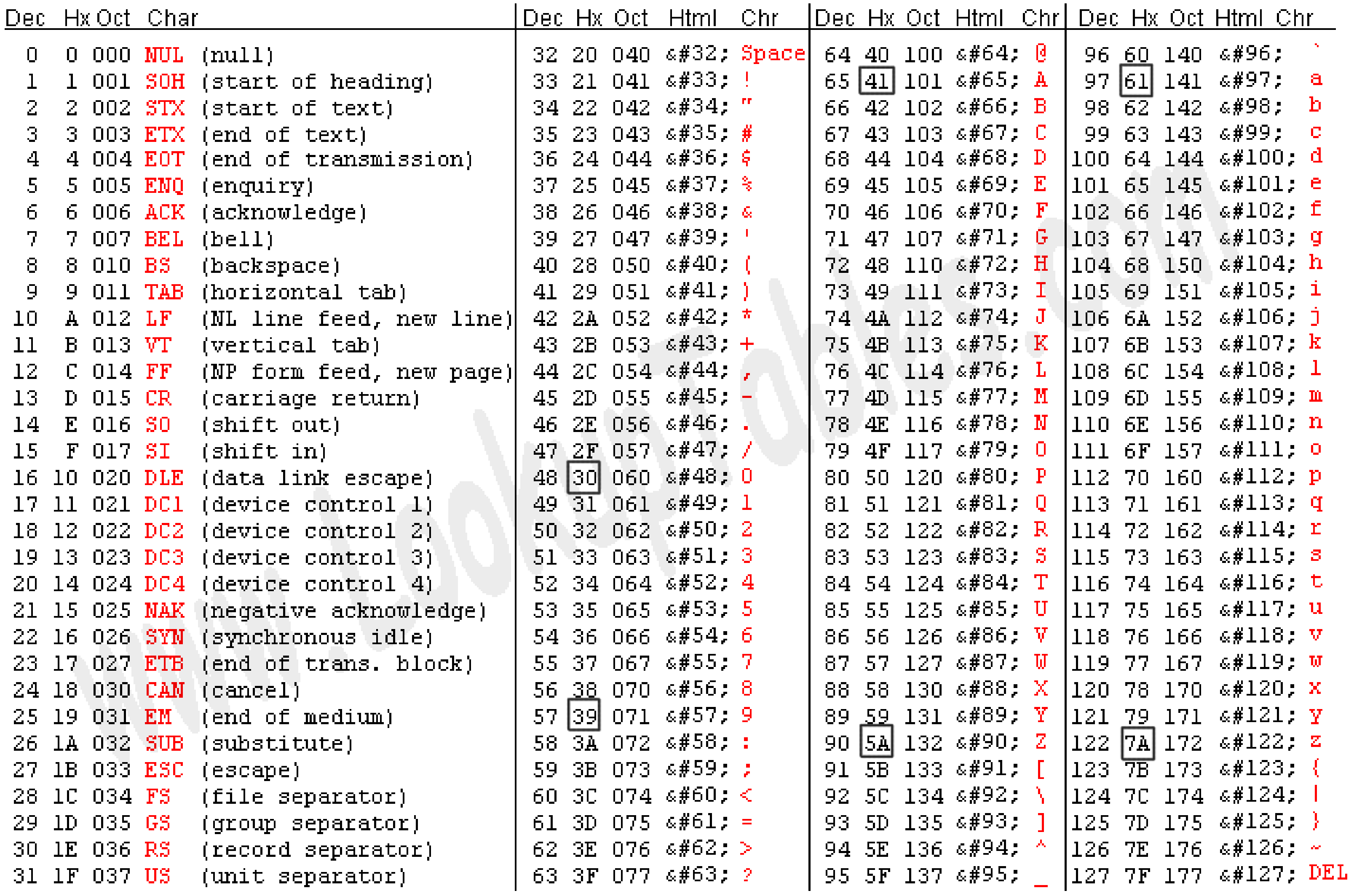
#	i	n	c	l	u	d	e	SP	<	s	t	d	i	o	.
35	105	110	99	108	117	100	101	32	60	115	116	100	105	111	46
h	>	\n	\n	i	n	t	SP	m	a	i	n	(	)	\n	{
104	62	10	10	105	110	116	32	109	97	105	110	40	41	10	123
\n	SP	SP	SP	SP	p	r	i	n	t	f	(	“	h	e	l
10	32	32	32	32	112	114	105	110	116	102	40	34	104	101	108
l	o	,	SP	w	o	r	l	d	\	n	”	)	;	\n	SP
108	111	44	32	119	111	114	108	100	92	110	34	41	59	10	32
SP	SP	SP	r	e	t	u	r	n	SP	0	;	\n	}	\n	
32	32	32	114	101	116	117	114	110	32	48	59	10	125	10	

Source File

Notes)

- Text file
  - File that consists exclusively of ASCII characters
- Binary file
  - …
- ASCII standard
  - Represents each character with a unique 8-bit integer value

Source File (ASCII Code Table)



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

Source : www.LookupTables.com

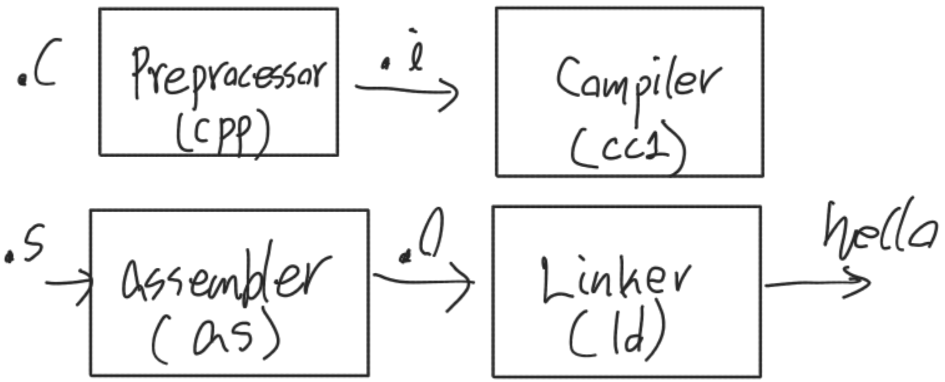
Translation

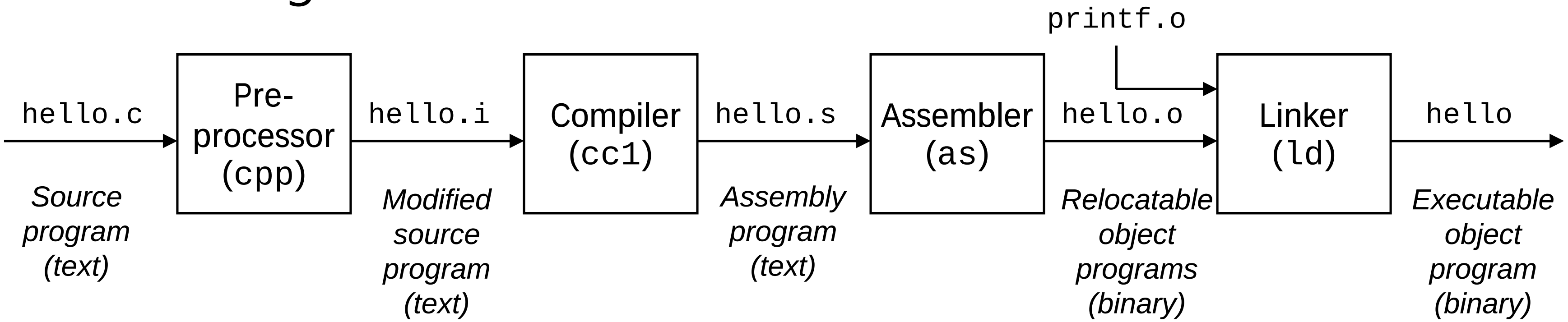
Compilation system

```
unix> gcc –o hello hello.c
```

