

实验五-RV64缺页异常处理以及fork机制

姓名:王晶晶 学号:3200104880 学院:计算机科学与技术学院 课程名称: 计算机系统Ⅲ

实验时间: 2022.5.18 实验地点: 紫金港东4-509 指导老师: 周亚金

一、实验目的和要求

- 通过 `vm_area_struct` 数据结构实现对进程多区域虚拟内存的管理。
- 在 Lab4 实现用户态程序的基础上，添加缺页异常处理 Page Fault Handler。
为进程加入 fork 机制，能够支持通过 fork 创建新的用户态进程。

二、实验原理

2.1 `vm_area_struct`

`vm_area_struct` 是虚拟内存管理的基本单元，`vm_area_struct` 保存了有关连续虚拟内存区域(简称vma)的信息。

每个进程都有自己对应的vma链表。它将该进程对应的所有虚拟地址空间按一定的顺序串联起来。从而维护进程的虚拟地址空间。一个vma存储了一段虚拟地址区域的起始和结束地址，以及对应的地址读写和执行的权限。

通过这种机制，可以让编程者在编写程序时只要关注虚拟地址，而不用关注具体虚拟地址映射到的实际物理地址。

2.2 缺页异常 Page Fault

缺页异常是一种正在运行的程序访问当前未由内存管理单元（MMU）映射到虚拟内存的页面时，由计算机硬件引发的异常类型。访问未被映射的页或访问权限不足，都会导致该类异常的发生。处理缺页异常通常是操作系统内核的一部分。当处理缺页异常时，操作系统将尝试使所需页面在物理内存中的位置变得可访问（建立新的映射关系到虚拟内存）。而如

果在非法访问内存的情况下，即发现触发 Page Fault 的虚拟内存地址（ Bad Address ）不在当前进程 vm_area_struct 链表中所定义的允许访问的虚拟内存地址范围内，或访问位置的权限条件不满足时，缺页异常处理将终止该程序的继续运行。

2.3 fork 系统调用

fork()通过复制当前进程创建一个新的进程，新进程称为子进程，而原进程称为父进程。子进程和父进程在不同的内存空间上运行。父进程fork成功时返回：子进程的pid，子进程返回：0。fork失败则父进程返回：-1。创建的子进程需要拷贝父进程 task_struct、pgd、mm_struct 以及父进程的 user stack 等信息。

三、实验代码实现

3.1 维护vma虚拟内存管理结构

- 初始化进程vma链表，将进程代码段和用户栈的虚拟空间加入vma结构中

```
do_mmap(task[i]->mm,USER_START,uapp_end - uapp_start,VM_READ | VM_WRITE | VM_EXEC
do_mmap(task[i]->mm,USER_END - PGSIZE,PGSIZE,VM_READ | VM_WRITE);
```

- 如果传入的虚拟地址空间已经在vma中存在，则需要另外寻找空闲的虚拟地址空间加入链表

```
while(!add_link(mm,addr,length,prot)){
    addr=get_unmapped_area(mm,length);
}
```

3.2 page fault异常处理

- 设定异常入口处理程序，遇到系统调用号12，13，15转入page_fault处理程序

Interrupt	Exception Code	Description
0	12	Instruction Page Fault
0	13	Load Page Fault
0	15	Store/AMO Page Fault

```
if(x==0xC || x==0xD || x==0xF)
    do_page_fault(scause, sepc, regs);
```

- do_page_fault先检查发生缺页异常的虚拟地址是否在vma链表中

```
struct vm_area_struct* found=find_vma(current->mm, stval);
```

- 并检查其权限是否和Exception Code对应的权限相吻合

```
if(scause ==12&&(found->vm_flags&VM_EXEC)==0) {
    printk("Invalid vm area in page fault\n");
    return;
} else if(scause==13&&(found->vm_flags&VM_READ)==0){
    printk("Invalid vm area in page fault\n");
    return;
} else if(scause==15&&(found->vm_flags&VM_WRITE)==0){
    printk("Invalid vm area in page fault\n");
    return;
}
```

