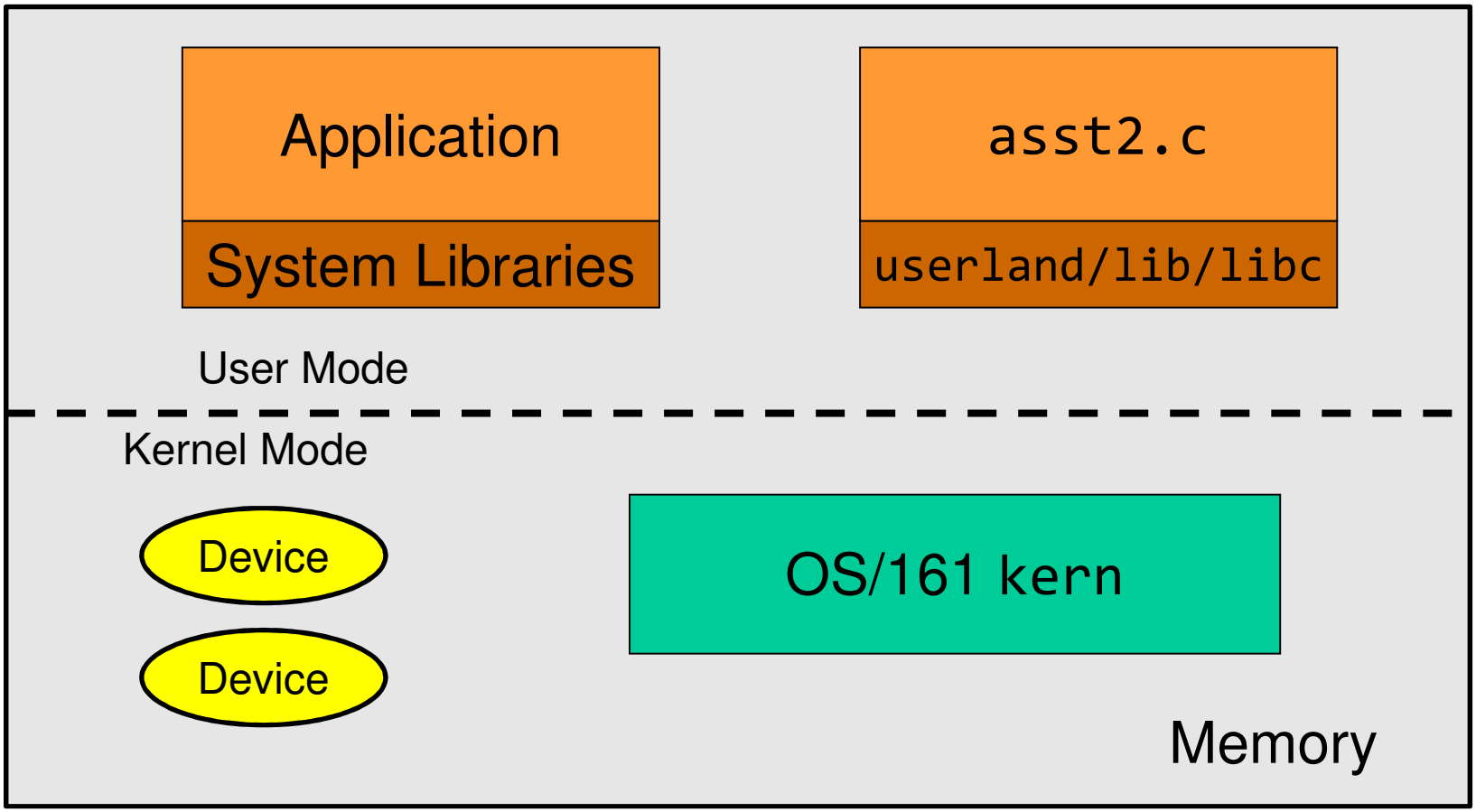
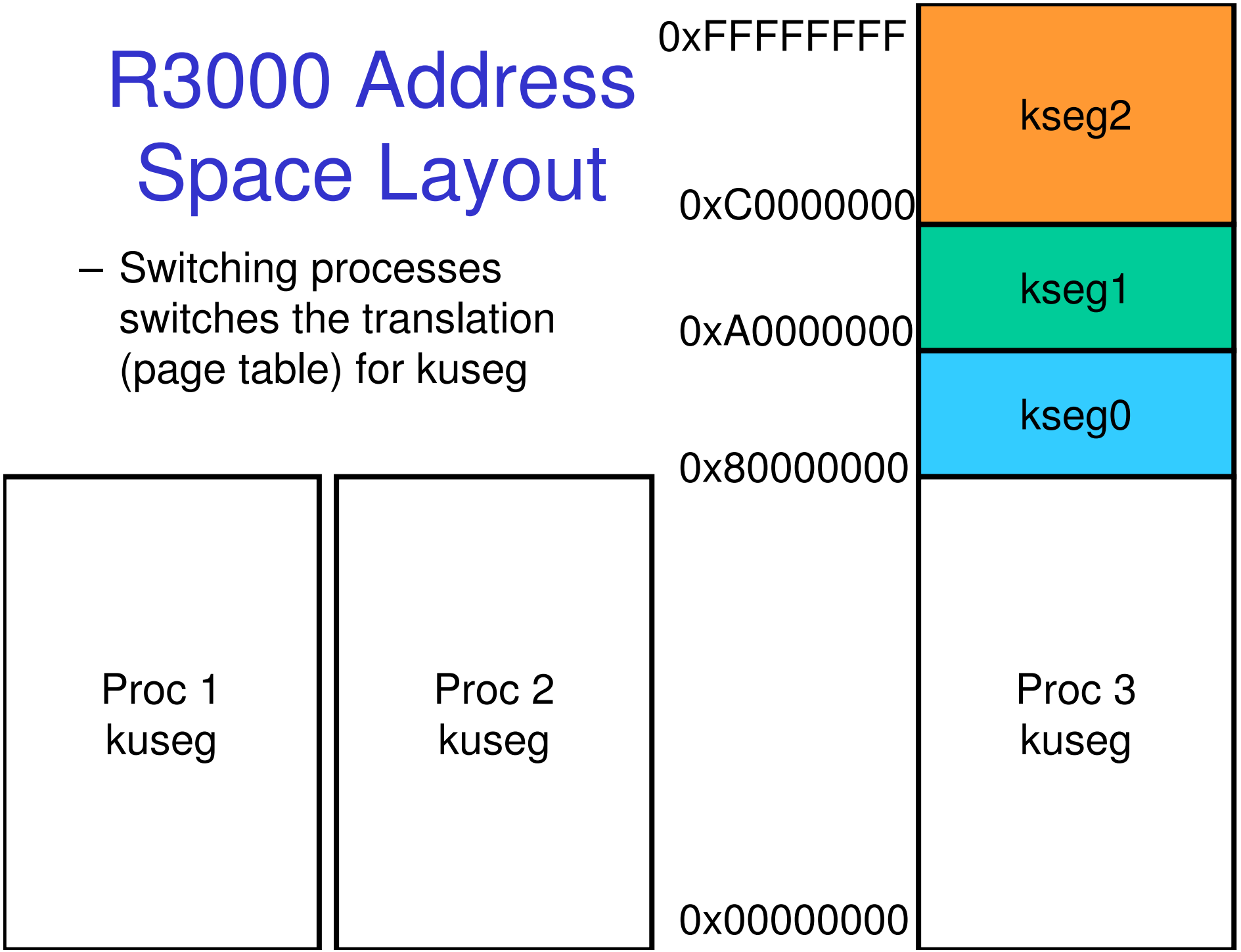
Assignment 2 tips

