Operating Systems

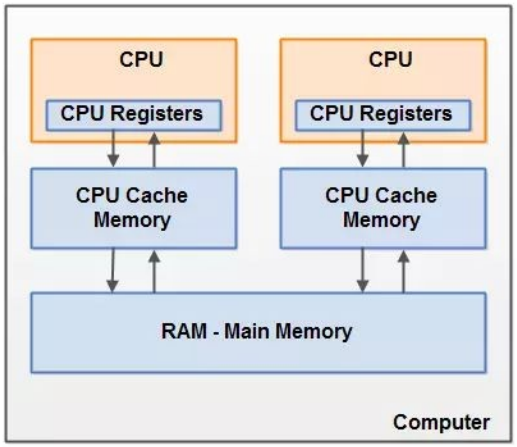
Isfahan University of Technology
Electrical and Computer Engineering Department
1400-1 semester

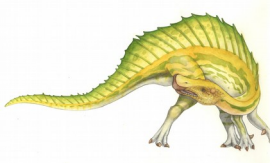
Zeinab Zali

Session 21: Main Memory: logical Addresses, Segmentation

Background

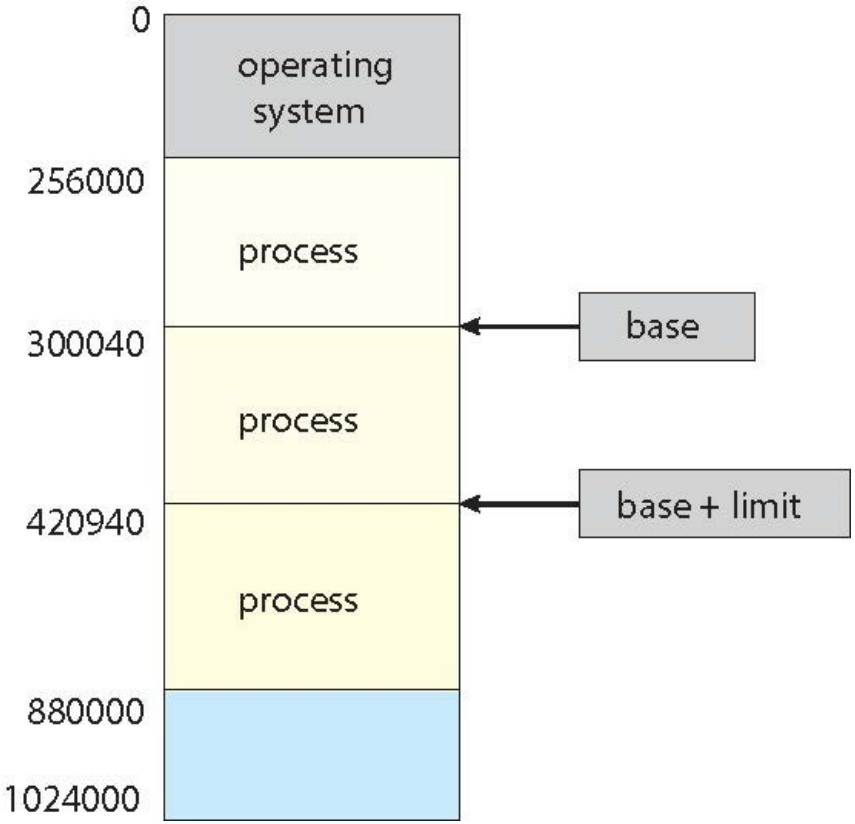
- Program must be brought (from disk) into memory and placed within a process for it to be run
- **Main memory** and **registers** are only storage CPU can access directly
- Memory unit only sees a stream of addresses + read requests, or address + data and write requests
- Register access in one CPU clock (or less)
- But Main memory can take many cycles, causing a **stall**
- **Cache** sits between main memory and CPU registers
- Protection of memory required to ensure correct operation



