

EEE3052 Introduction to Operating Systems
prof. Jinkyu Jeong

Project 2. Virtual memory

2017.9.25 (Mon.)

TAs

안민우(minwoo.ahn@cs.skku.edu) /

송재현(jaehyun.song@cs.skku.edu)

Project plan

- Total 4 projects

- 1) Process management

- System call

- Scheduling

- 2) Virtual memory

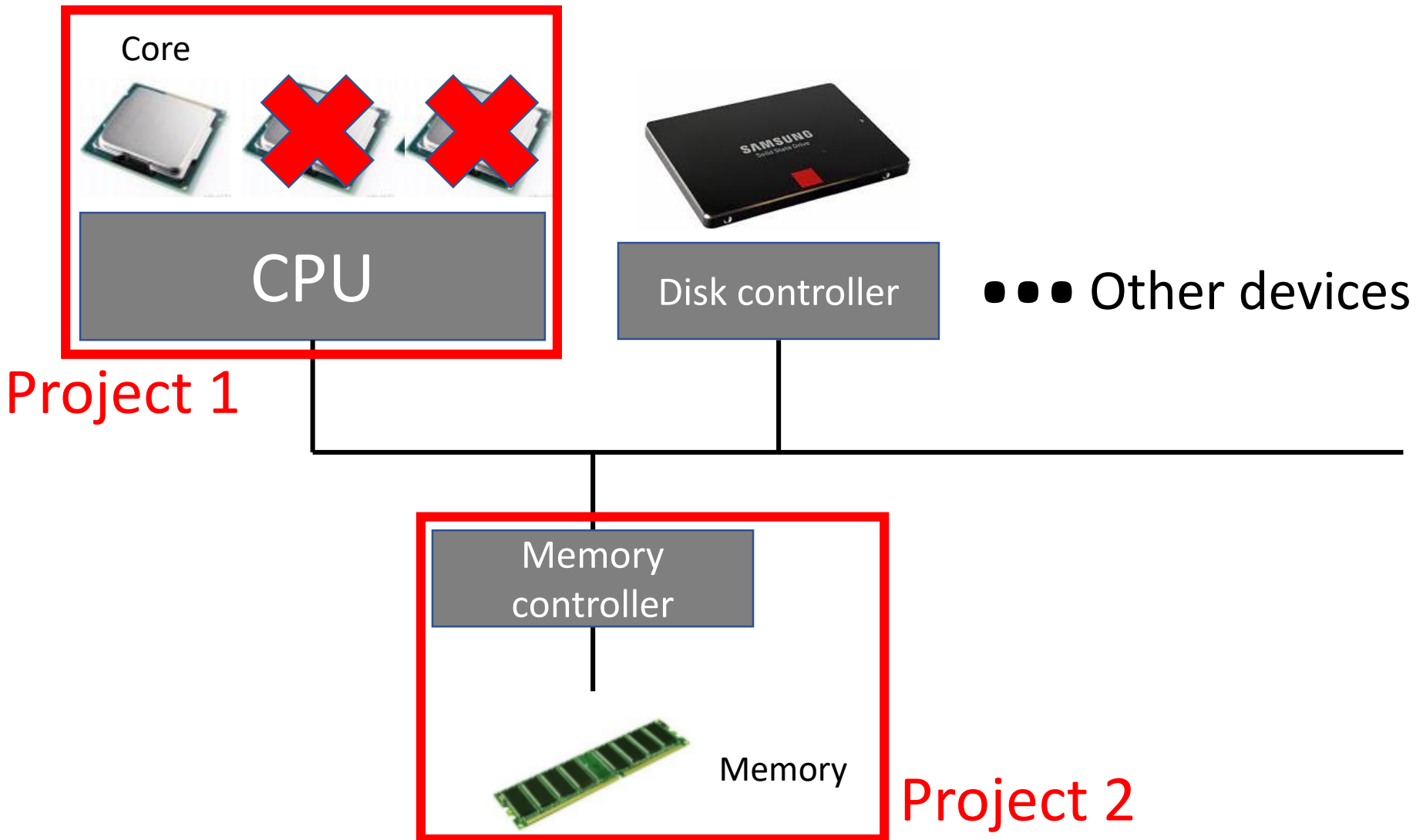
- Stack growth (Due: 9/25 ~ 10/8)

