CSE3241: Operating System and System Programming

Class-18

Sangeeta Biswas, Ph.D.
Assistant Professor
Dept. of Computer Science and Engineering (CSE)
Faculty of Engineering
University of Rajshahi (RU)
Rajshahi-6205, Bangladesh
E-mail: sangeeta.cse@ru.ac.bd / sangeeta.cse.ru@gmail.com

December 4, 2017

A Simple C Program

- What we simply do:
  - $ gedit **helloWorld.c**
  - $ gcc **helloWorld.c**
    $ ./**a.out**
    OR
  - $ gcc **helloWorld.c** -o **helloWorld**
    $ ./**helloWorld**

helloWorld.c

```
/* First C Program
   By: EtaSetaMix
   Date: 1.1.1953
*/
#include<stdio.h>

#define MSG "Hello World!!!"

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

Output

```
Hello World!!!
```

Workers Working Behind

- What we simply do:
  - $ gcc **helloWorld.c**
    $ ./**a.out**

    OR

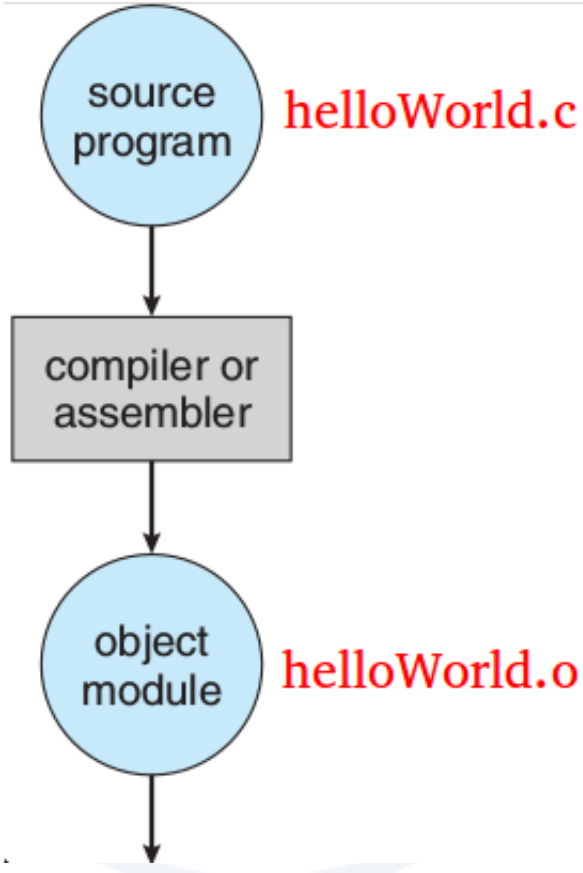
  - $ gcc **helloWorld.c** -o **helloWorld**
    $ ./**helloWorld**

- Who work behind:
  1. **Preprocessor**
  2. **Compiler**
  3. **Assembler**
  4. **Linker / Lingkage Editor**
  5. **Loader**

Journey of a C Program

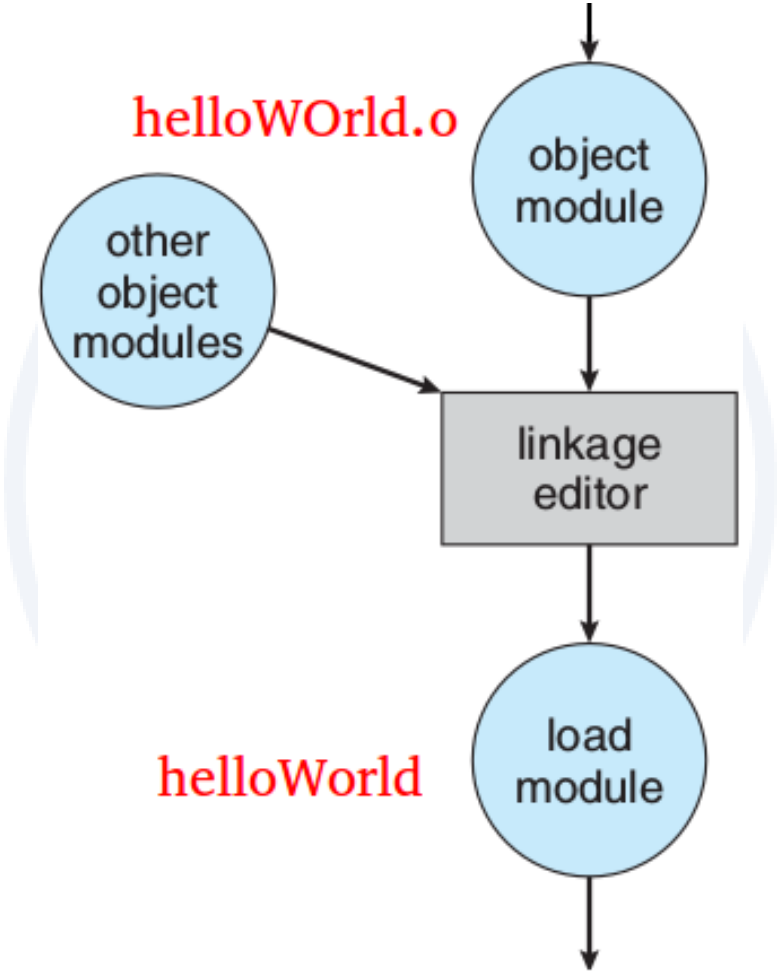
- **Step-1: Pre-Processing**
  - pre-processed code is generated by removing comments, including contents of header files and expanding macros.
