COP 3402 Systems Software

OS Intro

Thanks to Euripides Montagne

COP 3402 System Software

1.- Absolute Loader

2.- Bootstrap Loader

3.- Operating System: Loading application programs
- 3.1- The Process concept.
- 3.2- Creating a process.

4.- Relocating Loaders
- 4.1- Relocation bits
- 4.2- Relocation maps (modification records)

COP 3402 System Software

Assembly program

	Label	opcode	address
01		; This is	
02		; a comment	
03	start	.begin	x200
04	here	LOAD	sum
05		ADD	a
06		STORE	sum
07		LOAD	b
08		SUB	one
09		STORE	b
0A		SKIPZ	
0B		JMP	here
0C		LOAD	sum
0D		HALT	
0E	sum	.data	x000
0F	a	.data	x005
10	b	.data	x003
11	one	.data	x001
12		.end	start

object code file

Header	Program name: start Starting address text: x200 Length of text in bytes: x14 Starting address data: x20A Length of data in bytes: 8
Text	0001000000001001 0010000000001001 0011000000000111 0001000000001000 0100000000001000 0011000000000110 1001000000000000 1000111111111000 0001000000000001 0111000000000000
Data	0000000000000000 0000000000000101 0000000000000011 0000000000000001

COP 3402 System Software

Assembler object code file

Program name: start **Starting address text: x200** **Length of text in bytes: x14** **Starting address data: x20A** **Length of data in bytes: 8**	Header
0001000000001001 **0010000000001001** **0011000000000111** **0001000000001000** **0100000000001000** **0011000000000110** **1001000000000000** **1000111111111000** **0001000000000001** **0111000000000000**	Text section
0000000000000000 **0000000000000101** **0000000000000011** **0000000000000001**	Data section

Absolute loader:

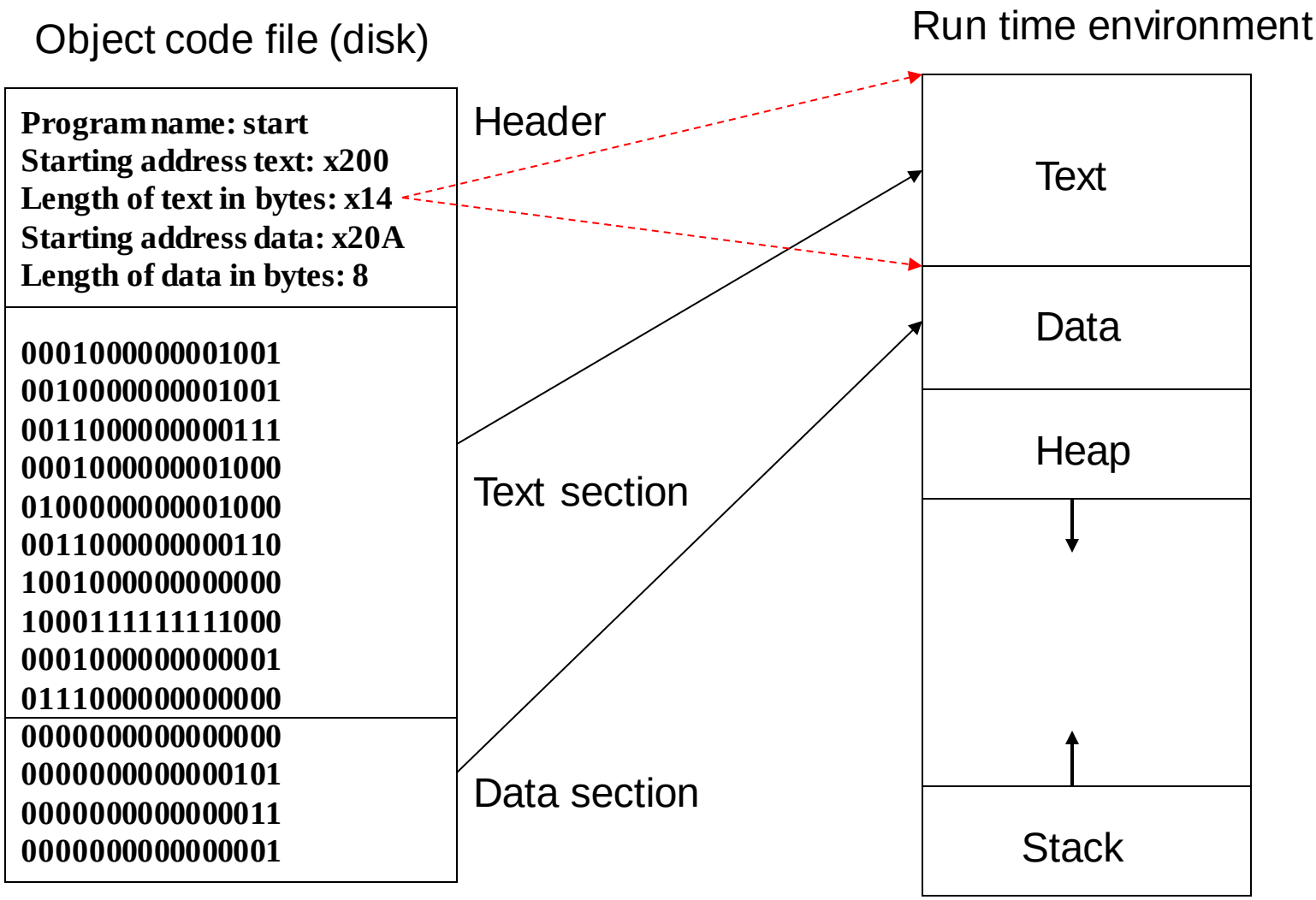
The absolute loader will load the program at memory location x200:

1.- The header record is checked to verify that the correct program has been presented for loading.