- COW(Copy-on-write) (Due: 10/9 ~ TBD)

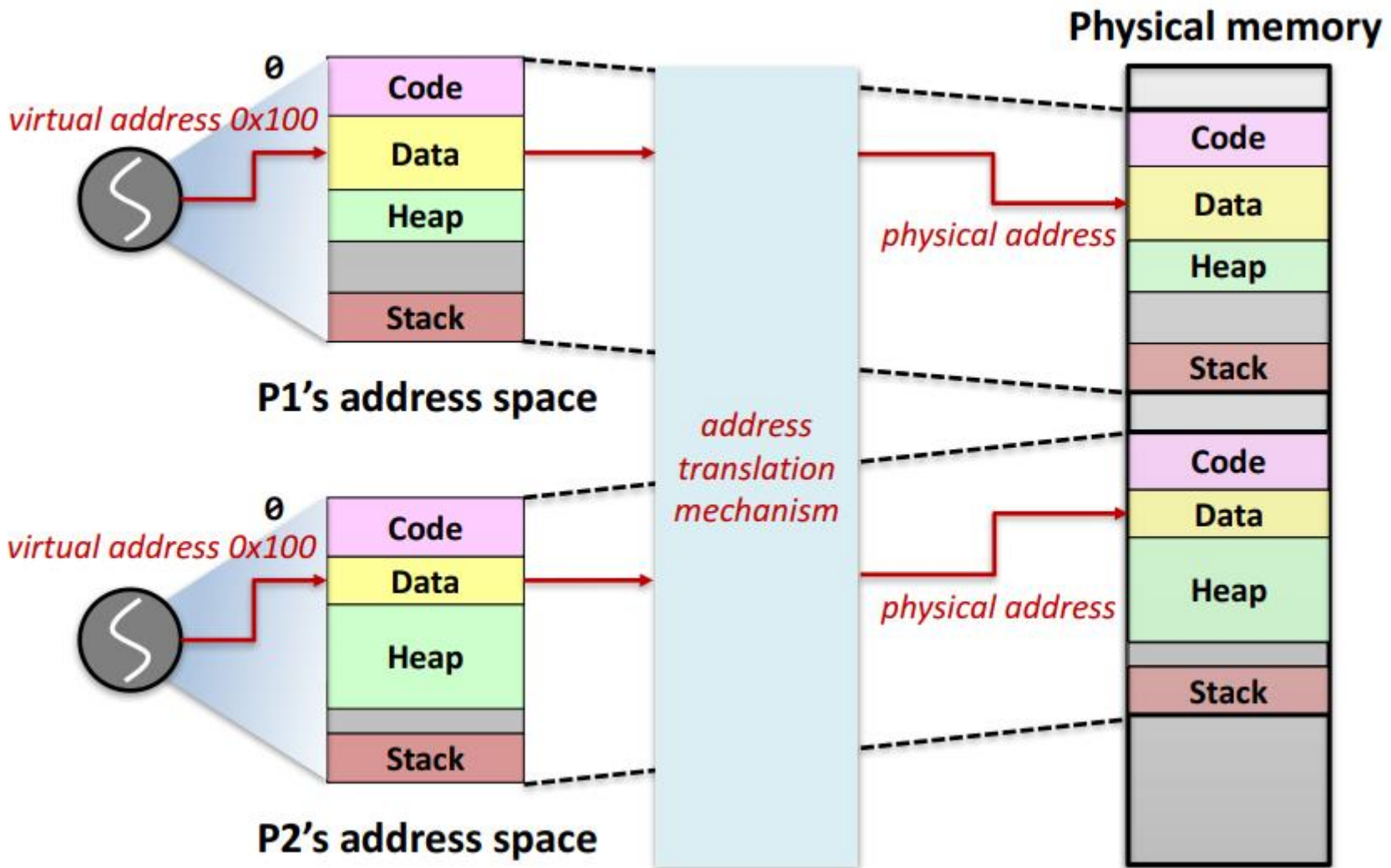
- 3) Synchronization

