# Segmentation

How to decouple address space from physical memory?

By Matthew Tancreti
For COM S 352
Iowa State University

# Segmentation

Base and bounds requires direct translation from address space to physical memory

Assumes program knows in advance how much dynamic memory will be required

Growing process's memory dynamically is very difficult

How to remove limitations of physical memory from address space?

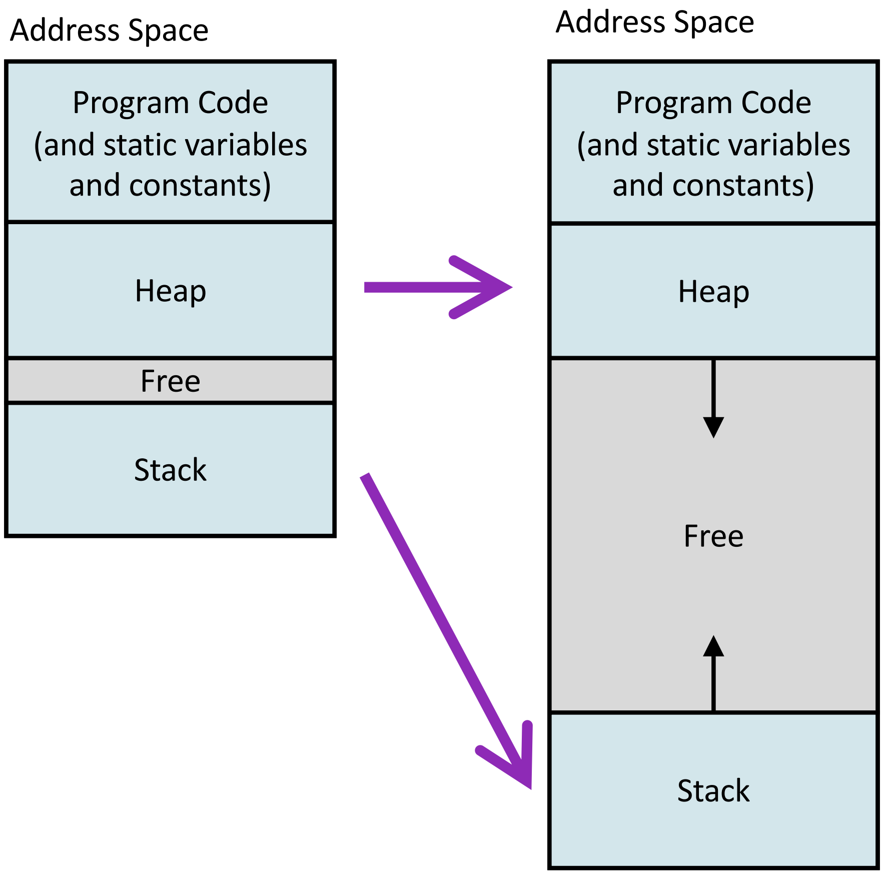
*Kowloon Walled City, 1993* [source]

# Problem

What happens when address space is full?

Using base and bounds, process needs to be copied to larger region in memory

All pointers to stack need to be updated!?!?


Address Space
Program Code (and static variables and constants)
Heap
Free
Stack
Address Space
Program Code (and static variables and constants)
Heap
Free
Stack

# Example Address Space in Linux

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    printf("code  %p\n", main);
    printf("heap  %p\n", malloc(1));
    printf("stack %p\n", &argc);
    return 0;
}
```

