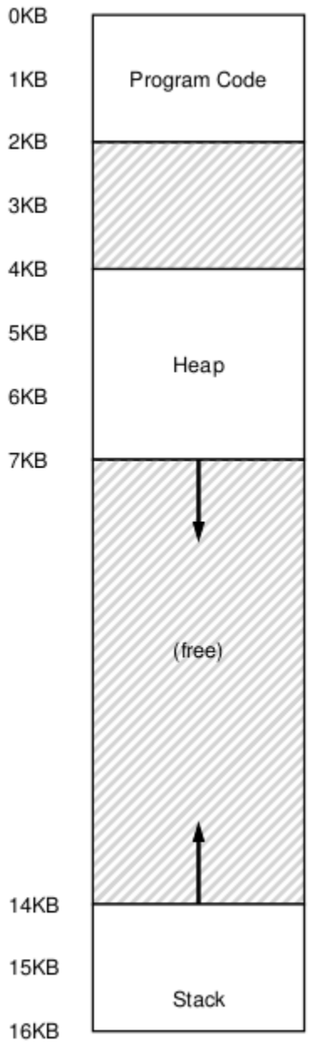
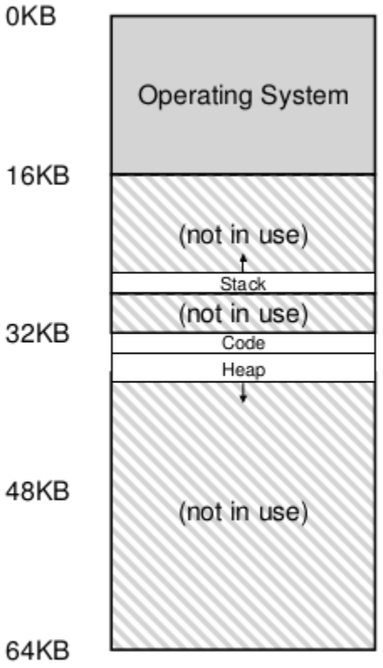
OSTEP Chapter 16

ECE 3600, Fall 2022

Table of Contents

1. Segmentation





Segment	VA	Base	Size
Code	0-2K	32K	2K
Heap	4-7K	34K	3K (grows positive)
Stack	16-14K	28K	2K (grows negative)

Figure 16.1: **An Address Space (Again)** Figure 16.2: **Placing Segments In Physical Memory**

2. Address Translation Examples

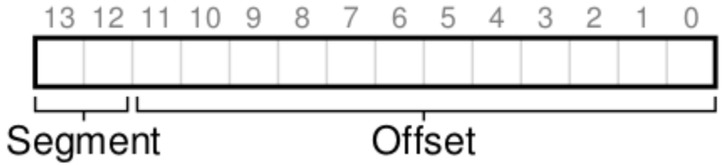
```
Segment    VA     Base  Size
-------  ------  -----  ----
   Code   0-2K   32768   2K
   Heap   4-7K   34816   3K (grows positive) [4K = 4096] [34816 + 3K = 37888]
  Stack  16-14K  28672   2K (grows negative) [16K = 16384]

Virtual Address 100 (Code) --> Physical Address 32768 + 100 = 32868

Virtual Address 4200 (Heap) --> Physical Address 34816 + (4200 - 4096) = 34920

Virtual Address 15360 (Stack) --> Physical Address 28672 - (16384 - 15360) = 27648

Segmentation Violation = Segmentation Fault = Illegal Virtual Address:

  Virtual Address 8000 (Heap) --> Physical Address 34816 + (8000 - 4096) = 38720 ≥ 37888
```