Structure of a Computer System



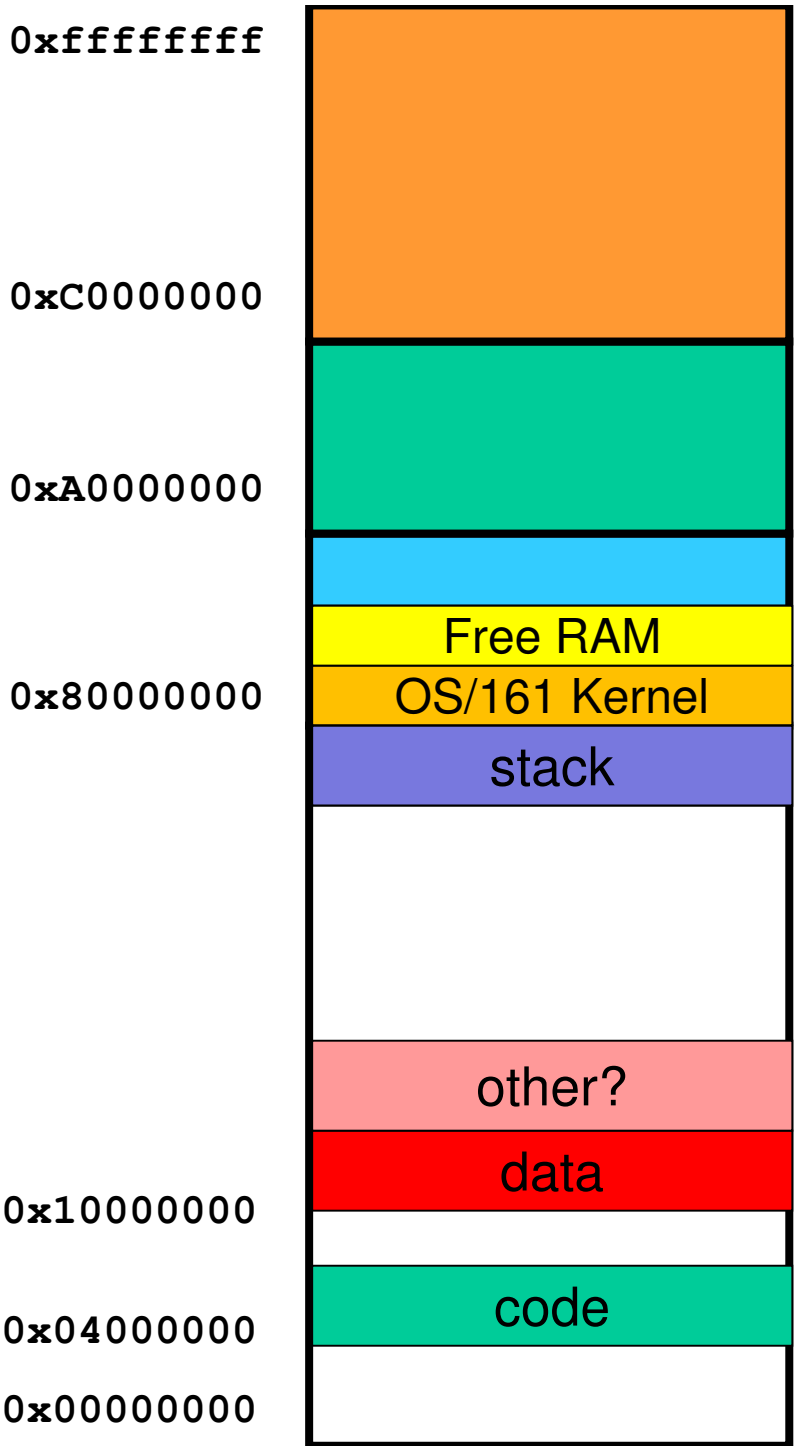
R3000 Address Space Layout

- Switching processes switches the translation (page table) for kuseg



Process Layout

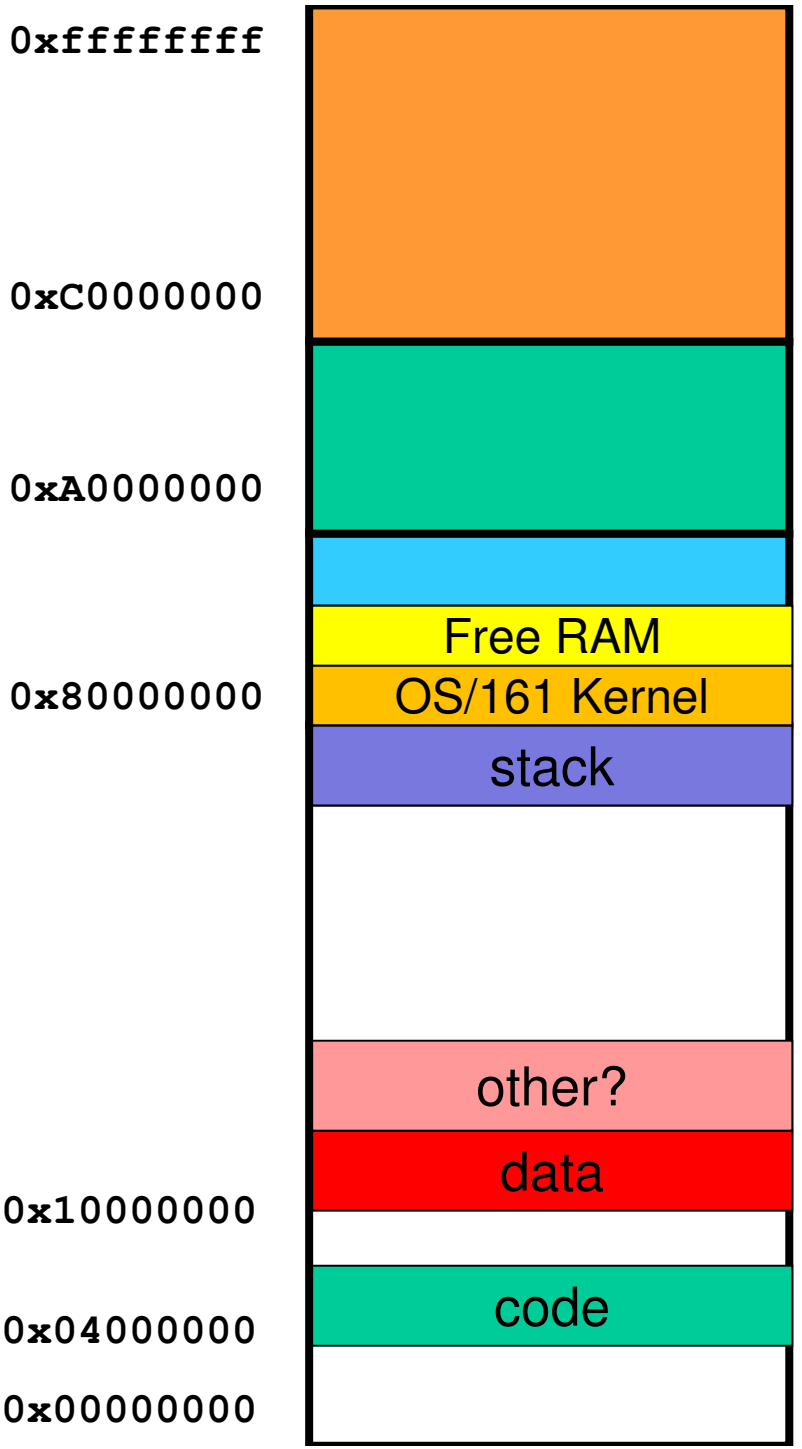
- Where is asst2 code/data (from `asst2.c`)?