- 4) File system

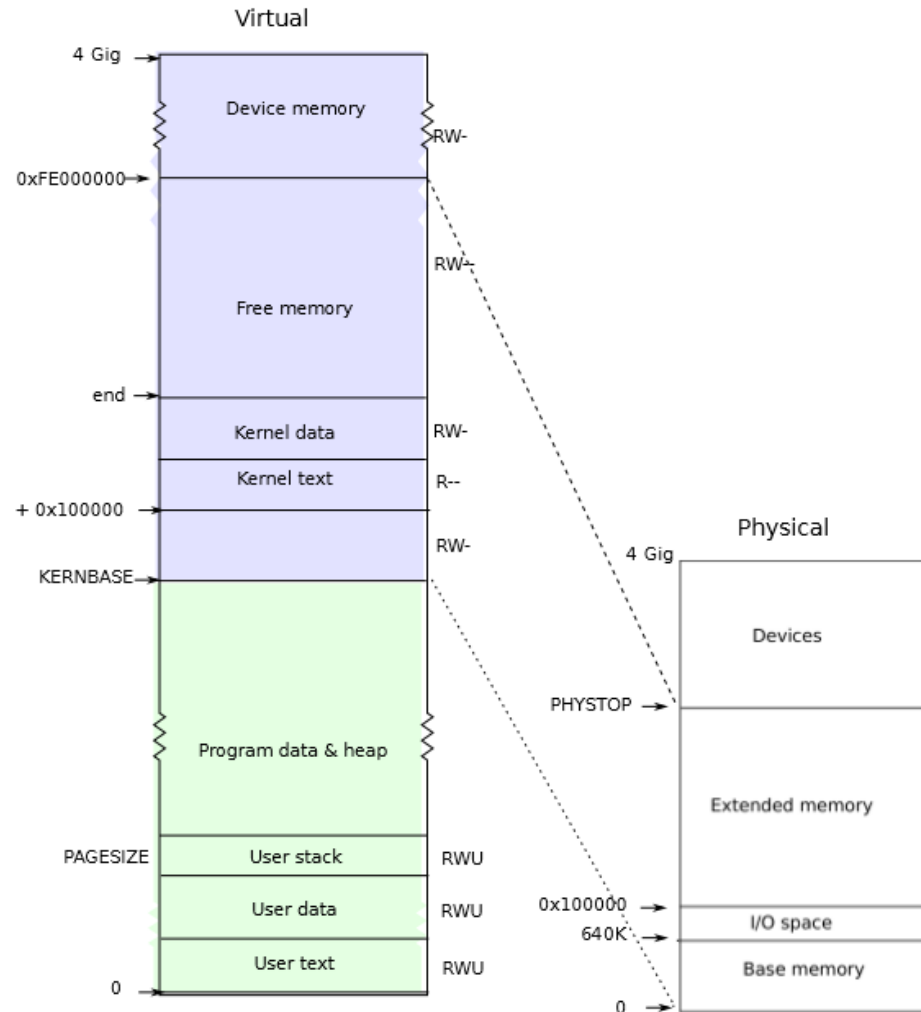
Computer system organization



What is VM(Virtual memory)?