```
code  0x401136        =           4,198,710
heap  0x1d096b0       =          30,447,280
stack 0x7ffc6a46bf4c  = 140,722,091,507,532    140TB address space size!
```

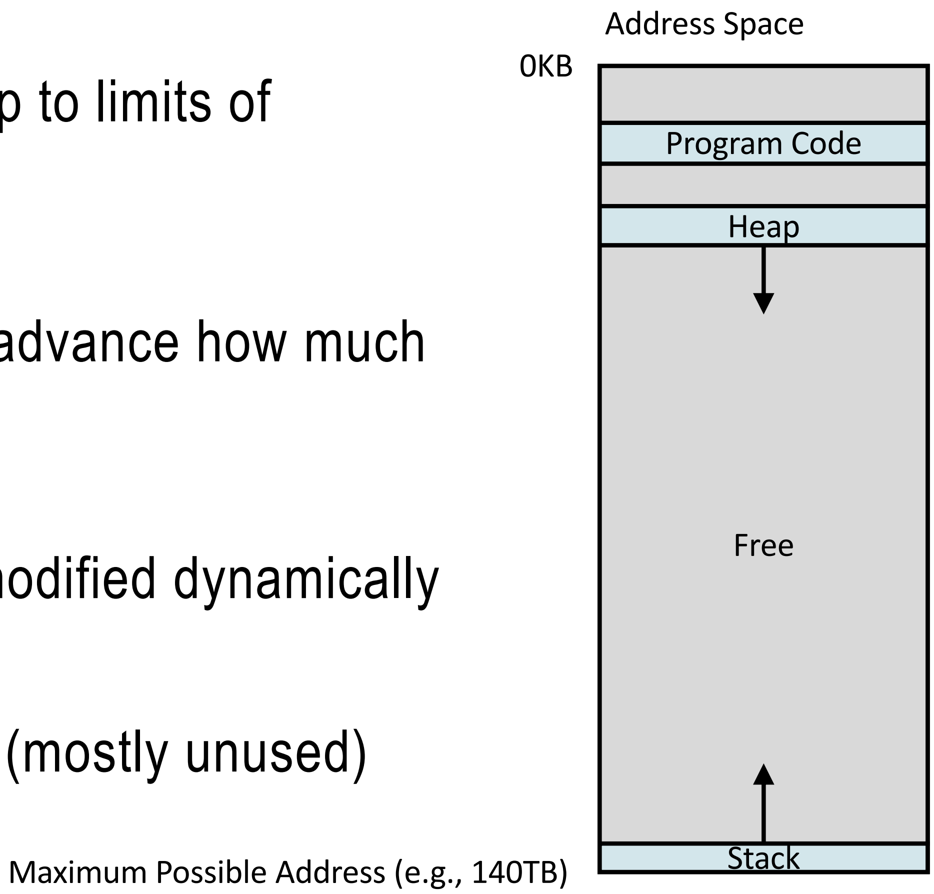
# Goal: Remove Physical Limitations from Address Space

Make the address space massive (up to limits of addressable size)

Program doesn't need to declare in advance how much memory it will need

Address space doesn't need to be modified dynamically

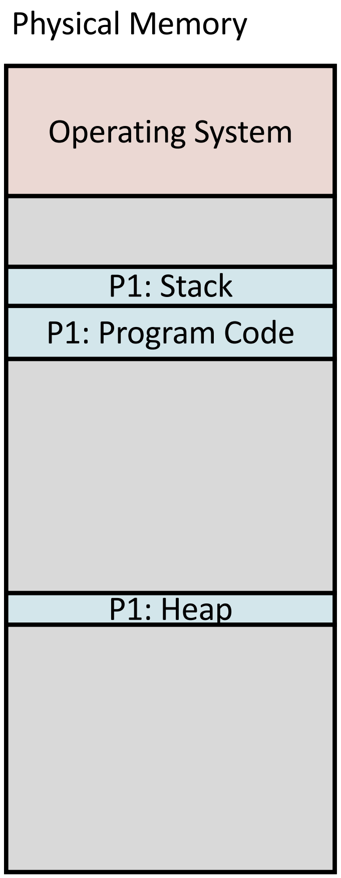
Known as a **sparse address space** (mostly unused)



# Segmentation

How to allow for sparce address space in physical memory?

**Segmentation** means we can locate parts of the address space independently in physical memory

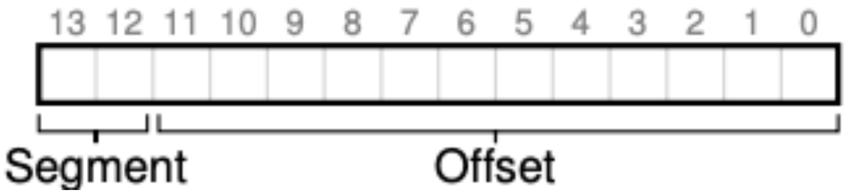


# Hardware Requirements for Segmentation

Registers for the start and size of each segment

| Segment | Base register | Size register |
|---|---|---|
| Code | 32K | 2K |
| Heap | 34K | 2K |
| Stack | 28K | 2K |

# How to Translate Addresses?

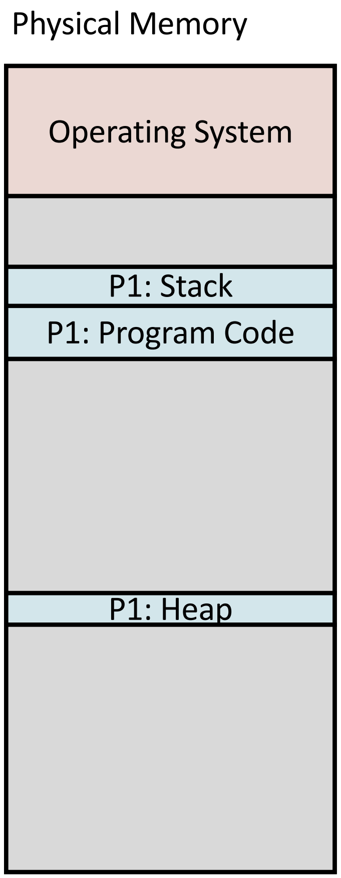