4 phases

- Preprocessing
- Compilation
- Assembly
- Linking





Translation

Preprocessing

- Produces pure C-language program
  - By processing preprocessor directives such as…

```
#include …
#define …
#ifdef …
```

etc…

Translation

Compiling the source program

- Produces assembly-language program

```
main:
  subq   $8, %rsp
  movl   $.LCO, %edi
  call   puts
  movl   $0, %eax
  addq   $8, %rsp
  ret
```

Translation

Relocatable object

Executable object

Assembling and Linking

- Produces binary executable code

```
53 | 48 89 d3 | e8 00 00 00 00 | 48 89 03 | 5b | c3
```

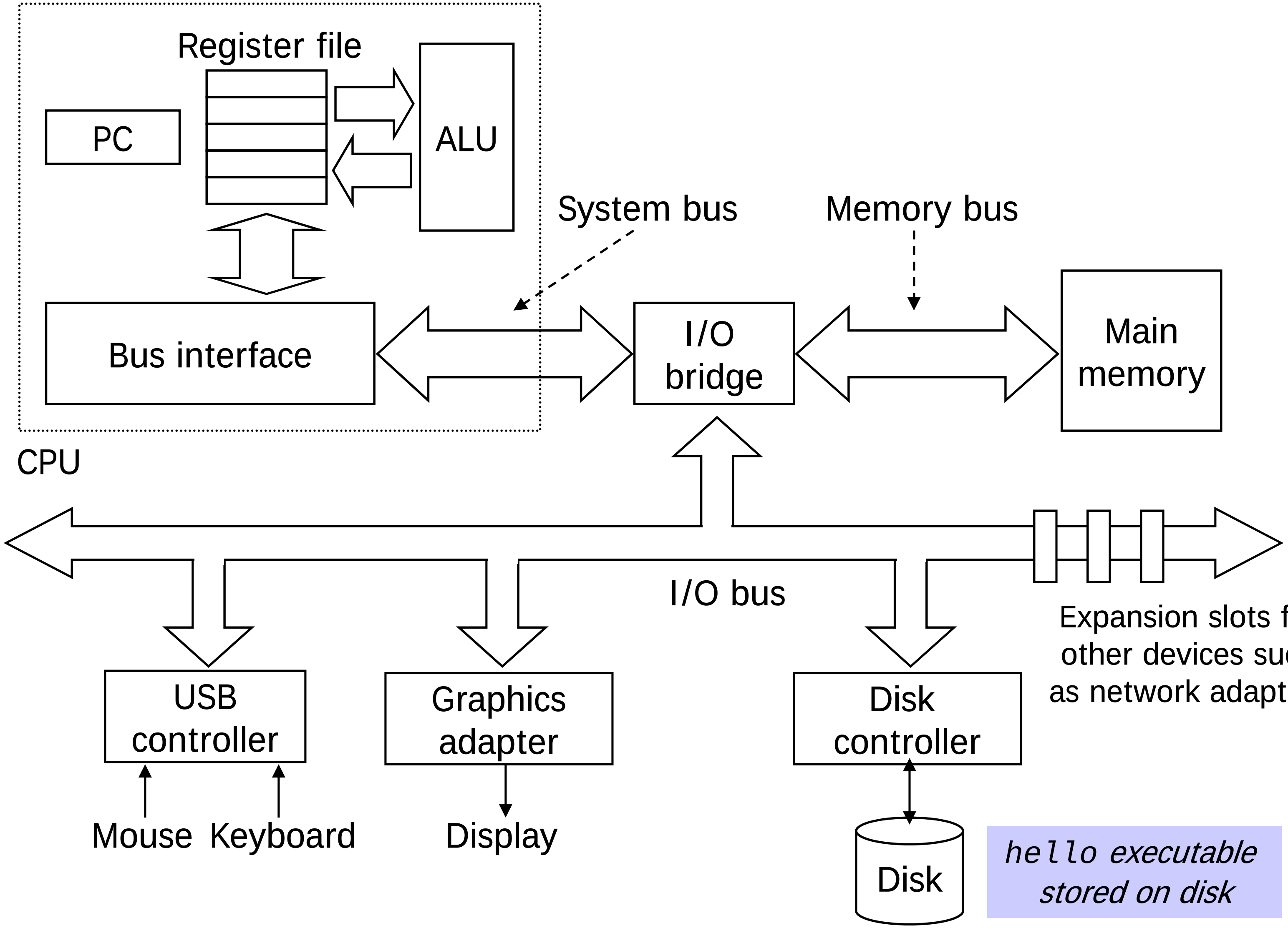
Running

Running the executable

```
unix> ./hello
hello, world
unix> _
```

- The shell interprets the command line
  - Checks the first word in the command line corresponds to built-in shell command
  - When it is, runs the command by itself
  - When it is not, loads the executable file and runs it
  - Waits for it to terminate
  - Prompts again

Running (HW organization)



hello executable stored on disk

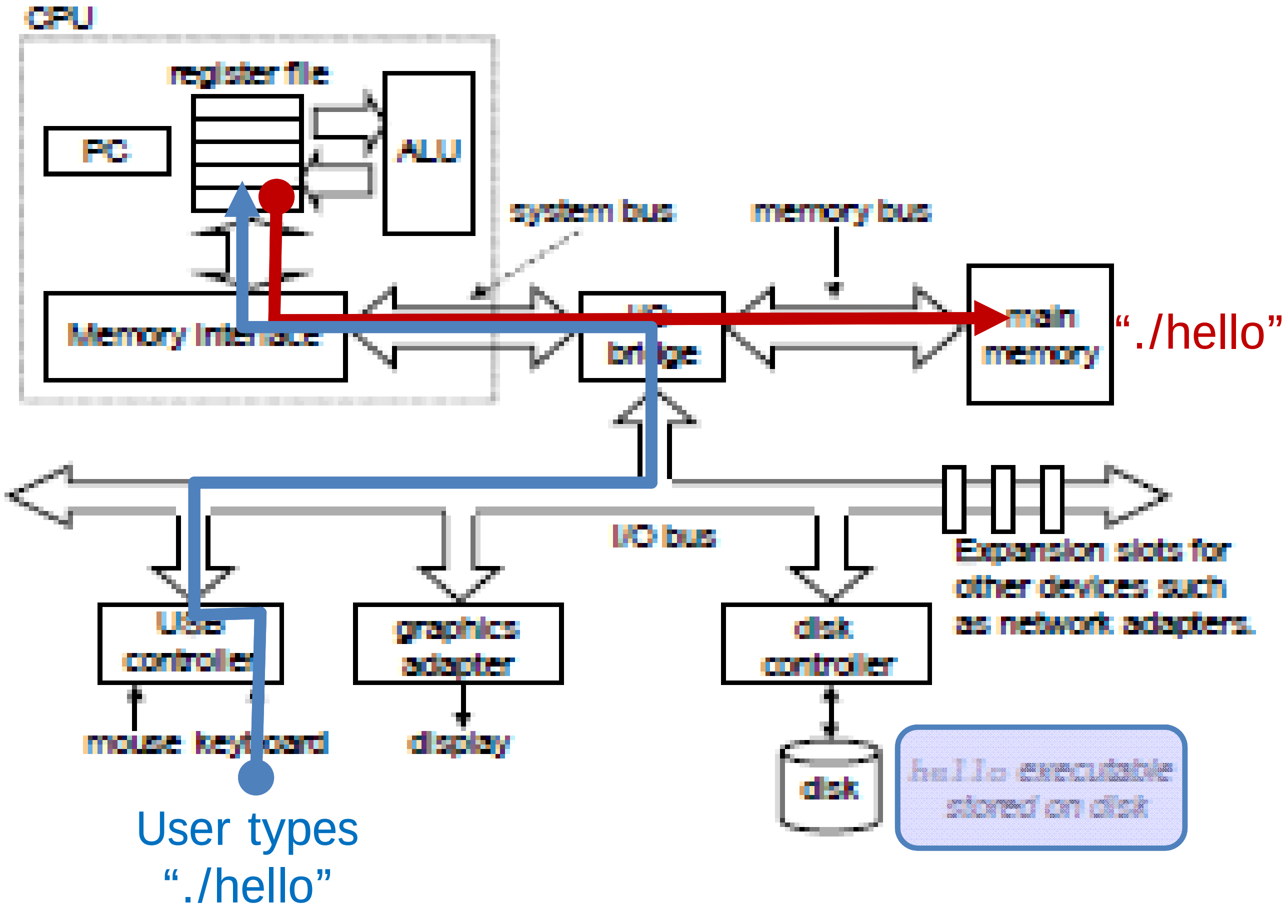
Running (HW organization)