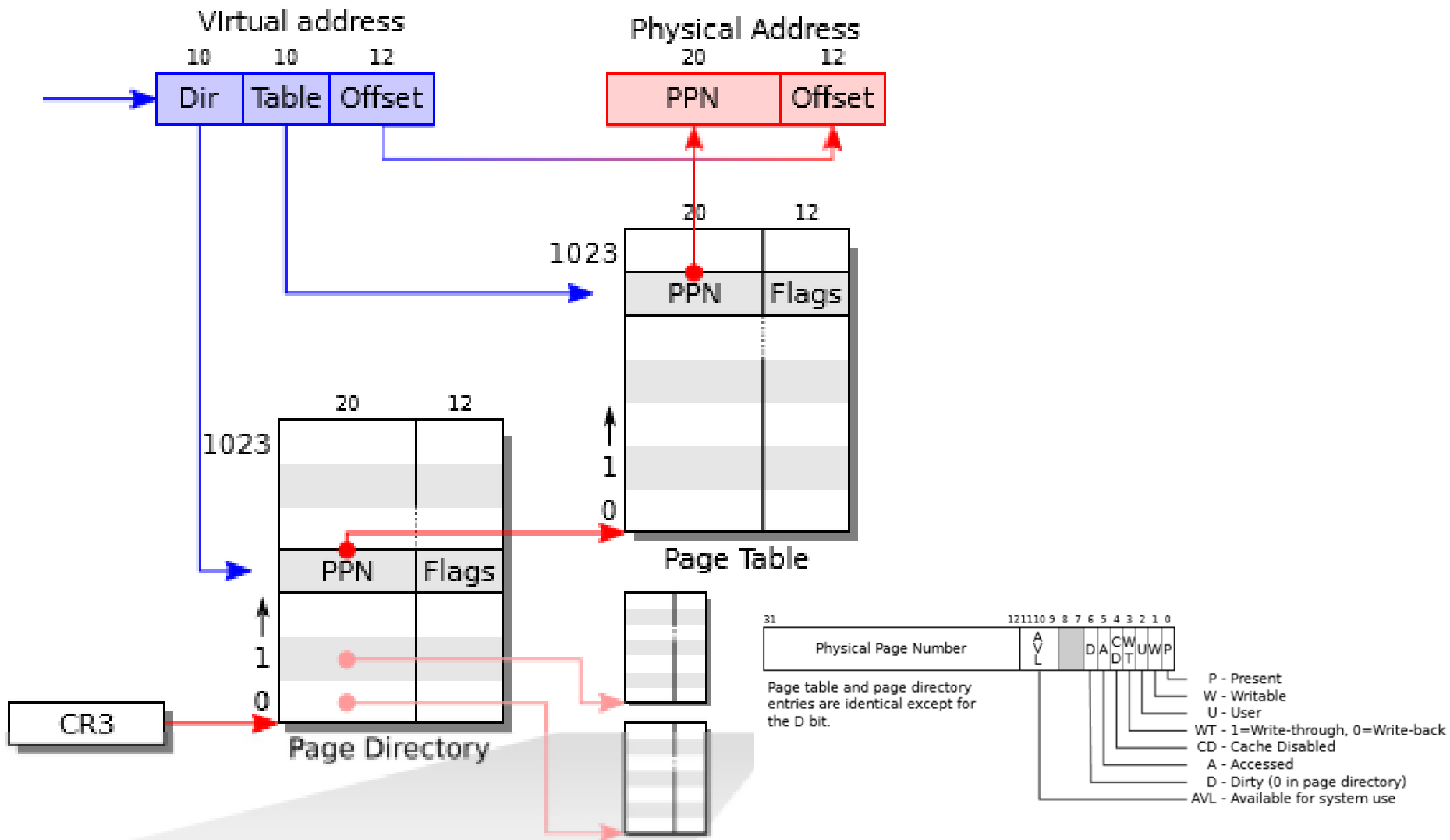


Virtual address space

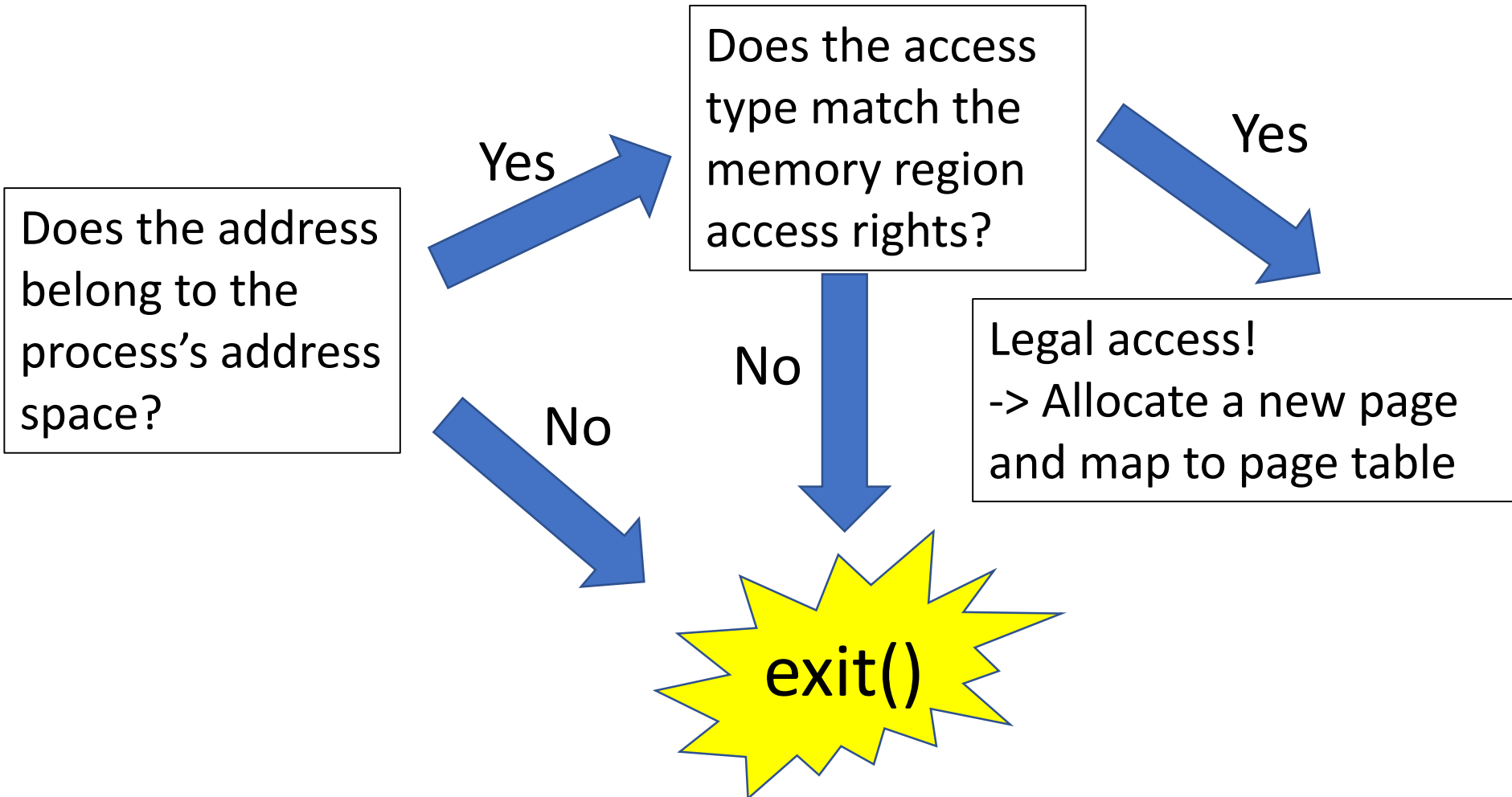


Layout of a virtual address space and the physical address space

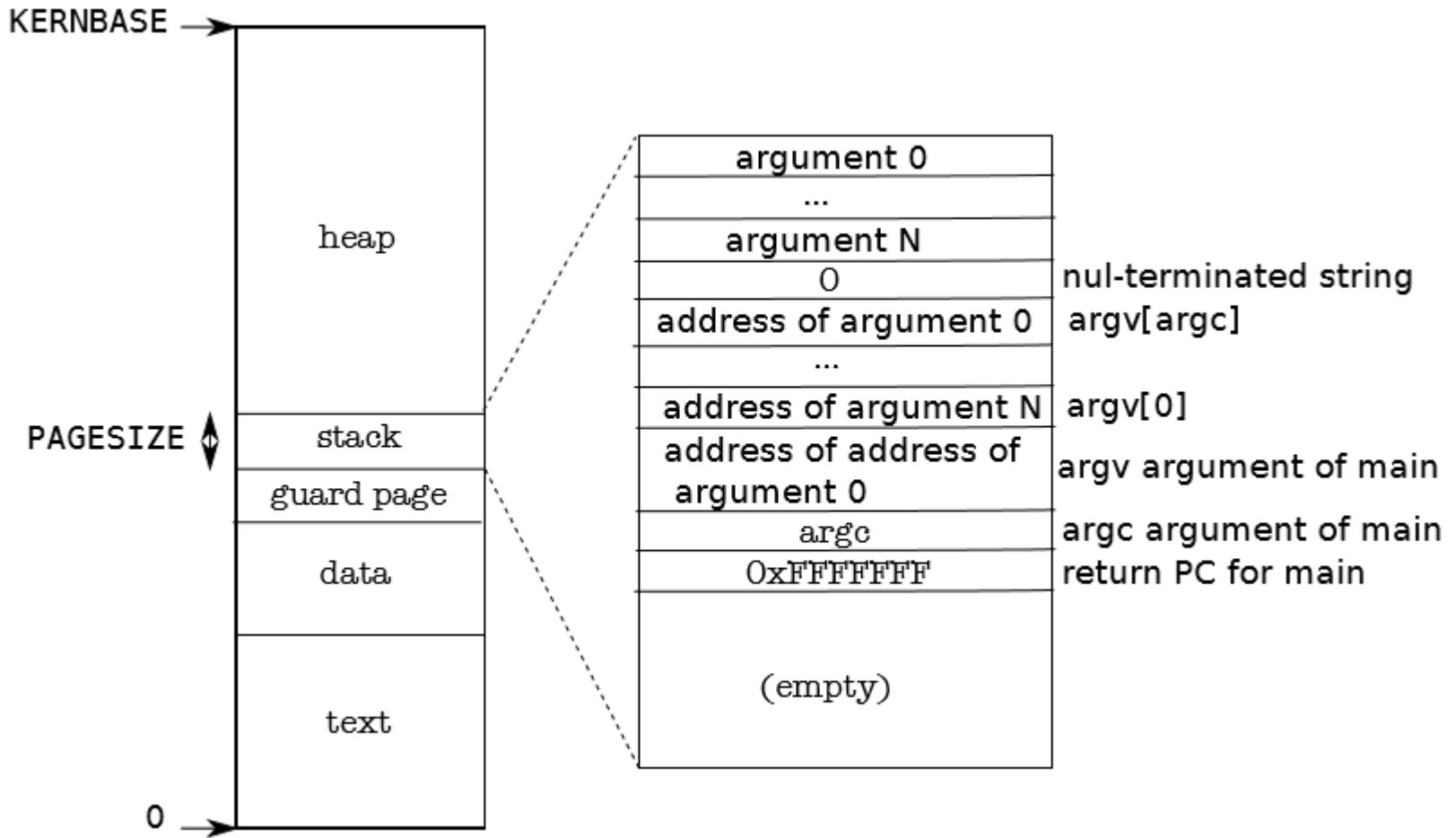
Address translation in Intel x86



Page fault scenario



User address space in xv6



Memory layout of a user process with its initial stack

proc.h

```
// Per-process state
struct proc {
    uint sz;                // Size of process memory (bytes)
    pde_t* pgdir;          // Page table
    char *kstack;          // Bottom of kernel stack for this process
    enum procstate state;  // Process state
    int pid;               // Process ID
    struct proc *parent;   // Parent process
    struct trapframe *tf; // Trap frame for current syscall
    struct context *context; // swtch() here to run process
    void *chan;            // If non-zero, sleeping on chan
    int killed;            // If non-zero, have been killed
    struct file *ofile[NOFILE]; // Open files
    struct inode *cwd;     // Current directory
    char name[16];         // Process name (debugging)
    int nice;
};
```

