CS2106 Tutorial 9

AY 25/26 Sem 1 — github/omgeta

Q1. (a.) TLB Miss:

```
use PTE for P#
if (valid == 0) segfault
replace TLB entry
return
```

Page Fault for P#:

```
choose victim page
writeback victim page into disk if dirty
locate P# in HDD
load P# into RAM
update P# PTE
update TLB with PTE
```

(b.) Page 3: TLB Hit $\rightarrow$ Memory access
Page 1: TLB Miss $\rightarrow$ PTE Access (Resident) $\rightarrow$ Update TLB (Evict page 0) $\rightarrow$ Memory access
Page 5: TLB Miss $\rightarrow$ PTE Access (Non-resident) $\rightarrow$ Page Fault (Evict page 0 in frame 4, replace with swap page 15) $\rightarrow$ Update PTE $\rightarrow$ Update TLB (Evict page 0) $\rightarrow$ Memory access

(c.) Best (Memory resident in TLB): 1ns (TLB) + 30ns (Memory) = 31ns
Worst (Page fault): 1ns (TLB) + 30ns (PTE) + 5ns (Locate page) + 5ns (Write-back victim page) + 30ns (Update PTE) + 30ns (Memory) = 101ns

(d.) Yes, as on page fault, two separate page table accesses are required.

Q2. (a.) Pages = $2^{48}/2^{12} = 2^{36}$, so bytes needed = $2^{36} \times 8 = 2^{39}$ bytes (512GiB)

(b.) # of PTE per page (store each page table in a page) = $2^{12}/8 = 2^9 = 512$
So we use 4-Level page table, each of 2^9 PTE so represent the 2^{36} pagess

(c.) 1 entry (covering the array)

(d.) 1 frame (for the page directory)

(e.) 1 frame (for the final page directory)

(f.) 513 frames (to deal with the misalignment)

Q3. (a.) (i) Frame 2

(ii) Set A04 PTE to Non-resident, write back if dirty and remove any TLB entry. Update A08 PTE to resident and set frame 2.

(b.) (i.) Frame 0

(ii.)

Frame	Page	Ref
0	B13	1
1	A31	0
2	A8	1
3	A17	0

(iii.) On page fault, non-resident is not in the inverted page table and we replace the victim page with the non-resident information.