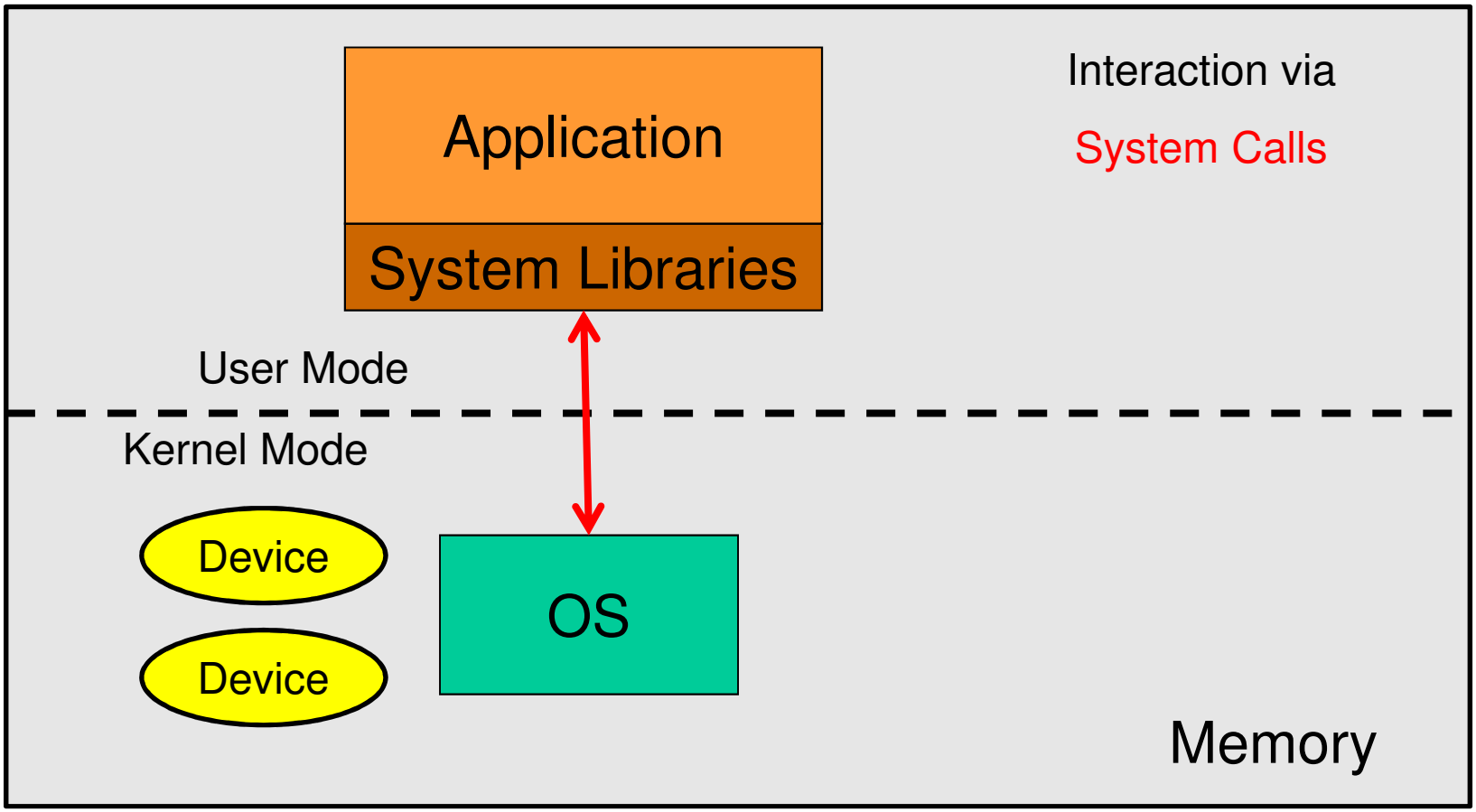
Calling open()

```
int open(const char *filename,
                int flags, ...);
```

- Where is “open()”?