- **Step-2: Compilation**
  - assemply code is generated.
- **Step-3: Assembly**
  - object file with relocatable address is generated.
- **Step-4: Linking**
  - Complete (almost complte) executable code is generated.
- **Step-5: Loading**
  - binary memory image is generated and loaded into memory.
- **Step-6: Execution**
  - process runs.

From Source Code to Object Code



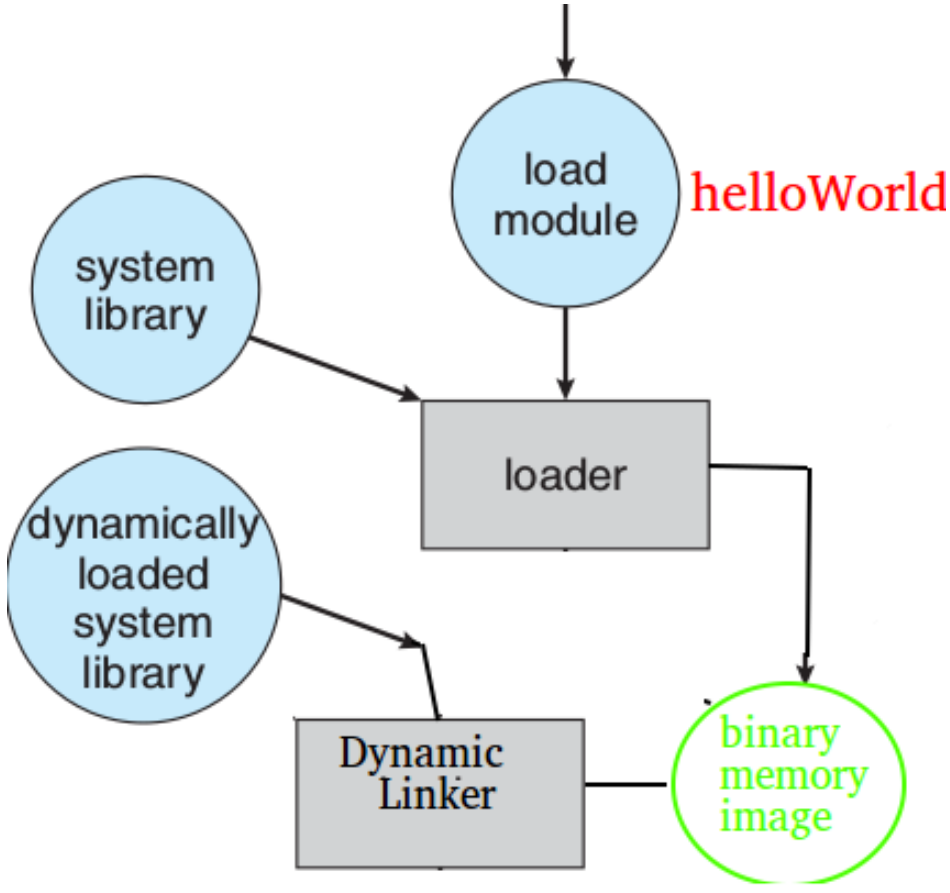
- $ gcc -E helloWorld.c -o helloWorld.i [Preprocessing]
- $ gcc -S helloWorld.i -o helloWorld.s [Compilation]
- $ as helloWorld.s -o helloWorld.o [Assembly]

From Object Code to Executable Code



- $ gcc helloWorld.o -o helloWorld [Linking]

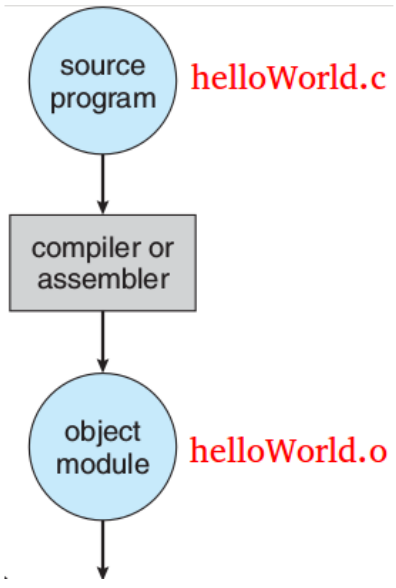
From Executable Code to Binary Memory Image



