SSE3044 Introduction to Operating Systems

Prof. Jinkyu Jeong

Project 2. Virtual Memory

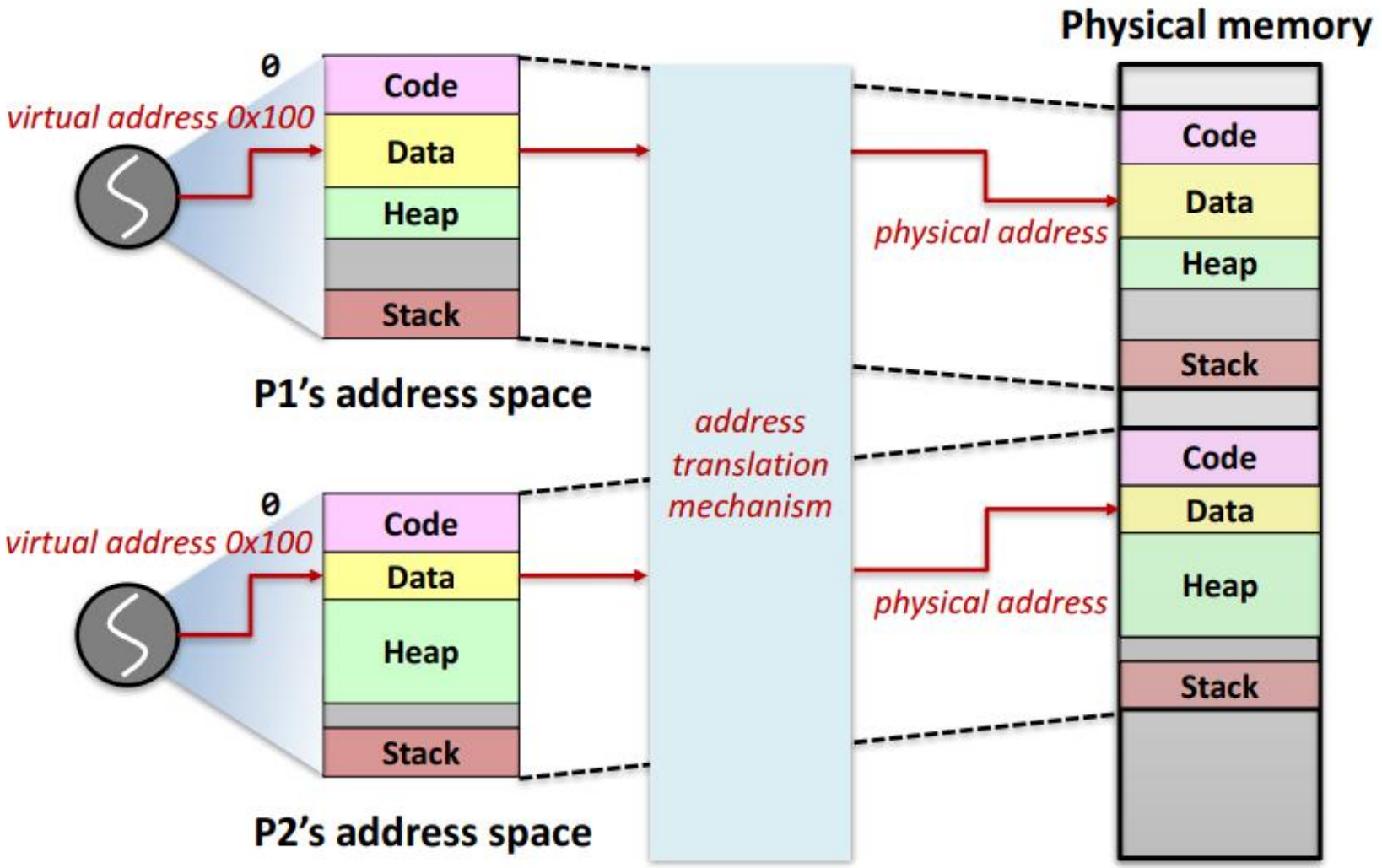
2018.04.11 (Wed.)