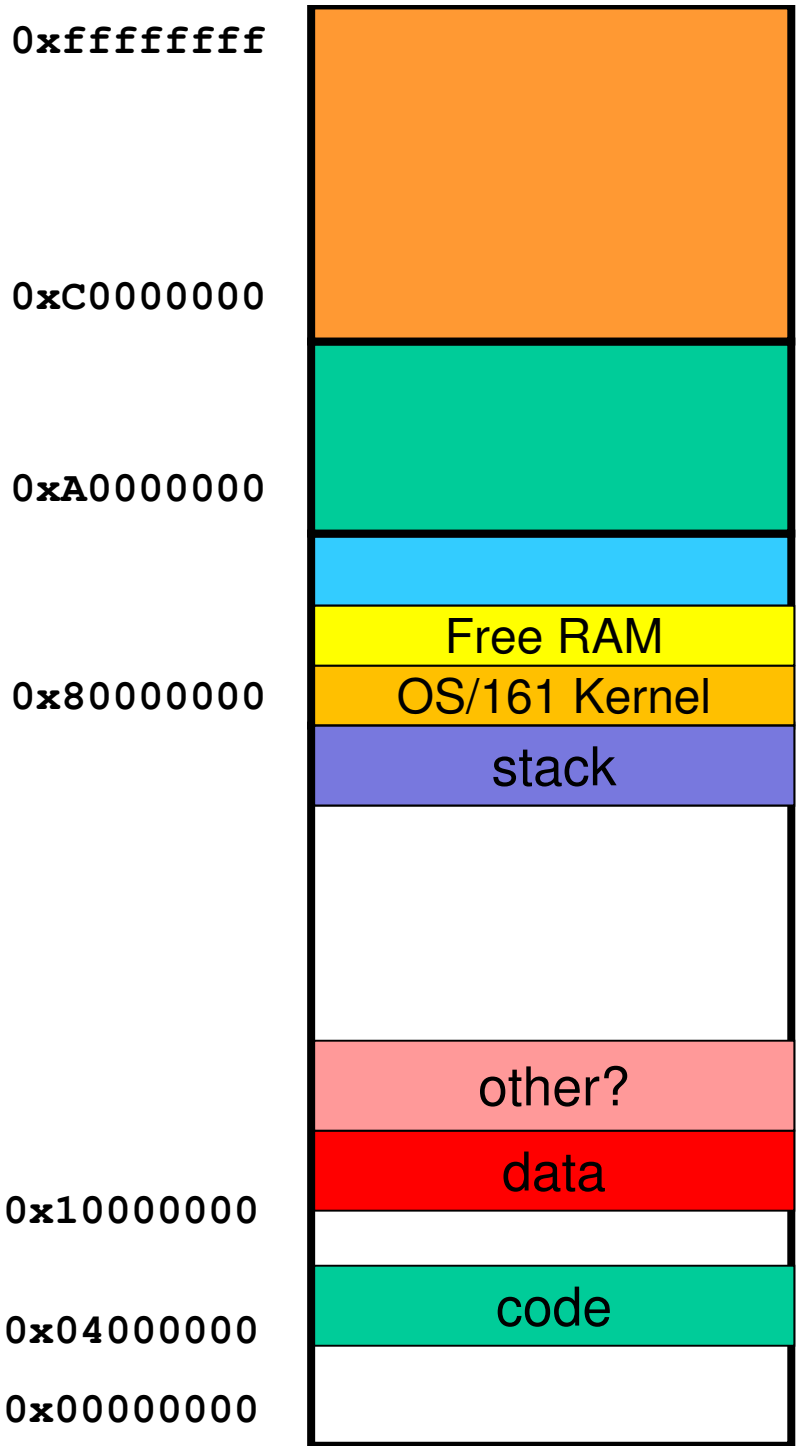
Structure of a Computer System

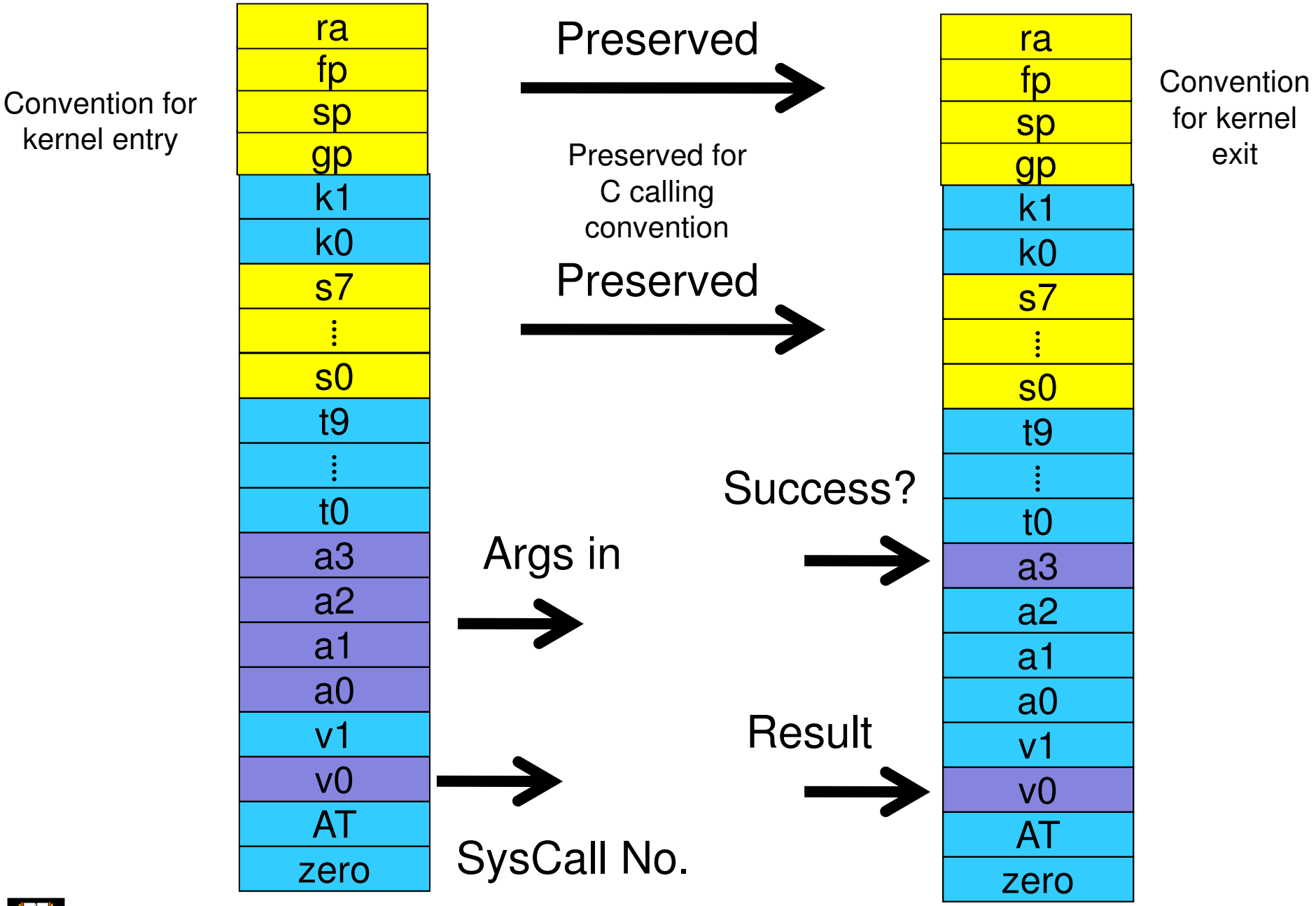


open()?

```
int open(const char *filename,
                int flags, ...);
```

- Where is “open()’s” implementation?
- By convention, it’s called `sys_open()` in the kernel.