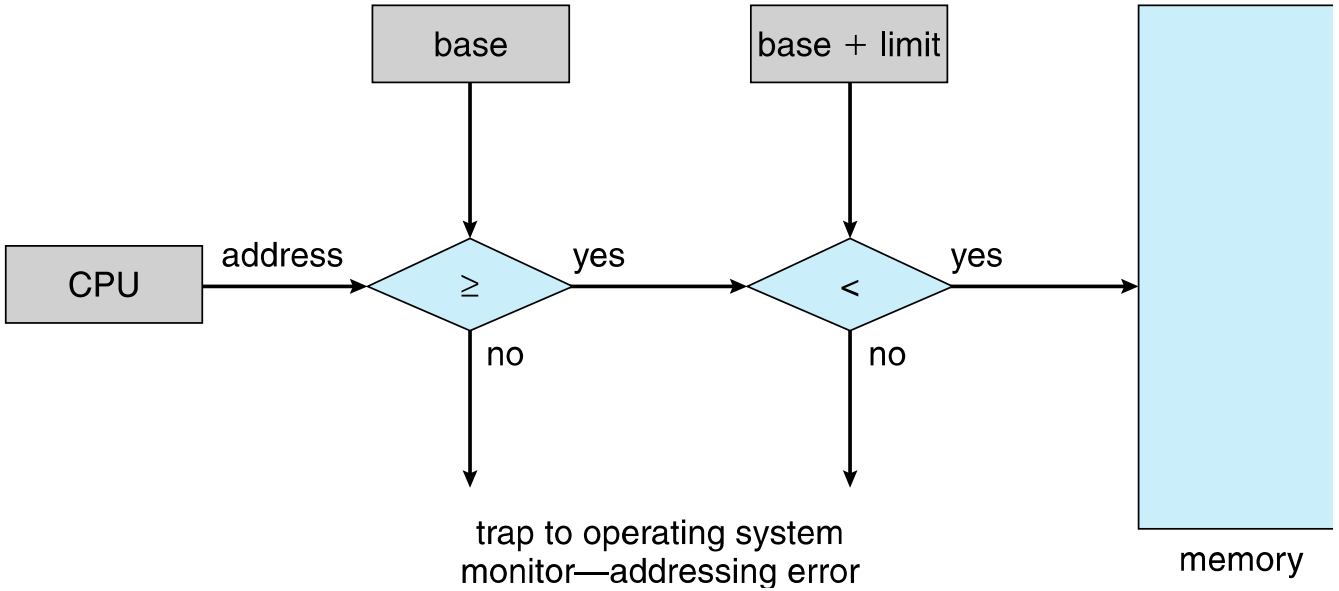


Base and Limit Registers

- A pair of **base** and **limit registers** define the logical address space
- CPU must check **every memory access generated in user mode** to be sure it is between base and limit for that user