2.- Each text record is read and moved to the indicate address in memory

3.- When the "end" record (EOF) is encountered, the loader jumps to the specified address to begin execution.

Loading object code into memory



COP 3402 System Software

Bootstrapping:

Computers execute programs stored in main memory, and initially the operating system is on the hard disk.

When the computer is turned on it does not have an operating system loaded in memory and the hardware alone cannot do the operations of an OS. To solve this paradox a special program called bootstrap loader is created.

Bootstrapping continued…

- This program does not have the full functionality of an operating system, but it is capable of loading into memory a more elaborated software (i.e. loader2) which in its turn will load the operating system.

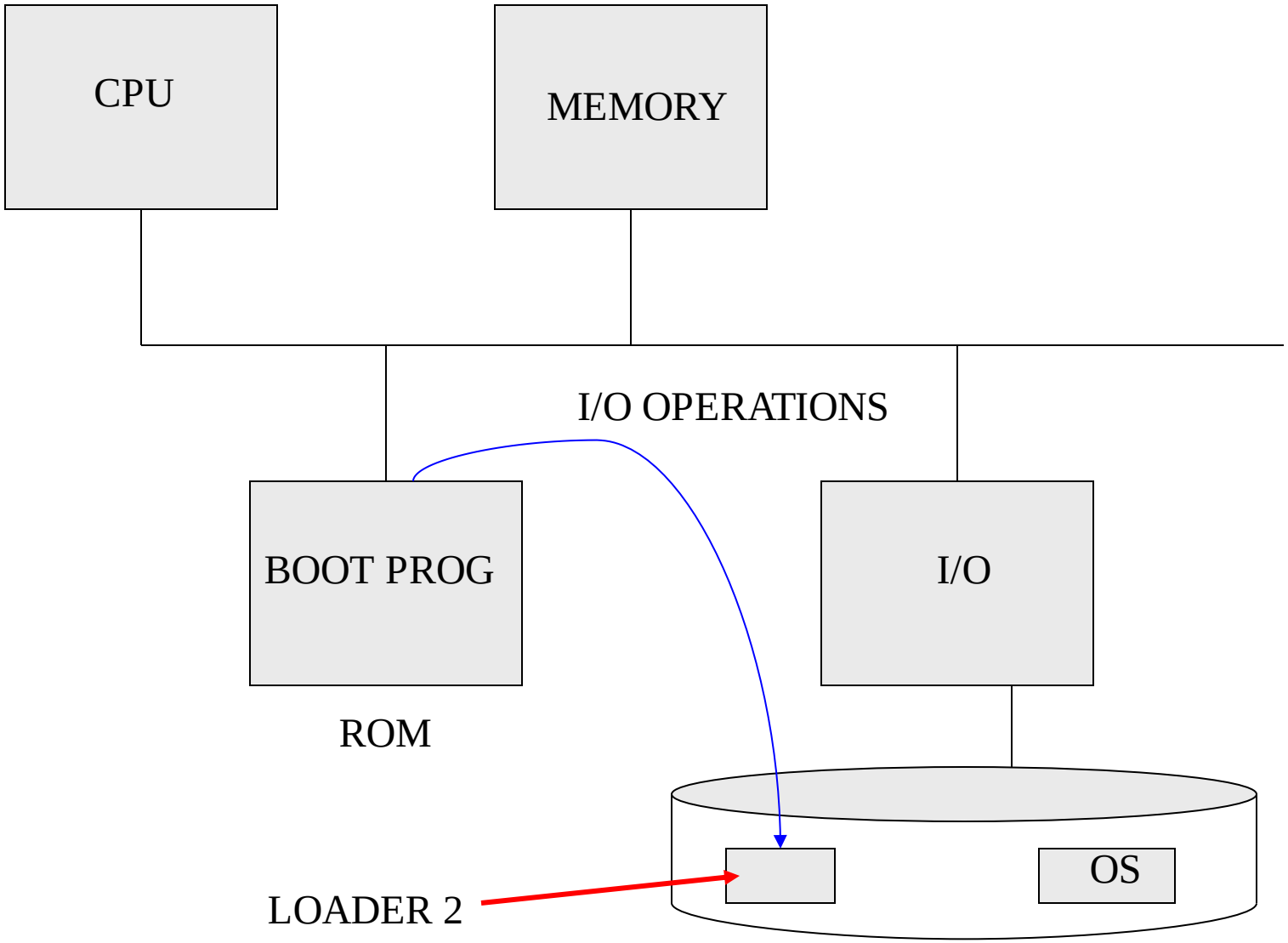
- Once the OS has been loaded the loader transfers the control of the computer system to the operating system.

