CS2106 Tutorial 8

AY 25/26 Sem 1 — github/omgeta

Q1. (a.)

Page#	Frame#	Valid
0	5	1
1	2	1
2	10	1
3	-	0

Processor Action	Logical Addr.	Physical Addr.
Fetch 1st instruction	0	$5 \times 4 + 0 = 20$
Load the 2nd Data word	7	$2 \times 4 + 3 = 11$
Load the 3rd Data word	8	$10 \times 4 + 0 = 40$
Load the 6th Data word	11	$10 \times 4 + 3 = 43$(in valid page)

(b.)

Segment#	Base#	Limit
0	50	6
1	23	5

Processor Action	Logical Addr.	Physical Addr.
Fetch 1st instruction	$< 0, 0 >$	$5 \times 50 + 0 = 50$
Load the 2nd Data word	$< 1, 1 >$	$2 \times 24 + 1 = 24$
Load the 3rd Data word	$< 1, 2 >$	$10 \times 23 + 2 = 25$
Load the 6th Data word	$< 1, 5 >$	segfault

(c.)

Segment#	Page Table Addr.#	Page Limit
0	Seg 0 PT Addr	2
1	Seg 1 PT Addr	2

Segment 0 (Text) Page Table:

Page#	Frame#	Valid
0	7	1
1	4	1
2	-	0
3	-	0

Segment 1 (Data) Page Table:

Page#	Frame#	Valid
0	9	1
1	3	1
2	-	0
3	-	0

Processor Action	Logical Addr.	Physical Addr.
Fetch 1st instruction	$< 0, 0, 0 >$	$7 \times 4 + 0 = 28$
Load the 2nd Data word	$< 1, 0, 1 >$	$9 \times 4 + 1 = 37$
Load the 3rd Data word	$< 1, 0, 2 >$	$9 \times 4 + 2 = 38$
Load the 6th Data word	$< 1, 1, 1 >$	$3 \times 4 + 1 = 13$

Q2. (a.) Keep heap at the end of the memory space, then heap space can be increased by increasing used parititon space. In fixed partitioning, heap just needs to use remaining unused space (internal fragmentation) in the partition. In dynamic partioning, expansion to free adjacent partition can just take over, while if it is occupied then rellocation is necessary.

(b.) In pure paging, take up the internal fragmentation space used within a page. If we exceed the page boundary, OS must look for a new physical frame and update the page table.

(c.) In pure segmentation, the OS updates the limit of the segment and reduce the size of the free partition. If there is no free memory, relocation is required and updating of the segmentation table.

Q3. (a.) 50ns (access page table) + 50ns (access item) = 100ns

(b.) 0.75 * 50ns + 0.25 * 100ns = 62.5ns

(c.) Given page size of $2^12 = 4MB$, there are $2^{32-12} = 2^20$ page table entries. Under uniform distribution of memory accesses, we would need $0.75 \times 2^20 = 786432$ entries in the TLB which is unreasonably high due to cost. However, it is still possible to achieve high hit rate thanks to locality of memory accesses.