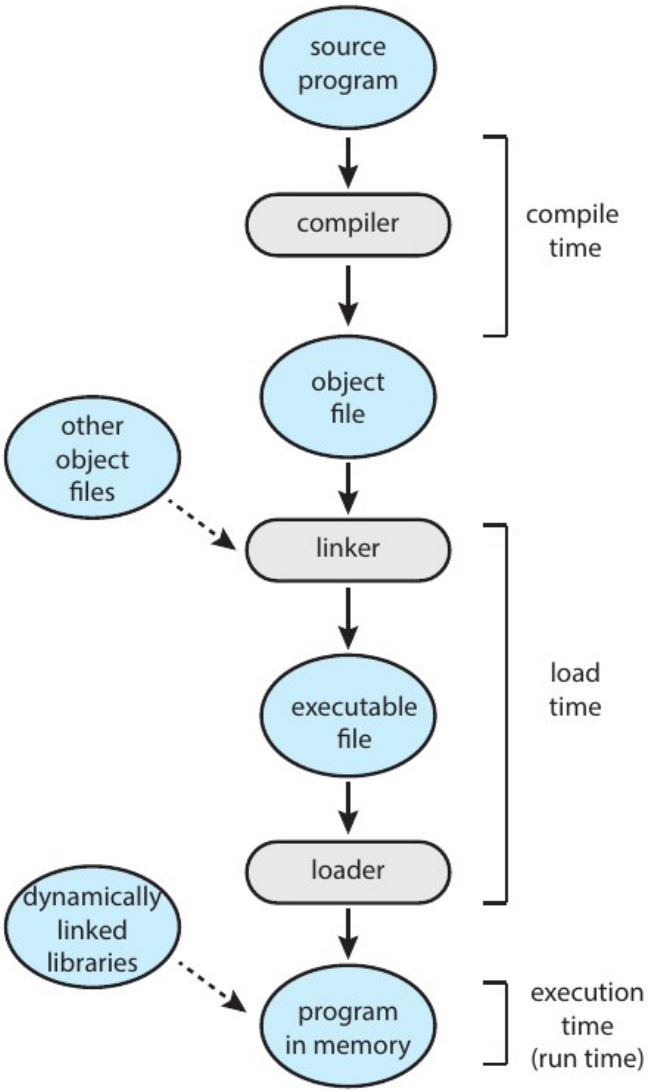
Hardware Address Protection



Address representation

- Address are represented in different ways during different steps
  - **Source program:** addresses are symbolic like variables
  - **Compiler**: binds symbolic addresses to **relocatable**
    - Ex: 14 bytes from the beginning of this module
  - **Loader**: binds relocatable addresses to **absolute,** Ex. 74014

Multistep processing of a user program

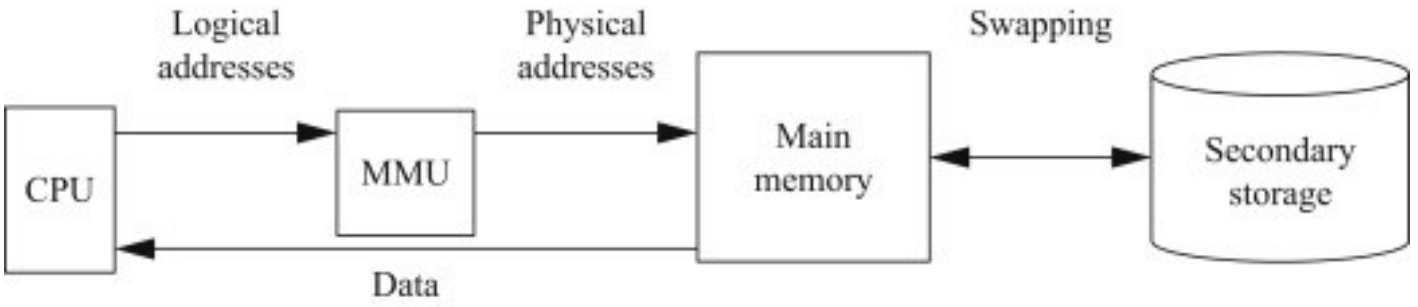


Binding of Instructions and Data to Memory

- Address binding of instructions and data to memory addresses can happen at different steps
  - **Compile time:** if at some later time the starting location changes, then recompiling is necessary
  - **Load time**: binding relocatable addresses (from compile time) to absolute addresses
  - **Execution time**: Binding delayed until run time if the process can be moved during its execution from one memory segment to another
    - Need hardware support for address maps (e.g., base and limit registers)