Bootstrapping continued…

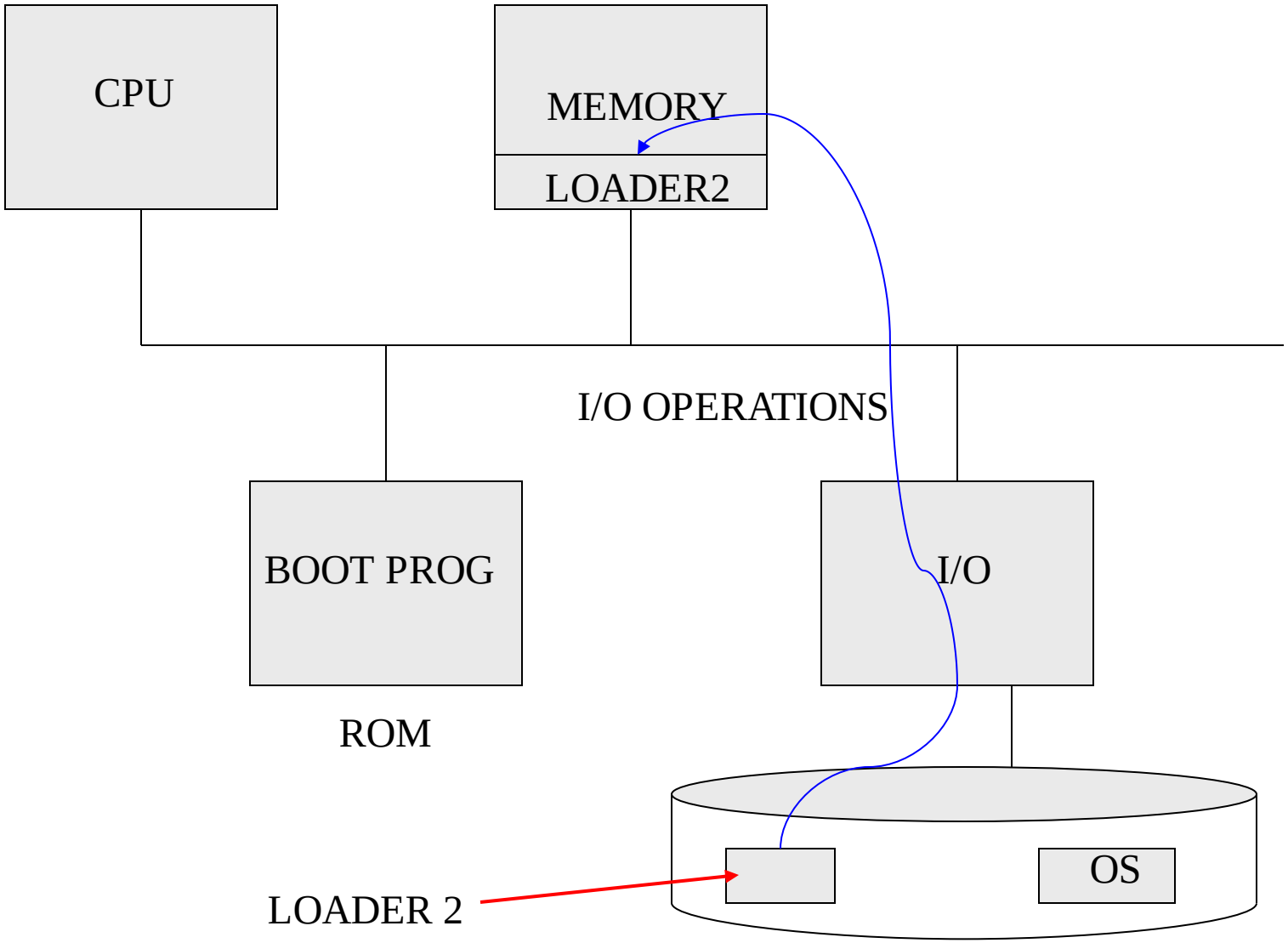
- Early programmable computers had toggle switches on the front panel to allow the operator to place the bootloader into the program store before starting the CPU.

- In modern computers the bootstrapping process begins with the CPU executing software contained in ROM at a predefined address whose elementary functionality is to search for devices eligible to participate in booting, and load a small program from a special section of a device.