TAs
이규선(gyusun.lee@csl.skku.edu) /
안민우(minwoo.ahn@csl.skku.edu)

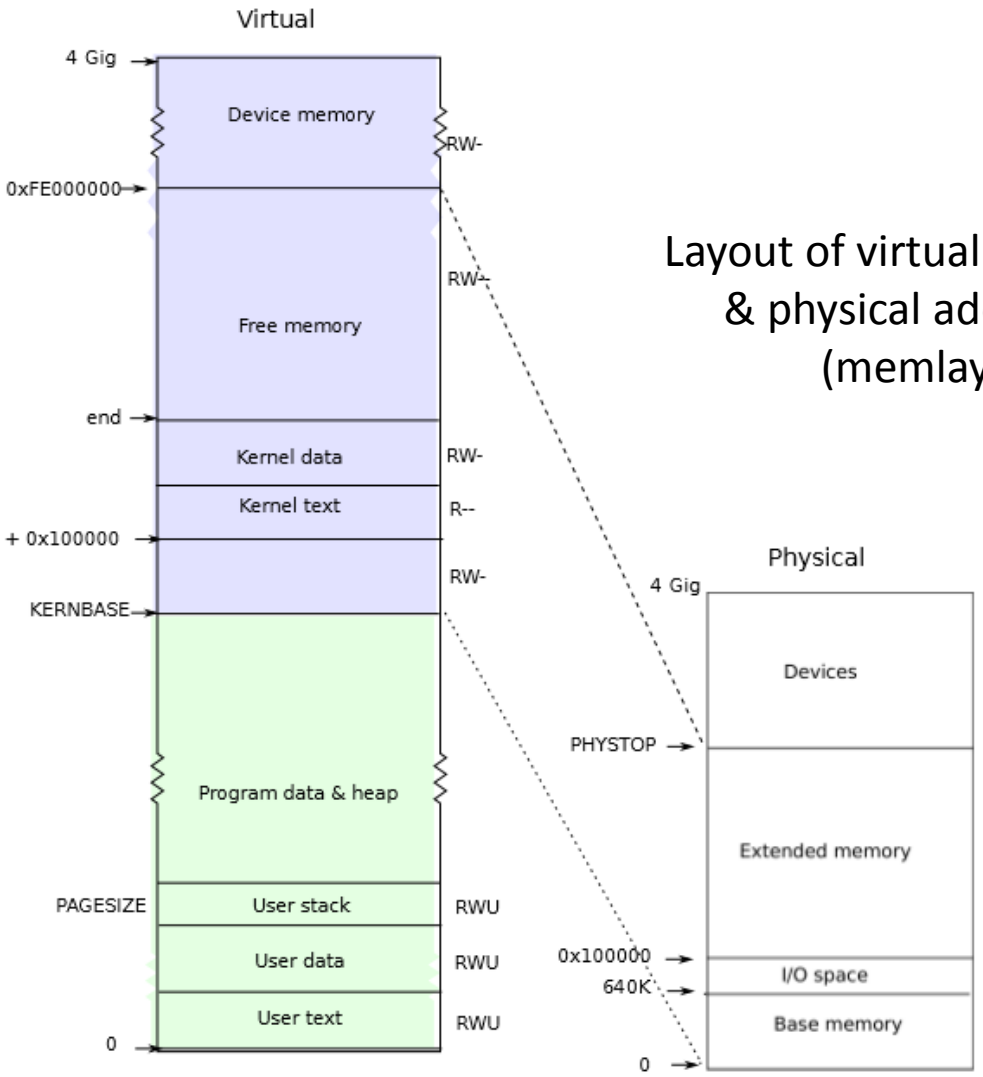
Project Plan

- Total 4 projects
  1) Process management
     - System call
     - Priority scheduler
  2) Virtual memory
     - Stack growth (Due: 4/11 ~ 5/1)
     - COW(Copy-on-write) (Due: 5/2 ~ TBD)
  3) Synchronization
  4) File system

What is VM(Virtual memory)?