Notes)

- Bus and word
  - Word size: 1/2/4/8 bytes
- Controller and adapter
  - Controller
    - ✓ Chipset in the device itself or on the system's main PCB
  - Adapter
    - ✓ Card that plugs into a slot on the PCB
- Instruction cycle
  - IF/ID/OF/EX/IC

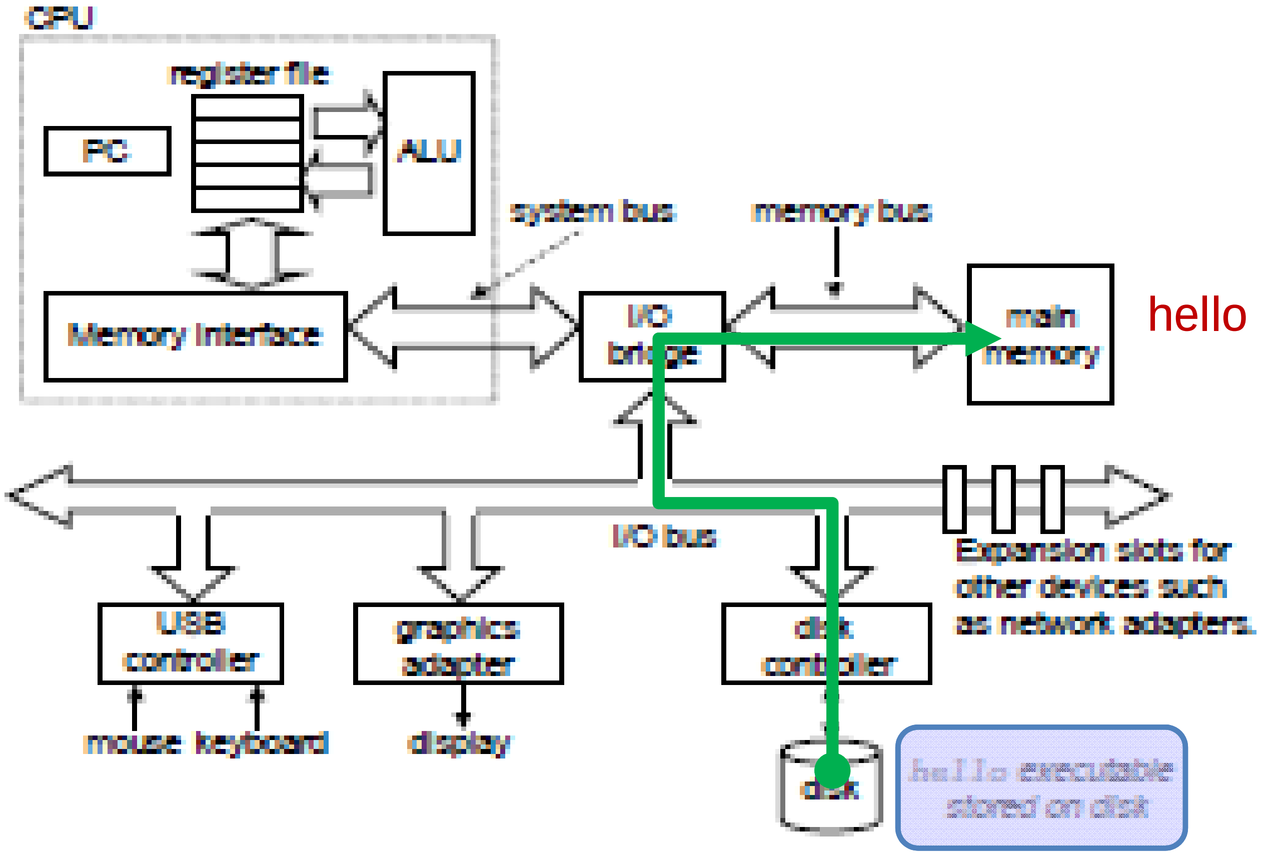
Running

- **Storing the command into memory**



Running

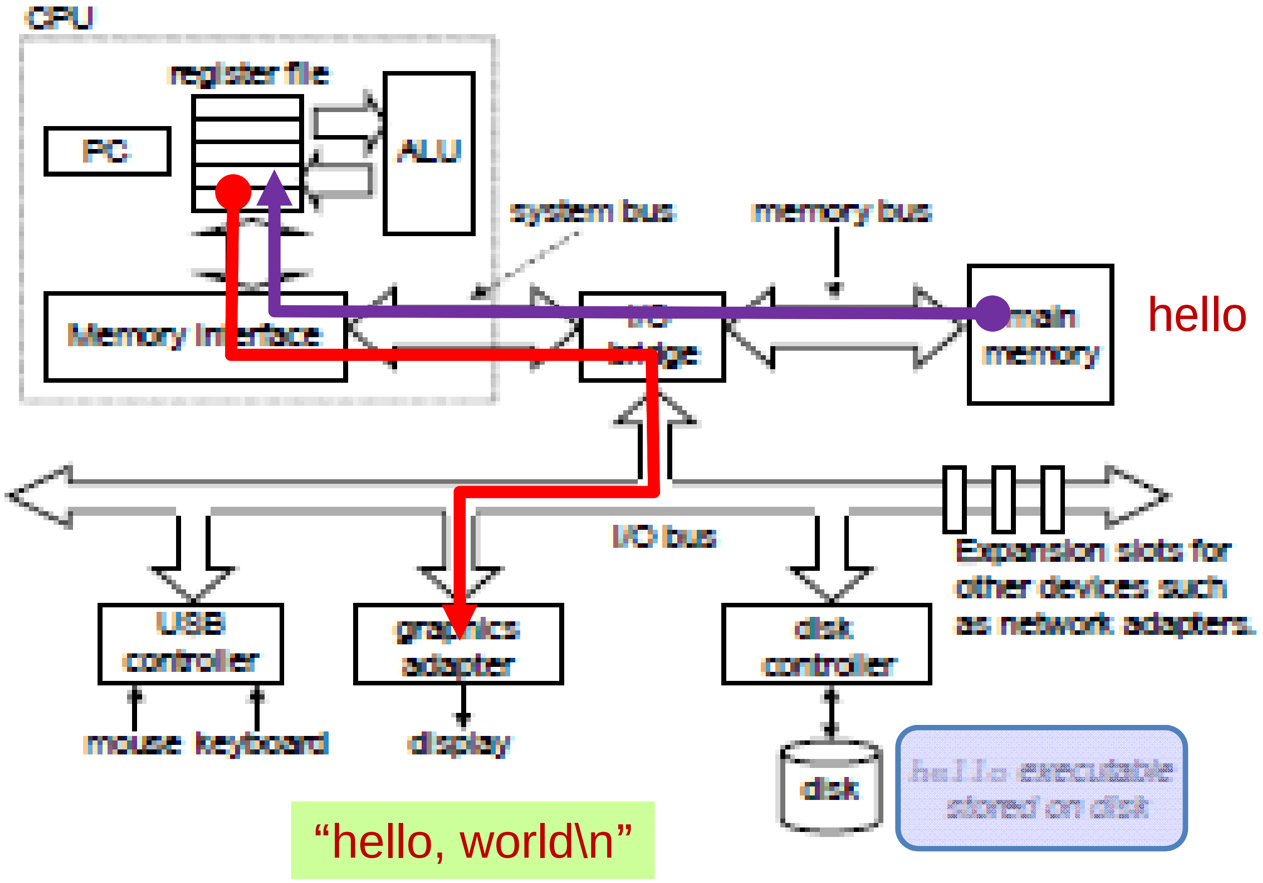
Loading the executable



hello code+data

Running

- Executing the program