- 如果以上检查都满足，就建立页表映射

- 如果是用户代码段，则由于代码已经load进物理地址中，无需kalloc，可直接建立页表映射

```
create_mapping((unsigned long)current->pgd+PA2VA_OFFSET,USER_START,(unsigned
```

- 如果是用户栈段，考虑到**fork出子进程的用户栈**在fork的时候就已经被分配了物理页，考虑处理的统一性，在初始时就为进程分配好了用户栈的物理页，并将**物理页的虚拟栈底地址**存入user_sp中，也无需进行kalloc()

```
create_mapping((unsigned long)current->pgd+PA2VA_OFFSET,USER_END-PGSIZE,pa,Pt
```

- 其它情形则需要先kalloc()再进行页表设置

```
create_mapping((unsigned long)current->pgd+PA2VA_OFFSET,found->vm_start,pa,Pt
```

3.3 fork处理函数设置

- 设置fork系统调用入口处理

```

if(a7==SYS_CLONE){
    regs->x10=clone(regs);
}

```

- 在fork处理程序中创建子进程并加入所有进程的task数组中

```
task[sum]=(struct task_struct*)kalloc();
```

这里采用的是全局变量sum

- 初始化子进程的一些状态
 - sp应和父进程一致，由于父进程进入trap后，由于csrrw将sp和sscratch的值进行了交换，故而子进程的sp应从当前sscratch中获取

```
unsigned long sscratch=csr_read(sscratch);
```

- 拷贝用户栈的所有值

```

for(int i=0;i<512;i++){
    user_stack[i]=((unsigned long*)(USER_END-PGSIZE))[i];
    //printf("child user stack:0x%lx 0x%lx\n",(unsigned long*)(USER_END-PG'
}

```

- 拷贝vma链表

```

while(tmp){
    child=kalloc();
    child->vm_mm=task[sum]->mm;
    child->vm_start=tmp->vm_start;
    child->vm_end=tmp->vm_end;
    child->vm_flags=tmp->vm_flags;
    if(prev==NULL){
        task[sum]->mm->mmap=child;
    }
    else{
        prev->vm_next=child;
    }
    child->vm_prev=prev;
    prev=child;
    tmp=tmp->vm_next;
}
prev->vm_next=NULL;

```

- 设置返回值a0为0，并修改sp为当前用户栈的值

```
task[sum]->trapframe->x2=csr_read(sscratch);
task[sum]->trapframe->x10=0;
```

3.4 fork返回

- 需要根据传入的regs结构，来恢复子进程的上下文

```
ld t0, 0(a0)
csrw sstatus, t0
ld t0, 8(a0)
addi t0,t0,4 # sepc = ecall+4
csrw sepc, t0
ld x31, 16(a0)
ld x30, 24(a0)
ld x29, 32(a0)
ld x28, 40(a0)
ld x27, 48(a0)
ld x26, 56(a0)
ld x25, 64(a0)
...
ld x3, 240(a0)
ld x2, 248(a0)
ld x1, 256(a0)
ld x10, 184(a0)
sret
```