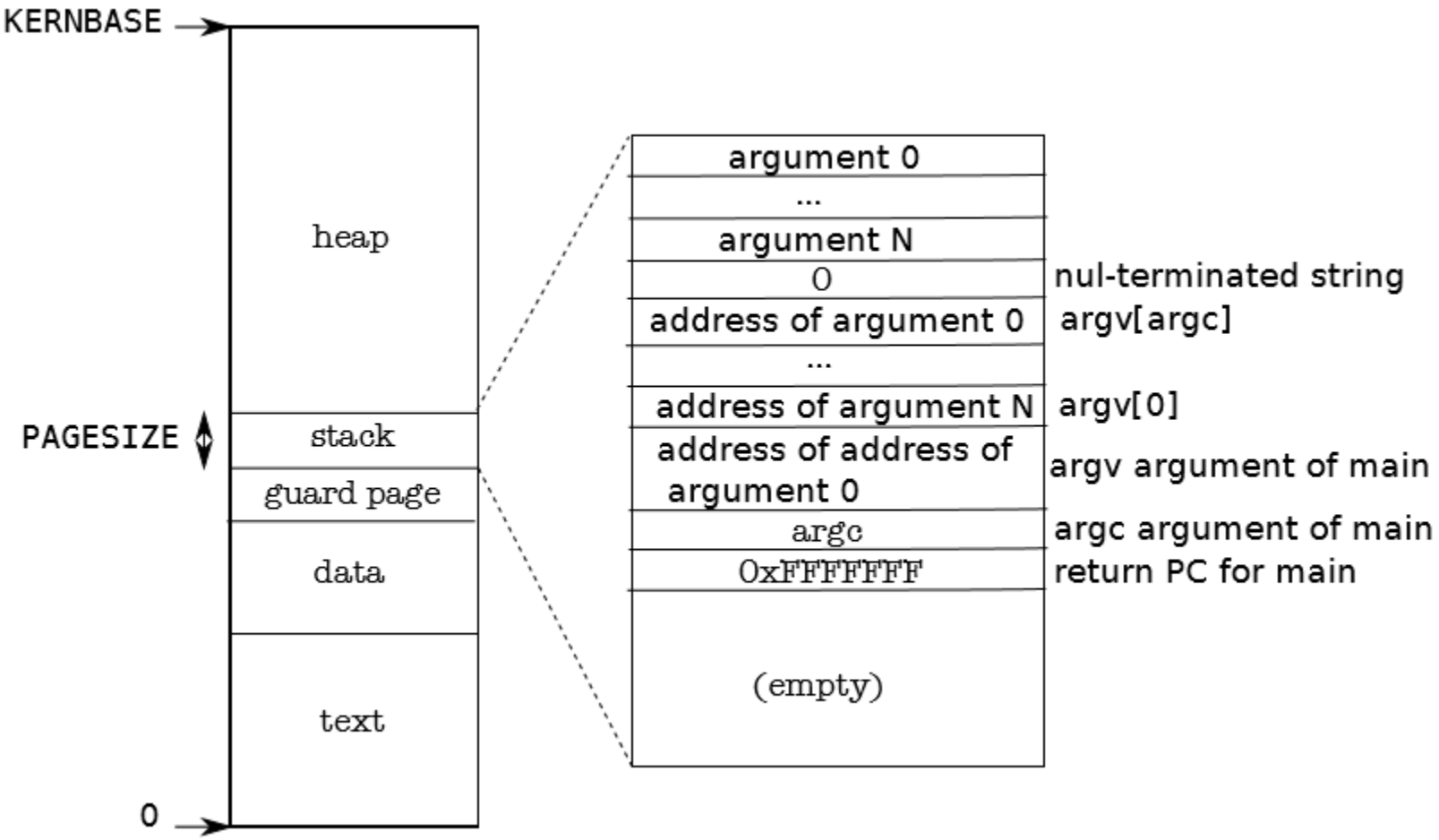
Virtual Address Space (32-bit)



Layout of virtual address space
& physical address space
(memlayout.h)