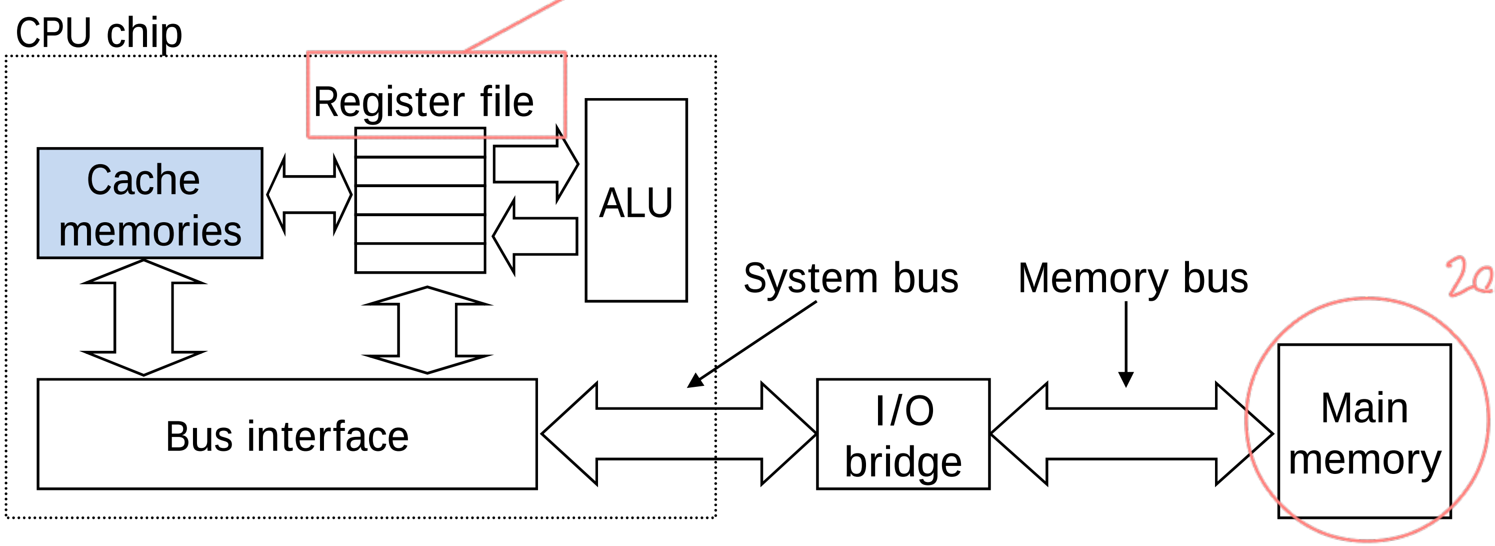
Caches

Cache memories

- Deals with the processor-memory gap
  - L1 cache (on the processor chip) 4 cycles
  - L2 cache (connected to the processor by a special bus) 10 cycles
  - L3 cache 50 cycles
- Exploits locality

한 번 접근된 명령, 데이터는 재접근될 확률↑ + 2 근처의 데이터도 접근할 확률↑

1 clock cycle

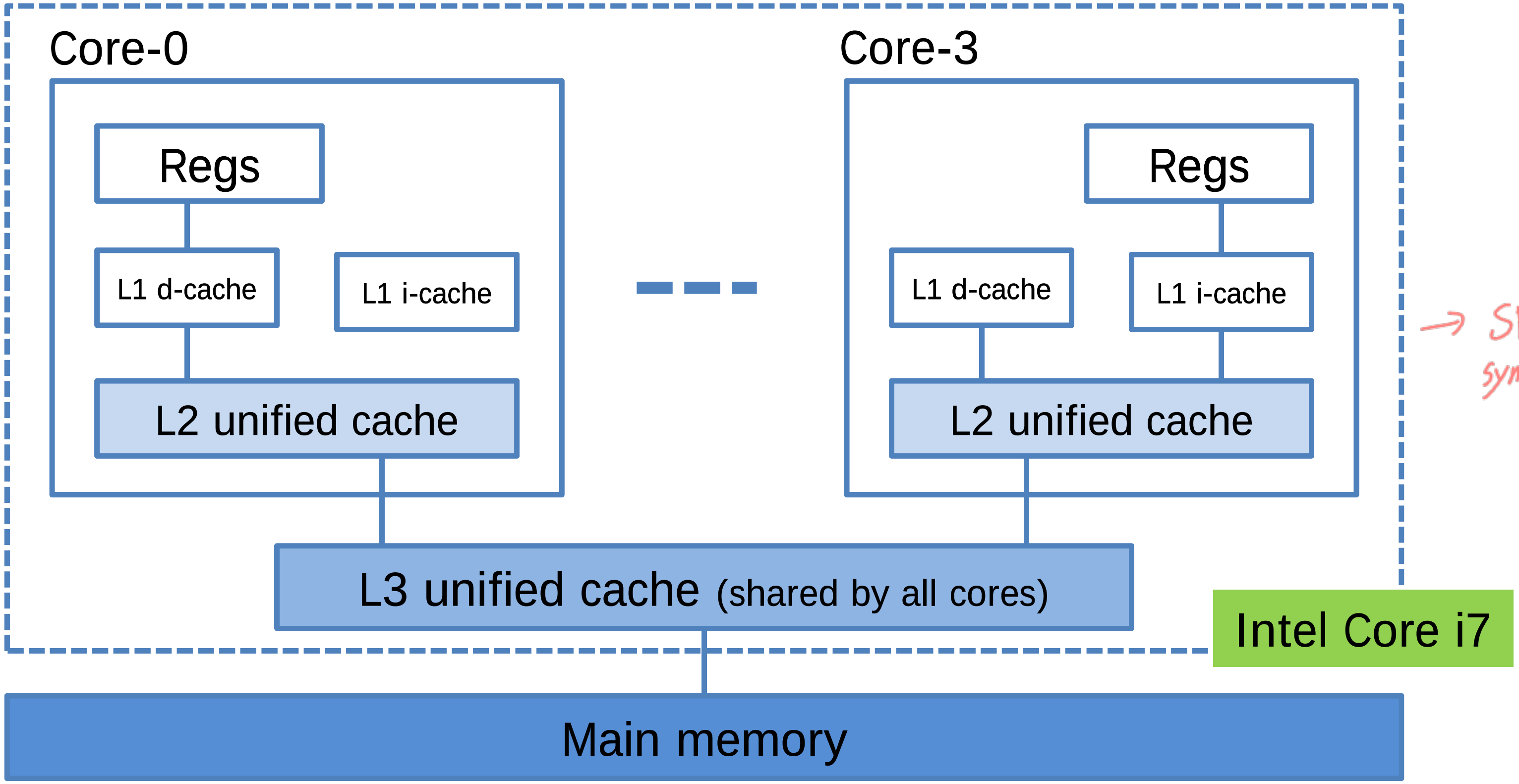


200 cycles

Multi-core Architectures

Multi-core processor

- Have several processors (cores) integrated into a single integrated-circuit chip

