Lazy

Design

> implement lazy alloction in xv6

What is lazy allocation?

User process uses `sbrk()` to allocate more heap memory for it's space, however, memory allcation is an expensive action when it comes to a relatively large scale like 1GB. So the lazy allocation is that when a user process use `sbrk()` to ask for more free space, we promise it free space without allocating real physical memory for the process.

When this process wants to access to these newly-allocated memory, it will activate the **user trap**,. In **user trap**, os will allocate physical memory for that fault page.

Implementation

1. promise new space in `sbrk()`

   `sbrk()` call `growproc()` in kernel space, `growproc()` will allocate new space for that process. But in lazy situation, we just add the `proc.sz`

```
myproc()->sz+=n;
  if(n<0)
  {
     uvmdealloc(myproc()->pagetable,addr,myproc()->sz);
  }
```

2. handle page fault in `trap.c`

   In riscv arch, whether a fault is a page fault by seeing if r_scause() is 13 or 15 in usertrap().

   In this situation, we allocate a new physical page and mapping it in va

```
 if (r_scause()==13 || r_scause() == 15){
      uint64 va = r_stval();
      if (va >p->sz){
        // va out of range
        printf("usertrap(): va out of memory size\n");
        p->killed = 1;
      }else if (va < PGROUNDDOWN(p->trapframe->sp)){
        // va in read only address (under stack)
        printf("usertrap(): va under guard page\n");
        p->killed = 1;
      }else{
```

```
12          va = PGROUNDDOWN(va);
13          char *mem = kalloc();
14          if(mem == 0){
15            // no enough memory for alloc
16            //printf("usertrap(): no enough memory for new page\n");
17            p->killed = 1;
18            goto end;
19          }
20          memset(mem, 0, PGSIZE);
21          if(mappages(p->pagetable, va, PGSIZE, (uint64)mem,
    PTE_W|PTE_X|PTE_R|PTE_U) != 0){
22            // map err
23            printf("usertrap(): mapping va to pa fail\n");
24            kfree(mem);
25            p->killed = 1;
26          }
27        }
28      }
```

3. make lazy allocation capatible to all tests
    1. As we just add `proc.sz`, there are some null page under sz, so in `uvmunmap` or `uvmcopy`, we need to omit no page panic

```
1   if((pte = walk(pagetable, a, 0)) == 0)
2         continue;
3       if((*pte & PTE_V) == 0){
4         *pte = 0;
5         continue;
6       }
```

    2. As there are some invalid PPE in a pagetable, so we need to modify `walkaddr()` to handle problems

Result

pass all test

```
make[1]: Leaving directory '/home/vielo/code/xv6lab'
== Test running lazytests ==
$ make qemu-gdb
(7.1s)
== Test   lazy: map ==
  lazy: map: OK
== Test   lazy: unmap ==
  lazy: unmap: OK
== Test usertests ==
$ make qemu-gdb
(100.2s)
== Test   usertests: pgbug ==
  usertests: pgbug: OK
== Test   usertests: sbrkbugs ==
  usertests: sbrkbugs: OK
== Test   usertests: argptest ==
  usertests: argptest: OK
== Test   usertests: sbrkmuch ==
  usertests: sbrkmuch: OK
== Test   usertests: sbrkfail ==
  usertests: sbrkfail: OK
== Test   usertests: sbrkarg ==
  usertests: sbrkarg: OK
== Test   usertests: stacktest ==
  usertests: stacktest: OK
== Test   usertests: execout ==
  usertests: execout: OK
== Test   usertests: copyin ==
  usertests: copyin: OK
== Test   usertests: copyout ==
  usertests: copyout: OK
== Test   usertests: copyinstr1 ==
  usertests: copyinstr1: OK
== Test   usertests: copyinstr2 ==
  usertests: copyinstr2: OK
== Test   usertests: copyinstr3 ==
  usertests: copyinstr3: OK
== Test   usertests: rwsbrk ==
  usertests: rwsbrk: OK
```

```
== Test   usertests: kernmem ==
  usertests: kernmem: OK
== Test   usertests: validatetest ==
  usertests: validatetest: OK
== Test   usertests: opentest ==
  usertests: opentest: OK
== Test   usertests: writetest ==
  usertests: writetest: OK
== Test   usertests: writebig ==
  usertests: writebig: OK
== Test   usertests: createtest ==
  usertests: createtest: OK
== Test   usertests: openiput ==
  usertests: openiput: OK
== Test   usertests: exitiput ==
  usertests: exitiput: OK
== Test   usertests: iput ==
  usertests: iput: OK
== Test   usertests: mem ==
  usertests: mem: OK
== Test   usertests: pipe1 ==
  usertests: pipe1: OK
== Test   usertests: preempt ==
  usertests: preempt: OK
== Test   usertests: exitwait ==
  usertests: exitwait: OK
== Test   usertests: rmdot ==
  usertests: rmdot: OK
== Test   usertests: fourteen ==
  usertests: fourteen: OK
== Test   usertests: bigfile ==
  usertests: bigfile: OK
== Test   usertests: dirfile ==
  usertests: dirfile: OK
== Test   usertests: iref ==
  usertests: iref: OK
== Test   usertests: forktest ==
  usertests: forktest: OK
== Test time ==
time: OK
Score: 119/119
(base)
```