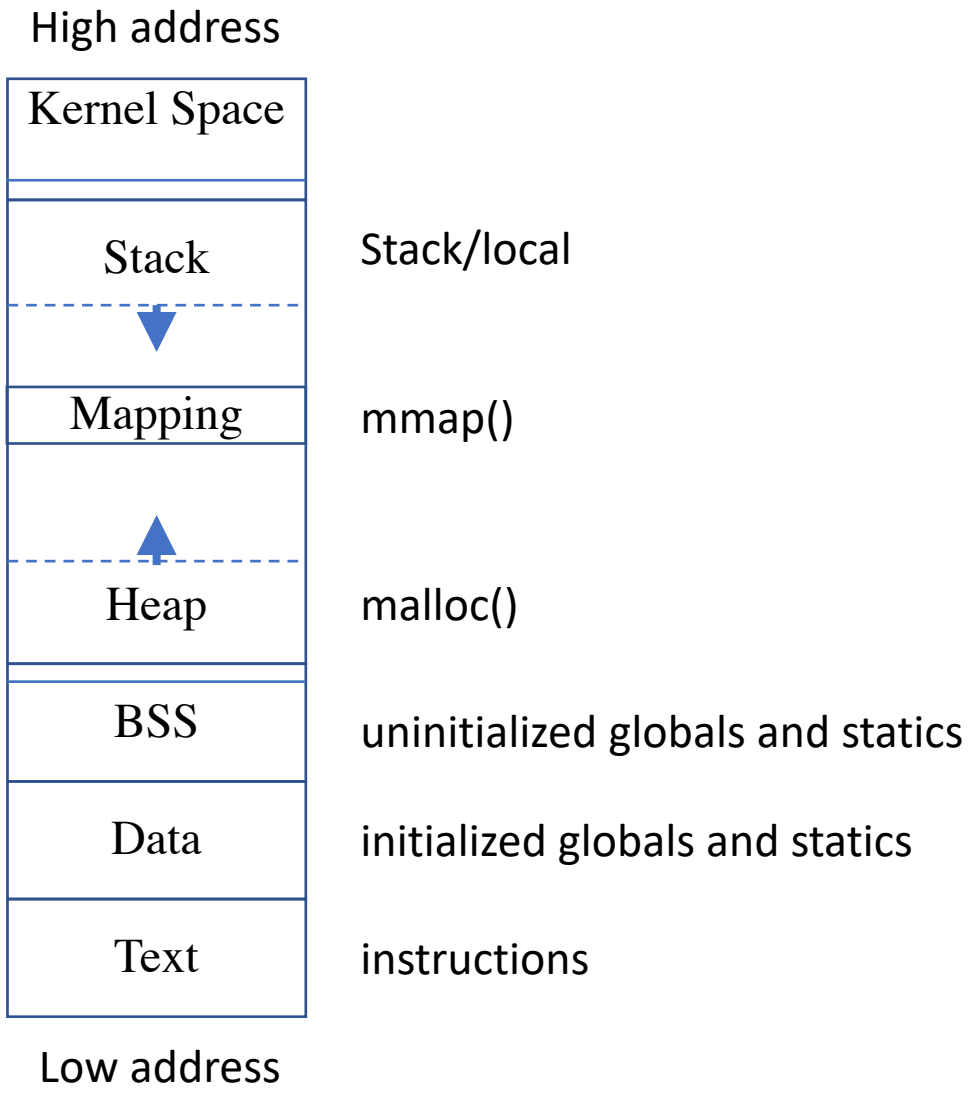