Convention for kernel entry
ra
fp
sp
gp
k1
k0
s7
⋮
s0
t9
⋮
t0
a3
a2
a1
a0
v1
v0
AT
zero
Preserved
Preserved for C calling convention
Preserved
Args in
SysCall No.
Success?
Result
ra
fp
sp
gp
k1
k0
s7
⋮
s0
t9
⋮
t0
a3
a2
a1
a0
v1
v0
AT
zero
Convention for kernel exit

```
syscall(struct trapframe *tf)
{
   callno = tf->tf_v0;
   retval = 0;

   switch (callno) {
       case SYS_reboot:
        err = sys_reboot(tf->tf_a0);
        break;

       /* Add stuff here */

       default:
        kprintf("Unknown syscall %d\n", callno);
        err = ENOSYS;
        break;
   }
```

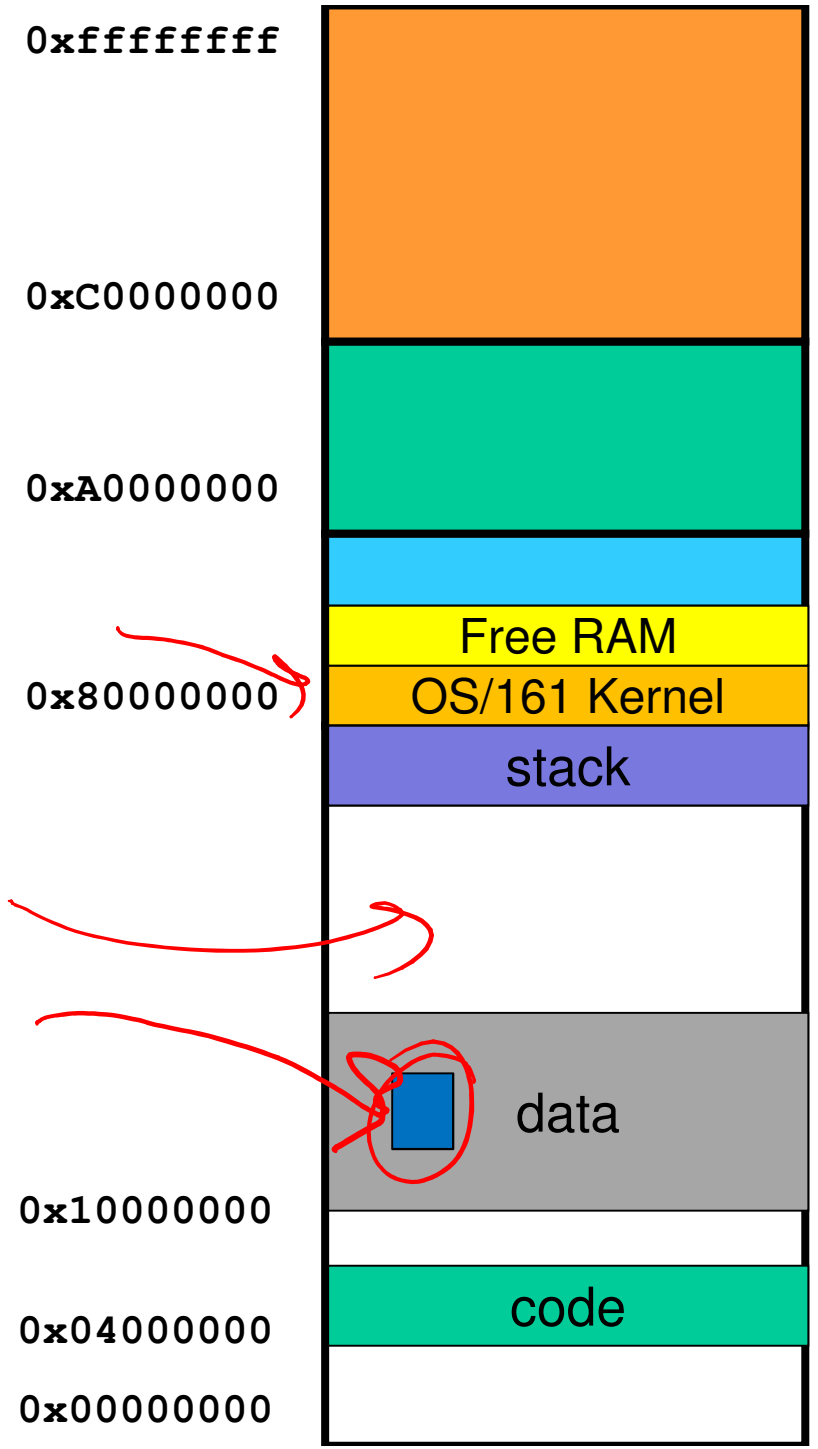
```
    if (err) {
        tf->tf_v0 = err;
        tf->tf_a3 = 1;      /* signal an error */
    }
    else {
        /* Success. */
        tf->tf_v0 = retval;
        tf->tf_a3 = 0;      /* signal no error */
    }

    tf->tf_epc += 4;

}
```

