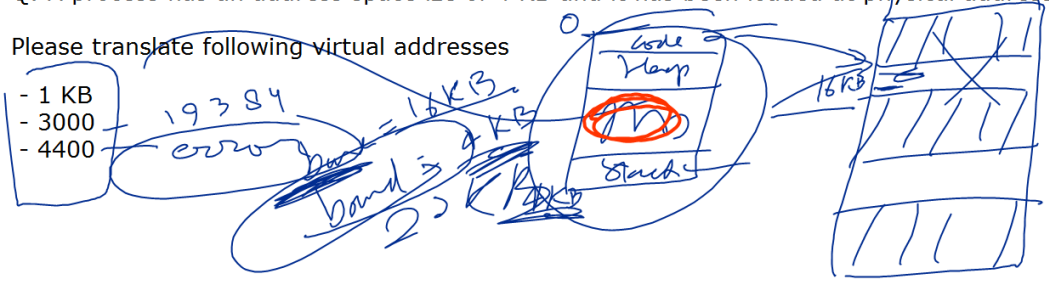
Memory

Q: A process has an address space ize of 4 KB and it has been loaded at physical address 16 KB.

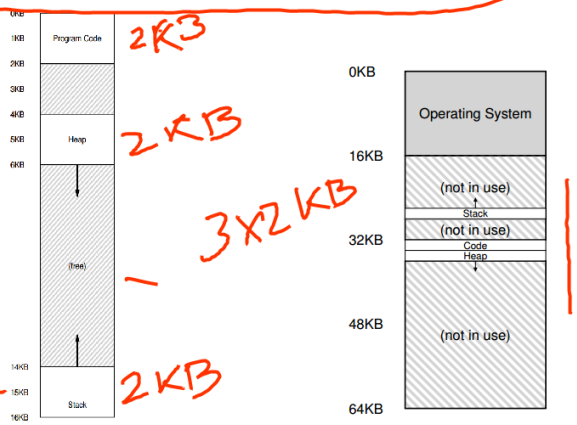
Please translate following virtual addresses

- 1 KB
- 3000
- 4400



HOW TO SUPPORT A LARGE ADDRESS SPACE

Segmentation



Segment	Base	Size
Code	32K	2K
Heap	34K	2K
Stack	28K	2K

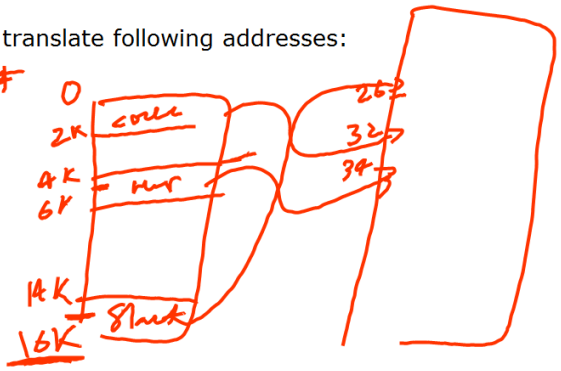
bound = 2K

How would you translate following addresses:

100 ~~16824~~ 32868

4200

7KB



32k = 32768

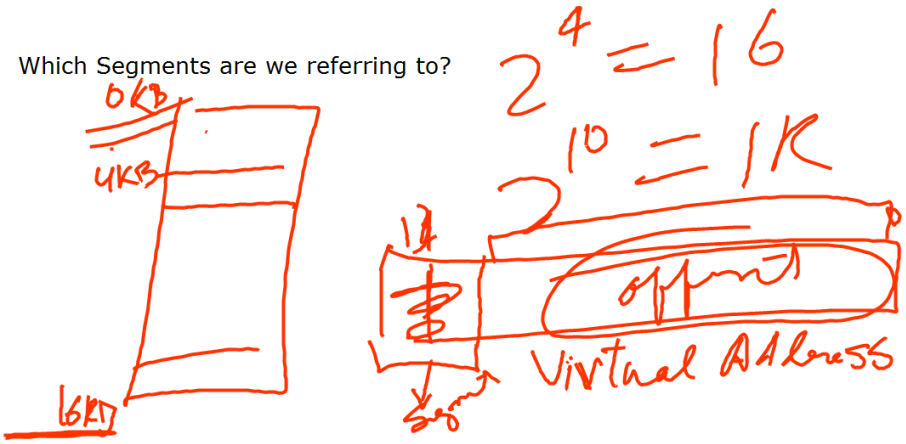
34k = 34816

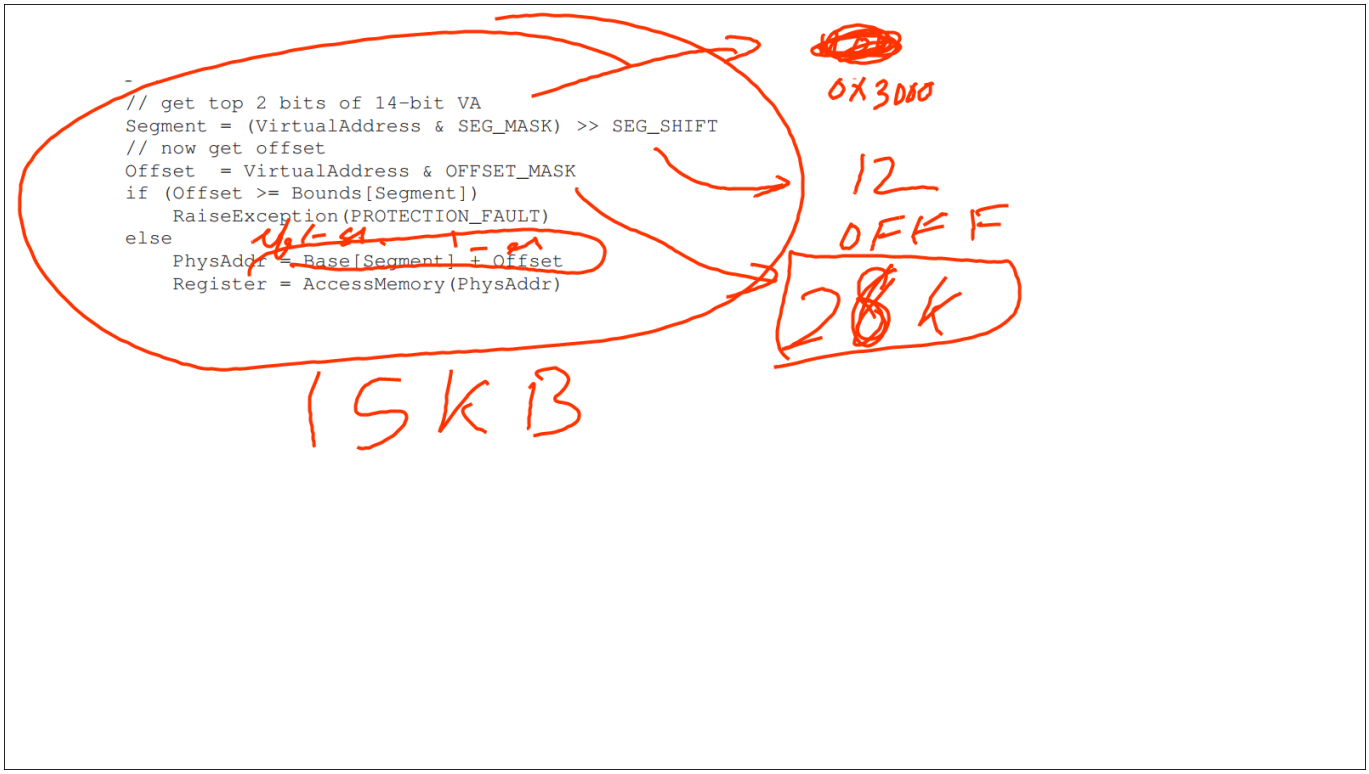
26k = 26624

4200

4096