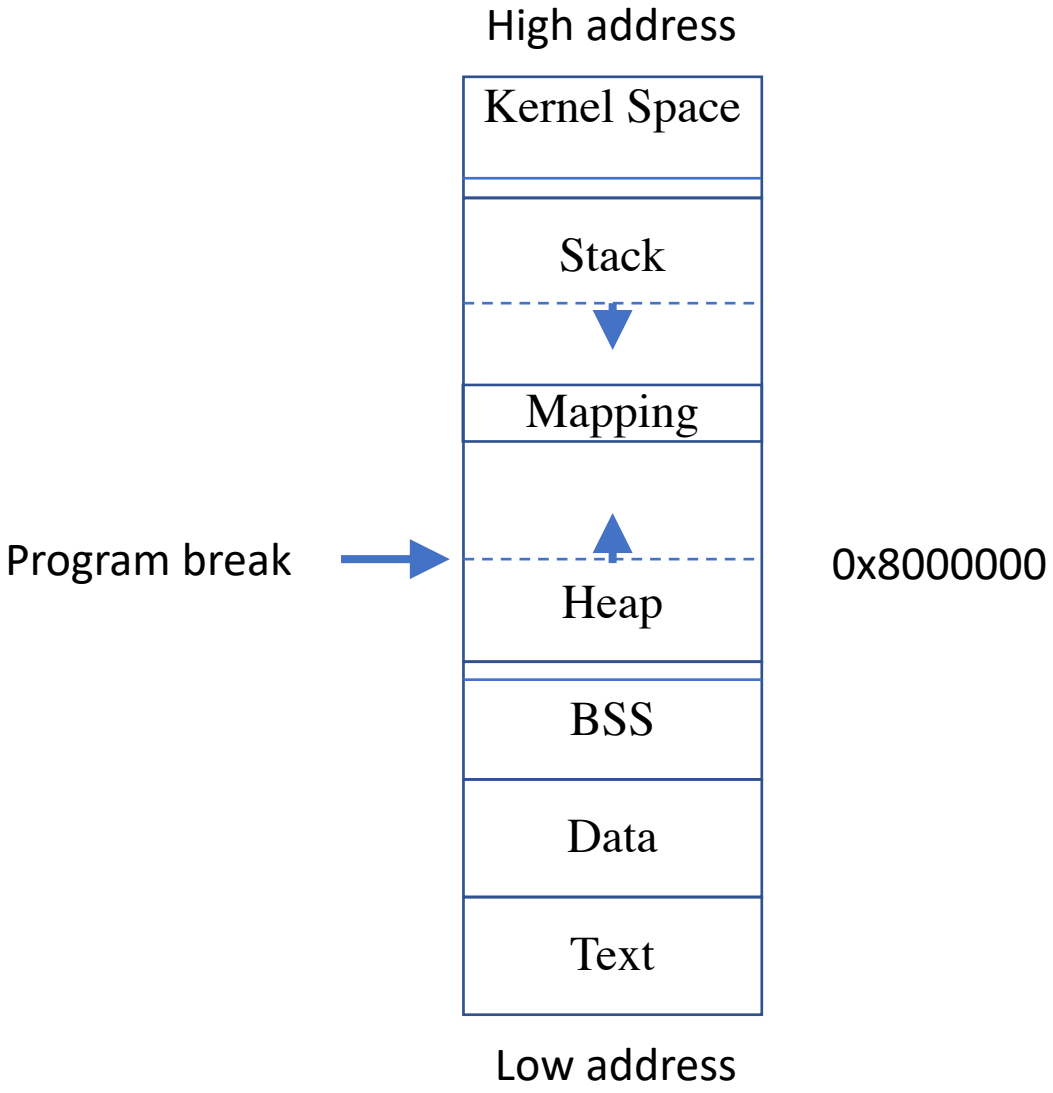
CS1550 Lab 3