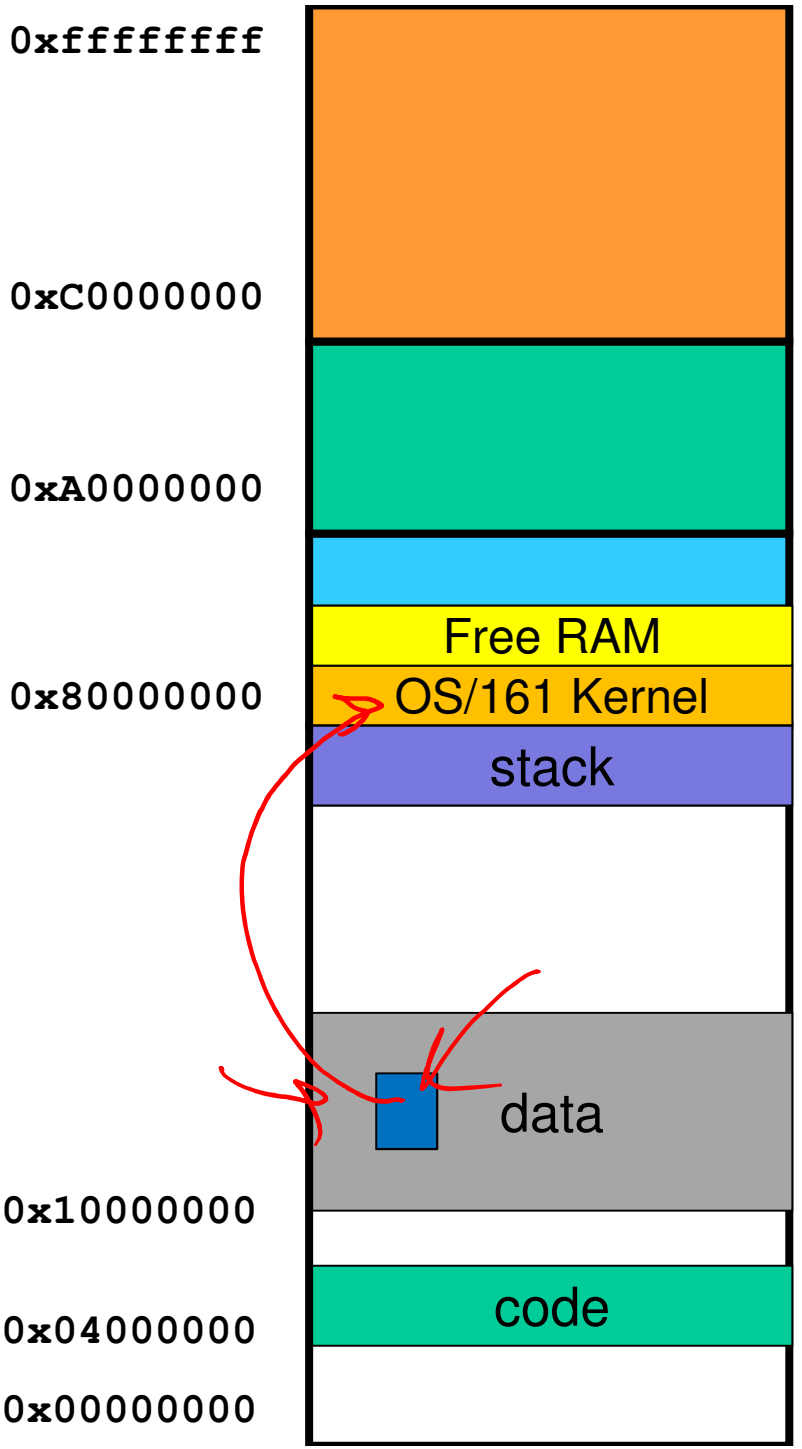
Pointers

- What about the first argument to open()
  - It's a string?



Copy in/out(str)