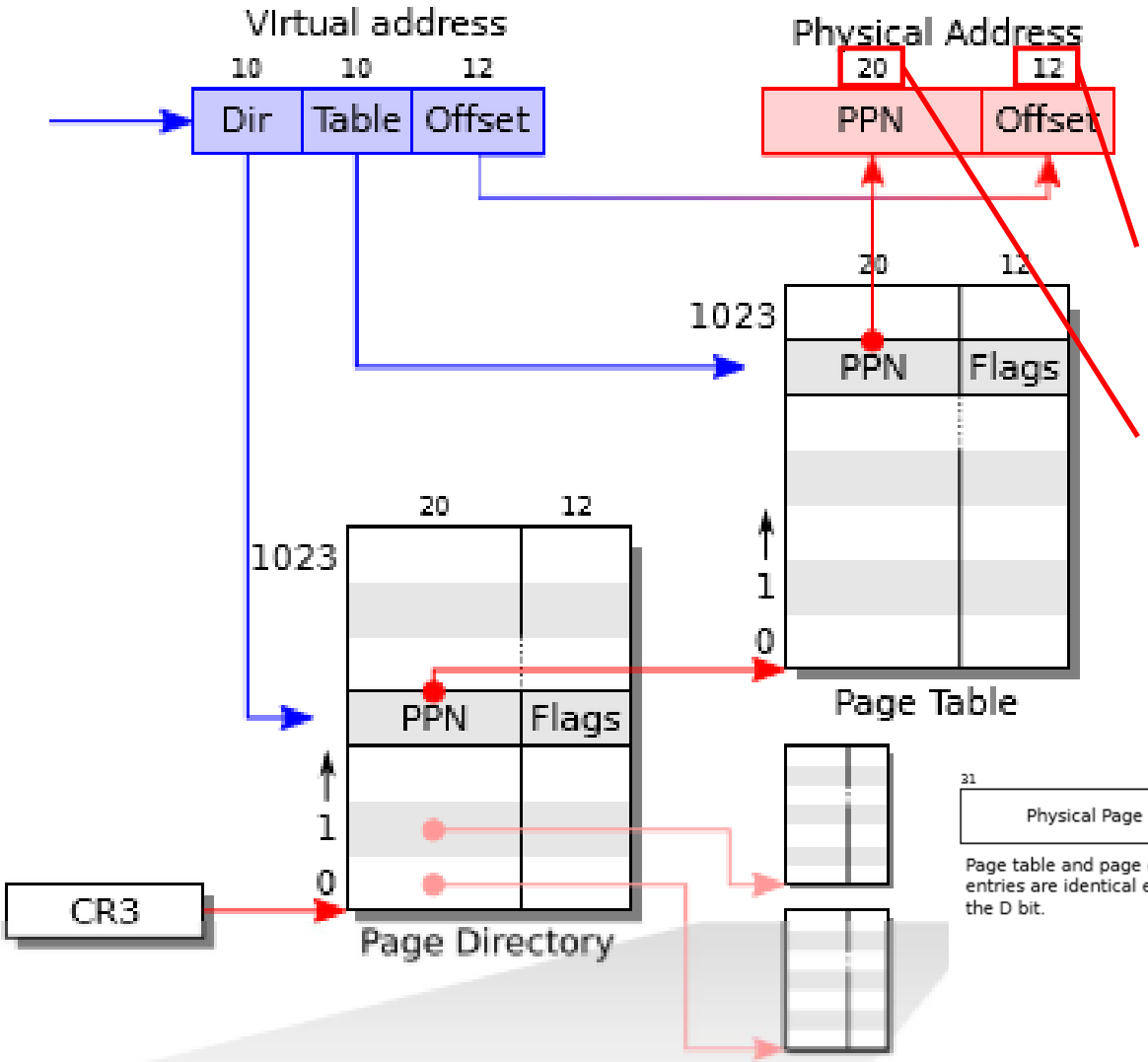
32bit machine
& 4GB memory

User Address Space in Xv6



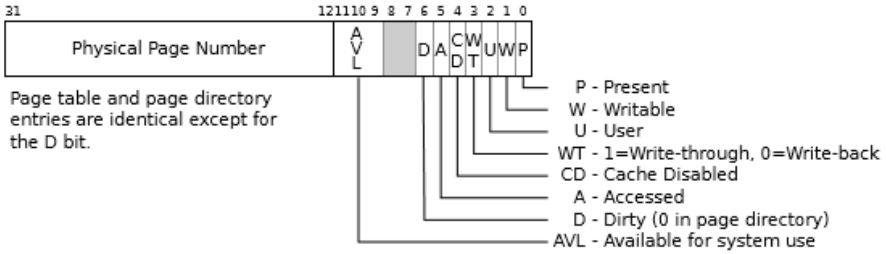
Memory layout of a user process with its initial stack

