5.1.1

1. pte可以透過pagetable的每個entry的address取得，其中一個page的size為4096 Byte，而一個entry的size為8 Byte，因此一個page共有512個entry，而pa可以透過pte作為macro PTE2PA()的參數取得。由於pagetable的每個entry皆會指向一個4KB大的table，並且最後一層table的entry所指到的frame size和page size相同為4096 Byte，因此在第一層的page table每走一個entry，virtual address會增加(512 \* 512 \* 4096)，在第二層的page table每走一個entry，virtual address會增加(512 \* 4096)，在第三層的page table每走一個entry，virtual address會增加4096，且virtual address從0開始計算，因此va可以透過該page table第一個entry的virtual address，以及每走一個entry要增加多少virtual address計算。

2. entry 0-0-0：由於在mp2\_1.c中新增一個global variable，該global variable的virtual address位於這個範圍中，因此entry 0-0-0對應到text and data。

entry 0-0-1：由於PTE\_U bit沒有set，因此entry 0-0-1對應到guard page。

entry 0-0-2：由於在mp2\_1.c中新增一個local variable，該local variable的virtual address位於這個範圍中，因此entry 0-0-2對應到stack。

entry 255-511-510：由於PTE\_X bit沒有set，且mp2\_1.c中沒有呼叫sbrk()，因此不會產生heap，因此entry 255-511-510對應到trapframe。

entry 255-511-511：由於該entry為最高的address，因此entry 255-511-511對應到trampoline。

3.

(a) 若一個process在physical memory有n個frame，則inverted page table只需要n個entry分別對應到每個frame，而multilevel page table最後一層的page table也是n個entry指到每一個frame，但除此之外還需要前幾個level的page table，因此inverted page table的memory space usage較小。

(b) inverted page table需要對page table做linear search，而multilevel page table則需要一層一層access每個page table的entry，若是frame的數量很多，則multilevel page table的lookup time會較小。

5.2

1. page table會在第5步更改。在更改以前，page table中的entry存的是secondary storage的block number，且PTE\_S bit有set而PTE\_V bit沒有set。在更改以後，page table中的entry存的是physical address，且PTE\_V bit有set而PTE\_S bit沒有set。

2.

Step 1: 檢查page table的entry的PTE\_V bit是否有set。

Step 2: 如過PTE\_V bit沒有set，便會產生page fault，在usertrap()中會透過r\_scause()查看錯誤的原因，若是因為讀寫不存在的page，便會呼叫handle\_pgfault()。

Step 3: 在handle\_pgfault()會透過r\_stval()取得發生錯誤的virtual address，透過walk()取得該virtual address對應的pte，並查看該pte的PTE\_S bit是否有set。

Step 4: 如果PTE\_S bit有set，便透過read\_page\_from\_disk()到secondary storage讀取該page。

Step 5: 將page table entry中的block number改為physical address，將PTE\_V bit set並且將PTE\_S bit unset。

Step 6: 重新執行產生page fault的instruction。