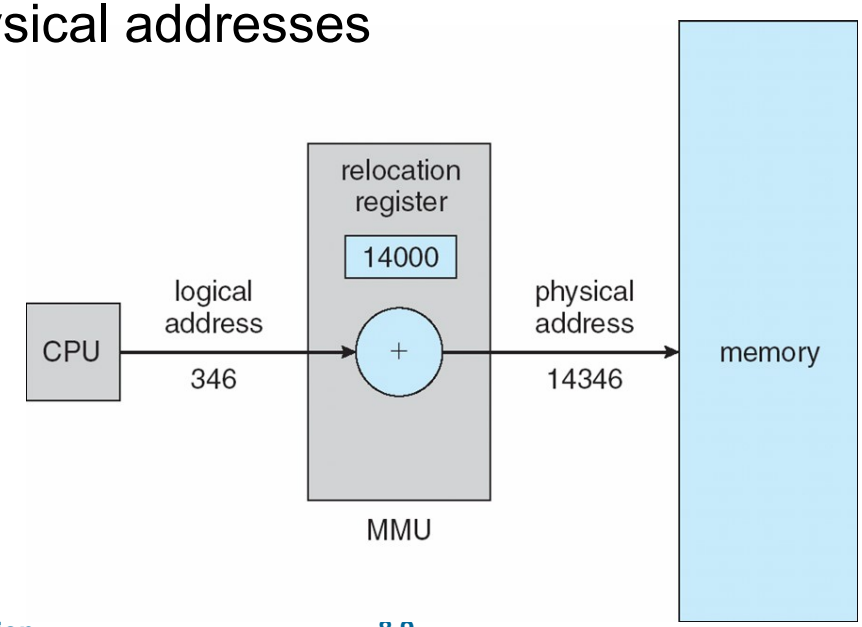
Logical vs. Physical Address Space

- **Logical address** – generated by the CPU; also referred to as **virtual address**
- **Physical address** – address seen by the memory unit
- **Logical address space** is the set of all logical addresses generated by a program
- **Physical address space** is the set of all physical addresses corresponding to the program logical addresses