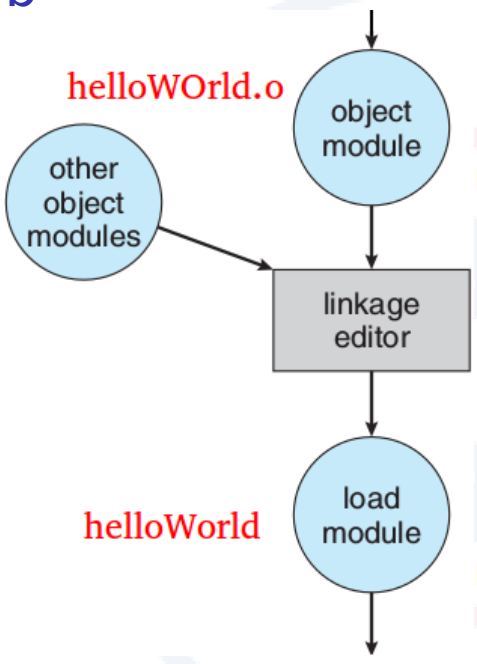
- $ ./helloWorld [Dynamic Linking & Loading]

Multistep Processing of a User Program

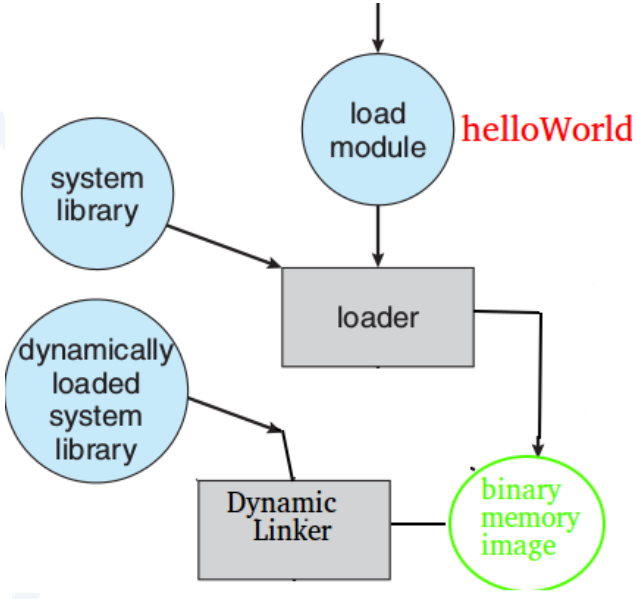
a



b



c




```
$ cpp helloWorld.c >helloWorld.i
$ gcc -S helloWorld.i -o helloWorld.s
$ as helloWorld.s -o helloWorld.o
$ gcc helloWorld.o -o helloWorld
$ ./helloWorld
```

Linker and Loader

- **Linker** is a part of compiler tool which:
  - takes one or more objects generated by a compiler
  - links them to static library functions and
  - makes executable programs.

- **Loader** is a part of OS which:
  - loads executable code and standard libraries into memory.
  - prepares them for execution.

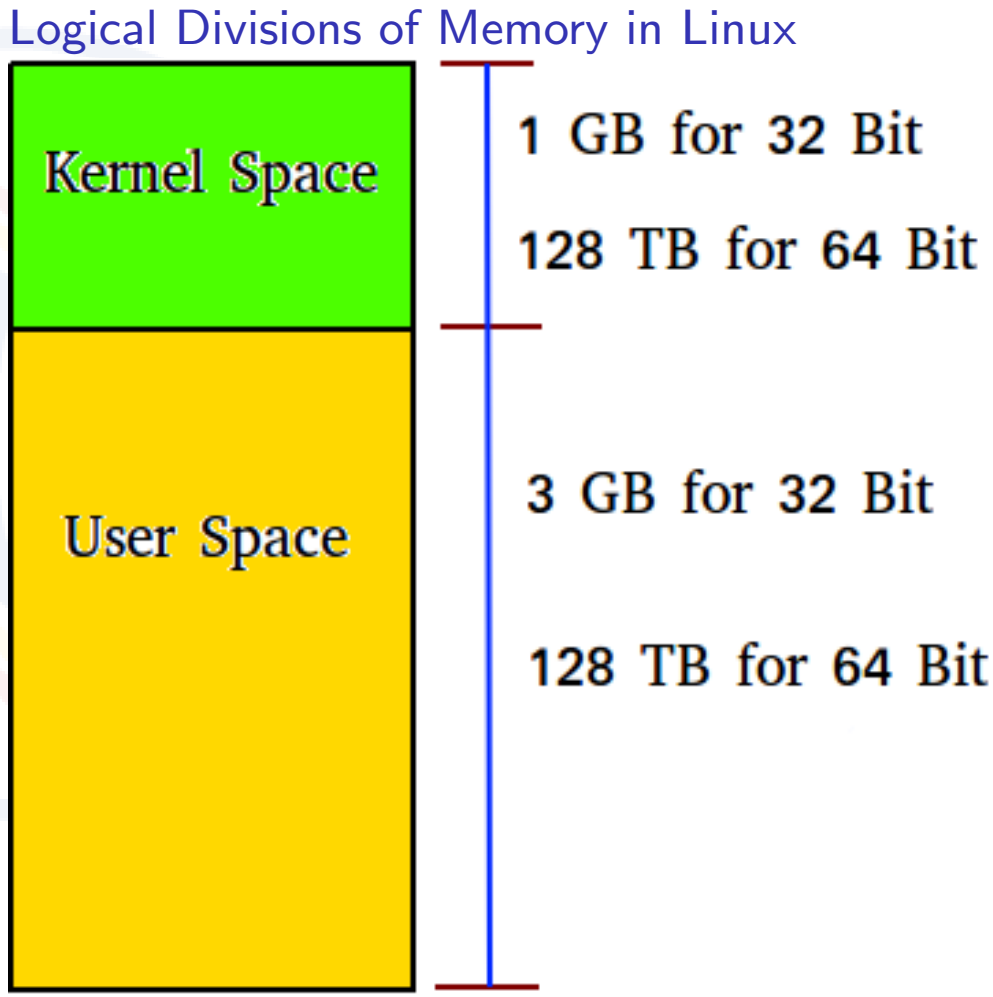
- **Dynamic linker** is a special part of OS, which.