四、实验结果分析

4.1 page fault

不存在fork的时候，在一个进程刚刚被调度的时候，它的代码段发生缺页异常，无法取指，故scause=12,从第一条指令的地址0处第一次缺页异常。

对代码段建立页表映射后，等到用户栈的访问。

```
sd      ra,24(sp)
```

发生scause=15的访存写入缺页异常，发生后对用户栈建立页表映射。

当进程再次被调度时，由于所有映射均已建立，不会再发生缺页异常，可以正常执行。

```
[S-MODE] Hello RISC-V
SET [PID = 1 PRIORITY = 1 COUNTER = 10]
SET [PID = 2 PRIORITY = 4 COUNTER = 10]
SET [PID = 3 PRIORITY = 10 COUNTER = 5]
SET [PID = 4 PRIORITY = 4 COUNTER = 2]
switch to [PID = 4 PRIORITY = 4 COUNTER = 2]
[S] PAGE_FAULT: scause: 12, sepc: 0x0000000000000000, badaddr: 0x0000000000000000
vm found! badaddr:0x0000000000000000 is of vm:0xffffffe007fab000 with start:0x0000000000000000 and end:0x0000000000000738
into create_mapping... pgtbl=0xffffffe007fa9000 va=0x0000000000000000 pa=0x0000000000000000 sz=0x0000000000000738
third level pa: 000000002008101f
[S] mapped PA: 0x0000000000000000 to VA: 0x0000000000000000 with size: 0x0000000000000738
[S] PAGE_FAULT: scause: 15, sepc: 0x000000000000003c, badaddr: 0x00000003ffffff8
vm found! badaddr:0x00000003ffffff8 is of vm:0xffffffe007fab000 with start:0x00000003ffffff000 and end:0x0000000400000000
into create_mapping... pgtbl=0xffffffe007fa9000 va=0x00000003ffffff000 pa=0x00000000087fb1000 sz=0x0000000000001000
third level pa: 0000000021fec417
[S] mapped PA: 0x00000000087fb1000 to VA: 0x00000003ffffff000 with size: 0x0000000000001000
[PID = 4] is running!
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
switch to [PID = 3 PRIORITY = 10 COUNTER = 5]
[S] PAGE_FAULT: scause: 12, sepc: 0x0000000000000000, badaddr: 0x0000000000000000
vm found! badaddr:0x0000000000000000 is of vm:0xffffffe007fb1000 with start:0x0000000000000000 and end:0x0000000000000738
into create_mapping... pgtbl=0xffffffe007faf000 va=0x0000000000000000 pa=0x00000000080204000 sz=0x0000000000000738
third level pa: 000000002008101f
[S] mapped PA: 0x00000000080204000 to VA: 0x0000000000000000 with size: 0x0000000000000738
[S] PAGE_FAULT: scause: 15, sepc: 0x000000000000003c, badaddr: 0x00000003ffffff8
vm found! badaddr:0x00000003ffffff8 is of vm:0xffffffe007fb1000 with start:0x00000003ffffff000 and end:0x0000000400000000
into create_mapping... pgtbl=0xffffffe007faf000 va=0x00000003ffffff000 pa=0x00000000087fb7000 sz=0x0000000000001000
third level pa: 0000000021fec417
[S] mapped PA: 0x00000000087fb7000 to VA: 0x00000003ffffff000 with size: 0x0000000000001000
[PID = 3] is running!
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
switch to [PID = 1 PRIORITY = 1 COUNTER = 10]
```

```
[S] mapped PA: 0x00000000087fbd000 to VA: 0x00000003ffffff000 with size: 0x0000000000001000
[PID = 2] is running!
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[PID = 2] is running!
[S] Supervisor Mode Timer Interrupt
SET [PID = 1 PRIORITY = 1 COUNTER = 9]
SET [PID = 2 PRIORITY = 4 COUNTER = 4]
SET [PID = 3 PRIORITY = 10 COUNTER = 4]
SET [PID = 4 PRIORITY = 4 COUNTER = 10]
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
switch to [PID = 3 PRIORITY = 10 COUNTER = 4]
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
switch to [PID = 1 PRIORITY = 1 COUNTER = 9]
[PID = 1] is running!
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
```

4.2 fork-main 1

一开始仅有1个进程pid=1，当该进程执行到fork()之前，发生了代码段和用户栈段的缺页异常。

到了fork()后初始化了pid=2的进程，由于counter=0，该进程一开始不会被调度。pid=1的进程运行完所有时间片，对两个进程重新初始化counter进行调度。