```
// get top 2 bits of 14-bit VA
Segment = (VirtualAddress & SEG_MASK) >> SEG_SHIFT
// now get offset
Offset = VirtualAddress & OFFSET_MASK
if (Offset >= Bounds[Segment])
    RaiseException(PROTECTION_FAULT)
else
    PhysAddr = Base[Segment] + Offset
    Register = AccessMemory(PhysAddr)
```

# Question

What if physical memory runs out of space for a segment and needs to relocate it? Will pointers in the program need to be updated?

No, address space does not depend on where segments are located in physical memory.

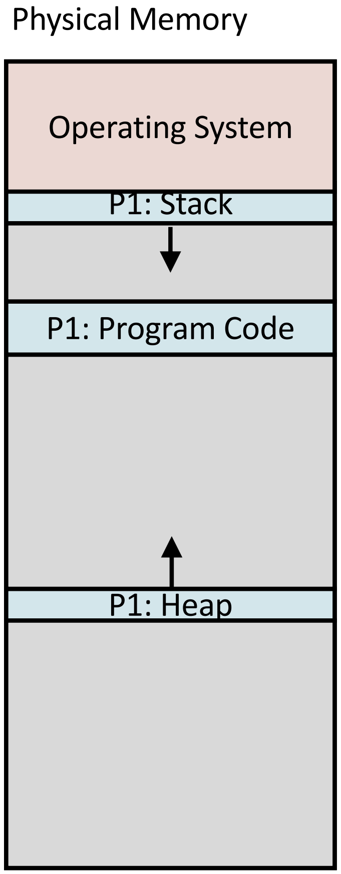


# Independent Direction of Segment Growth

We can even allow segments to grow in different directions

A set of registers can indicate if a segment grows up or down

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



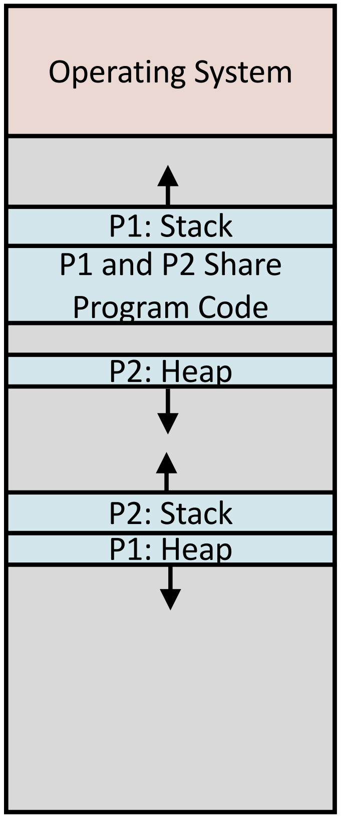
# Sharing

Protection registers can enable **sharing**

Example: two processes are executing the same code. If code segment is read-only no danger of processes corrupting each other

| Segment | Base register | Size register | Grows Positive? | Protection |