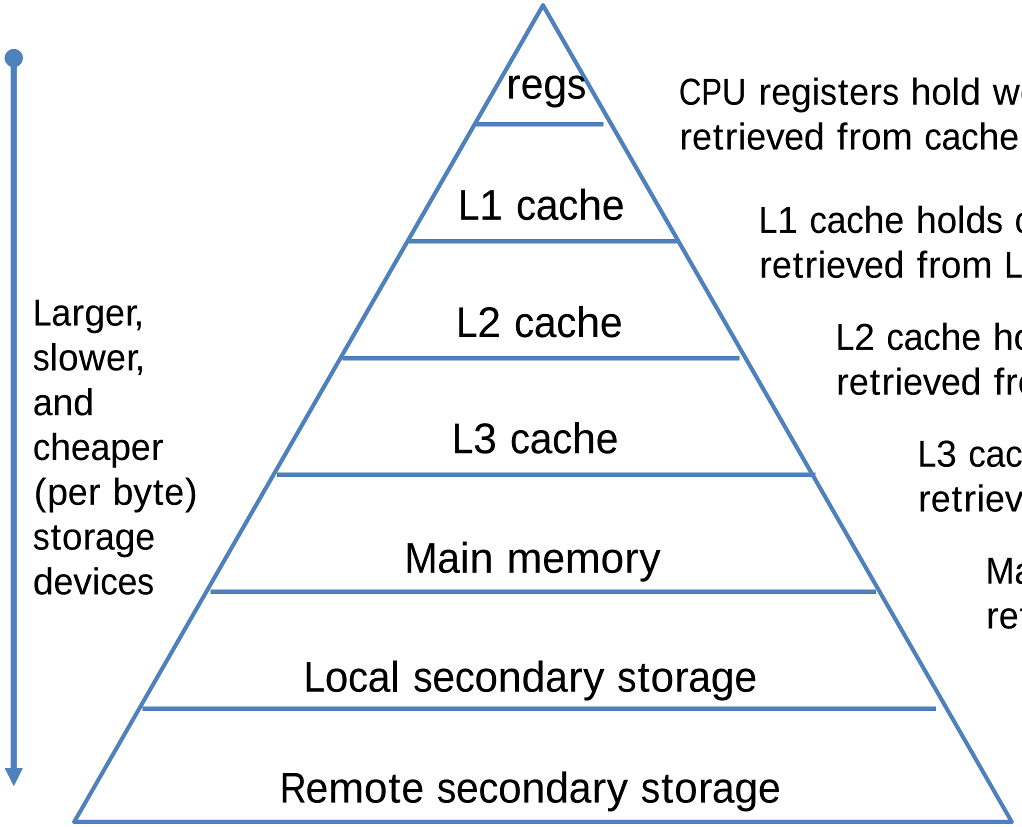


→ SMP
symmetric - multiprocessor

Storage Hierarchy

Hierarchy of storage devices

- Storage at one level serves as a cache for storage at the next lower level



CPU registers hold words retrieved from cache memory

L1 cache holds cache lines retrieved from L2 cache

L2 cache holds cache lines retrieved from L3 cache

L3 cache holds cache lines retrieved from main memory

Main memory holds disk blocks retrieved from local disks

Local disks hold files retrieved from disks on remote network server

Storage Hierarchy

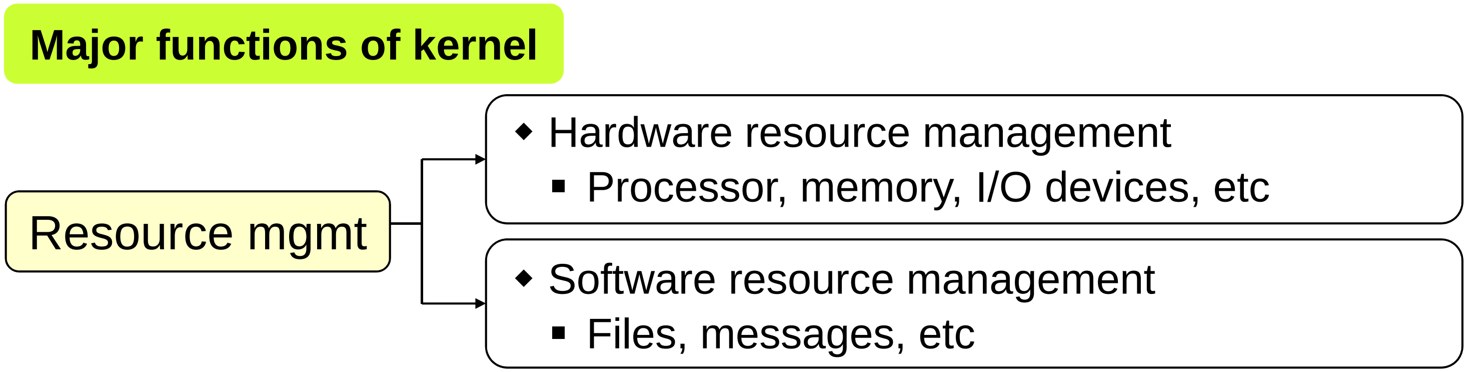
Storage hierarchy

Type	What cached	Where cached	Latency (#cycles)	Managed by
CPU registers	4B or 8B words	On-chip CPU registers	0~1	Hardware
L1 cache	64B blocks	On-chip L1 cache	4	Hardware
L2 cache	64B blocks	On-chip L2 cache	10	Hardware
L3 cache	64B blocks	On-chip L3 cache	50	Hardware
Main memory	4KB pages	Main memory	200	Hardware + OS
Storage (HDD)	Parts of files	Local HDD	10,000,000	OS
Web cache	Web pages	Remote server disks	1,000,000,000	Web proxy server

Operating Systems

OS

- Functions of OS
  - User interface (for user convenience)
  - Resource management (for efficiency)
  - Process management
  - + Networking / Security&Protection