调度到子进程，由于返回地址为父进程ecall后的一条指令，会先后发生代码段和用户栈段的缺页异常。需要建立页表映射后，能够继续运行。

0000000000000038 <fork>:

```
38: fe010113      addi    sp,sp,-32
3c: 00813c23      sd      s0,24(sp)
40: 02010413      addi    s0,sp,32
44: fe843783      ld      a5,-24(s0)
48: 0dc00893      li      a7,220
4c: 00000073      ecall
50: 00050793      mv      a5,a0
54: fef43423      sd      a5,-24(s0)
58: fe843783      ld      a5,-24(s0)
5c: 00078513      mv      a0,a5
60: 01813403      ld      s0,24(sp)
64: 02010113      addi    sp,sp,32
68: 00008067      ret
```

反汇编可知，子进程在0x50处发生scause=12的取指缺页异常，在0x54处发生scause=15的用户栈写入缺页异常。

```
[U-PARENT] pid: 1 is running!
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
SET [PID = 1 PRIORITY = 1 COUNTER = 4]
SET [PID = 2 PRIORITY = 10 COUNTER = 10]
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
switch to [PID = 2 PRIORITY = 10 COUNTER = 10]
[S] PAGE_FAULT: scause: 12, sepc: 0x0000000000000050, badaddr: 0x0000000000000050
third level pa: 000000002008141f
[S] mapped PA: 0x0000000000000050 to VA: 0x0000000000000050 with size: 0x000000000000008c
[S] PAGE_FAULT: scause: 15, sepc: 0x0000000000000054, badaddr: 0x00000003ffffffc8
third level pa: 0000000021fec017
[S] mapped PA: 0x0000000000000054 to VA: 0x00000003ffffff00 with size: 0x0000000000000100
[U] pid: 0
[U-CHILD] pid: 2 is running!
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
SET [PID = 1 PRIORITY = 1 COUNTER = 10]
SET [PID = 2 PRIORITY = 10 COUNTER = 5]
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
switch to [PID = 1 PRIORITY = 1 COUNTER = 10]
```

4.3 fork-main 2

一开始仅有1个进程，该进程在0处发生scause=12的取指缺页异常。

```
0000000000000000 <_start>:
    0:    06c0006f                j        6c <main>
```

建立代码段的页表映射后，在0x70处发生用户栈写入的缺页异常，并建立用户栈的页表映射。

```
000000000000006c <main>:
    6c:    fe010113                addi     sp,sp,-32
    70:    00113c23                sd      ra,24(sp)
```

而后调用fork，建立一个子进程。

对于第二次调度，该子进程被分配时间片后参与调度。在0x50和0x54处发生缺页异常并建立页表映射。

```
0000000000000038 <fork>:
    38:    fe010113                addi     sp,sp,-32
    3c:    00813c23                sd      s0,24(sp)
    40:    02010413                addi     s0,sp,32
    44:    fe843783                ld       a5,-24(s0)
    48:    0dc00893                li       a7,220
    4c:    00000073                ecall
    50:    00050793                mv       a5,a0
    54:    fef43423                sd      a5,-24(s0)
    58:    fe843783                ld       a5,-24(s0)
    5c:    00078513                mv       a0,a5
    60:    01813403                ld      s0,24(sp)
    64:    02010113                addi     sp,sp,32
    68:    00008067                ret
```

一共fork了4个进程。


```

[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
switch to [PID = 1 PRIORITY = 1 COUNTER = 9]
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
switch to [PID = 4 PRIORITY = 2 COUNTER = 10]
[S] PAGE_FAULT: scause: 12, sepc: 0x0000000000000050, badaddr: 0x0000000000000050
third level pa: 000000002008141f
[S] mapped PA: 0x0000000000205000 to VA: 0x0000000000000000 with size: 0x0000000000000818
[S] PAGE_FAULT: scause: 15, sepc: 0x0000000000000054, badaddr: 0x0000003fffffff8
third level pa: 0000000021fe6817
[S] mapped PA: 0x0000000087f9a000 to VA: 0x0000003fffffff000 with size: 0x0000000000001000
[PID = 4] is running!
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
SET [PID = 1 PRIORITY = 1 COUNTER = 5]
SET [PID = 2 PRIORITY = 10 COUNTER = 10]
SET [PID = 3 PRIORITY = 4 COUNTER = 4]
SET [PID = 4 PRIORITY = 2 COUNTER = 7]
switch to [PID = 3 PRIORITY = 4 COUNTER = 4]

