Segmentation

CS 537: Introduction to Operating Systems

Louis Oliphant

University of Wisconsin - Madison

Fall 2023

Administrivia

- **Project 2** Due Sep 26th @ 11:59pm
- **Only use handin directory to submit solution** – don't compile or run code from your handin directory.
- **Slip Days** Clarification - meant to cover for minor emergencies – No need to ask or tell me.
  - The handin directory is copied on the due date.
  - For the next 3 days, directories that have changed will be copied.
  - During grading, after a grade is calculated, if you have a slipdays.txt file (and you have slip days left) points will be added back on.
- **Project 1** Support
  - Tickets 183 completed / 191 Tickets = **95.81% Completion Rate**
  - Piazza 242 posts – **Avg. Response time: 39 Minutes**

Agenda

- Recall disadvantages of base/bound address translation
- Extend address translation to segmentation
- Advantages / Disadvantages of Segmentation

Review: Memory Virtualization

- The **Address Space** is the running program's view of memory. It includes **(1) code and static data**, **(2) the stack**, and **(3) the heap**.
- Understand how these memory segments (stack, heap) are used.
- Request dynamically allocated memory using `malloc()` and `free()`
- **Dynamic (hardware-based) Memory Relocation** using **base and bounds** is a form of **address translation**, requiring hardware support and OS configuration and management.

Review: Address Translation (Base/Bounds)

./relocation.py

```
address space size 1k
phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x00003082 (decimal 12418)
  Limit  : 472

Virtual Address Trace
  VA  0: 0x000001ae (decimal:  430) --> PA or segmentation violation?
  VA  1: 0x00000109 (decimal:  265) --> PA or segmentation violation?
  VA  2: 0x0000020b (decimal:  523) --> PA or segmentation violation?
  VA  3: 0x0000019e (decimal:  414) --> PA or segmentation violation?
  VA  4: 0x00000322 (decimal:  802) --> PA or segmentation violation?
```

Review: Address Translation (Base/Bounds)

./relocation.py -c

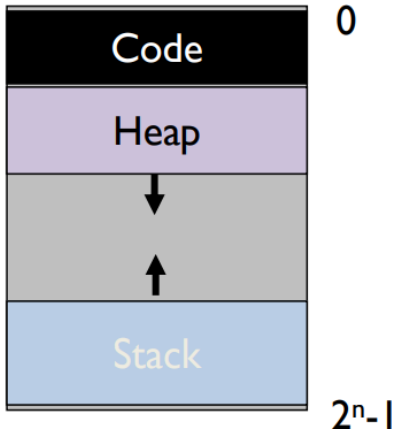
```
address space size 1k
phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x00003082 (decimal 12418)
  Limit  : 472

Virtual Address Trace
  VA  0: 0x000001ae (decimal:  430) --> VALID: 0x00003230 (decimal: 12848)
  VA  1: 0x00000109 (decimal:  265) --> VALID: 0x0000318b (decimal: 12683)
  VA  2: 0x0000020b (decimal:  523) --> SEGMENTATION VIOLATION
  VA  3: 0x0000019e (decimal:  414) --> VALID: 0x00003220 (decimal: 12832)
  VA  4: 0x00000322 (decimal:  802) --> SEGMENTATION VIOLATION
```

Quiz 4 – Address Space & Base/Bounds Translation

https://tinyurl.com/cs537-fa23-q4

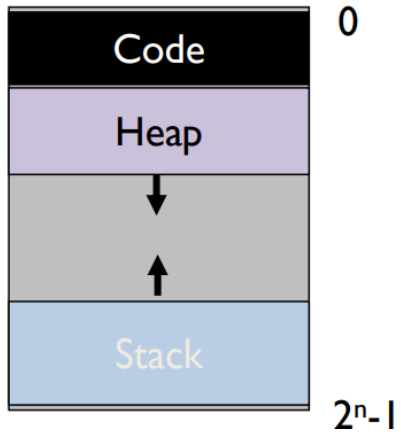
Base/Bounds Translation Disadvantages

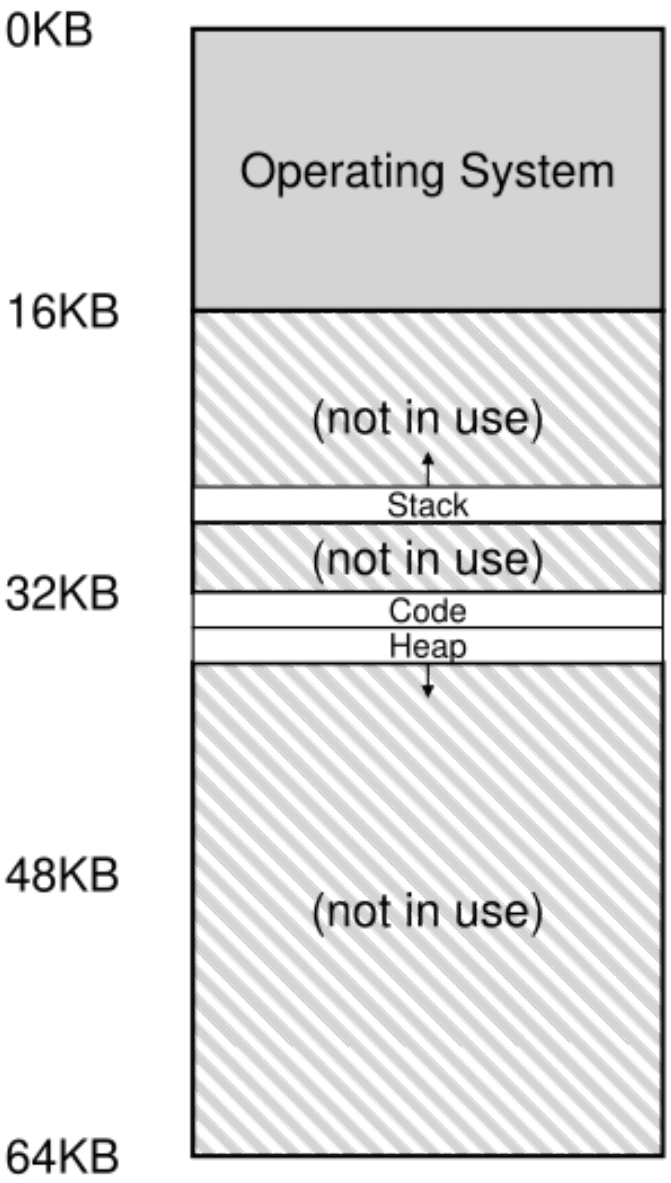
- Each process must be allocated contiguously
- Must allocate memory that may not all be used by process
  - Example: a 32-bit address space (4GB in size); A typical program will only use megabytes of memory, but still would demand that the entire address space be resident in memory.
- No partial sharing: Cannot share parts of address space