Memory layout

```
int  t = 0;  // Data
int m;       // BSS
…
int main() {
        …
        int  i;                 // Stack
        static  int  j;         // BSS

        // ptr: Stack
        // 4B pointed by ptr: Heap
        char * ptr = (char*)malloc(4);

         // mptr: Stack
        // 4K pointed by mptr: memory Mapping
        char * mptr = (char*)mmap(…,4096,…);
        …
}
```



Program break

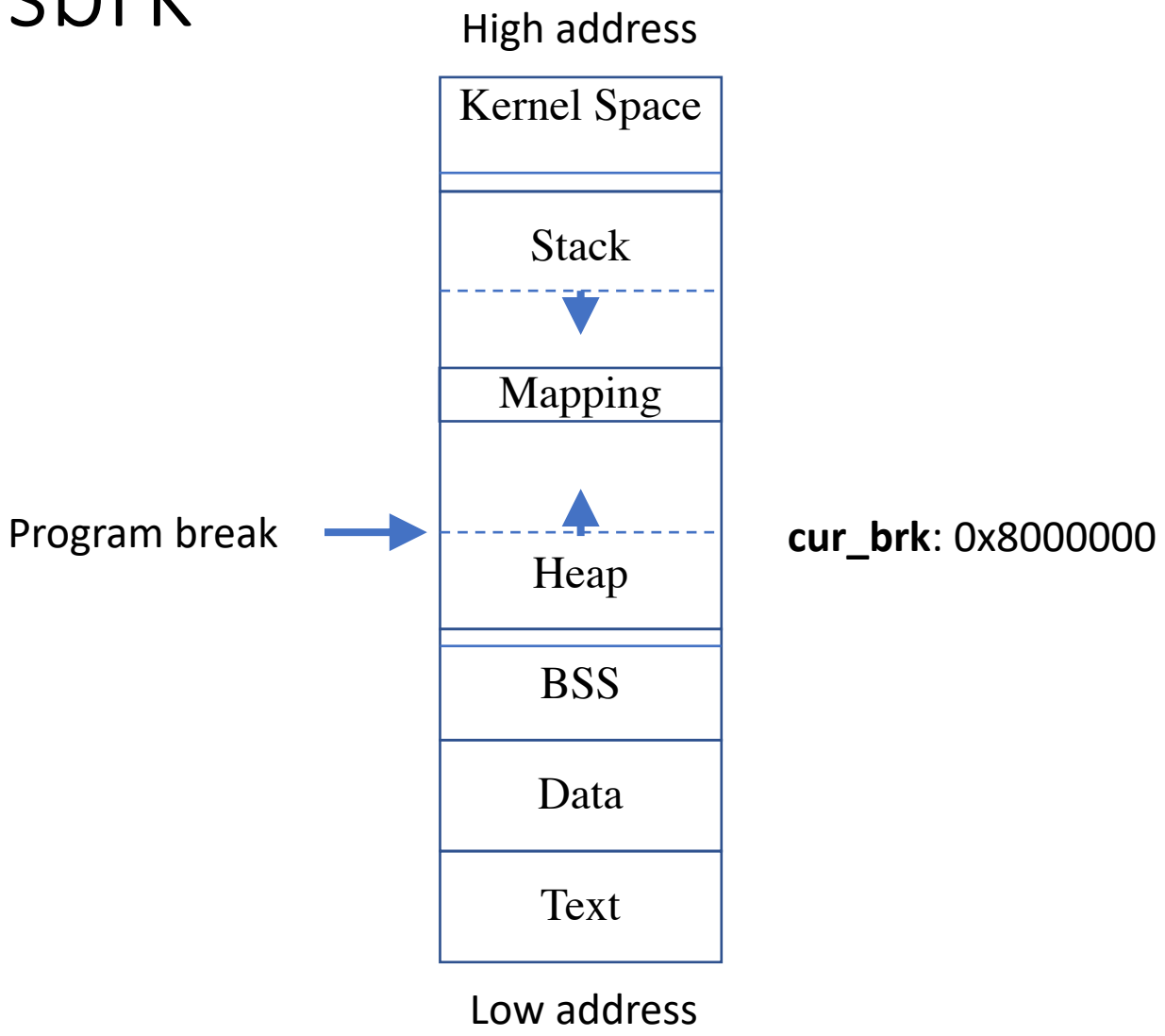


Program break: sbrk

void *cur_brk = sbrk(0);

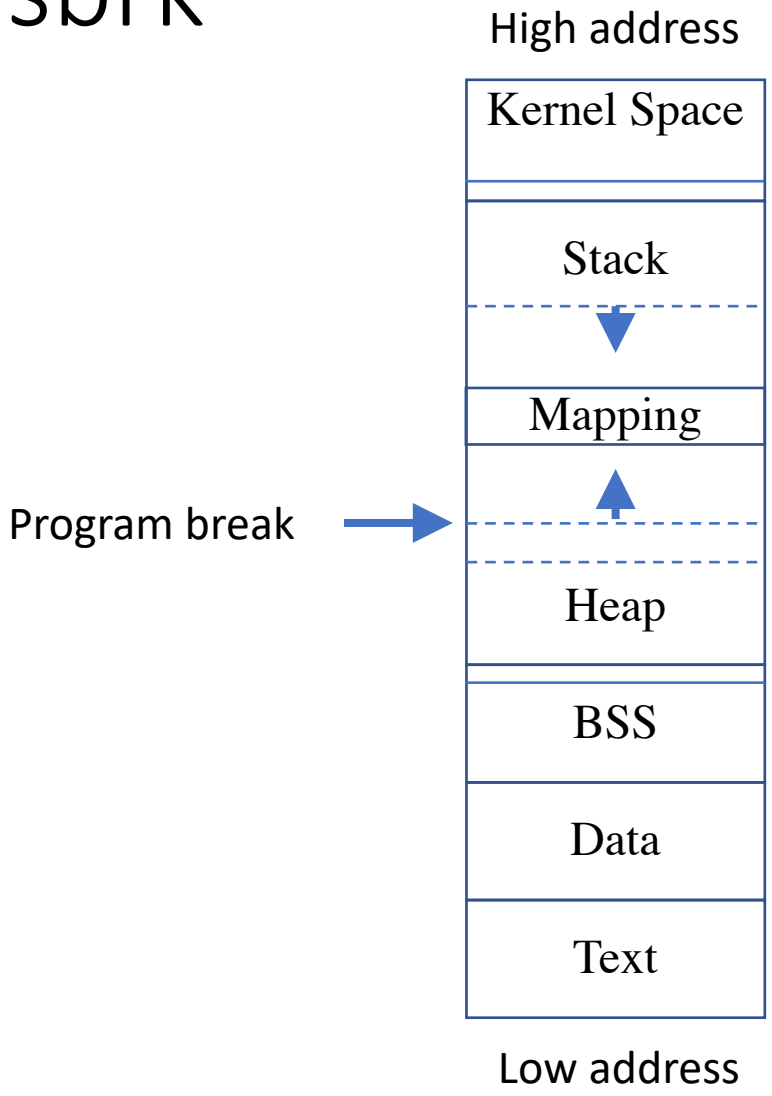
void *old_brk = sbrk(1024);

