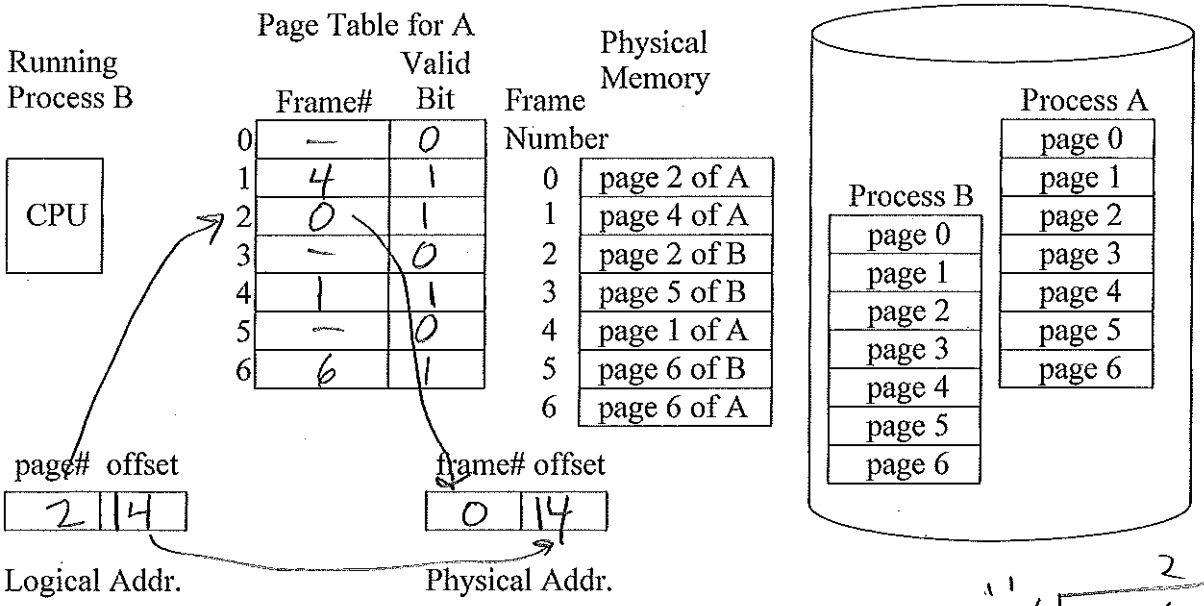
Comp. Arch. Homework #2 Due: Friday (9/19) 4 PM

1. Consider the demand paging system with **4096-byte pages**.



4096 | 8206 = 2 R14
8192
14

.25 a) Complete the above page table for **Process A**.

0.25/0.5 b) If process A is currently running and the CPU generates a logical/virtual address of 8206_{10}, then what would be the corresponding physical address? addr. 14_{10}

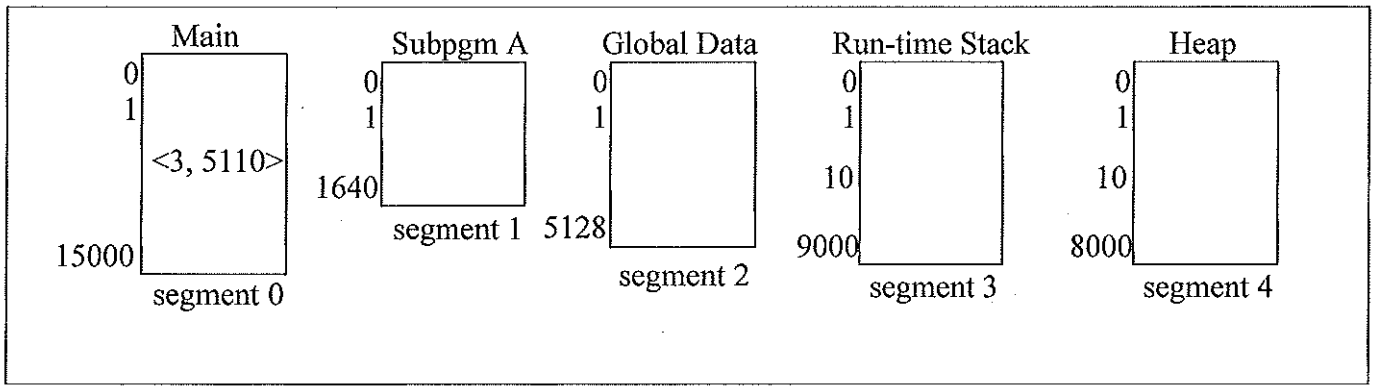
.25 2. Explain how a TLB (translation-lookaside buffer) speeds the process of address translation? The TLB is a cache of page table entries (PTE's). When CPU generates a logical addr, the TLB might be able to supply the frame # to build the physical addr. without going to the slower Page table.