  - loads external shared libraries into a running process and
  - then binds those shared libraries dynamically to the running process.

Division of Memory Space

- To protect OS from user interference, memory (RAM) is divided into two spaces:
  - Kernel Space
  - User Space

- Kernel space is where the OS processes execute and provide their services.

- User space is where user processes execute.



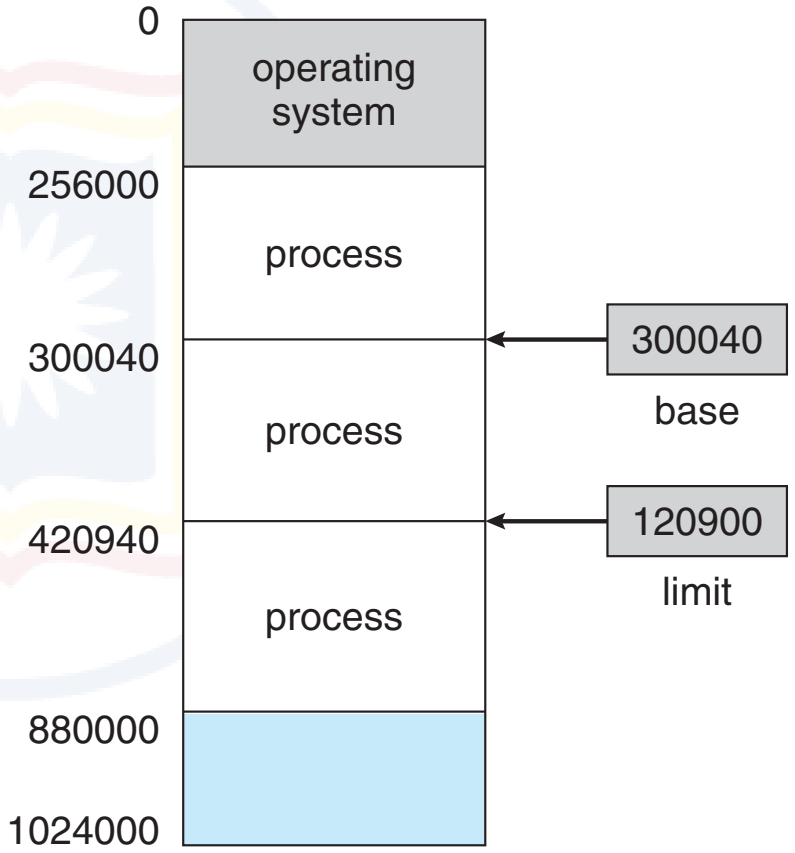
Hardware Support for Memory Divisions

- To protect user processes from each others, **user processes are kept in separate memory spaces** by the help of two registers:
  1. Base Register
  2. Limit Register

- **Base Register** holds the smallest legal physical memory address.