main.c

```
int
main(void)
{
    kinit1(end, P2V(4*1024*1024)); // phys page allocator
    kvmalloc(); // kernel page table
    mpinit(); // detect other processors
    lapicinit(); // interrupt controller
    seginit(); // segment descriptors
    picinit(); // disable pic
    ioapicinit(); // another interrupt controller
    consoleinit(); // console hardware
    uartinit(); // serial port
    pinit(); // process table
    tvinit(); // trap vectors
    binit(); // buffer cache
    fileinit(); // file table
    ideinit(); // disk
    startothers(); // start other processors
    kinit2(P2V(4*1024*1024), P2V(PHYSTOP)); // must come after startothers()
    userinit(); // first user process
    mpmain(); // finish this processor's setup
}
```

Page fault handler

traps.h

```

x86 trap and interrupt constants.

// Processor-defined:
#define T_DIVIDE      0      // divide error
#define T_DEBUG      1      // debug exception
#define T_NMI        2      // non-maskable interrupt
#define T_BRKPT      3      // breakpoint
#define T_OFLOW      4      // overflow
#define T_BOUND      5      // bounds check
#define T_ILLOP      6      // illegal opcode
#define T_DEVICE      7      // device not available
#define T_DBLFLT     8      // double fault
// #define T_COPROC   9      // reserved (not used since 486)
#define T_TSS        10     // invalid task switch segment
#define T_SEGNP      11     // segment not present
#define T_STACK      12     // stack exception
#define T_GPFLT      13     // general protection fault
#define T_PGFLT      14     // page fault
// #define T_RES      15     // reserved
#define T_FPERR      16     // floating point error
#define T_ALIGN      17     // alignment check
#define T_MCHK       18     // machine check
#define T_SIMDERR    19     // SIMD floating point error

// These are arbitrarily chosen, but with care not to overlap
// processor defined exceptions or interrupt vectors.
#define T_SYSCALL    64     // system call
#define T_DEFAULT    500    // catchall

#define T_IRQ0       32     // IRQ 0 corresponds to int T_IRQ

#define IRQ_TIMER    0
#define IRQ_KBD      1
#define IRQ_COM1     4
#define IRQ_IDE      14
#define IRQ_ERROR    19
#define IRQ_SPURIOUS 31

```

- If page fault occurs, “trapno” of trapframe automatically filled with T_PGFLT and call trap function in trap.c
- You have to make your “own” page fault handler!
- Currently, implemented as below...
 - rcr2() -> page fault address