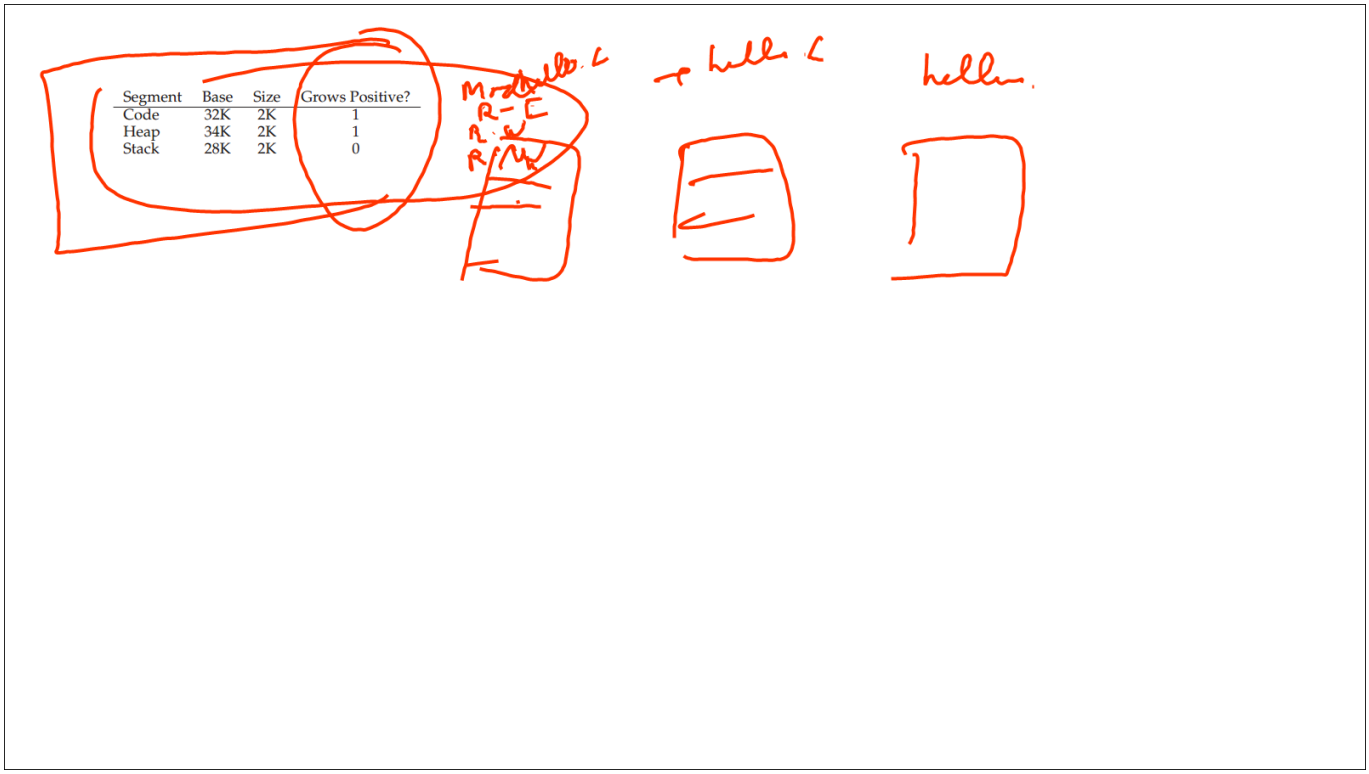
Which Segments are we referring to?




// 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)
0x3000
12
0FFF
28K
15KB