Segmentation

- Divide the address space into logical segments
  - Each segment corresponds to logical entity in address space
  - E.g. Code, Stack, Heap
- Each segment has separate base+bounds registers



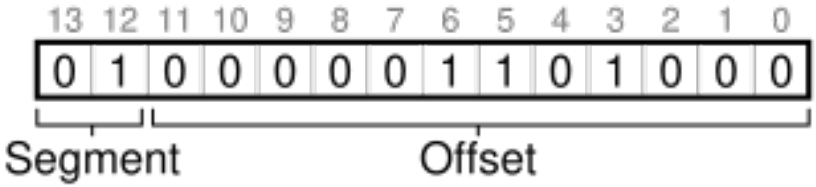


Physical Memory Layout

- Segments placed independently
- Segments grow/shrink independently

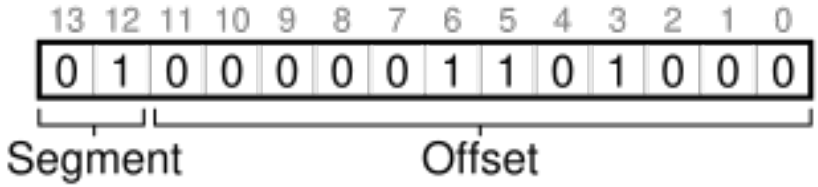
Segmented Addressing

- Explicit Approach – Logical address is divided:
  - top bits select the segment
  - remaining bits are the offset within the segment



- Implicit Approach
  - entire logical address is the offset
  - which segment is based off of how logical address was formed:
    - from PC then code segment
    - from SP then stack segment
    - Anything else is from heap segment

Segmented Addressing Translation



Segment: 0x01 (decimal: 1)

Offset: 0x68 (decimal: 104)

Segment (binary)	Base	Size (Bounds)
00	0x8000 (decimal: 32768)	0x800 (decimal: 2048)
01	0x8800 (decimal: 34816)	0x800 (decimal: 2048)
10	0x7000 (decimal: 28672	0x800 (decimal: 2048)
11	0x0000 (decimal: 0)	0x0000 (decimal: 0)

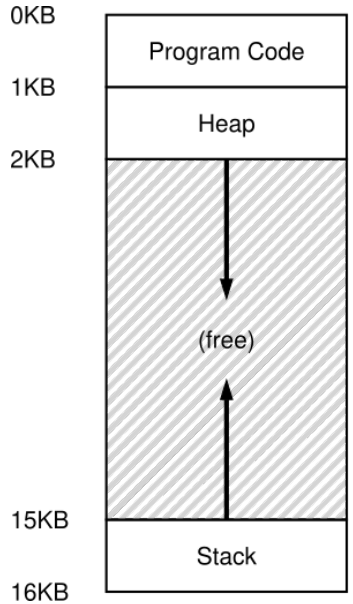
In Bounds? yes

Physical Address: 0x8800+0x68 = 0x8868 (decimal: 34920)

Sharing Segments / Growth Direction

- Additional **protection bits** record if segment is readable/writable.
- Bit records direction of growth – Changes translation process
  - The base is the largest address of the segment (instead of smallest)
  - Add **Negative offset** to bounds
  - **Negative offset** is offset - max segment size

Segment (binary)	Base	Size (Bounds)	Grows Positive?	R	W
00	0x8000 (decimal: 32768)	0x800 (decimal: 2048)	1	1	0
01	0x8800 (decimal: 34816)	0x800 (decimal: 2048)	1	1	1
10	0x7000 (decimal: 28672	0x800 (decimal: 2048)	0	1	1
11	0x0000 (decimal: 0)	0x0000 (decimal: 0)	1	0	0



Stack Translation Example

Segment	Base	Size	Grows Positive?
Code	32K	2K	1
Heap	34K	2K	1
Stack	28K	2K	0

Virtual Address (binary): 11 1100 0000 0000 (hex 0x3C00)
Top 2 bits indicate the stack base/bounds.
Offset: 3K
Negative Offset: 3K - 4K = -1K
Physical Address: 28K (base) + -1K (negative offset) = 27K

Segmentation Advantages

- Enables sparse allocation of address space
- Stack and Heap can grow independently
- Different protection for different segments
  - enables sharing of selected segments
  - protects code from modification
- Supports dynamic relocation of each segment

Segmentation Disadvantages

- Each segment must be allocated contiguously

- May not have sufficient physical memory for large segments

- External Fragmentation (makes managing free memory hard!)