Bootstrapping continued…

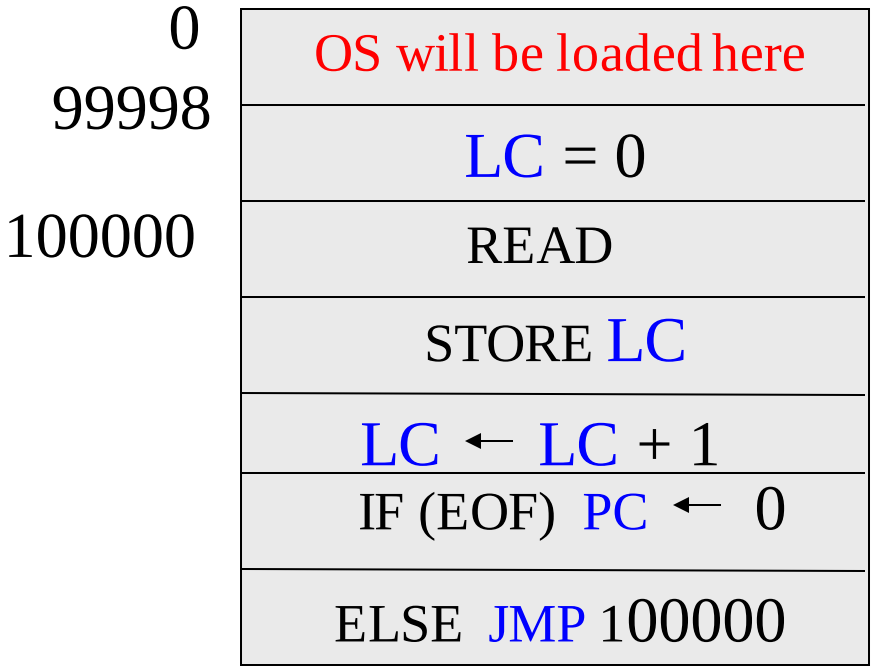


Bootstrapping continued…

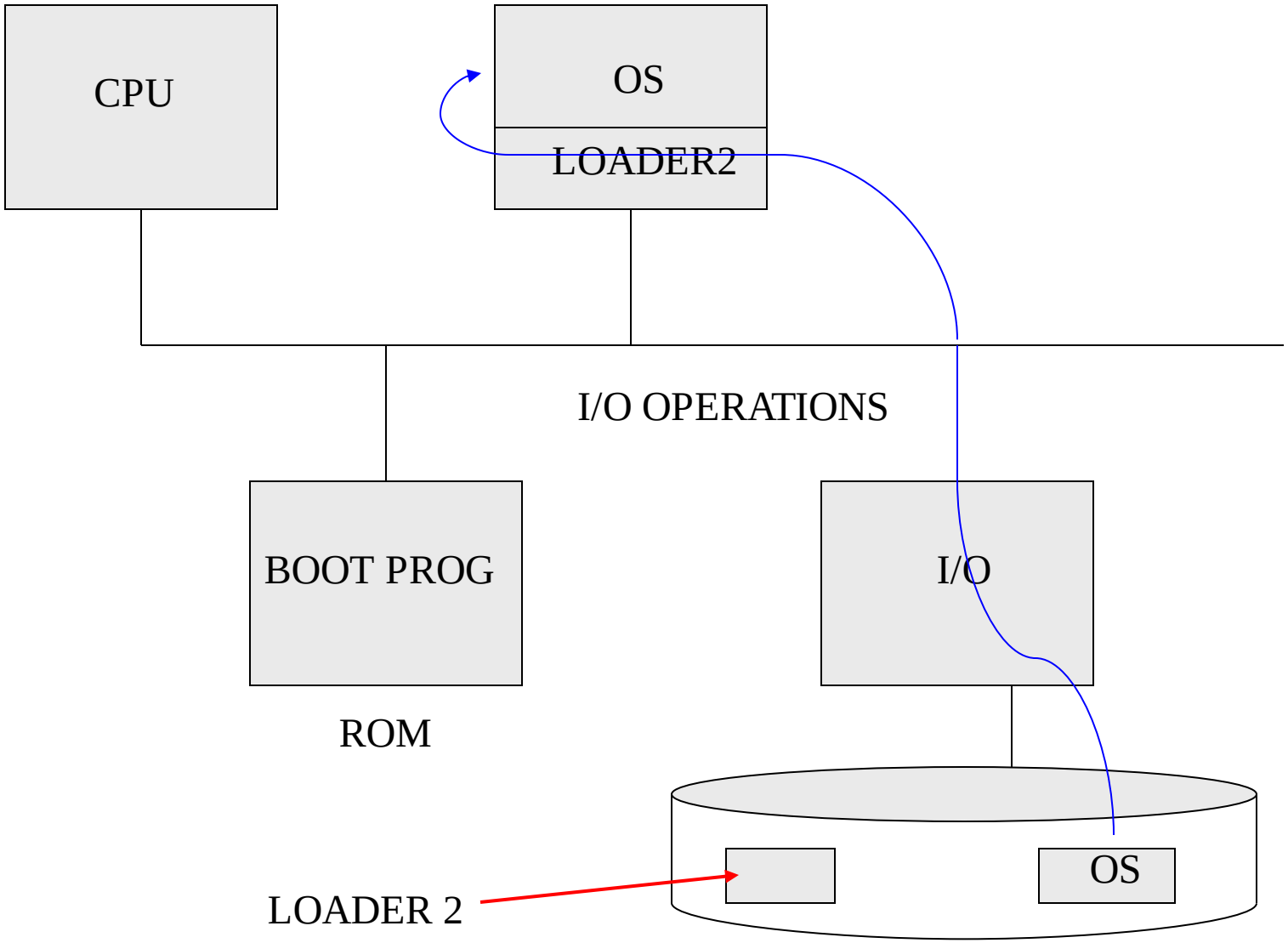


Bootstrapping continued…

- In earlier computers data had to be hand loaded as specified before, but nowadays a small piece of software called loader helps us to avoid the manual loading.