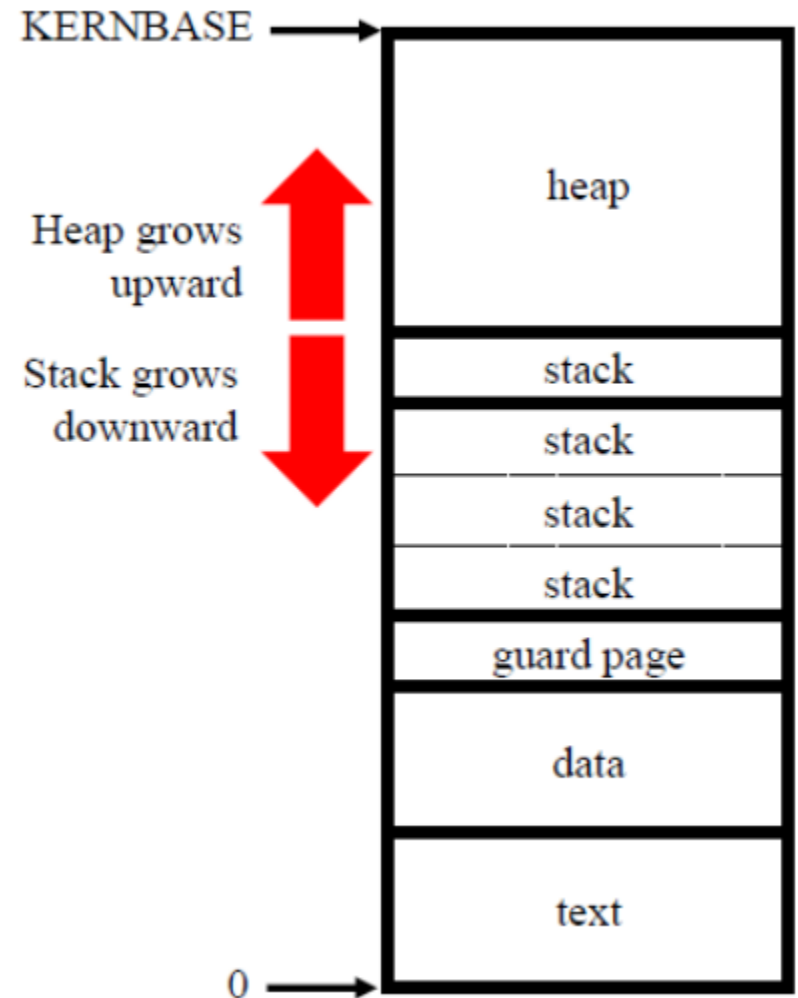
```

//PAGEBREAK: 13
default:
    if(myproc() == 0 || (tf->cs&3) == 0){
        // In kernel, it must be our mistake.
        printf("unexpected trap %d from cpu %d eip %x (cr2=0x%x)\n",
            tf->trapno, cpuid(), tf->eip, rcr2());
        panic("trap");
    }
    // In user space, assume process misbehaved.
    printf("pid %d %s: trap %d err %d on cpu %d "
        "eip 0x%x addr 0x%x--kill proc\n",
        myproc()->pid, myproc()->name, tf->trapno,
        tf->err, cpuid(), tf->eip, rcr2());
    myproc()->killed = 1;
}

```

Project 2-1. Stack growth

- Size of stack
 - 1 page initially
 - Can be grow up to 4 pages
- Growth of stack
 - `tf->esp` can move upto 32bytes
 - You page should be allocated to this process if current stack is full
- When stack pointer reaches guard page, stack overflow occurs and process is **killed**



Compile test cases

- Erase `-Werror` in Makefile

```
CC = $(TOOLPREFIX)gcc
AS = $(TOOLPREFIX)gas
LD = $(TOOLPREFIX)ld
OBJCOPY = $(TOOLPREFIX)objcopy
OBJDUMP = $(TOOLPREFIX)objdump
CFLAGS = -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer
#CFLAGS = -fno-pic -static -fno-builtin -fno-strict-aliasing -fvar-tracking -fvar-tracking-assignments -O0 -g -Wall -MD -gdwarf-2 -m32 -Werror -fno-omit-frame-pointer
CFLAGS += $(shell $(CC) -fno-stack-protector -E -x c /dev/null >/dev/null 2>&1 && echo -fno-stack-protector)
ASFLAGS = -m32 -gdwarf-2 -Wa,-divide
# FreeBSD ld wants `elf_i386_fbsd'
LDFLAGS += -m $(shell $(LD) -V | grep elf_i386 2>/dev/null | head -n 1)
```

Submission

- Compress your code as YourStudentID-2_1.tar.gz
- Send your file to “minwoo.ahn@csi.skku.edu”
- **PLEASE DO NOT COPY**
 - **YOU WILL GET -100 POINTS OF YOUR PROJECT IF YOU COPIED**
- Due date: 10/8(Sun.), 23:59:59 PM

Grade policy

- If you failed to pass “oral test”, you will get 0
- You have to explain reason why you pass each test cases (If not, we consider it as failed to pass test cases)

Tips

- Reading xv6-book will help you a lot
 - <http://csl.skku.edu/uploads/EEE3052F17/book-rev10.pdf>
 - Chapter 2. Page tables(p. 29 ~ 36)

Questions

- If you have “any” questions, please email to TA
- You can also visit Semiconductor Building #400509
 - Please email TA before visiting