Machine Problem 2 Report

資工四 胡材溢 B08208032

5.1

Q1. Explain how pte, pa and va values are obtained in detail. Write down the calculation formula for va.

Ans:

pte: 這裡要print的是page table entry的address。可將pagetable看成一array，並用 &pagetable[i] 來取。

pa: page table entry由0到9 bits的flag，10到53 bits的physical page number以及54 bits之後的reserved bits組成。因此將pte先往右移動10個bits，可以得到physical page number。而page size是4096，故再往左移動12個bits。可直接從PTE2PA這個Macro獲得。

va: 從vm.c的walk() function前面的敘述可知一個64 bits的virtual address分成五個部分。其中30到38是level 2的index；21到29是level 1的index；12到20是level的index。若給定level 0到level 2的index為i0, i1,i2，且PAGESHIFT 為12，則。

Q2. Write down the correspondences from the page table entries printed by mp2\_1 to the memory sections in Figure 1. Explain the rationale of the correspondence. Please take virtual addresses and flags into consideration.

Ans:

entry: 0-0-0是對應到text and data，當我們在mp2\_1.c中建立一個global variable並且把它的位址印出時即可看出他的virtual address在這個範圍內。

entry: 0-0-1是對應到guard page，因為他的PTE\_U bit沒有set。

entry: 0-0-2是stack，可以透過在mp2\_1.c中建立一個local variable並把它的位址印出來可發現他的virtual address在這個範圍內。

entry: 255-511-510是trapframe，因為它的PTE\_X bit沒有set，而且mp2\_1.c沒有sbrk()因此也不會有heap的virtual address，所以可知道是trapframe。

entry: 255-511-511是trampoline，因為他是所有的address最大的。

Q3. Make a comparison between the inverted page table in textbook and multilevel page table in the following aspects:

(a) Memory space usage

(b) Lookup time / efficiency of the implementation.

Ans:

(a) Memory space usage

inverted page table所需的空間比multilevel page table的少。如果是inverted page table，整個system只需要一個page table。若用multilevel page table，則每個process都會產生至少一個page table，因每個level都會需要至少一個page table。

(b) Lookup time / efficiency of the implementation

因為inverted page table需要一個個線性搜尋，因此會花比較久的時間才能找到virtual address對應到的physical address。而multilevel page table可以在constant time查出對應的frame，但它要花的memory access time較多。如果frame數量很多，由上述可知multilevel page table將會較efficiency。

5.2

Q1. In which steps the page table is changed? How are the addresses and flag bits modified in the page table?

Ans:

在第5步Reset Page Table時，page table會被改變。在一開始reference page table entry時，因為page儲存在backing store，所以valid bit為0、swapped bit為1、high bits的部分為block number。在第4步Bring in Missing Page與第5步Reset Page Table後，page會搬回physical memory，所以valid bit為1、swapped bit為0、high bits的部分為physical address。

Q2. Describe the procedure of each step in plain English in Figure 2. Also, include the functions to be called in your implementation if any.

Ans:

Step 1: Check an internal table to see if the reference is valid or invalid.

Step 2: When the reference is invalid, the process will be terminated, and usertrap() will be triggered. After adding the condition r\_scause() == 13 || r\_scause() == 15, handle\_pgfault() will be called.

Step 3: In handle\_pgfault(), if PTE\_S bit is set and the page is found to exist in the backing storage, use read\_page\_from\_disk() to retrieve the required page.

Step 4: In handle\_pgfault(), use kalloc() to obtain a free frame in physical memory, and store the page obtained in Step 3 in that space.

Step 5: In handle\_pgfault(), after the page is stored back into physical memory, modify the internal table and the page table to indicate that the page is now in memory.

Step 6: Restart the instruction.