Major functions of kernel

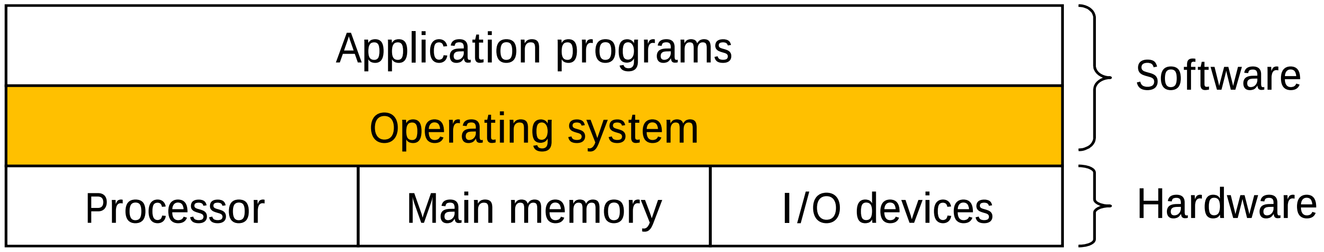
Resource mgmt

- Hardware resource management
  - Processor, memory, I/O devices, etc
- Software resource management
  - Files, messages, etc

Operating Systems

OS

- Provide applications with simple and uniform interface for manipulating complicated and often widely different low-level HW devices
- Protects the HW from misuse by applications