3. What advantages does combining paging and segmentation (i.e., paging of each segment) have over:

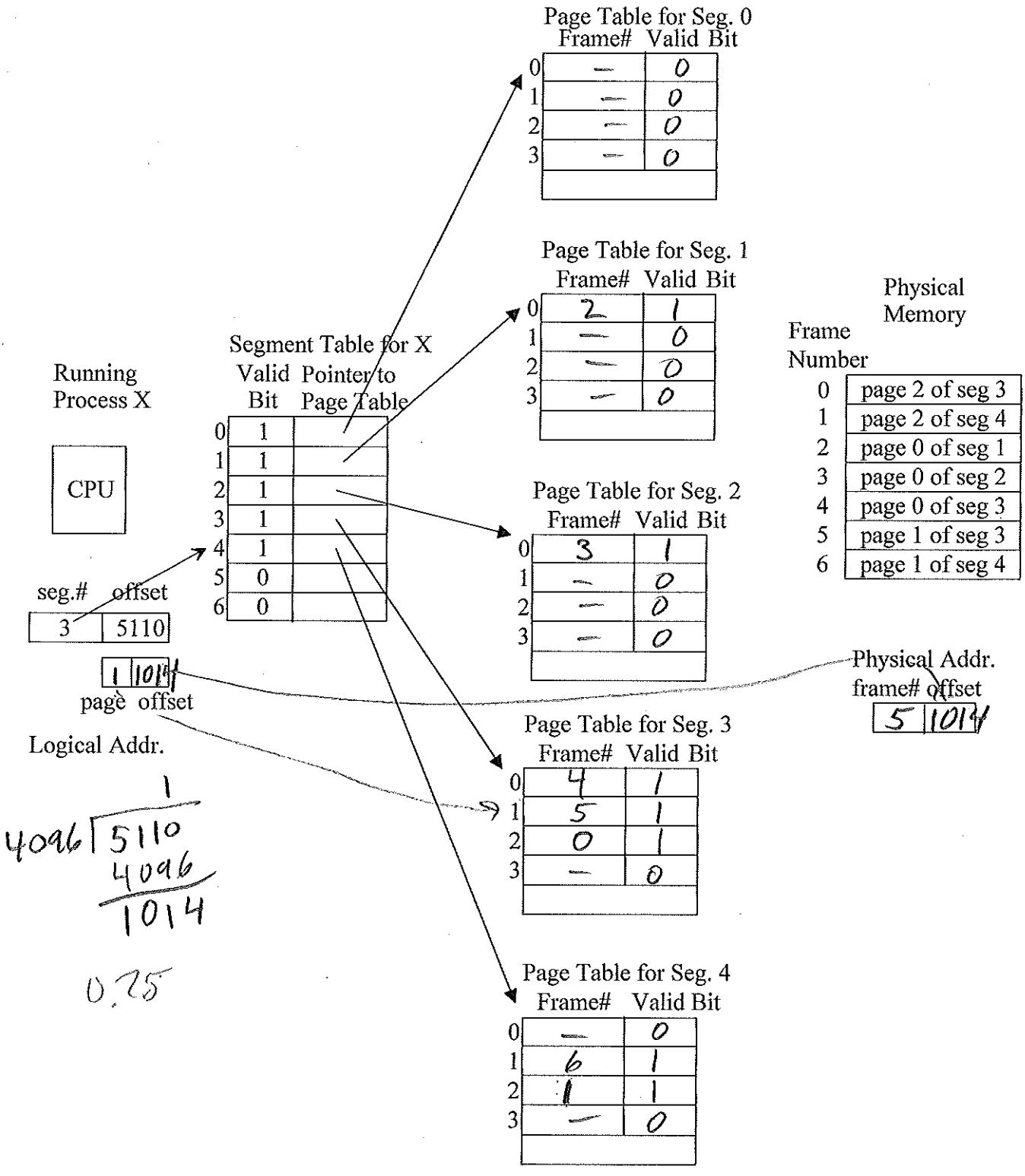
.25 a) only paging – protection of logical units of program.

0.25/0.5 b) only segmentation – don't need to load whole segments only part(s) needed.

4. Assuming a page size of 4096 bytes, complete the Page Tables for the pages in memory, and determine the physical address for the logical address <3, 5110>.



1.25


Page Table for Seg. 0
Frame# Valid Bit
0 — 0
1 — 0
2 — 0
3 — 0
Page Table for Seg. 1
Frame# Valid Bit
0 2 1
1 — 0
2 — 0
3 — 0
Page Table for Seg. 2
Frame# Valid Bit
0 3 1
1 — 0
2 — 0
3 — 0
Page Table for Seg. 3
Frame# Valid Bit
0 4 1
1 5 1
2 0 1
3 — 0
Page Table for Seg. 4
Frame# Valid Bit
0 — 0
1 6 1
2 1 1
3 — 0
Physical Memory
Frame Number
0 page 2 of seg 3
1 page 2 of seg 4
2 page 0 of seg 1
3 page 0 of seg 2
4 page 0 of seg 3
5 page 1 of seg 3
6 page 1 of seg 4
Running Process X
CPU
Segment Table for X
Valid Bit Pointer to Page Table
0 1
1 1
2 1
3 1
4 1
5 0
6 0
seg.# offset
3 5110
1 1014
page offset
Logical Addr.
Physical Addr.
frame# offset
5 1014
1
4096 | 5110
4096
1014
0.25
.25
.5