Bootstrapping continued…



Bootstrapping continued…

- The above diagram can be explained in the following steps.

  1. Check hardware

  2. Initiate I/O to load the loader 2 program into memory

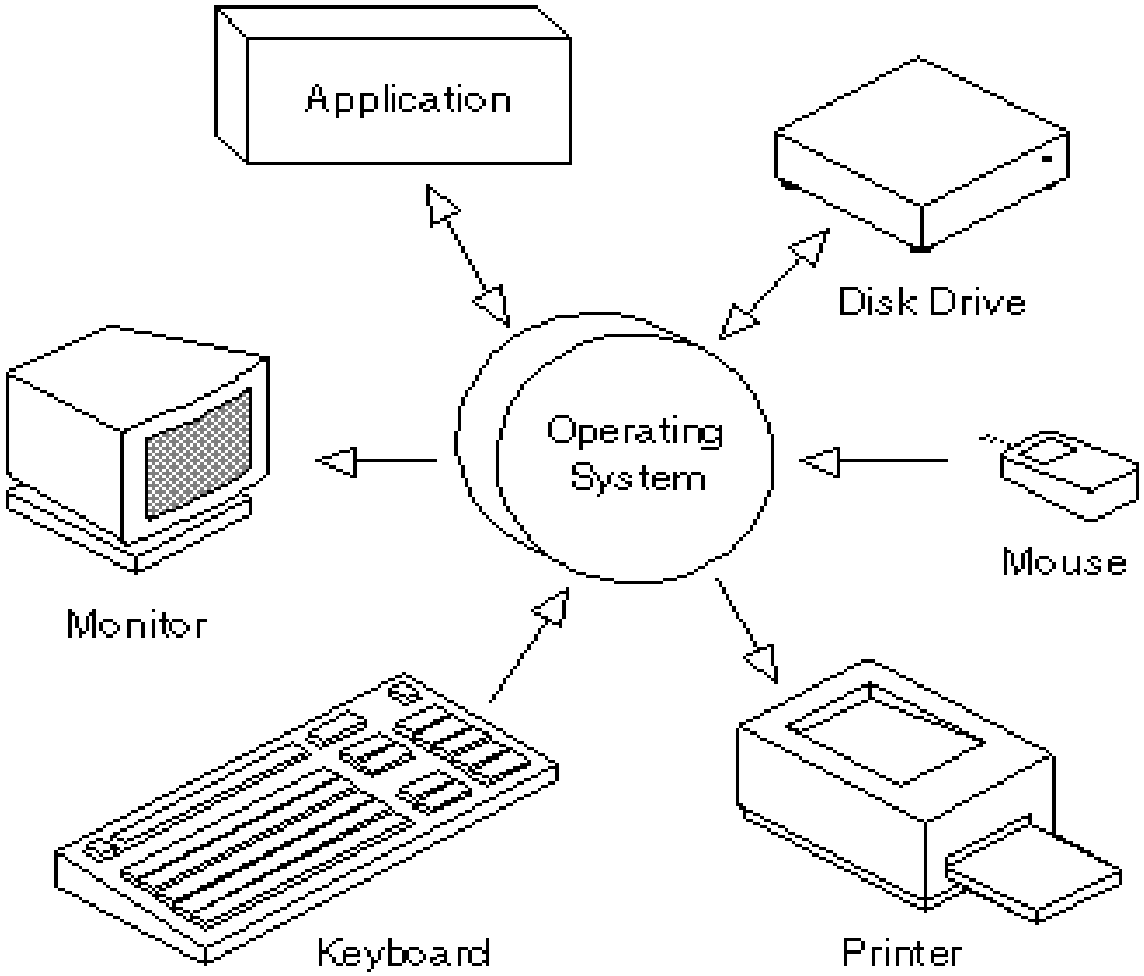
  3. Loader 2 loads the OS and passes control to it