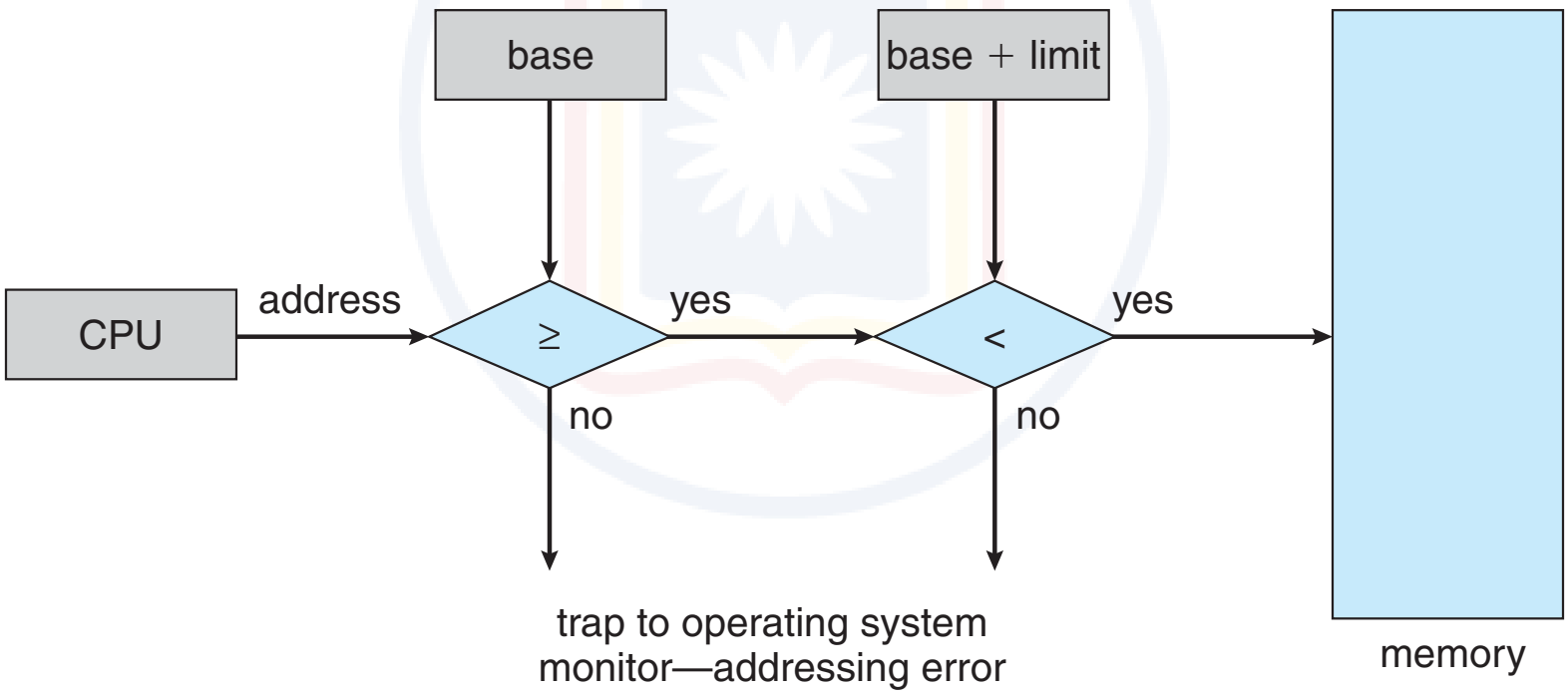
- **Limit Register** contains the size of the range.

Memory Access by Base and Limit Registers [1]



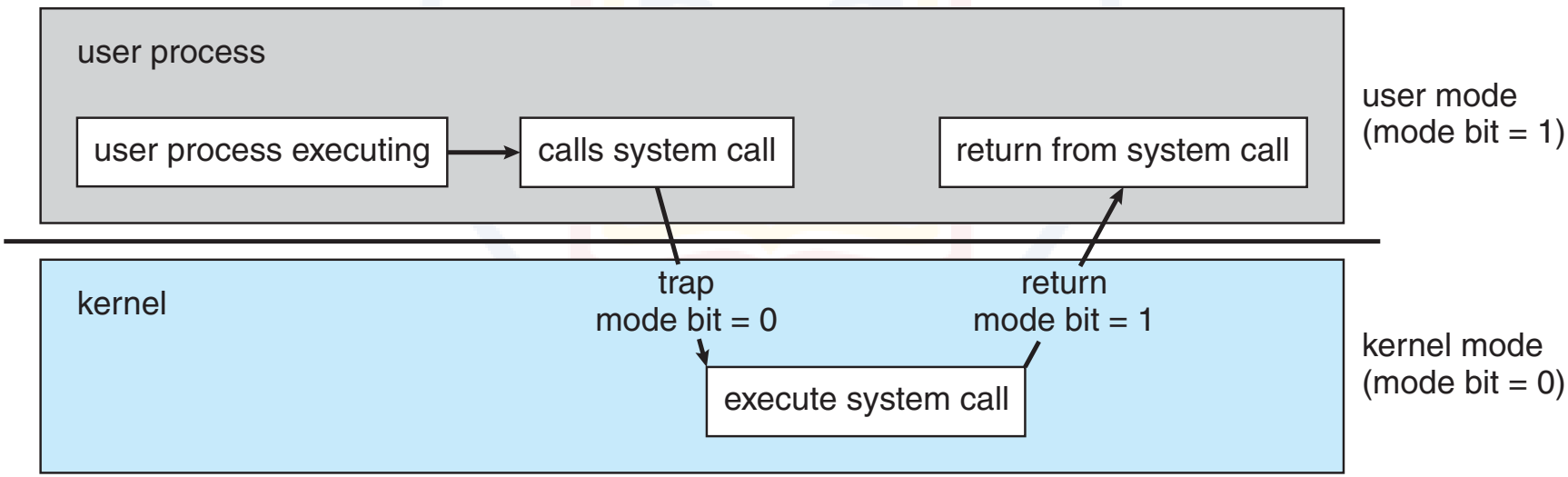
Hardware Address Protection [1]

- Any attempt by a process executing in user mode to access OS memory or other users' memory results in a trap.
- Base and limit registers can be loaded by only the OS in the kernel mode.



Dual-Mode Operation [1]

- **User Mode**: restricted mode.
- **Kernel Mode**: supervisor mode / system mode / monitor mode / privileged mode/ monitor mode/ unrestricted mode.



User Mode Vs. Kernel Mode

- At boot time, the hardware starts at kernel mode.