Application programs

Operating system

Processor | Main memory | I/O devices

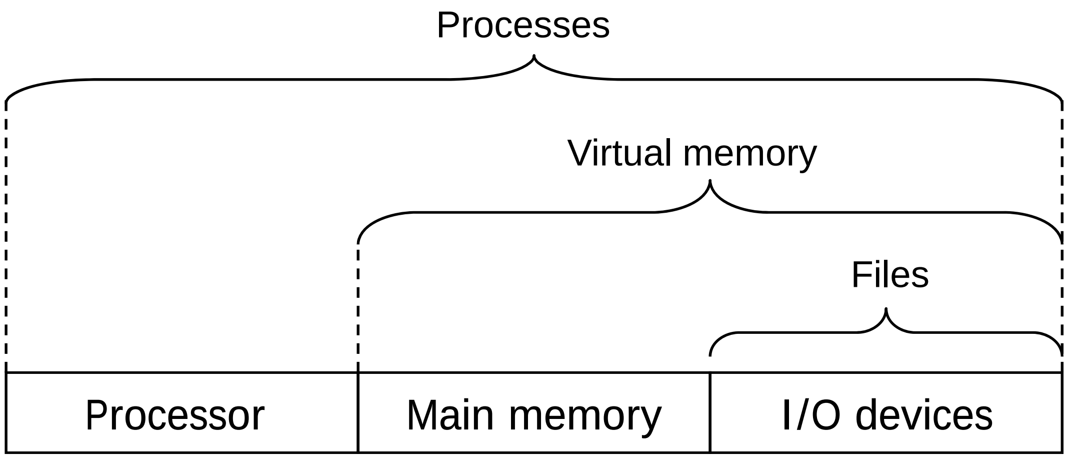
Software

Hardware

Operating Systems

OS abstractions

- Processes
- Virtual memory
- Files

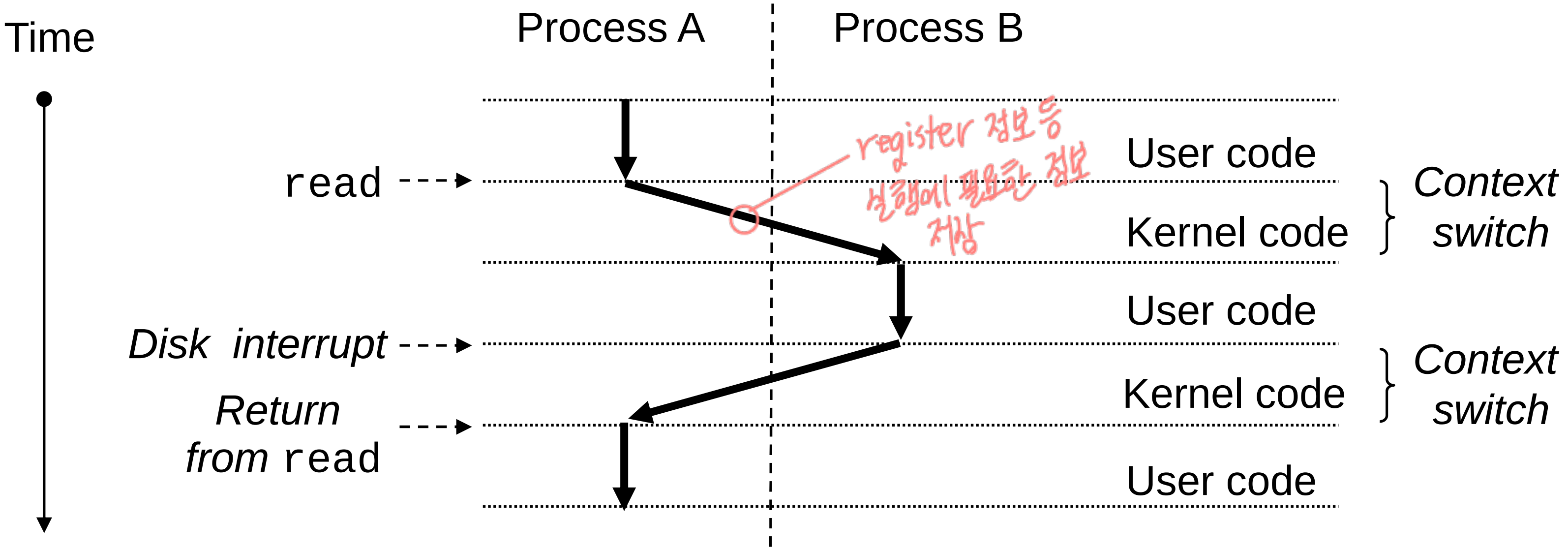


Operating Systems

OS abstractions

- Process
  - OS's abstraction for a running program

- Context switching

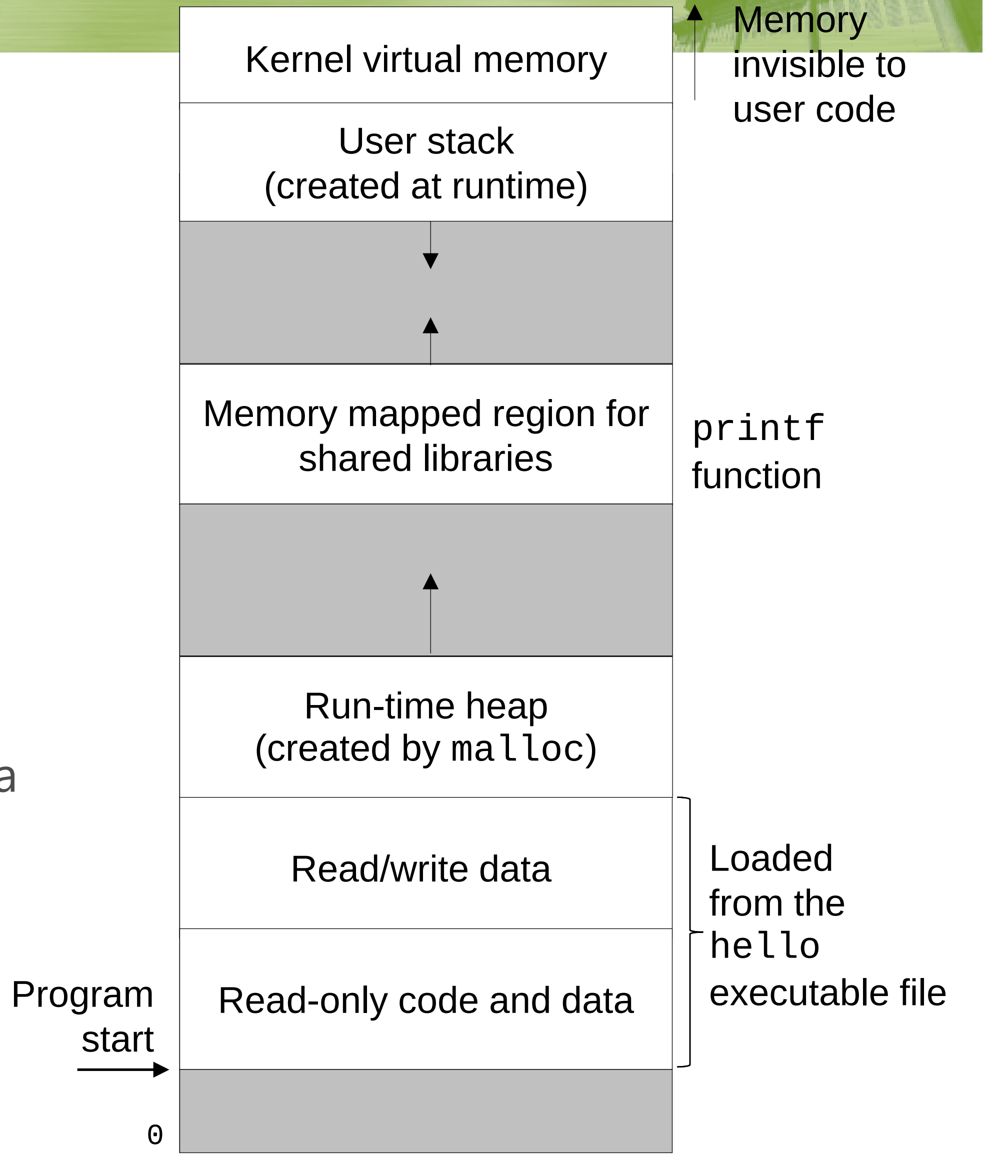


register 정보 등
실행에 필요한 정보
저장

Operating Systems

OS abstractions

- Virtual memory
  - Provides each process with the illusion that it has exclusive use of main memory
- Virtual address space
  - View of memory by each process
    - ✓ Program code and data
    - ✓ Heap
    - ✓ Shared libraries
    - ✓ Stack
    - ✓ Kernel space


Kernel virtual memory
Memory invisible to user code
User stack (created at runtime)
Memory mapped region for shared libraries
printf function
Run-time heap (created by malloc)
Read/write data
Loaded from the hello executable file
Read-only code and data
Program start
0

Operating Systems

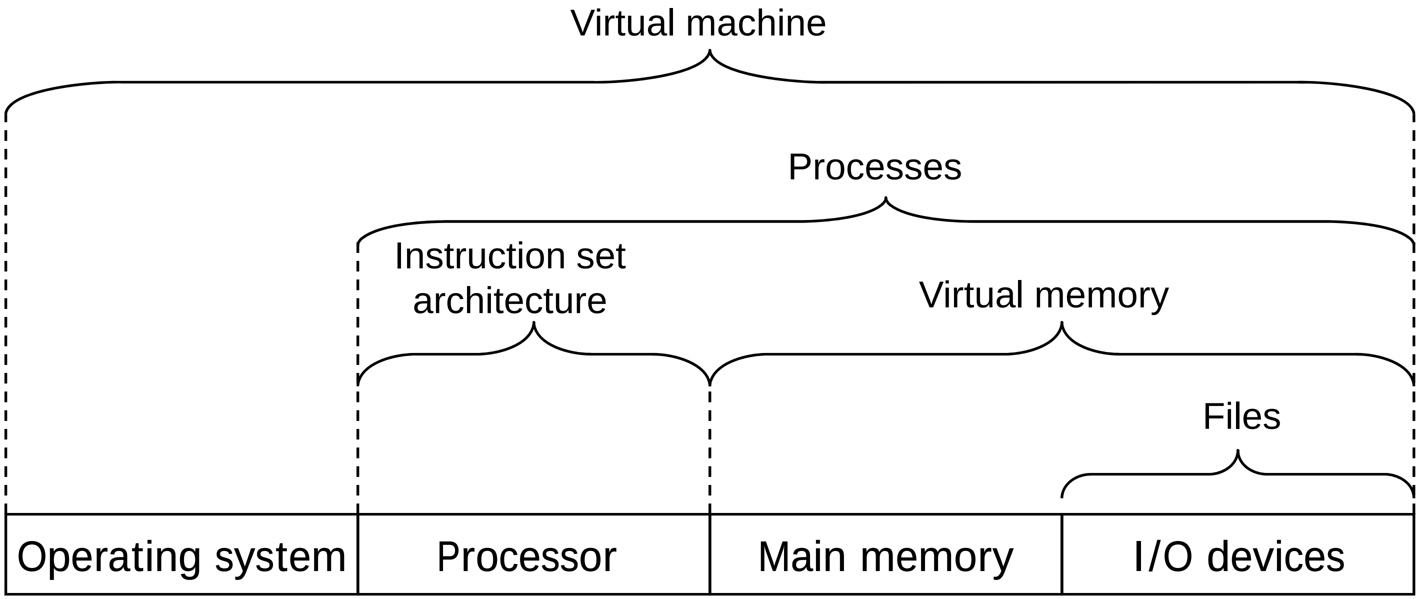
OS abstractions

- File
  - Sequence of bytes
  - Provides applications with a uniform view of all of the varied IO devices that might be contained in the system

Operating Systems

More abstractions: virtual machine

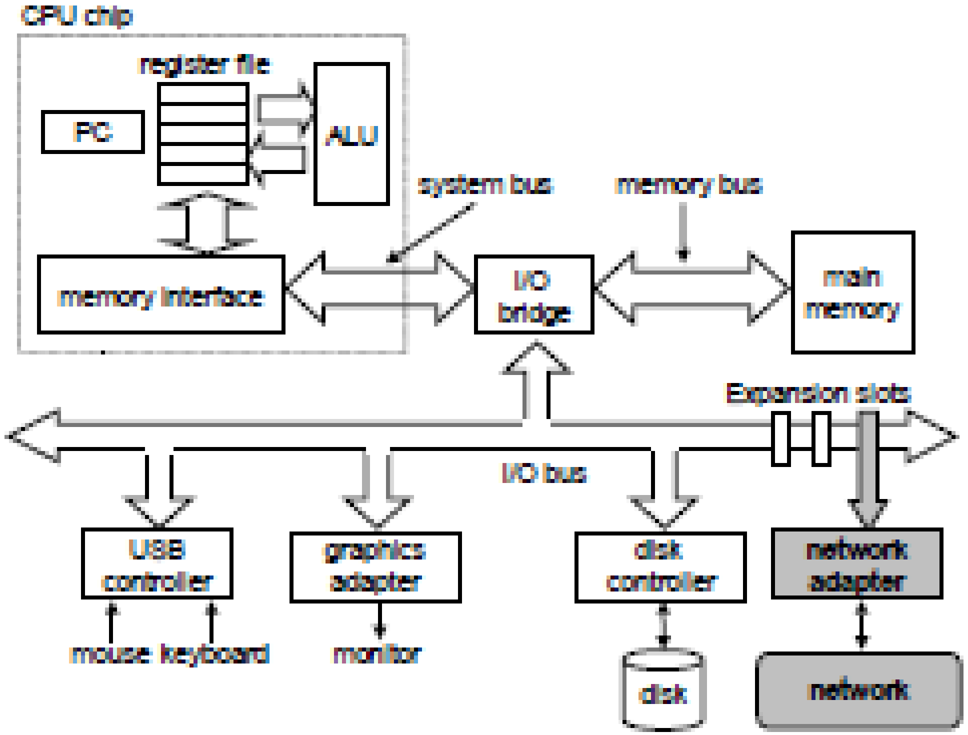
- An abstraction of the entire system, including operating system, the processor, and the programs (processes)