Conclusion

- We have seen that once the OS has control over the system , it can create an environment for programs to run.

- The operating system will load device drivers and other programs that are needed for the normal operation of the computer system.

Operating system



COP 3402 Systems Software

The Process Concept

COP 3402 Systems Software

Programs and processes

Once the operating system takes control of the computer system, an applications program (object module or ELF) can be loaded into memory to be executed.

When the program is loaded into memory a process is created.

What is a process?

COP 3402 System Software

Process

- Definition:
  - A program in execution
  - An asynchronous activity
  - The “locus of control” of a procedure in execution
  - It is manifested by the existence of a process control block (PCB) in the operating system.

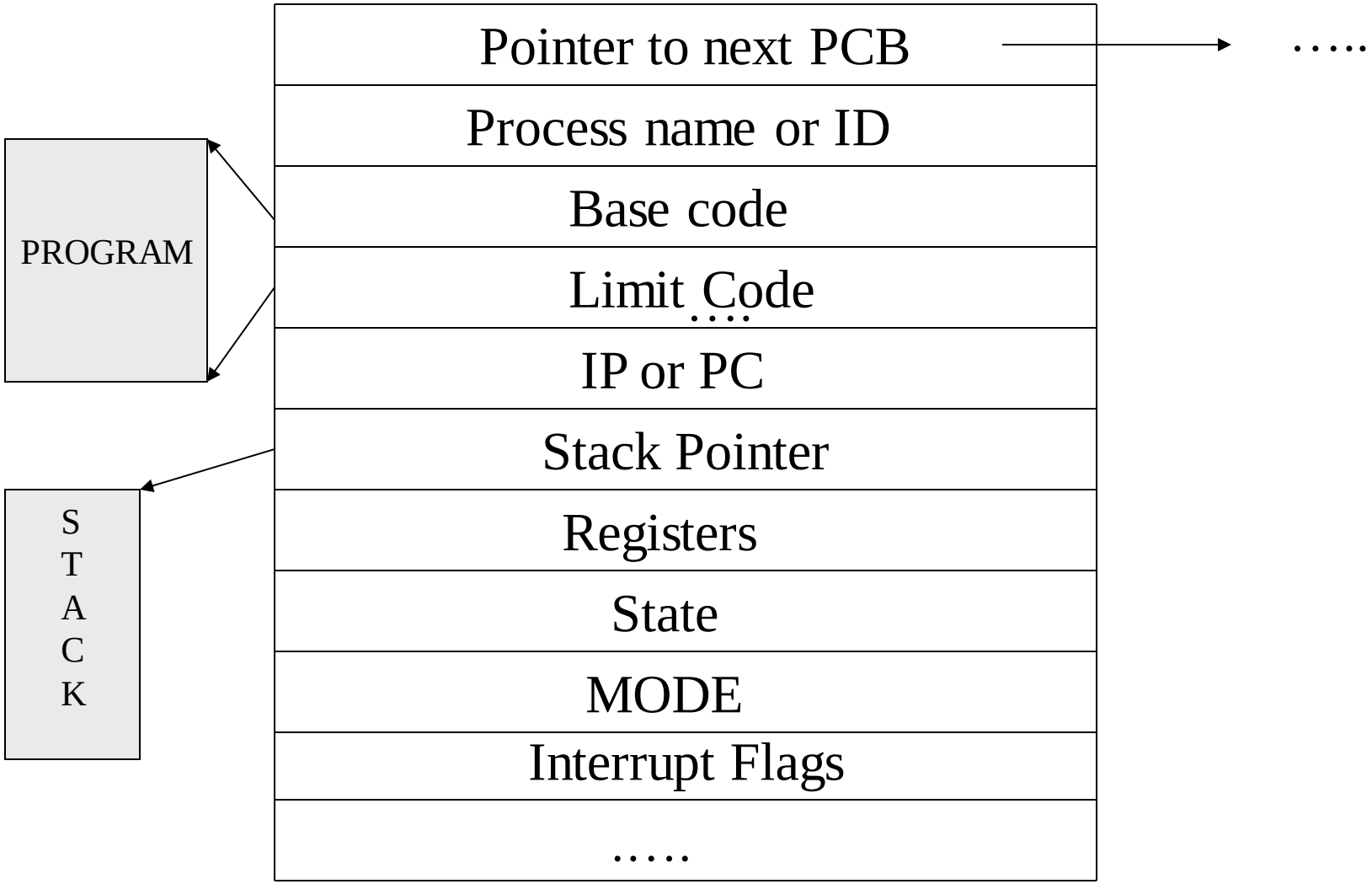
COP 3402 Systems Software

Process Continued…

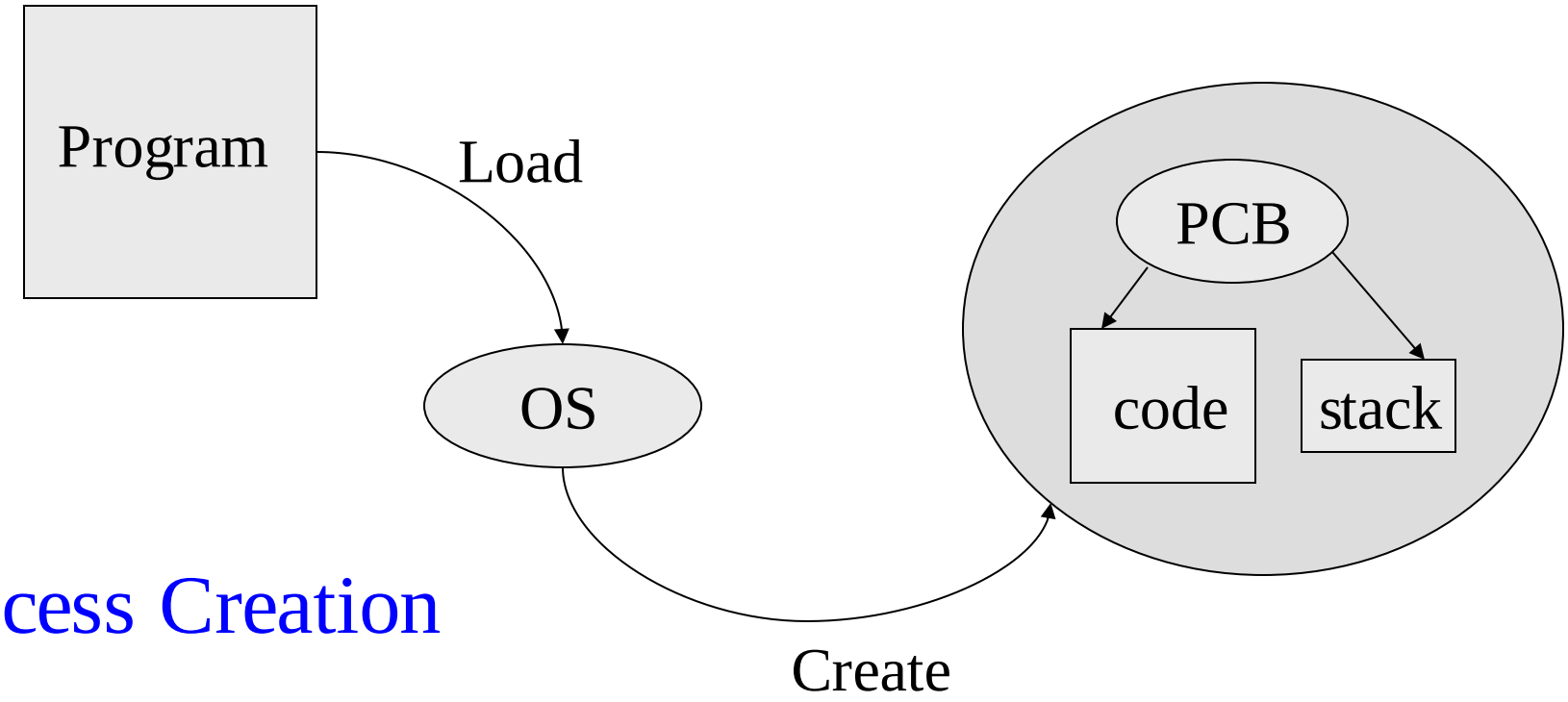
- The activity of a process is controlled by a data structure called **P**rocess **C**ontrol **B**lock(**PCB**).
- A PCB is created every time a program is loaded to be executed.
- So, a process is defined by a PCB-Program couple.