```

```

[PID = 4] is running!
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[PID = 4] is running!
[S] Supervisor Mode Timer Interrupt
SET [PID = 1 PRIORITY = 1 COUNTER = 5]
SET [PID = 2 PRIORITY = 10 COUNTER = 10]
SET [PID = 3 PRIORITY = 4 COUNTER = 4]
SET [PID = 4 PRIORITY = 2 COUNTER = 7]
switch to [PID = 3 PRIORITY = 4 COUNTER = 4]
[PID = 3] is running!
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
switch to [PID = 1 PRIORITY = 1 COUNTER = 5]
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
switch to [PID = 4 PRIORITY = 2 COUNTER = 7]
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
switch to [PID = 2 PRIORITY = 10 COUNTER = 10]

```

五、实验中遇到的问题及解决方法

- 奇怪的循环输出，create_mapping中死循环

```

switch to [PID = 4 PRIORITY = 4 COUNTER = 2]
[S] PAGE_FAULT: scause: 12, sepc: 0x0000000000000000, badaddr: 0x0000000000000000
vm found! badaddr:0x0000000000000000 is of vm:0xffffffff007fad000 with start:0x0000000000000000 and end:0x0000000000000738
[S] PAGE_FAULT: scause: 15, sepc: 0xffffffff0002000b8, badaddr: 0x00000003fffffffff8
vm found! badaddr:0x00000003fffffffff8 is of vm:0xffffffff007fad000 with start:0x00000003ffffff00 and end:0x0000000400000000
mapping is creating... va: 0x00000003ffffff00 => pa: 0x00000000087faa000
[S] PAGE_FAULT: scause: 15, sepc: 0xffffffff0002000b8, badaddr: 0x00000003fffffffff8
vm found! badaddr:0x00000003fffffffff8 is of vm:0xffffffff007fad000 with start:0x00000003ffffff00 and end:0x0000000400000000
mapping is creating... va: 0x00000003ffffff00 => pa: 0x00000000087fa9000
[S] PAGE_FAULT: scause: 15, sepc: 0xffffffff0002000b8, badaddr: 0x00000003fffffffff8
vm found! badaddr:0x00000003fffffffff8 is of vm:0xffffffff007fad000 with start:0x00000003ffffff00 and end:0x0000000400000000
mapping is creating... va: 0x00000003ffffff00 => pa: 0x00000000087fa8000
[S] PAGE_FAULT: scause: 15, sepc: 0xffffffff0002000b8, badaddr: 0x00000003fffffffff8
vm found! badaddr:0x00000003fffffffff8 is of vm:0xffffffff007fad000 with start:0x00000003ffffff00 and end:0x0000000400000000
mapping is creating... va: 0x00000003ffffff00 => pa: 0x00000000087fa7000

```

各种printf()结合gdb调试，发现是传入create_mapping()进程的根页表地址不对劲，导致在create_mapping()中一致发生缺页异常。

```

void create_mapping(uint64 *pgtbl, uint64 va, uint64 pa, uint64 sz, int perm){
    unsigned long i;
    unsigned long *first,*second,*third,*final;
    if(pgtbl!=swapper_pg_dir)
        printk("into create_mapping... va=0x%lx pa=0x%lx sz=0x%lx \n",va,pa,sz);
    for(i=0;i<sz;i+=PGSIZE){
        //first level
        first=&pgtbl[((va+i)>>30)&0x1FF];
        //if first level page is valid
        if((*first)&0x1)
            second=(unsigned long*)((unsigned long)((*first)>>10)<<12)+PA2VA_OFFSET);
        //if first level page is not valid, allocate one page for it and fill the page table entry
        else{
            second=(uint64*)kalloc();
            memset(second,0,PGSIZE);
            *first=(unsigned long)((*first)&0xffc0000000000000)|((((unsigned long)second-PA2VA_OFFSET)>>12)<<10)|((unsigned
        }
        //if(pgtbl!=swapper_pg_dir)
        //printf("first level finished...\n");
        //second level
        second=&second[((va+i)>>21)&0x1FF];
        //if second level page is valid
        if((*second)&0x1)
            third=(unsigned long*)((unsigned long)((*second)>>10)<<12)+PA2VA_OFFSET);
    }
}