Address Translation in Intel x86

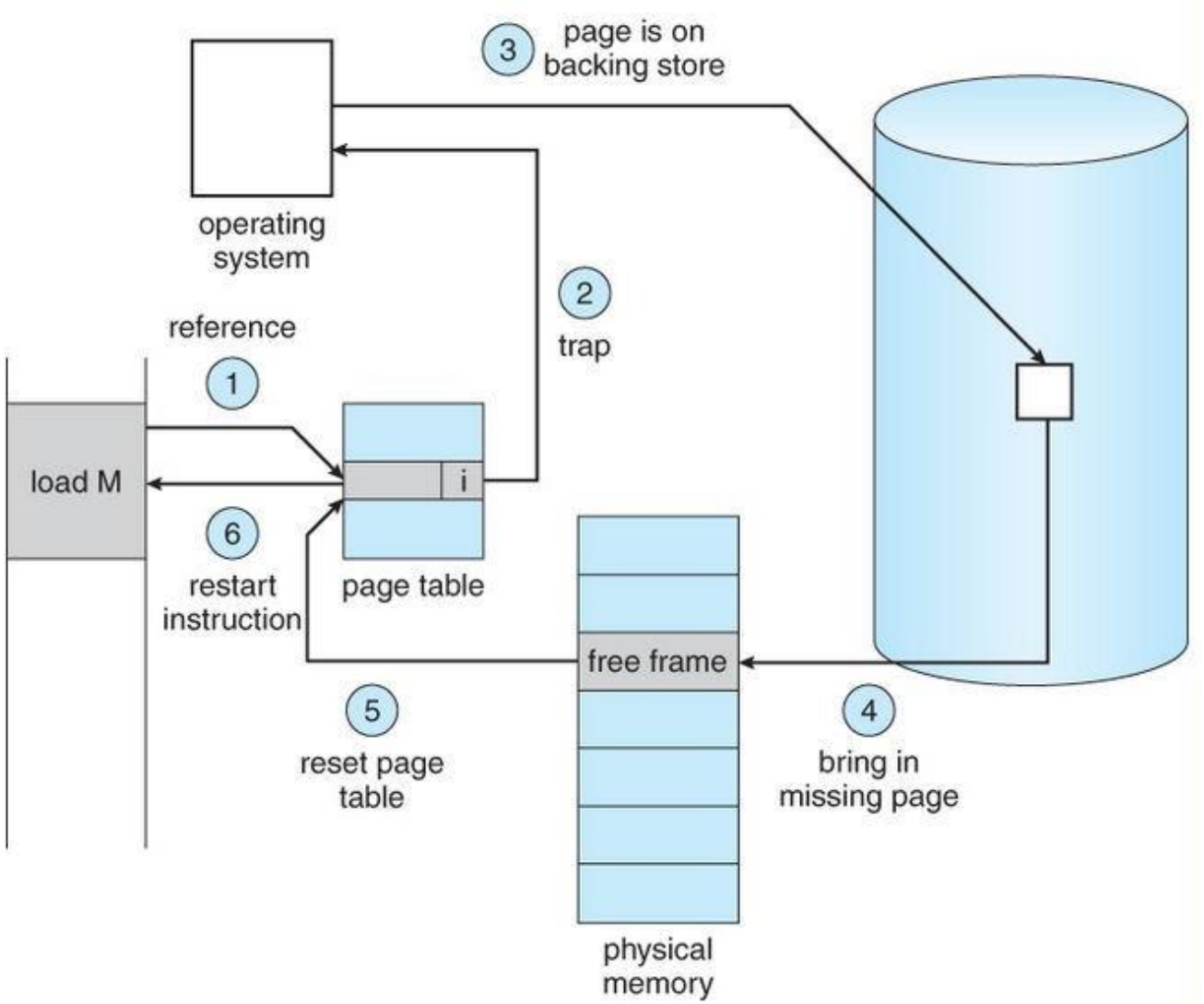


Why is the Offset index of PA 20?
- PAGESIZE 4096bytes

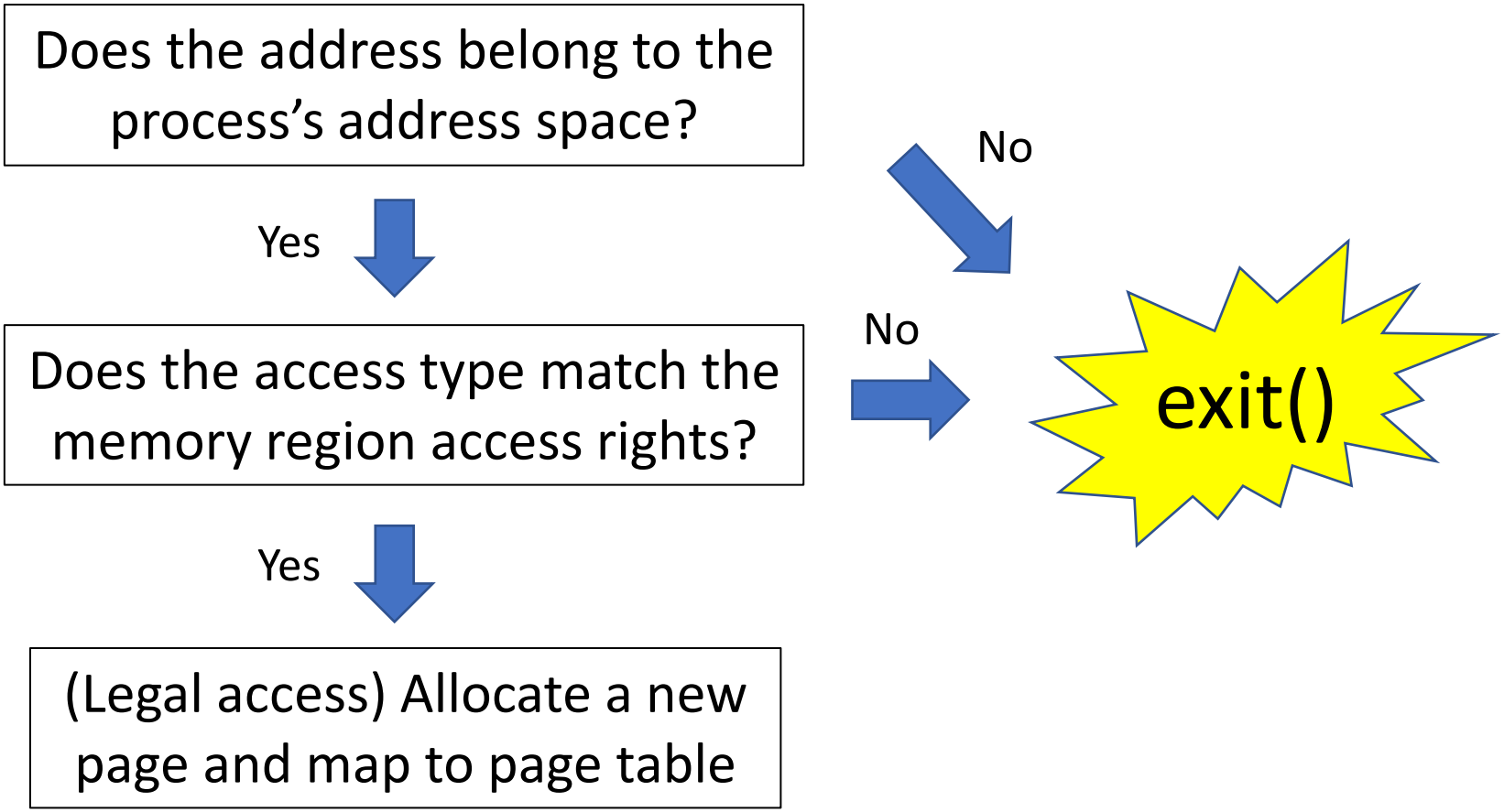
Why is the PPN index of PA 20?
- Memory(4G) / PAGESIZE



Page-fault Scenario



Page-fault Scenario



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 in Xv6

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      // aligment 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