COP 3402 Systems Software

Structure of the PCB

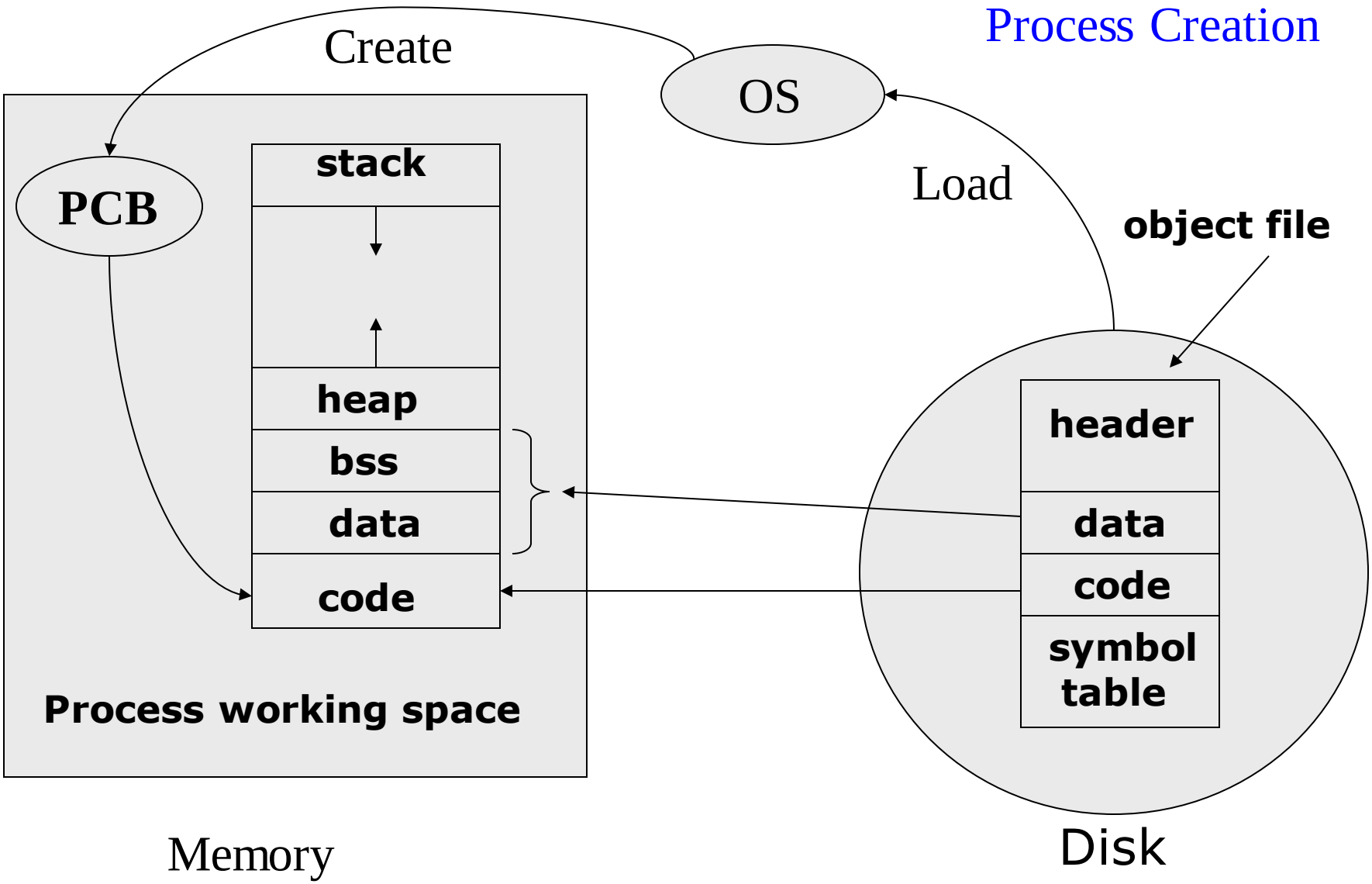


COP 3402 Systems Software



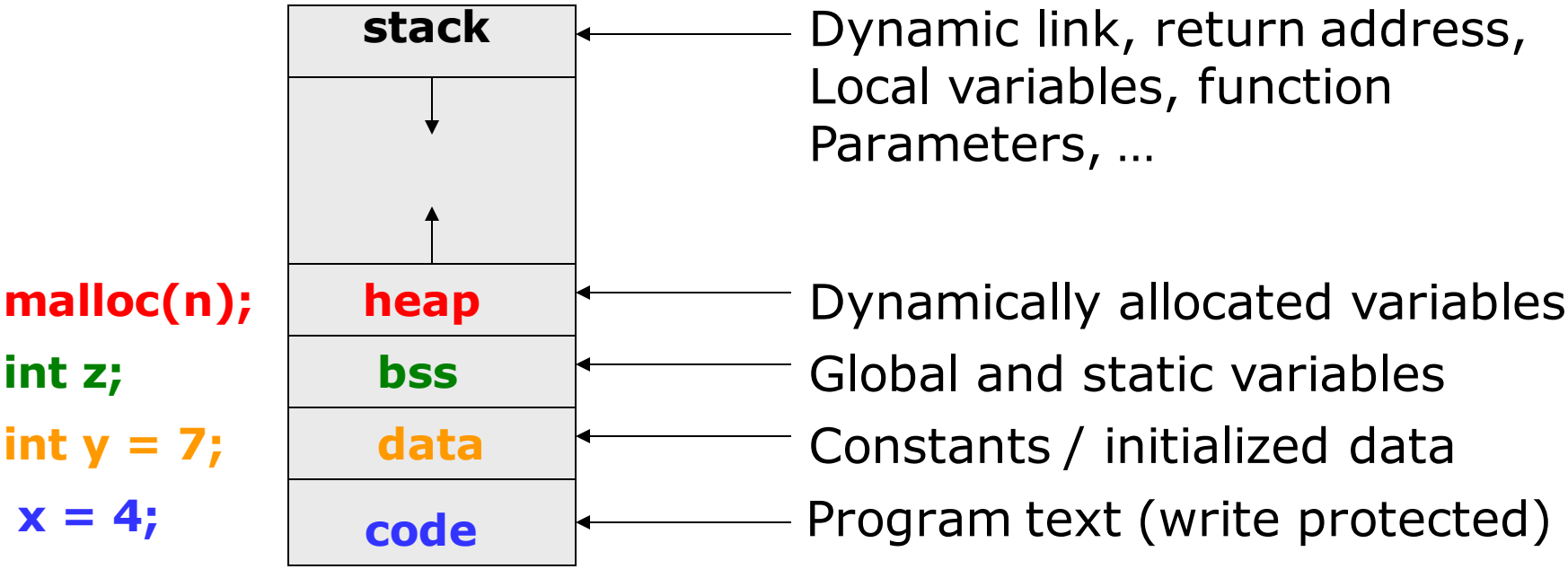
Process Creation

COP 3402 Systems Software



COP 3402 Systems Software

Process working space (run-time environment)