```

```

0xffffffff0002015f8 <create_mapping+220> and     a1,a1,s10
0xffffffff0002015fc <create_mapping+224> slli    a5,a5,0xa
0xffffffff000201600 <create_mapping+228> or      a1,a1,s5
0xffffffff000201604 <create_mapping+232> or      a1,a1,a5
0xffffffff000201608 <create_mapping+236> ori     a1,a1,1
0xffffffff00020160c <create_mapping+240> lui     a5,0x1
0xffffffff000201610 <create_mapping+244> sd      a1,0(a0)
0xffffffff000201614 <create_mapping+248> add     s9,s9,a5
0xffffffff000201618 <create_mapping+252> bgeu    s9,s2,0xffffffff0002016bc <create_mapping+416>
0xffffffff00020161c <create_mapping+256> add     s0,s3,s9
0xffffffff000201620 <create_mapping+260> srli    s7,s0,0x1e
0xffffffff000201624 <create_mapping+264> andi    s7,s7,511
0xffffffff000201628 <create_mapping+268> slli    s7,s7,0x3
0xffffffff00020162c <create_mapping+272> add     s7,s1,s7
>0xffffffff000201630 <create_mapping+276> ld      a5,0(s7)
0xffffffff000201634 <create_mapping+280> srli    s8,a5,0xa
0xffffffff000201638 <create_mapping+284> slli    s8,s8,0xc
0xffffffff00020163c <create_mapping+288> andi    a5,a5,1

```

```

remote Thread 1.1 In: create_mapping
0xffffffff00020161c in create_mapping ()
(gdb) si
0xffffffff000201620 in create_mapping ()
0xffffffff000201624 in create_mapping ()
0xffffffff000201628 in create_mapping ()
0xffffffff00020162c in create_mapping ()
0xffffffff000201630 in create_mapping ()
(gdb) i r s7
s7          0xfffffffffc87fa9000    -1114410151936
(gdb)

```

一方面是, kalloc()出来的根页表虚拟地址和物理地址转换的错误, 另一方面(unsigned int*)没转换成(unsigned int), 导致了奇怪的地址

```
create_mapping((unsigned long)current->pgd+PA2VA_OFFSET,found->vm_start,pa,PGSIZE,(found->vm_flags<<1));
```

