```
                    LAB6 HELP
0. ts.s file for kernel: focus on svc_entry,goUmode:

   svc_entry:
       stmfd sp!, {r0-r12, lr}     // save Umode registers in kstack
       save Umode cpsr in PROC.spsr;
                  usp  in PROC.usp
       call svc_hanler to execute a kernel function;
       put return value r into saved R0 in kstack

   goUmode:
       restore spsr from PROC.spsr;
               usp  from PROC.usp;
       ldmfd sp!, {r0-r12, pc}^   // back to Umode with r in R0

1. Each process, except P0, has a 2MB user mode image at
   1st  1MB at   7MB + pid*1MB
   2nd  1MB at  16MB + pid*1MB

SO each process needs TWO user mode pgdir entries [2048] -> 1st 1MB
                                                  [2049] -> 2nd 1MB

These TWO 1MB areas are not contiguous physically but are contiguous as VA.

Refer to the following diagram

    PROC

    pgdir ----> 4MB+pid*16KB
                ====================
                 0 -----> PA 0   MB
                 1 -----> PA 1   MB
                 ....
                257 ----> PA 257 MB
                ====================
   2MB VA
 -----------
 |  1MB    |    2048 ---->1st 1MB of Uimage, e.g. at 8MB
 |  2MB    |    2049 ->
 -----------          |   they are NOT adjacent !!!
                      |
                      |-->2nd 1MB of Uimage, e.g. at 16MB

2. kfork("u1") creates P1 with 2MB Umode image: one at 8MB, another at 16MB
   Load u1 to its first 1MB is good enough since u1 size is only about 5KB.

   HOWEVER, P1's usp must point at the HIGH end of its VA, i.e.
            p->usp = VA(2MB) = 0x80200000;

   You may get a data exception when P1 runs u1 in Umode if its R0 points to a
   invalid VA. Try to set R0 at p->kstack[SSIZE-14]=VA(2MB - 4);


   To test YOUR code used the 2MB VA or not, in u1,
      define a LOCAL varialbe
               int p;
               printf("&p = %x\n", &p);
      it should be 0x801FFFxx
                                          i
3. fork(): Caller Umode image is 2MB (two separate piceces of 1MB PA).
           MUST copy both 1MB areas to child's Umode image.


4. exec(commandLine):
   load file to 1st MB area.
   Make ustack conatin  ------------------------------------------------
                                |commandLine           |
                        --------|-----------------------
                               -128                    2MB VA

   Umode sp:      p->usp = VA(2MB-128)               // ustack pointer VA

   Umode R0:      p->kstack[SSIZE-14] = VA(2MB-128)  // passed to main0(s)

   Umode lr:      p->kstack[SSIZE- 1] = VA(0)        // entry of us.s at VA 0


5. Use the same technique to pass a line "u1 start" to P1 when it starts to
   execute u1 in Umode.
```