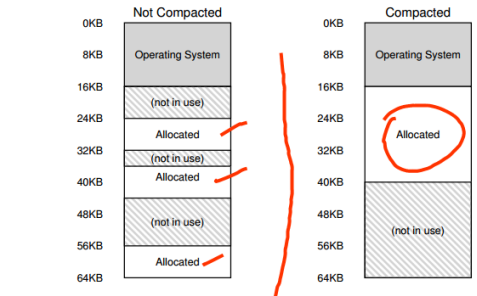
What about the Stack

27 KB?



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

Support for Sharing



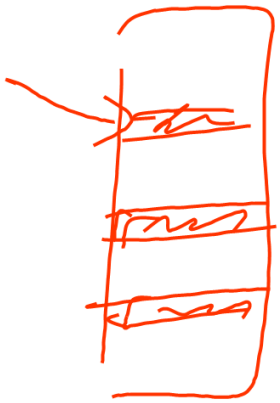
Fragmentation

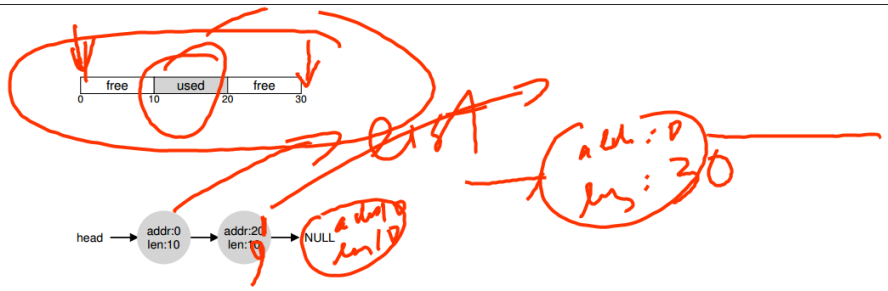
Free Space Management

void *malloc(size t size)

void free(void *ptr)

External Fragmentation





Splitting and Coalescing

Tracking The Size Of Allocated Regions

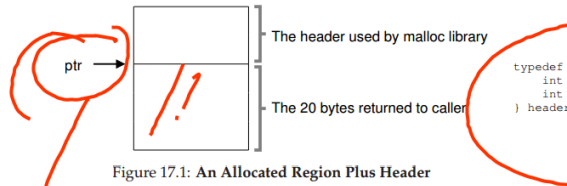


Figure 17.1: **An Allocated Region Plus Header**

