Homework 3

Rayala Harichandan
(2016CS10320)

1) Paging

Given virtual address is 0x80100000.

In this, first 10 bits are used to dereference page directory.

Next 10 bits are used as offset for page directory to get page table.

The last 12 bits are used as offset for page table to get the page.

First 10-bits → 0x200
Next 10-bits → 0x100
Last 12-bits → 0x000

Now, dereference 0x200 and add 0x100 to the obtained value to get PDE to be dereference.
Set its PPN (First 20 bits) to point to any page table.
Set Present bit to 1 and writable to 0.
Now in that page table, take the entry at offset 0x000 and set the first 20 bits same as the physical address. Set flags same as in directory.

So, the first 20 bits in page table are 0x00100.

2) Page table reload

a) kpgdir[0] gives 0 as address 0 belongs to boot sector and boot sector directly matches VA to PA.

b) To translate 0x80107beb to physical address, we will first take first 10 bits and find the page directory at that address.

0x80107beb >> 22 = 0x200

kpgdir[0x200].

Then we will add offset which is equal to next 10 bits, i.e, 0x107 to the page directory entry and dereference it to get page table.

Then we will add the remaining 12 bits offset to the page table entry and get the physical address.

c) 0x114007 is the page directory ~~address~~ entry (PDE) dereferenced at 0x200.

d) The first 20-bits of PDE is the PPN of page table.

∴ PPN = 0x114007 >> 12 = 0x114

e) Last three bits of PDE are flags Present, User and writable. 7 means that all of them are set to 1.

f) 0x107001 is the page table entry (PTE) ~~corresponding to the~~ dereferenced at an offset 0x107 from 0x114000.

g) The 1 in the low bits means page is present.

h) Physical address worked because switchkvm is not called and it is still running in kernel address space.

i) As switchkvm is called, it is user address space now. So, physical address won't work.

5) Context switching.

a) sched() executes on the process stack. It is in process ~~memory~~ address space

b) scheduler() executes on main stack which is in kernel address space.

c) Yes, the swtch() returns. It returns after the next call to the function swtch(). (which is done by scheduler())

d) Yes, swtch() can do less work and still be correct. Size of struct context can be reduced by not saving the caller save registers.

e) badc is the four letter sequence.

f) First when Scheduler() is called, 'a' will be printed and context switch happens to process stack. The process calls sched() when it wants to return the control. sched() prints 'c' first. So, the first characters are ac.

After swtch() call in sched(), the swtch() in scheduler returns and 'b' will be printed and this process goes on.

4) Traps

a) It is possible to have two context structures and one 'trapframe' on a kstack.
When a function is called, one context structure will be saved in kstack. When an interrupt occurs in middle of the function, a trapframe and context structure will be save. So, it is possible.

b) It is <u>not</u> possible to have two trapframes and one context structure in kstack.
When an interrupt occurs, interrupts will be disabled until it is handled. So, when there is one trap frame, interrupts won't occur which means semeond trap frame will not be save.

c) It is not possible to have more than three sets of saved registers.