void *new_brk = sbrk(0);



Program break: sbrk

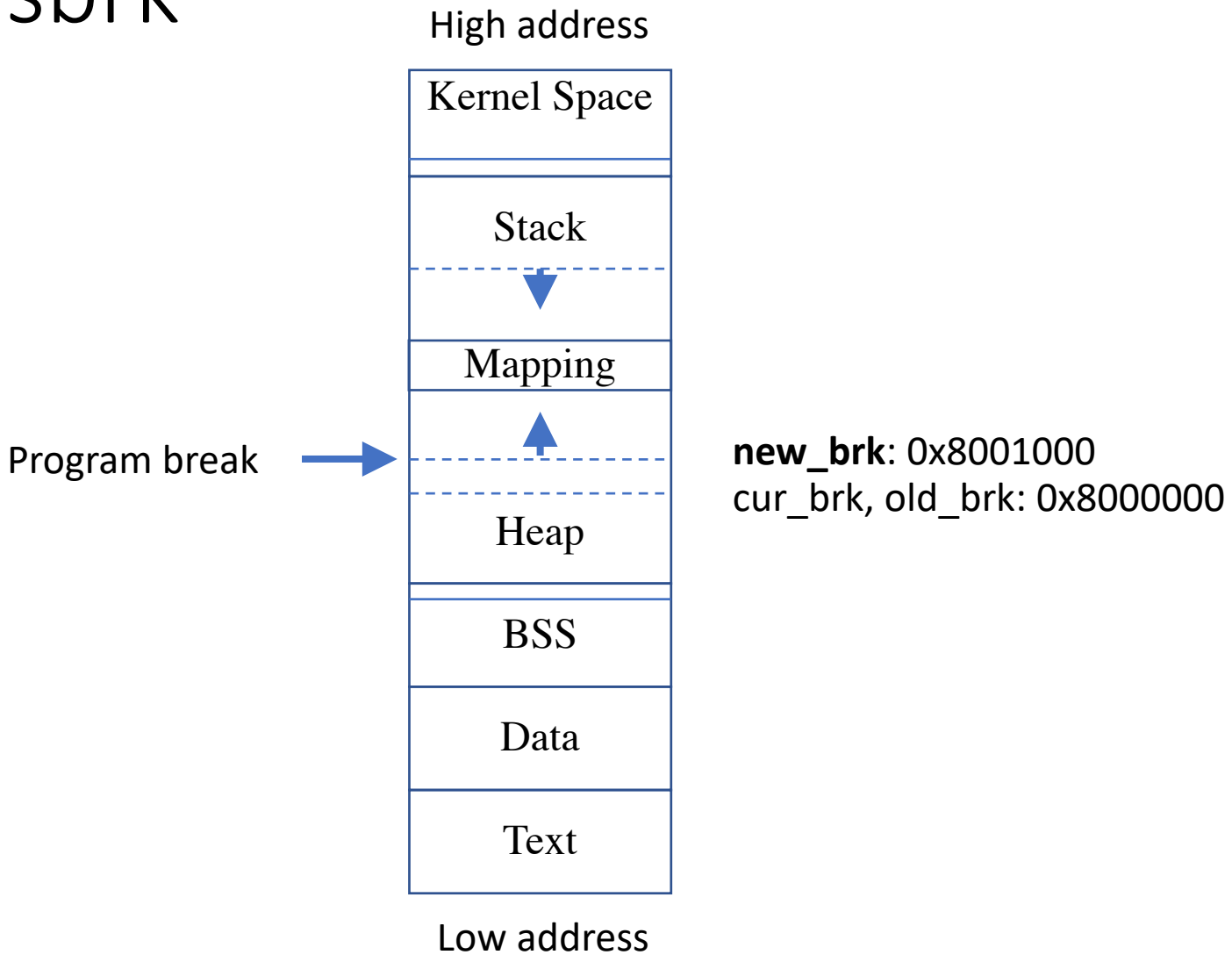
```
void *cur_brk = sbrk(0);

void *old_brk = sbrk(4096);

void *new_brk = sbrk(0);
```



