**MP2-Preliminary**

**Preliminary**

**Print a Page Table**

Page 0 which is saved at pte 0x0000000021fdb01f, pa 0x0000000087f6c000 is corresponding to “data” and “text” in figure.

Page 1 which is saved at pte 0x0000000021fda40f, pa 0x0000000087f69000 is

corresponding to “guard page” in figure

Page 2 which is saved at pte 0x0000000021fda01f, pa 0x0000000087f68000 is corresponding to “stack” in figure

Page 510 which is saved at pte 0x0000000021fddc07, pa 0x0000000087f77000 is corresponding to “trapframe” in figure

Page 511 which is saved at pte 0x0000000020001c0b, pa 0x0000000080007000 is corresponding to “trampoline” in figure

1. Page 0 contains things like functions, variables and structs.

2. Argument information is saved in page 2. It shows the initial contents as created by exec which containing the command-line arguments, and also an array of pointers to them.

3. No. If Page 1 is going to be read/wrote, that means stack layer overflows. Therefore, the hardware will stop it and generate a page-fault exception because the mapping is not valid.

**Generate a page fault**

The kernel handles physical memory by allocating, zeroing, and mapping. Since lazy-allocation just marks addresses as invalid in the user page table instead of really calling kalloc() to allocate physical memory to use, system can’t find the corresponding memory when any process try to use the content of the file. Therefore, an exception will be generated from user space, and trigger error message in uvmunmap() while cleaning action.