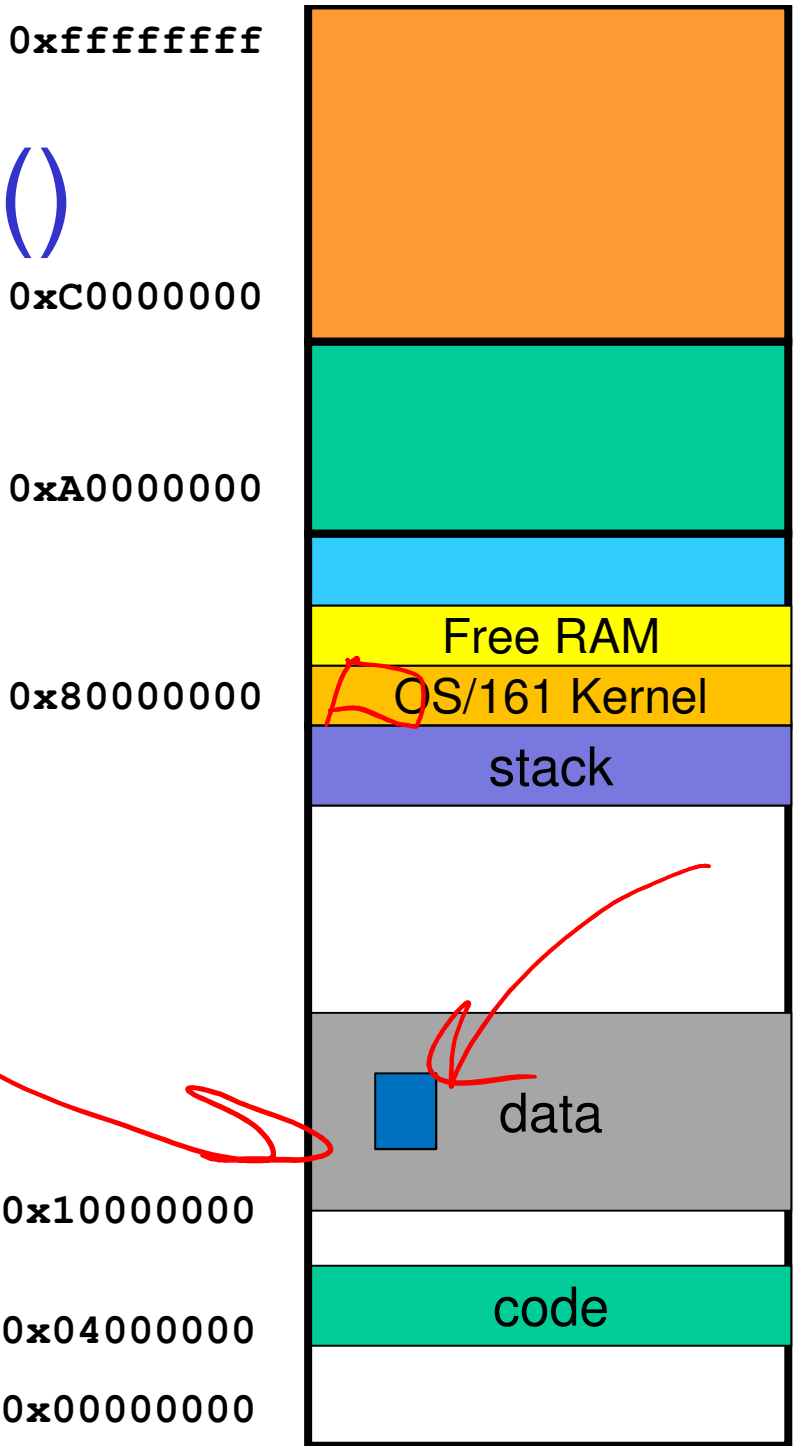
```
int copyin(const_userptr_t usersrc, void *dest,
        size_t len);
int copyout(const void *src, userptr_t userdest,
        size_t len);
int copyinstr(const_userptr_t usersrc, char
        *dest, size_t len, size_t *got);
int copyoutstr(const char *src, userptr_t
        userdest, size_t len, size_t *got);
```



Buffers – e.g. read()

- Kernel framework for safely handling buffers
  - Does error/range/validity checking for you

```
struct iovec {
        union {
        userptr_t  iov_ubase;   /* user-supplied pointer */
        void       *iov_kbase;  /* kernel-supplied pointer */
        };
        size_t iov_len;         /* Length of data */
};
```



UIO

```
/* Source/destination. */
enum uio_seg {
        UIO_USERISPACE,                 /* User process code. */
        UIO_USERSPACE,                  /* User process data. */
        UIO_SYSSPACE,                   /* Kernel. */
};

struct uio {
        struct iovec    *uio_iov;       /* Data blocks */
        unsigned         uio_iovcnt;    /* Number of iovecs */
        off_t            uio_offset;    /* Desired offset into object */
        size_t           uio_resid;     /* Remaining amt of data to xfer */
        enum uio_seg     uio_segflg;    /* What kind of pointer we have */
        enum uio_rw      uio_rw;        /* Whether op is a read or write */
        struct addrspace *uio_space;    /* Address space for user pointer */
};
```

Sample Helper function

```
uio_uinit(struct iovec *iov, struct uio *u, userptr_t buf,
size_t len, off_t offset, enum uio_rw rw)
{
        iov->iov_ubase = buf;
        iov->iov_len = len;
        u->uio_iov = iov;
        u->uio_iovcnt = 1;
        u->uio_offset = offset;
        u->uio_resid = len;
        u->uio_segflg = UIO_USERSPACE;
        u->uio_rw = rw;
        u->uio_space = proc_getas();
}
```

System call implementation

1. sys_open()
2. sys_close()
3. sys_read()
4. sys_write()
5. sys_lseek()
6. sys_dup2()

1. vfs_open()
   - copyinstr()
2. vfs_close()
3. VOP_READ()
4. VOP_WRITE()
5. VOP_ISSEEKABLE()
   - VOP_STAT()
6.

lseek() Offset

```
uint64_t offset;
int whence;
off_t retval64;

join32to64(tf->tf_a2, tf->tf_a3, &offset);

copyin((userptr_t)tf->tf_sp + 16, &whence,
sizeof(int));

split64to32(retval64, &tf->tf_v0, &tf->tf_v1);
```