Networks

Network

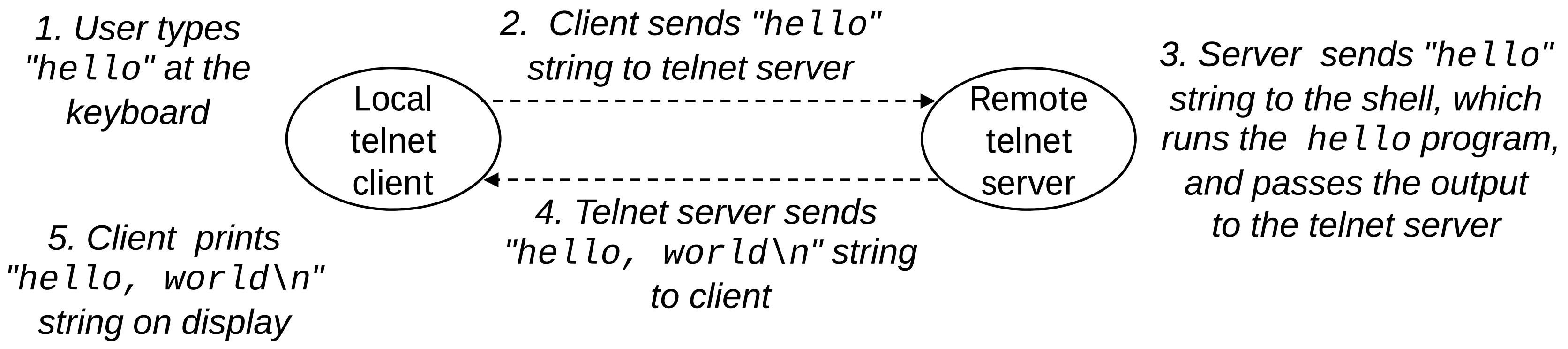
- Can be viewed as just another IO device, in the view of individual system



Networks

Remote execution (by telnet)

- 5 steps



1. User types "`hello`" at the keyboard

2. Client sends "`hello`" string to telnet server

3. Server sends "`hello`" string to the shell, which runs the `hello` program, and passes the output to the telnet server

4. Telnet server sends "`hello, world\n`" string to client

5. Client prints "`hello, world\n`" string on display

Concurrency and Parallelism

Concurrency

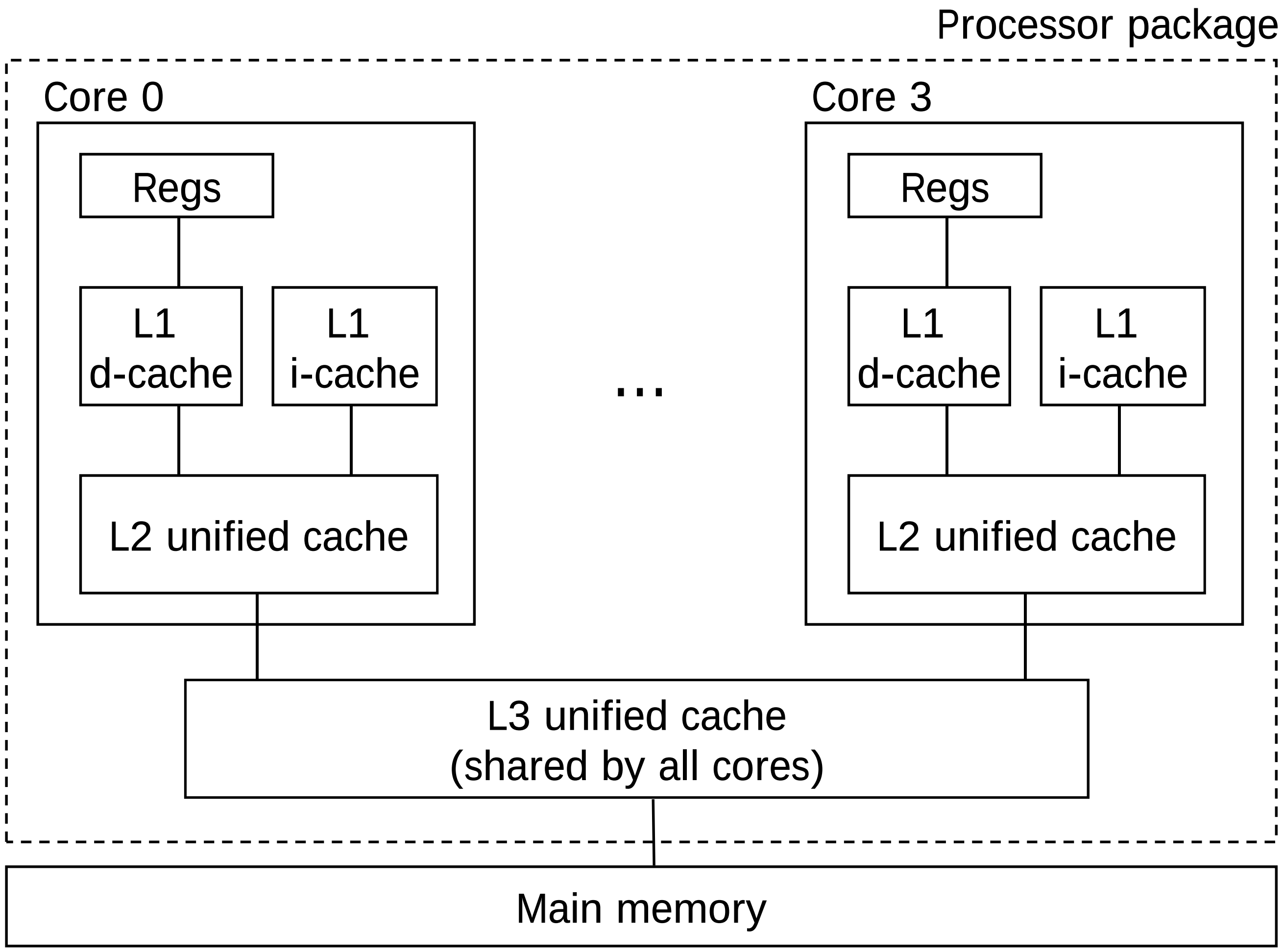
- Multiple processes execute at the same time, using processors alternately ☆
  - Allows multiple users to interact with the system at the same time
  - Allows a single user to engage in multiple tasks concurrently

Parallelism

- Supports multiple processors for the processes
- Multiprocessor systems
  - Multicore processors
  - Hyperthreading

Concurrency and Parallelism

- **Thread-level parallelism: Multicore processors**



Concurrency and Parallelism

Thread-level parallelism: Hyperthreading

- A technique that allows a single CPU to execute multiple flows of control
  - Multiple copies of some of the CPU hardware (PCs and register files)
  - Only single copy of the other parts of the hardware (floating-point arithmetic units)
- Very short thread-switching time
- Intel Core i7
  - Supports 4 cores and 2 threads for each core
  - Can actually execute 8 threads in parallel

Concurrency and Parallelism

Instruction-level parallelism

- Pipelining
- Superscalar processors

SIMD parallelism

- Single-instruction multiple-data architecture
  - Allows a single instruction to cause multiple operations to be performed in parallel
  - In recent Intel/AMD processors
    - ✓ Add 8 single-precision floating-point numbers in parallel

Summary

backward - campatible