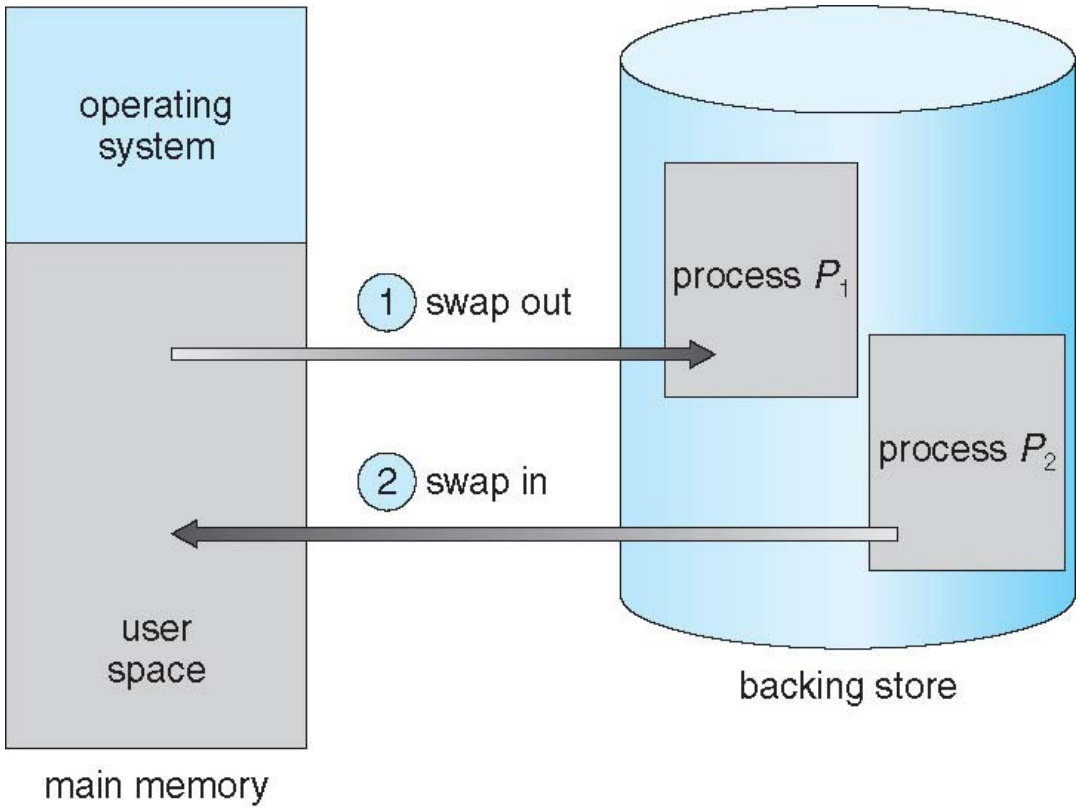


Memory-Management Unit (MMU)

- Hardware device that at run time maps virtual to physical address
  - Usually a part of CPU
- The value in the relocation register is added to every address generated by a user process at the time it is sent to memory
  - Base register now called **relocation register**
- The user program deals with *logical* addresses; it never sees the *real* physical addresses



Schematic View of Swapping



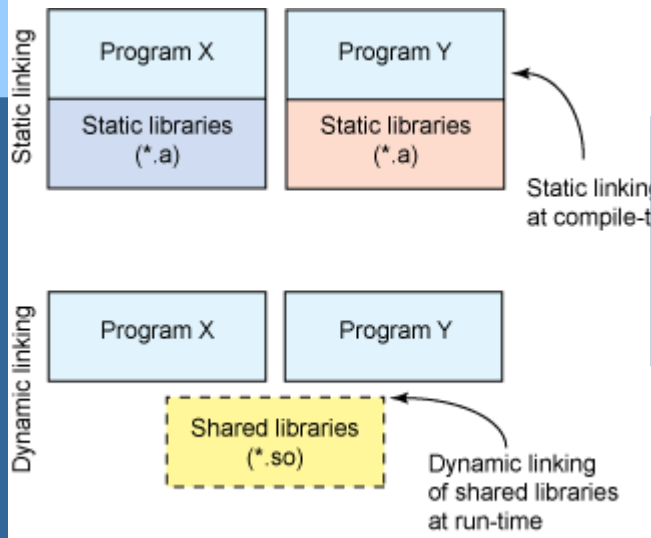
Dynamic Loading

- Until now we assumed that the entire program and data has to be in main memory to execute
- **Dynamic loading** allows a routine (module) to be loaded into memory only when it is called (used)
- Results in better memory-space utilization; an unused routine is never loaded
- All routines kept on disk in relocatable load format
- Useful when large amounts of code are needed to handle infrequently occurring cases (e.g., exception handling)
- No special support from the operating system is required
  - It is the responsibility of the users to design their programs to take advantage of such a method
  - OS can help by providing libraries to implement dynamic loading

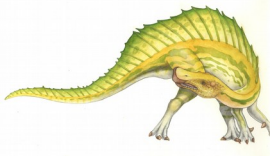
```
Void * hndl dlopen("libname.so", RTLD_NOW);
Void * lib_func = dlsym(hndl, "func_name");
```

Dynamic Linking

- **Dynamically linked libraries (DLL)** – system libraries that are linked to user programs when the programs are run.
  - Similar to dynamic loading. But, linking rather than loading is postponed until execution time
- Operating system checks if routine is in processes' memory address
  - If not in address space, add to address space
- Dynamic linking is particularly useful for libraries
- System also known as **shared libraries**