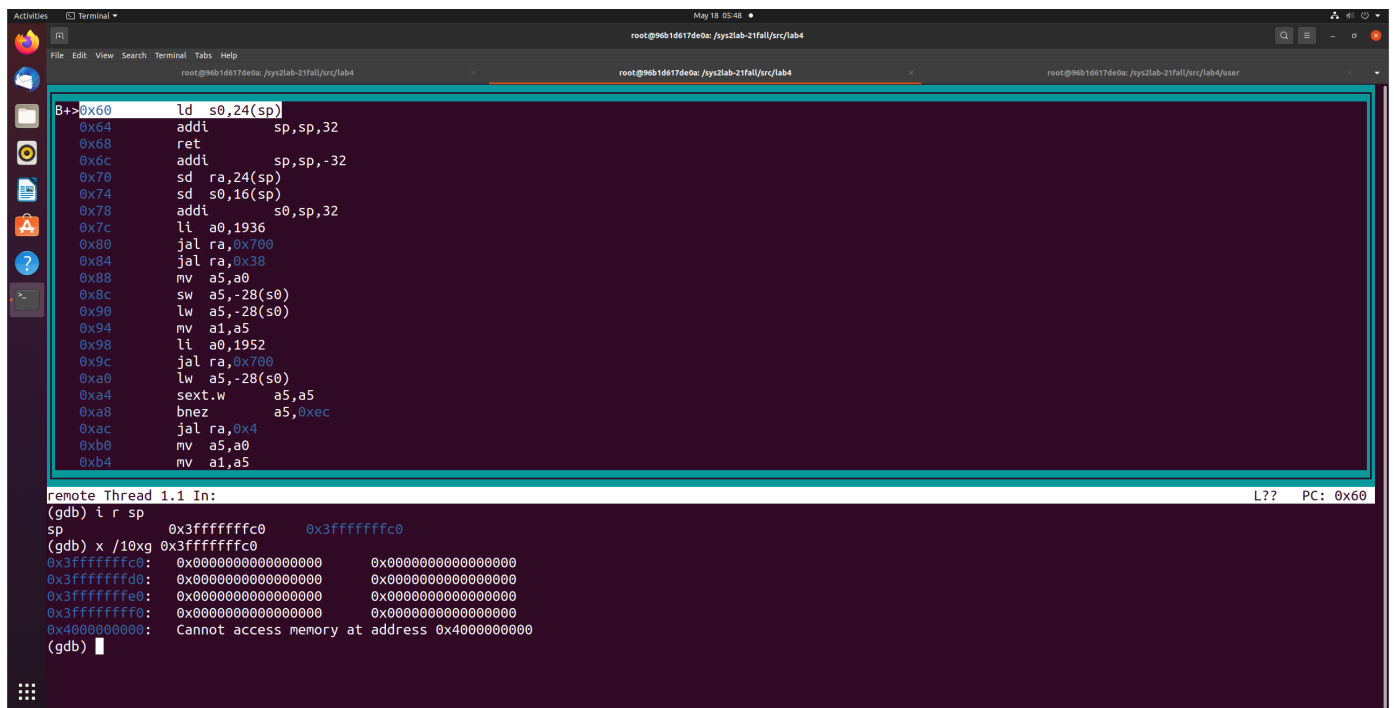
- 奇怪的循环输出, fork后子进程一直循环缺页错误, 捕获bad_address

0xffffffff007fc3018

```
vm->vm_length:0x000000000000084c vm->vm_flags:0x0000000000000007
vm->vm_mm :0xffffffff007fb2000 vm->vm_start:0x00000003ffffff000 vm->vm_end:0x0000004000000000
vm->vm_length:0x0000000000001000 vm->vm_flags:0x0000000000000003
[U] pid: 2
[U-PARENT] pid: 1 is running!
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
SET [PID = 1 PRIORITY = 1 COUNTER = 4]
SET [PID = 2 PRIORITY = 10 COUNTER = 10]
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
switch to [PID = 2 PRIORITY = 10 COUNTER = 10]
[S] PAGE_FAULT: scause: 12, sepc: 0x0000000000000050, badaddr: 0x0000000000000050
third level pa: 000000002008141f
[S] mapped PA: 0x0000000080205000 to VA: 0x0000000000000000 with size: 0x000000000000084c
[S] PAGE_FAULT: scause: 15, sepc: 0x0000000000000054, badaddr: 0x00000003ffffffc8
third level pa: 0000000021fedc17
[S] mapped PA: 0x0000000087fb7000 to VA: 0x00000003ffffff000 with size: 0x0000000000001000
[S] PAGE_FAULT: scause: 13, sepc: 0x0000000000000060, badaddr: 0xffffffff007fc3018
vm is not found with badaddr:0xffffffff007fc3018
[S] PAGE_FAULT: scause: 13, sepc: 0x0000000000000060, badaddr: 0xffffffff007fc3018
vm is not found with badaddr:0xffffffff007fc3018
[S] PAGE_FAULT: scause: 13, sepc: 0x0000000000000060, badaddr: 0xffffffff007fc3018
vm is not found with badaddr:0xffffffff007fc3018
[S] PAGE_FAULT: scause: 13, sepc: 0x0000000000000060, badaddr: 0xffffffff007fc3018
vm is not found with badaddr:0xffffffff007fc3018
[S] PAGE_FAULT: scause: 13, sepc: 0x0000000000000060, badaddr: 0xffffffff007fc3018
vm is not found with badaddr:0xffffffff007fc3018
[S] PAGE_FAULT: scause: 13, sepc: 0x0000000000000060, badaddr: 0xffffffff007fc3018
vm is not found with badaddr:0xffffffff007fc3018
[S] PAGE_FAULT: scause: 13, sepc: 0x0000000000000060, badaddr: 0xffffffff007fc3018
vm is not found with badaddr:0xffffffff007fc3018
```