0x8001000: increase 0x8000000 by 4K
old_brk, cur_brk: 0x8000000

Program break: sbrk

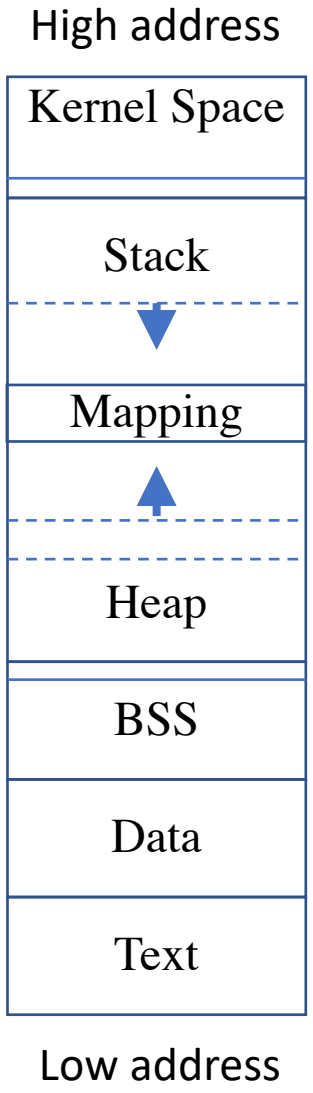
```
void *cur_brk = sbrk(0);

void *old_brk = sbrk(4096);

void *new_brk = sbrk(0);
```



Program break: brk

```
int ret =  brk(0x8001000);
if (ret != 0) {
        // error
}
```

Set program break to



New: **0x8001000**
Old: 0x8000000

Sbrk on XV6

The sys_sbrk() in syspro.c is a XV-6 implementation for sbk.

```
…
addr = myproc()->sz;          // get current brk
if(growproc(n) < 0)           // increase brk by n
        return -1;
…
return addr;                  // return old brk
```

growproc

The growproc() in proc.c:

```
…
if(n > 0) {                    // allocation
        allocuvm();            // allocate physical pages, update page table
} else if (n < 0) {            // deallocation
        deallocuvm();          // update page table, free physical page
}
```

Physical memory allocation

Given 4KB per page, and allocating an array with size of 100 pages:

```
char * ptr = (char*) malloc (4096 * 100);
```

Note:

This only allocates virtual memory: ptr to ptr+4096*100

How about physical memory?

XV6: Immediately allocate all 100 physical page frames

Problems?

Physical memory allocation

Given 4KB per page, and allocating an array with size of 100 pages:

```
char * ptr = (char*) malloc (4096 * 100);
// assume ptr is 0x8000000, i.e., a page-aligned virtual address
```

Note: malloc() only allocates virtual memory: ptr to ptr+4096*100

How about physical memory?

Lab 3: allocate physical page frame upon the 1st access on that page.

```
ptr[4096*99 + 50] = 'a';  //  the 1st access on the 100th page
```

Page Fault

Page Table: Page table stores mapping from virtual page to physical page frame

E.g., Virtual Page 0x8000000 -> Physical 0x400000

Translating a virtual address to physical address:

Virtual address → (TLB →) Page Table → Physical address

Translating virtual address 0x8000005:

1. Get its page-start-address 0x8000000, and **offset**-in-page **5**.
2. Search (TLB &) Page Table to find the mapping of 0x8000000 (how it works in multi-level page tables?)
3. If found, e.g., 0x8000000->0x4000000, then physical address is 0x400000**5**. If not found, Page Fault

Page Fault

char *ptr = (char*) malloc(4096*100);

ptr[4096*99 + 50] = ‘a’; // 1st access, no physical page frame: **Page Fault**

In XV6, Page Fault on ptr[4096*99 + 50] (inside 100th page):

1. Issue Page Fault trap. All traps are handled by trap() in trap.c.
2. **Handle Page Fault (*Hint: T_PGFLT, how to find the faulting addr*) in trap():**
   1) **Allocate a physical page frame for this 100th page**
   2) **Update page table**

Allocate physical pages, update page table

Recall the growproc() in proc.c.

```
if(n > 0) {                    // allocation
        allocuvm();            // allocate physical pages, update page table
} else if (n < 0) {            // deallocation
        deallocuvm();          // update page table, free physical page
}
```

Check how allocuvm & deallocuvm work.

(*Hint: PGROUNDDOWN, mappages)*