|---|---|---|---|---|
| Code | 32K | 2K | 1 | Read-execute |
| Heap | 34K | 2K | 1 | Read-Write |
| Stack | 28K | 2K | 0 | Read-Write |

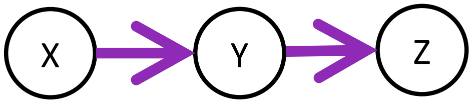
Physical Memory



# Free Memory

Segments are in contiguous regions of physical memory

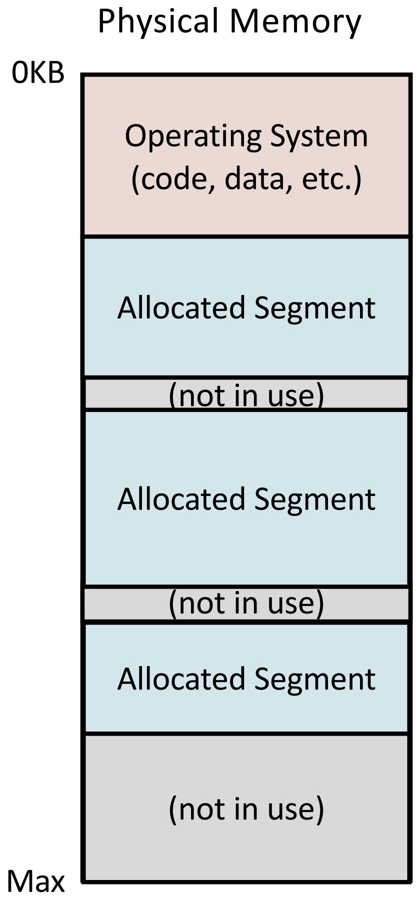
To allocate a new segment, OS must keep a list of free memory



Simple solution is a linked list of free regions of memory

On new allocation search for first open spot that has sufficient memory (**first fit** strategy)

**Best fit** strategy searches for smallest region of free memory that will fit the segment



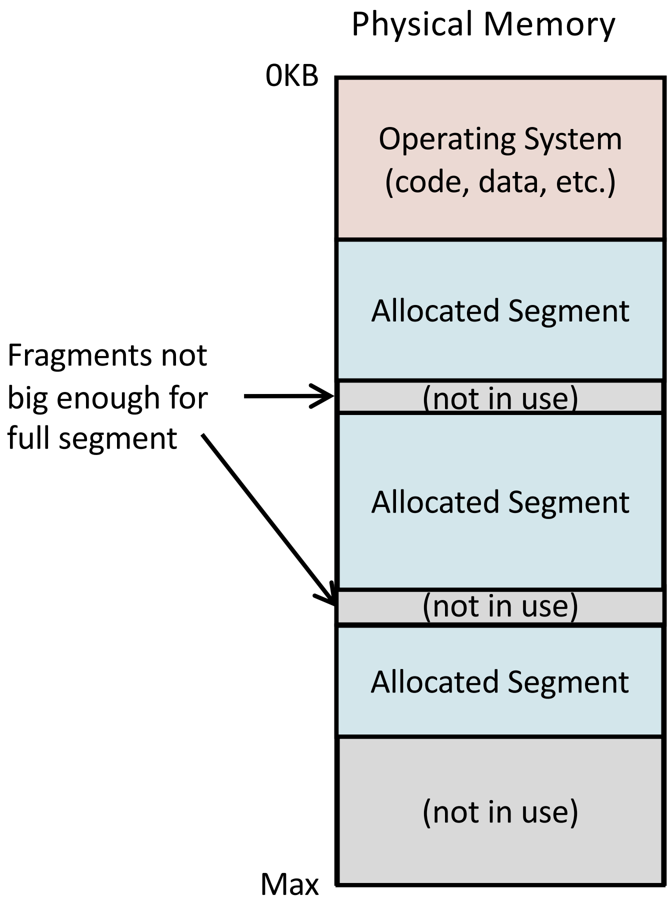
# Fragmentation

Segments are in contiguous regions of physical memory

Gaps result it **external fragmentation** (wasted physical memory)

Not big enough to fit a full segment, so can't be used

**Compaction** used to reclaim the fragments


Physical Memory
0KB
Operating System (code, data, etc.)
Allocated Segment
Fragments not big enough for full segment
(not in use)
Allocated Segment
(not in use)
Allocated Segment
(not in use)
Max