由于这个地址太奇怪了, debug一波发现是sp的问题。还有一开始是因为复制栈的时候, 循环了PGSIZE次, 但由于以unsigned long为单位复制, 其实只要循环512次。那么估计是子进程sp初值设错了, 一开始设成了user_sp, 这是不对。应该是sscratch一系列艰辛的debug历程, 把sp输出对比

- 还是奇怪的循环输出, sp是对的, 但s0从栈上取出来是0。调试发现, 栈上数据全是0。估计是栈没有copy成功, 最后发现还是分配出来的页表地址进行加减VA2PA_OFFSET的时候(unsigned int*)没转换成(unsigned int)



```
root@9db1d617de0a: /sys2lab-21fall/jrc/lab4
B->0x60 ld s0,24(sp)
0x64 addi sp,sp,32
0x68 ret
0x6c addi sp,sp,-32
0x70 sd ra,24(sp)
0x74 sd s0,16(sp)
0x78 addi s0,sp,32
0x7c li a0,1936
0x80 jal ra,0x700
0x84 jal ra,0x38
0x88 mv a5,a0
0x8c sw a5,-28(s0)
0x90 lw a5,-28(s0)
0x94 mv a1,a5
0x98 li a0,1952
0x9c jal ra,0x700
0xa0 lw a5,-28(s0)
0xa4 sext.w a5,a5
0xa8 bnez a5,0xec
0xac jal ra,0x4
0xb0 mv a5,a0
0xb4 mv a1,a5

remote Thread 1.1 In:
(gdb) t r sp
sp 0x3fffffff0 0x3fffffff0
(gdb) x /10xg 0x3fffffff0
0x3fffffff0: 0x0000000000000000 0x0000000000000000
0x3fffffff0d: 0x0000000000000000 0x0000000000000000
0x3fffffff0e: 0x0000000000000000 0x0000000000000000
0x3fffffff0f: 0x0000000000000000 0x0000000000000000
0x4000000000: Cannot access memory at address 0x4000000000
(gdb)
```

六、思考题与心得体会

fork子进程寄存器状态的变化过程

- sscratch和sp
父进程进入trap, sp变成sscratch中存的内核栈地址, sscratch存了发生异常的用户栈, 进行一波上下文保存, 进入fork后, 子进程的sp初始化成sscratch中的用户栈
- sepc
父进程进入trap,sepc为ecall的地址, 通过regs结构传给子进程。子进程被调用后fork_ret, 要将regs存的sepc+4后赋值给sepc,sret后刚好是ecall的后一条指令。而父进程是通过_traps的sret返回的, sepc在trap_handler后+4返回。
- a0
父进程的regs结构存入a0作为clone的参数传递给处理程序。处理程序将子进程的pid作为返回值返回给trap_handler,trap_handler将其赋值给a0。父进程异常处理完毕后恢复上下文返回。
子进程的a0在do_fork中被手动赋值为0, 通过fork_ret恢复后返回。
- 其它通用寄存器
regs struct父进程传递给子进程赋值。