- OS is then loaded and does its necessary tasks in kernel mode.
- After that OS starts user processes in user mode.
- Whenever a trap or interrupts occurs:
  - the hardware switches from user mode to kernel mode and
  - OS gains control.
- OS always switches to user mode before passing control to the user process.
- These two modes,known as **mode bits**, are controled by 1/2 bits of a special register.

Types of Addresses

- Logical Address: An address generated by the CPU.
- Physical Address: An address seen by the memory unit.
- Logical address can be called as:
  - virtual address.
  - relocatable address.
- Memory Management Unit (MMU):
  - It is a hardware.
  - It does run-time mapping from virtual to physical address

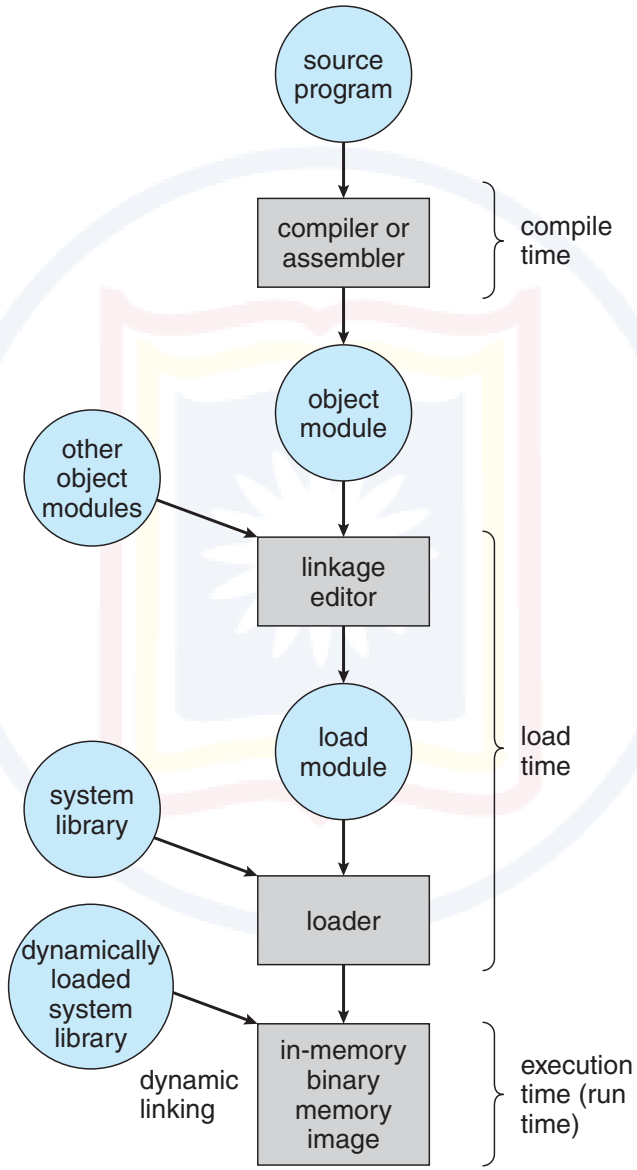
Address Binding

- Address binding is a mapping from one address space to another address apace.

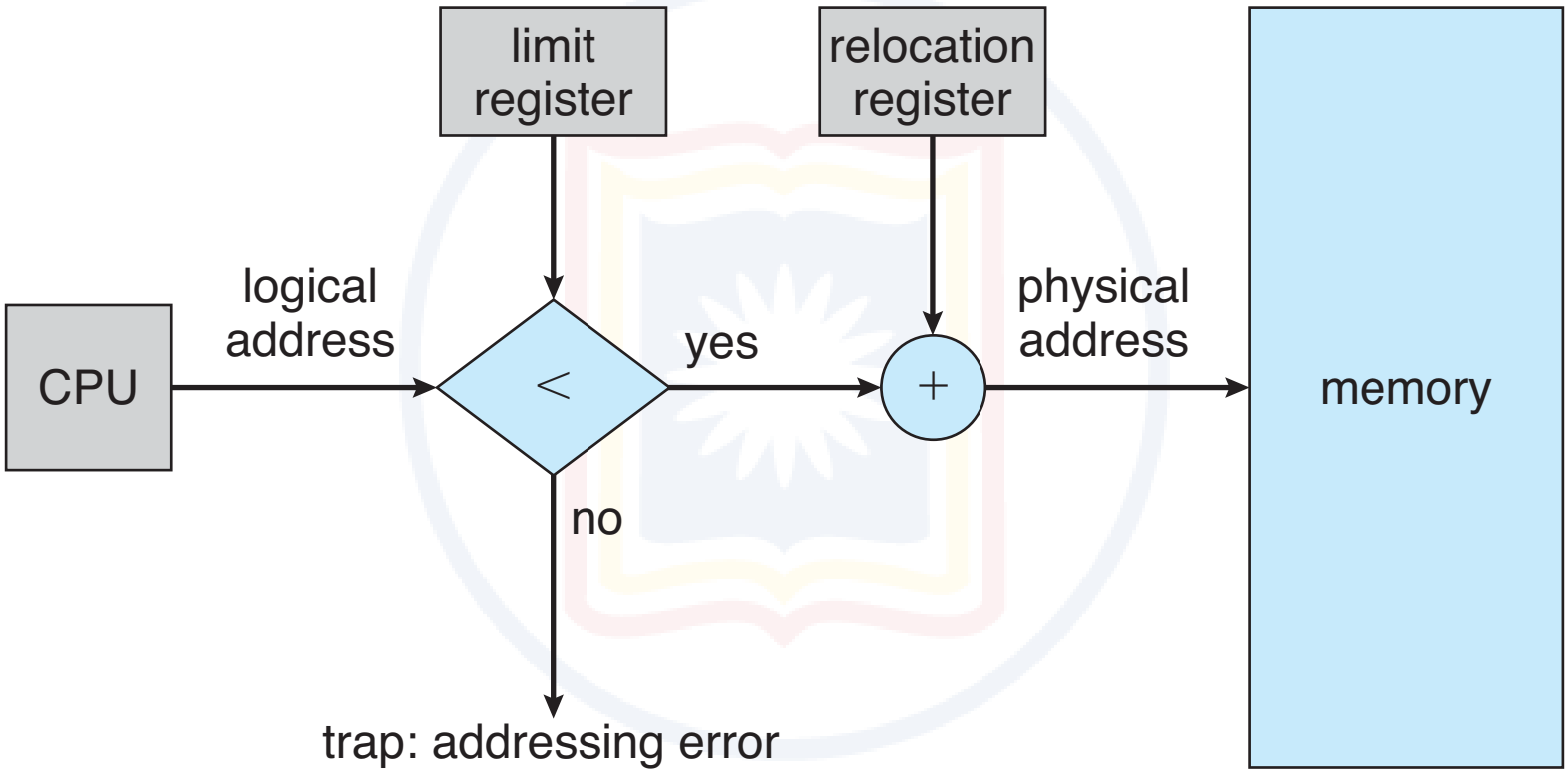
- Address binding can happen for:
  - instructions
  - data

- Address binding can happen at:
  - Compile Time
  - Load Time
  - Execution Time

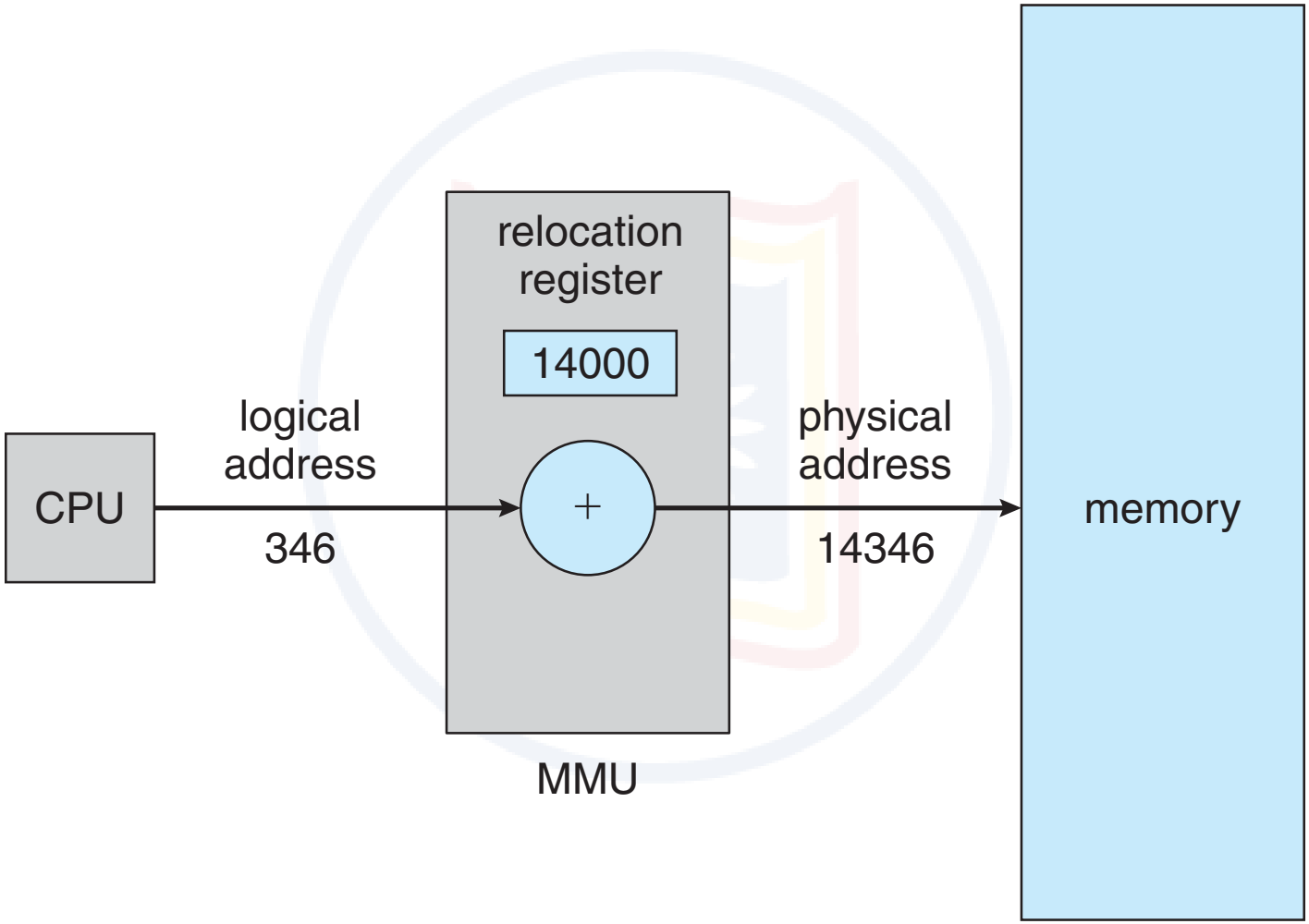
Multistep Binding of a User Program [1]



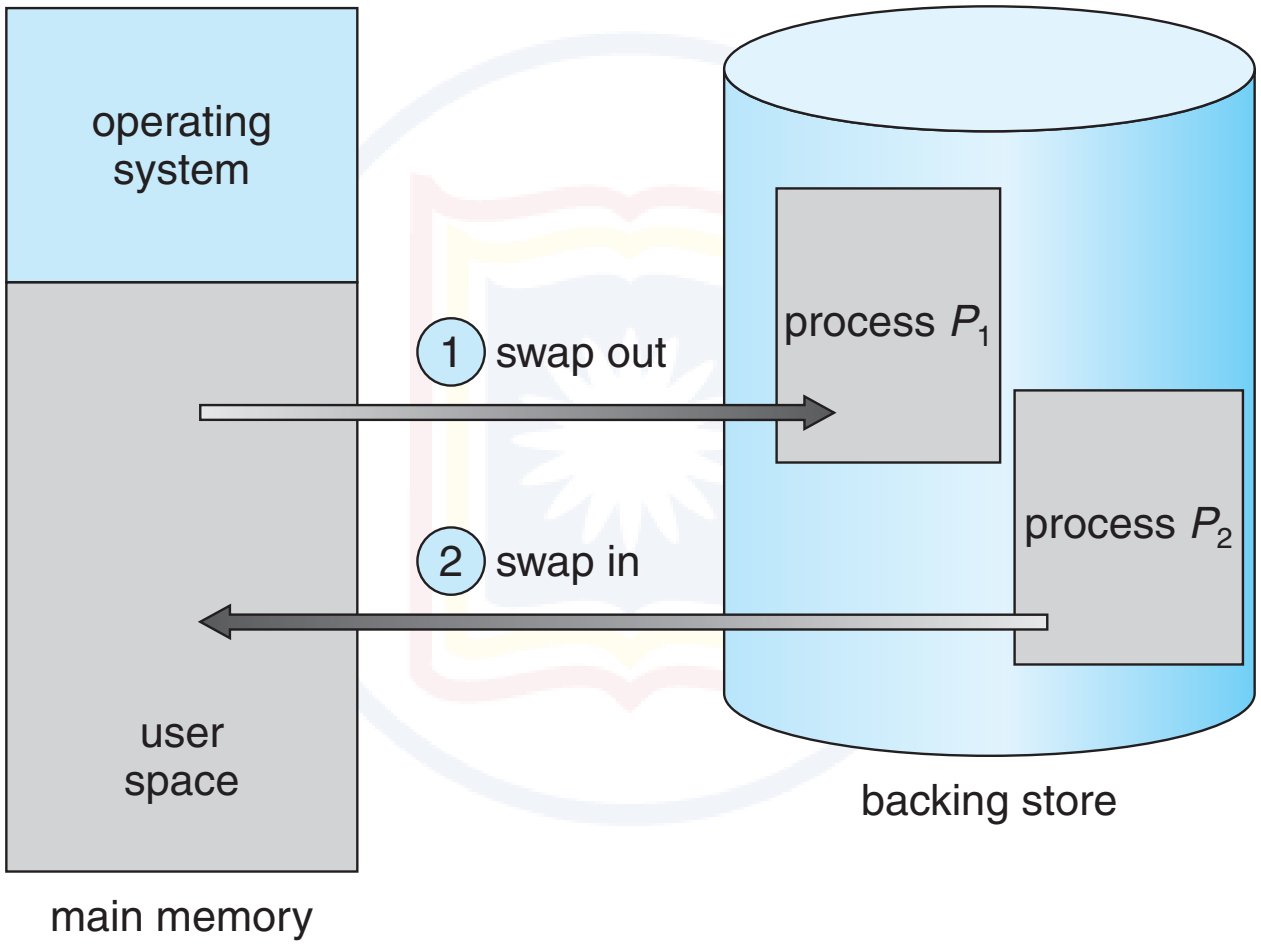
Dynamic Relocation using a Relocation Register [1]



Example of Dynamic Relocation [1]



Swapping of Two Processes [1]



References


P. B. Galvin A. Silbeschatz and G. Gagne.
Operating System Concepts.
John Wiley & Sons, 9 edition, 2012.