3. Segment Mapping Examples

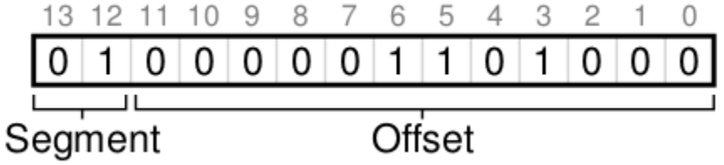
16K virtual address space --> 14-bit virtual address

max segment size 4K --> 12-bit offset

2-bit segment number



4200 = 01 0000 0110 1000




```
1  // get top 2 bits of 14-bit VA
2  Segment = (VirtualAddress & SEG_MASK) >> SEG_SHIFT
3  // now get offset
4  Offset  = VirtualAddress & OFFSET_MASK
5  if (Offset >= Bounds[Segment])
6      RaiseException(PROTECTION_FAULT)
7  else
8      PhysAddr = Base[Segment] + Offset
9      Register = AccessMemory(PhysAddr)
```

specify SEG_MASK, SEG_SHIFT, and OFFSET_MASK: _____

4. Segment Options and Protection

Segment	Base	Size (max 4K)	Grows Positive?
Code_{00}	32K	2K	1
Heap_{01}	34K	3K	1
Stack_{11}	28K	2K	0

Figure 16.4: **Segment Registers (With Negative-Growth Support)**

Segment	Base	Size (max 4K)	Grows Positive?	Protection
Code_{00}	32K	2K	1	Read-Execute
Heap_{01}	34K	3K	1	Read-Write
Stack_{11}	28K	2K	0	Read-Write

Figure 16.5: **Segment Register Values (with Protection)**

5. Fragmentation

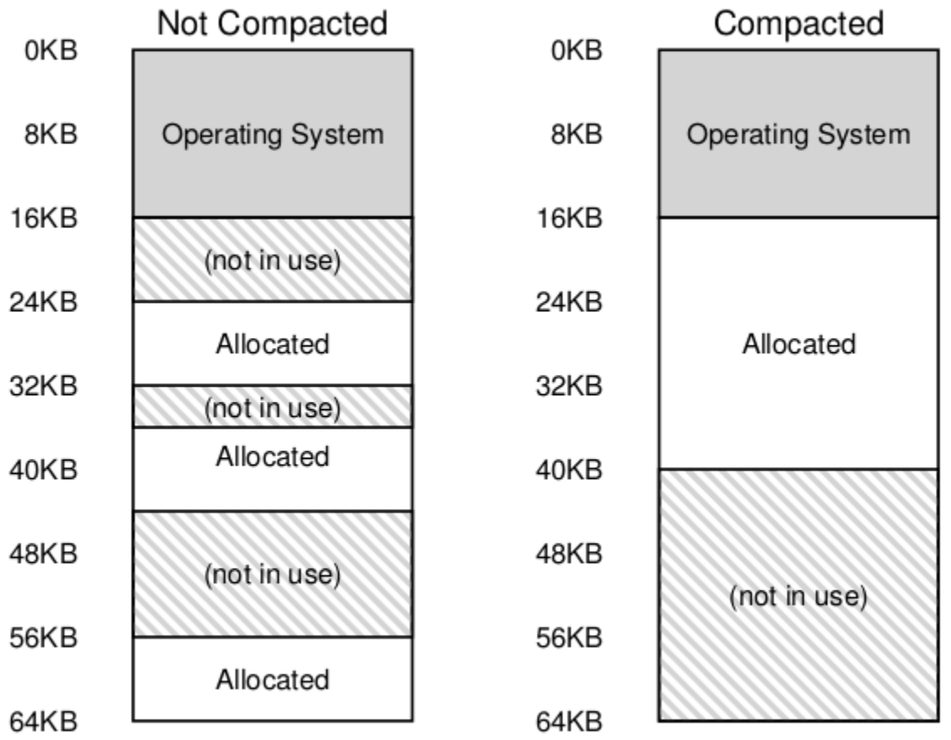


Figure 16.6: **Non-compacted and Compacted Memory**

6. Exercises

Exercises from the book using segmentation.py:

1. First let's use a tiny address space to translate some addresses. Here's a simple set of parameters with a few different random seeds; can you translate the addresses?

```
segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 0
segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 1
segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 2
```

2. Now, let's see if we understand this tiny address space we've constructed (using the parameters from the question above). What is the highest legal virtual address in segment 0? What about the lowest legal virtual address in segment 1? What are the lowest and highest illegal addresses in this entire address space? Finally, how would you run segmentation.py with the -A flag to test if you are right?