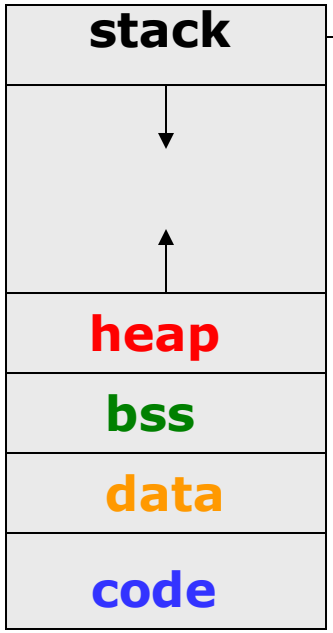


Process working space
or
Run-time environment

COP 3402 Systems Software

Process working space

bss: means "block started by symbol" and has that name for historical reasons.

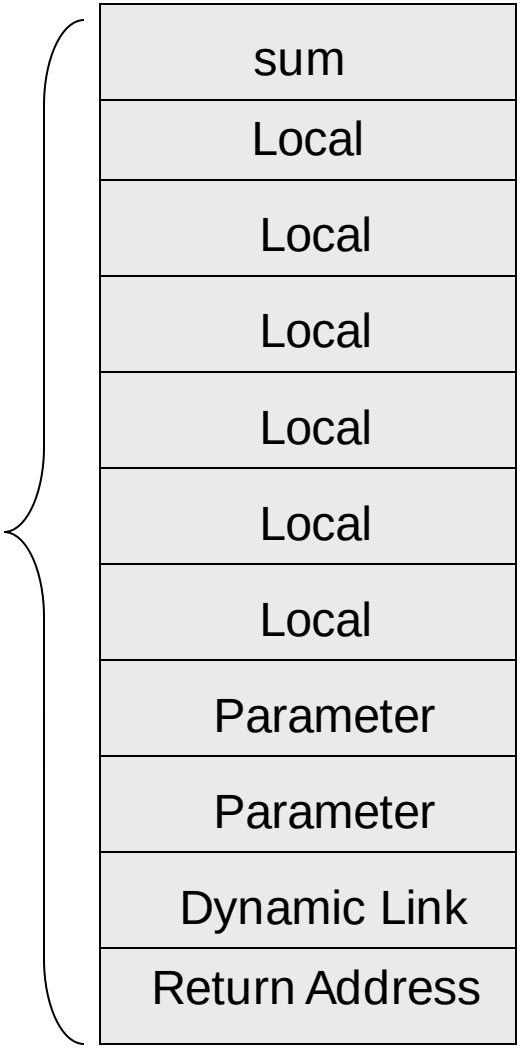


malloc(n);

int z;

int y = 7;

x = 4;




```
void sub(float total, int part ) {
int List[5];
float sum;
...

}
```

COP 3402 Systems Software

Relocating Loaders

Absolute loaders loads a program on a specific memory location but it is often desirable to have two or more programs residing in memory sharing the different resources of a computer system.

It would be impractical to assign starting addresses to each program to plan program execution.

A loader able to load a program into memory wherever there is room for it is called a relocating loader.

COP 3402 Systems Software

Relocation bits

Assemblers generate code that starts at address zero but it can also emit with each line of text (code) relocation bits indicating what fields in the object code must be modified when the program is loaded in an address different from zero.

For example, if the program will be loaded at address 40, a relocation bits equal to "1" indicates what part of the instruction must be modified:

Loc#	Len	reloc	text		Loc#	text
00	3	011	13 33 35	→	40	13 73 75

COP 3402 System Software

source program

	Label	opcode	address	address
00		copy	zero	older
03		copy	one	old
06		read	limit	
08		write	old	
10	comp	load	older	
12		add	old	
14		store	new	
16		sub	limit	
18		brpos	finalL	
20		write	new	
22		copy	old	older
25		copy	new	old
28		br	comp	
30	final	write	limit	
32		stop		
33	zero	CONST	0	
34	one	CONST	1	
35	older	SPACE		
36	old	SPACE		
37	new	SPACE		
38	limit	SPACE		

before relocation

Loc#	Len	reloc	text
00	3	011	13 33 35
03	3	011	13 34 36
06	2	01	12 38
08	2	01	08 36
10	2	01	03 35
12	2	01	02 36
14	2	01	07 37
16	2	01	06 38
18	2	01	01 30
20	2	01	08 37
22	3	011	13 36 35
25	3	011	13 37 36
28	2	01	00 10
30	2	01	08 38
32	1	0	11
33	1	0	00
34	1	0	01
35			
36			
37			
38			

after relocation

Loc#	text
40	13 73 75
43	13 74 76
46	12 78
48	08 76
50	03 75
52	02 76
54	07 77
56	06 78
58	01 70
60	08 77
62	13 76 75
65	13 77 76
68	00 50
70	08 78
72	11
73	00
74	01
75	
76	
77	
78	

Relocation constant to be added is 40

COP 3402 Systems Software

2.- Relocation maps (modification records)

Interleaving relocation bits with the program text makes cumbersome the process of loading the text directly into memory.

This problem can be resolve by collecting all relocation bits into a single contiguous relocation map that we will call the relocation section of the object code file (ELF).

The relocation section will be appended to the text and data sections.

The header will contain the entry point and length of the relocation section in the object module.

COP 3402 System Software