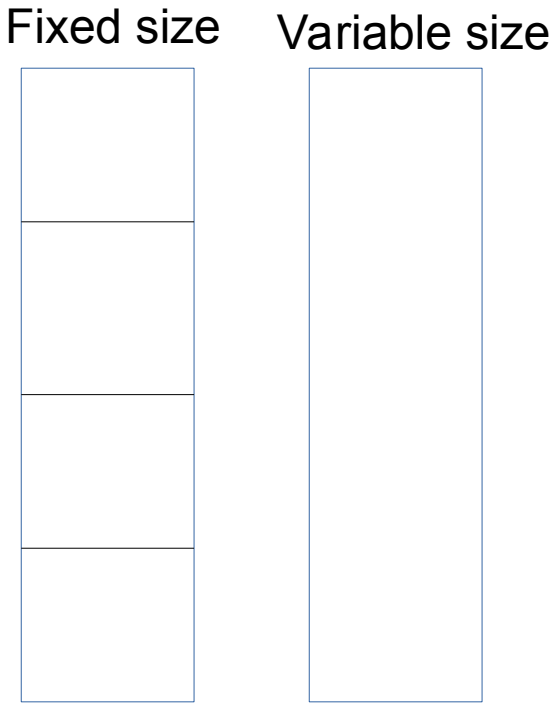



```
include library headers in your code.
In compile time, introduce the dynamic library
g++ sampleCode.c -ldynamicLibraryName
```



Memory allocation for process

- Contiguous Allocation
  - Fixed size partitions
    - Causes internal fragmentation
  - Variable size partitions
    - Causes external fragmentation
- **Segmentation**
  - Variable size segments
- **Paging**
  - Fixed size pages



Fragmentation

- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used

Solution to Fragmentation problem

- Reduce external fragmentation by **compaction**
  - Shuffle memory contents to place all free memory together in one large block
  - Compaction is possible *only* if relocation is dynamic, and is done at execution time

Segmentation

- Memory-management scheme that supports user view of memory
- A program is a collection of segments
  - A segment is a logical unit such as:

    main program

    procedure

    function

    method

    object

    local variables, global variables

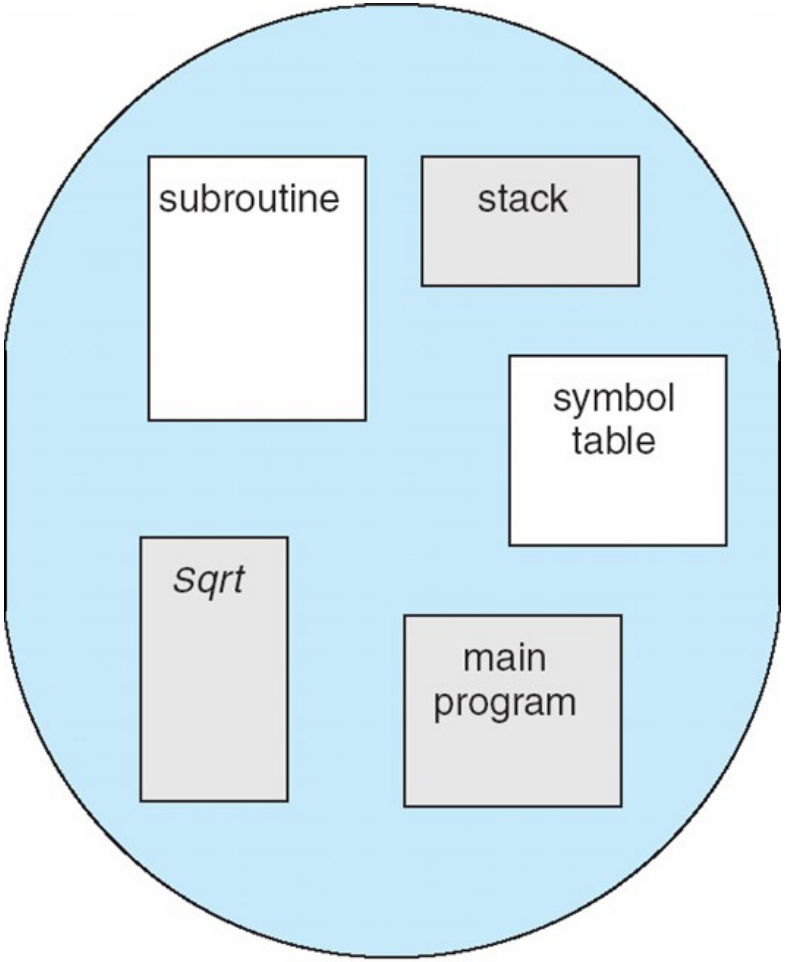
    common block

    stack

    symbol table

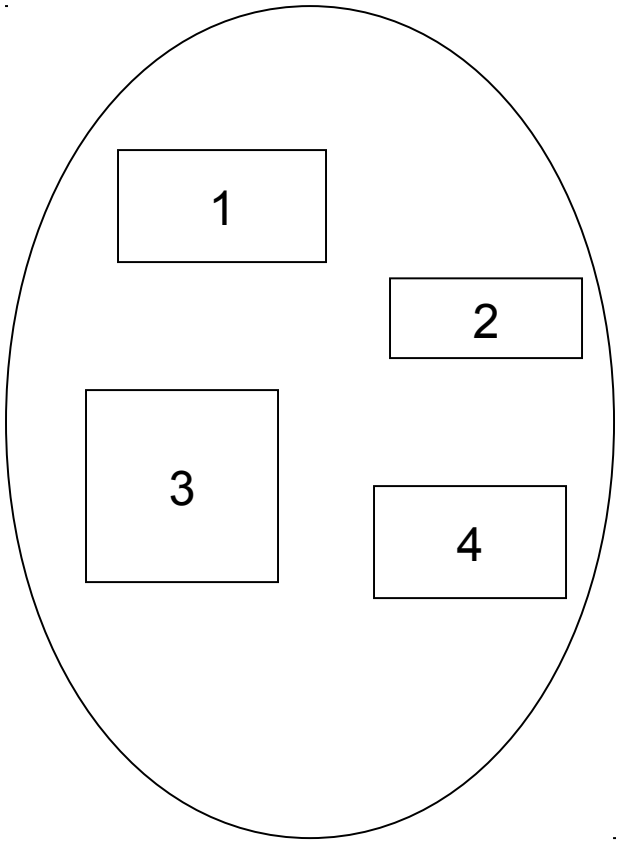
    arrays

User's View of a Program

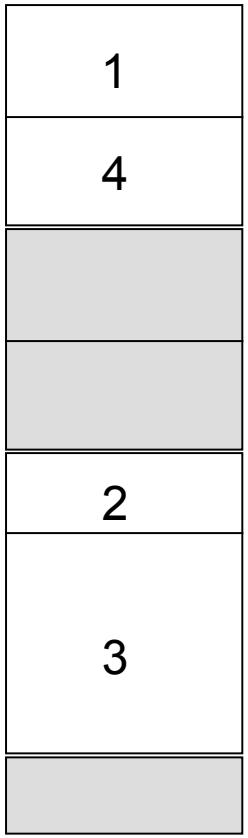


logical address

Logical View of Segmentation



user space



physical memory space

Segmentation Architecture

- Logical address consists of a two tuple:

  <segment-number, offset>,

- **Segment table** – maps two-dimensional physical addresses; each table entry has:
  - **base** – contains the starting physical address where the segments reside in memory
  - **limit** – specifies the length of the segment
- **Segment-table base register (STBR)** points to the segment table's location in memory
- **Segment-table length register (STLR)** indicates number of segments used by a program;

  segment number s is legal if $s <$ **STLR**

Segmentation: Example

- What is the physical addresses of below logical addresses according to the segment table?
  - 2,700
  - 0,150

Base	Length
500	200
700	1000
1800	600

Segmentation: Example

- What is the physical addresses of below logical addresses according to the segment table?
  - 2,700
  - 0,150

Base	Length
500	200
700	1000
1800	600

  - 2,700: Invalid
  - 0,150: 500+150=650