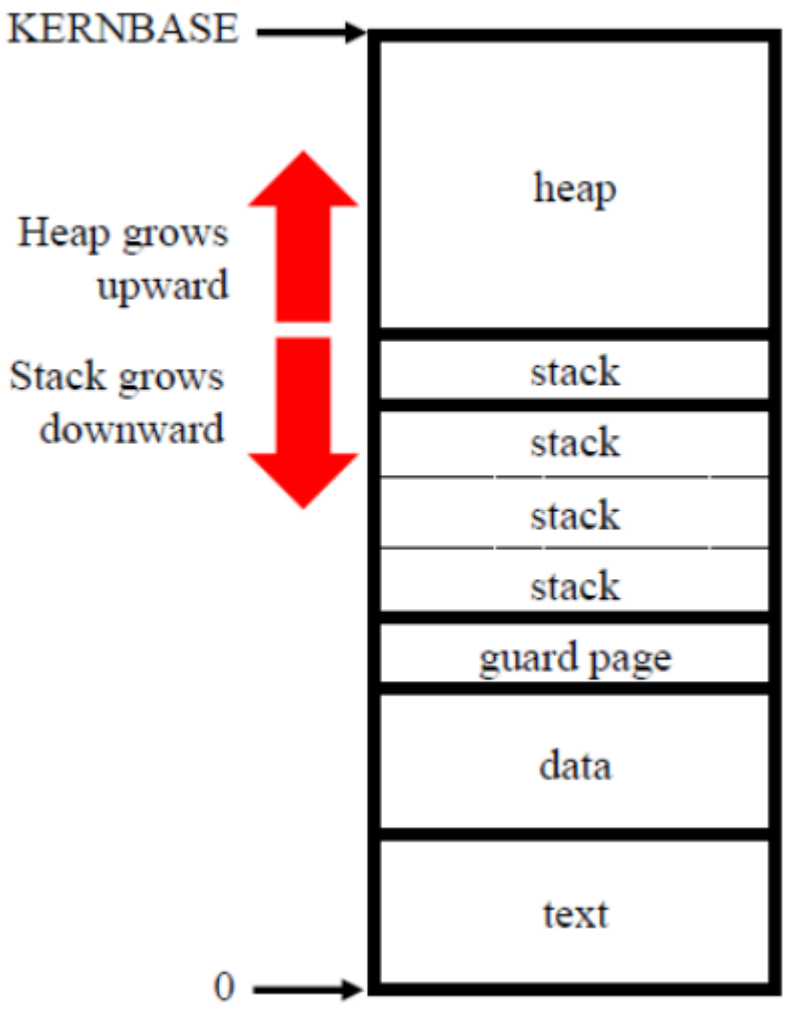
```
//PAGEBREAK: 13
default:
  if(myproc() == 0 || (tf->cs&3) == 0){
    // In kernel, it must be our mistake.
    cprintf("unexpected trap %d from cpu %d eip %x (cr2=0x%x)\n",
            tf->trapno, cpuid(), tf->eip, rcr2());
    panic("trap");
  }
  // In user space, assume process misbehaved.
  cprintf("pid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

Page fault handler in Xv6

- Requirement
  - Print out “Invalid access” when approaching abnormal address
  - Print out “Allocate page” if page-fault handler is excuted normally

Project 2-1. Stack growth

- Initial size of stack
  - Prepare 1 page initially
  - Can be grow up to 4 pages
- Growth of stack
  - tf->esp can move upto 32bytes
  - New page should be allocated to this process if current stack is full
- When stack pointer reaches guard page, occurs stack overflow and kill process



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 xv6 folder as YourStudentID-2-1.tar.gz
  - tar cvf 2016710580-2-1.tar.gz xv6-SSE3044
- Send your tar.gz file to gyusun.lee@csl.skku.edu
  - Please command $make clean, before submission
  - Please send mail with uniformized title
    - [SSE3044]YourStudentID-2-1
- PLEASE DO NOT COPY
  - YOU WILL GET F GRADE IF YOU COPIED
- Due date: 5/1(Tue.), 23:59:59 PM
  - -25% per day for delayed submission

Grade policy

- If you failed to pass “oral test”, you will got 0
- Decribe how you have implemented to 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 questions, please email to TA
- You can also visit Semiconductor Building #400509
  - Please email TA before visiting