Assignment6

1.[25 pts] Read Chapter 15 of “Three Easy Pieces” (https://pages.cs.wisc.edu/~remzi/OSTEP/vm-mechanism.pdf) and explain how do the CPU hardware and the operating system cooperate in the procedure of address translation.

A: The main idea of address translation is CPU hardware converts the virtual address to the physical address whenever memory access occurs. One of the efficient ways is base and bound. One register implemented in CPU hardware called base register is used to store the start of the physical address of current process and another register implemented in CPU hardware called bound register is used to limit the physical address of current process. What the OS does is to cooperate with MMU to implement address translation. OS should allocate memory for new processes and reclaim memory from terminated processes. The way to realize it is using a list to take all of the free memory in charge. If the process needs a free block of memory, then OS find approximate one in free-list and allocate it for this process. If the process is terminated, then OS adds the memory blocks which have been allocated for this dead process back into free-list and keeps them waiting to be allocated for other processes. What's more, OS should set base\bounds properly upon context switch. Otherwise it may cause memory conflicts during processes switching because each process has its own special base and bounds. Besides, OS should provide exception handler to handle the exception caused when the process does something wrong on memory like accessing illegal address. What CPU hardware does is just to efficiently map the virtual address used in OS to the physical address by adding the content of base register and virtual address together then comparing it with the content of bound register.

2.[25 pts] Read Chapter 16 ”(https://pages.cs.wisc.edu/~remzi/OSTEP/vm-segmentation.pdf)and chapter 18 (https://pages.cs.wisc.edu/~remzi/OSTEP/vm-paging.pdf) of “Three Easy Pieces” and compare segmentation and paging. Your answer should cover all aspects (e.g.,size of chunks, management of free space, context switch overhead, fragmentation, status bits and protection bits, etc.) and compare them side-by-side.

A: (paging based on single level paging introduced in chapter18)

	size of chunks	management of free space	context switch overhead	fragmentation	status bits and protection bits	Pros	Cons
segmentation	vary from different segments	(1)compact physical memory but expensive (2)maintain a free-list with fit algorithm	save and restore segment registers	cause external fragmentation but no internal fragmentation	(1)one bit referred to the direction this segment grows(2)protection bits with different mode such as only-read, read-exe,and so on,to implement segments sharing.	avoid internal fragmentation,efficient at low cost and provide sharing	cause severe external fragmentaion and can't support fully generalized sparse address space
paging	fixed-sized	maintain a free-list	save and restore page tables	no external fragmentaion but internal fragmentation exists	(1)valid bit to save memory for pages unused in address space.(2)protection bits to implement pages sharing. (3)present bit to indicate whether this page is in physical memory or on disk .(4)dirty bit to indicate whether the page has been modified since it was brought into memory. (5)reference bit to track whether a page has been accessed.(6)etc.	avoid external fragmentation and flexible enough to support sparse address space	may cause many extra memory accesses to access the page table and memory waste with memory filled with page tables instead of useful application data

---

3.[25 pts] Consider a system with the following specifications:

- 46-bit virtual address space
- Page size of 8 KBytes
- Page table entry size of 4 Bytes
- Every page table is required to fit into a single page

How many levels of page tables would be required to map the entire virtual address space? Please document the format of a virtual address under this translation scheme. Briefly explain your rationale.

A: As every page table is required to fit into a single page, so $Size_{page\ table} = Size_{page} = 8\ KB$. Then $Num_{PTE} = Size_{page\ table}/Size_{PTE} = 8KB/4B = 2048$. As $Size_{page} = 8\ KB$,so $Offset$ is 13 bits. As $Num_{PTE} = 2048$, which is 11 bits, so $Num_{level} = (46-13)/11 = 3$. Thus, 3 levels of page tables would be required to map the entire virtual address space

---

4.[25 pts] Consider a system with following specififications: Both virtual address space and physical address are 32bits. Page table entry size of 4Bytes.

(a) Suppose it uses 1-level page table. What is the page size? What is the maximum page table size?

A: As 12 bits offset, so the $Size_{page} = 4KB$. And the maximum $Size_{page\ table} = 2^{20} \times 4B = 4MB$.

(b) Suppose it uses 2-level page table.

- Please write down the 1-st level page number and its offset in decimal(base 10) of virtual address **0xC302C302**(base 16).
- Please write down the 2-nd level page number and its offset in decimal(base 10) of virtual address **0xEC6666AB** (base16).

A: (1) 1-st level page number: 780, offset: 770. (in decimal)

(2) 2-nd level page number: 614, offset: 1707. (in decimal)