```
typedef struct __header_t {
    int size;
    int magic;
} header_t;
```

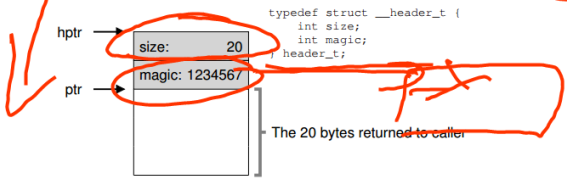


Figure 17.2: **Specific Contents Of The Header**

```
void free(void *ptr) {
    header_t *hptr = (void *)ptr - sizeof(header_t);
}
```

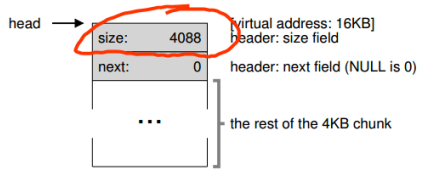
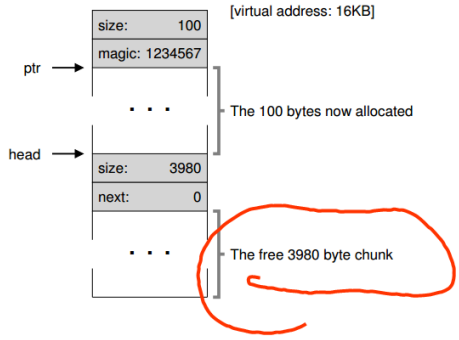
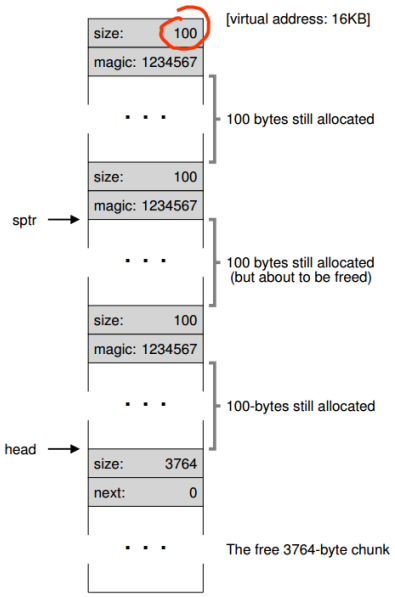
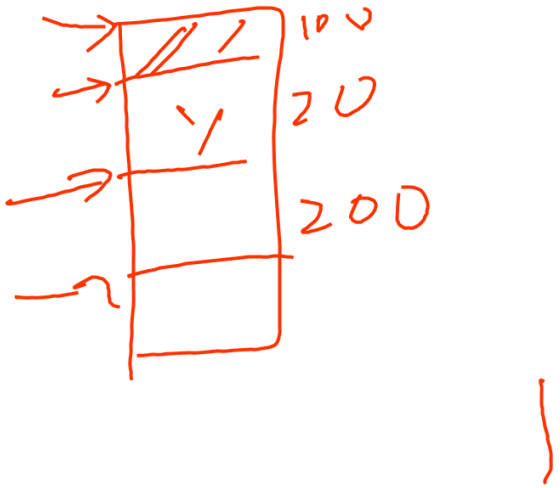
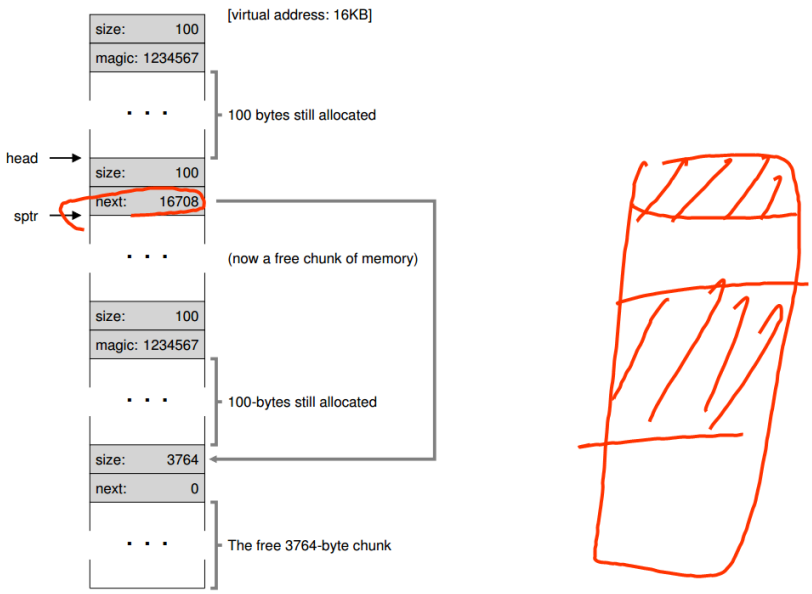


Figure 17.3: **A Heap With One Free Chunk**




size: 100
magic: 1234567
[virtual address: 16KB]
100 bytes still allocated
size: 100
magic: 1234567
sptr
100 bytes still allocated
(but about to be freed)
size: 100
magic: 1234567
100-bytes still allocated
head
size: 3764
next: 0
The free 3764-byte chunk


100
20
200


[virtual address: 16KB]
size: 100
magic: 1234567
100 bytes still allocated
head
size: 100
next: 16708
sptr
(now a free chunk of memory)
size: 100
magic: 1234567
100-bytes still allocated
size: 3764
next: 0
The free 3764-byte chunk

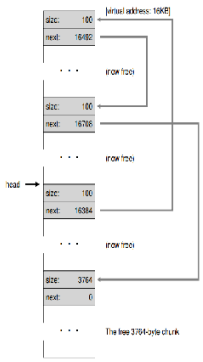


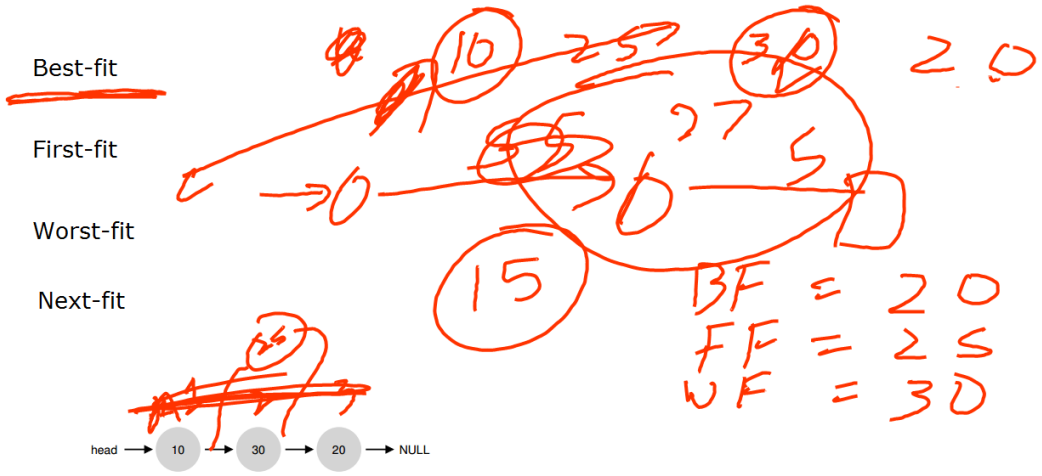
Figure 17.7: A Non-Coalesced Free List

Best-fit

First-fit

Worst-fit

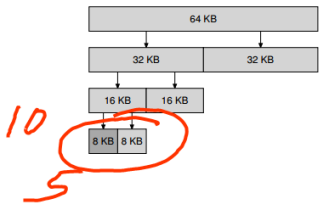
Next-fit

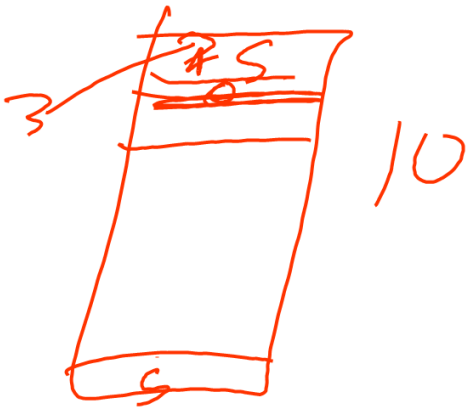


Segregated lists

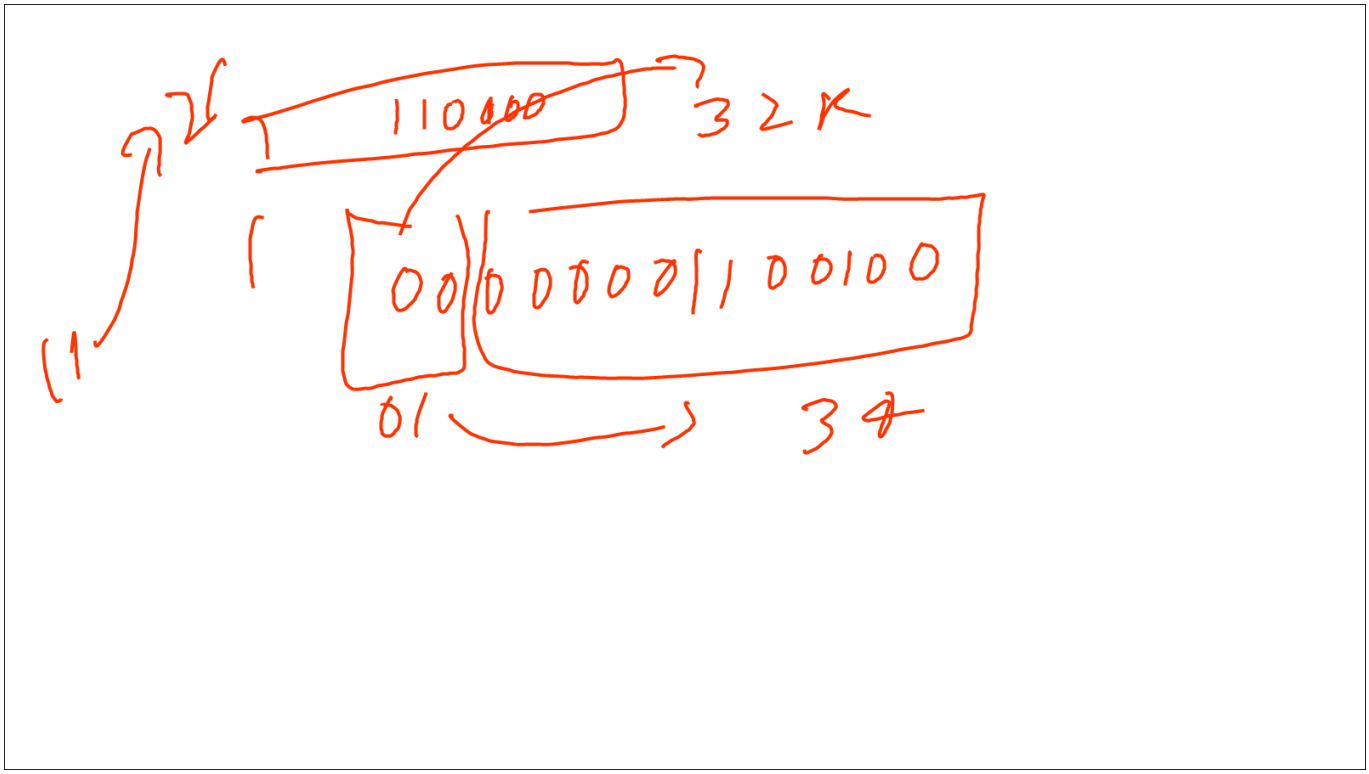
Slab Allocator

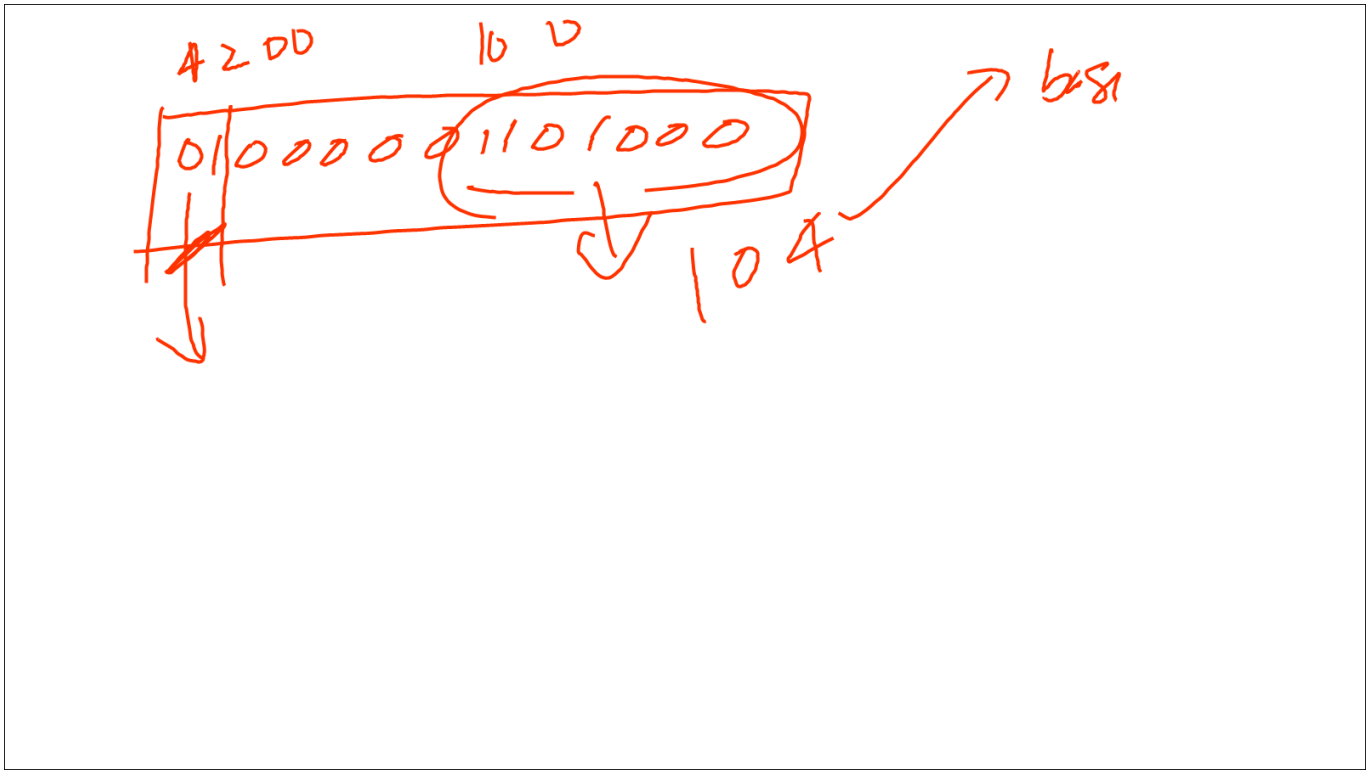
Buddy Allocator





Internal Fragmentation


110000
32K
00
00000|100100
01
34


4200
10
01000000 11010000
104
base