5. (This question deals with the following toy virtual memory system on the next page which is tiny...)
You have a byte-addressable memory with 8 bytes per memory block. The memory management unit has a two-entry TLB (fully-associate cache with a Page # as the tag) and a slower (vague I know) page-table for a process P. The cache is 2-way set-associative and has a total of 4 cache lines (tag bits shown in binary). Assume page size of 16 bytes, so two memory blocks per frame. In the diagram, memory is divided into blocks, where each block's content is represented abstractly by a letter.

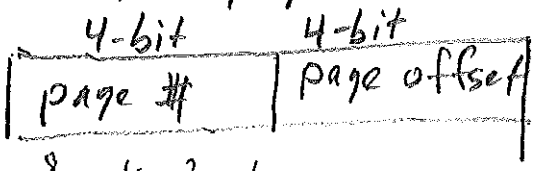
Given the system state as depicted above, answer the following questions:

a) How many bits are in a virtual address for process P?
b) How many bits are in a physical address?
c) Show the address format for a logical/virtual address including field names and number of bits.
d) Using your format in part (c), convert the virtual address 50_{10} to binary and put it in the appropriate fields. Now, explain how these fields are used to translate to the corresponding physical address.
e) Show the address format for a physical address including field names and number of bits that are used to check the cache. ~~Demonstrate using your answer to part (d)~~
f) Given that virtual address 12_{10} translates to physical address 60_{10}. Using your format in part (e), convert the physical address 60_{10} to binary and put it in the appropriate fields. Now, explain how these fields are used to locate physical address 54 in the cache.
g) Given that virtual address 100_{10} is located on virtual page 6, offset 4. Indicate exactly how this address would be translated to its corresponding physical address and how the data would be accessed. Include in your explanation how the TLB, the page table, cache, and memory are used.

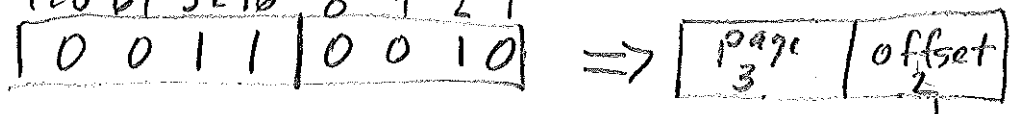
(a) .25 16 pages × 16 bytes/page = $2^4 \times 2^4 = 2^8$ bytes
8-bit virtual addr.

(b) .25 8 frames × 16 bytes = $2^3 \times 2^4 = 2^7$ bytes
7-bit physical addr.

(c) .25



(d) .25

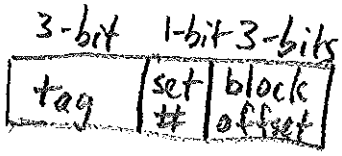


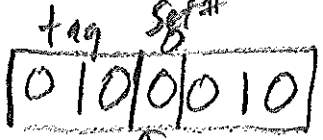
Use page #3 and check TLB to find frame #2

Build physical addr.

32 16 8 4 2 1
$010\ 0010_2$
$= 34_{10}$

(e) .25





check set 0 for matching tag of 010 ⇒ hit, cache can supply G

TLB

Page	Frame
3	2
6	5

Cache

	Tag	Data	Tag	Data
Set 0	010_2	G	101_2	M
Set 1	100_2	d	011_2	B

Page Table for P

	Frame	Valid Bit
0	3	1
1	0	1
2	-	0
3	2	1
4	1	1
5	-	0
6	5	1
7	-	0
8	7	1
9	-	0
10	6	1
11	-	0
12	-	0
13	-	0
14	4	1
15	-	0

Main Memory

Frame	Block	
0	C	0
	D	1
1	I	2
	J	3
2	G	4
	H	5
3	A	6
	B	7
4	c	8
	d	9
5	M	10
	N	11
6	U	12
	V	13
7	Q	14
	R	15

Virtual Memory for Process P

Page	Block	
0	A	0
	B	1
1	C	2
	D	3
2	E	4
	F	5
3	G	6
	H	7
4	I	8
	J	9
5	K	10
	L	11
6	M	12
	N	13
7	O	14
	P	15
8	Q	16
	R	17
9	S	18
	T	19
10	U	20
	V	21
11	W	22
	X	23
12	Y	24
	Z	25
13	a	26
	b	27
14	c	28
	d	29
15	e	30
	f	31

(f) physical addr. 60_{10} = 64 32 16 8 4 2 1 → 0 1 1 1 1 0 0

tag = 011_2, set # 1

Check set 1 for matching tag 011_2 which is a hit, so cache contains desired block "B".

(g) Virtual addr. 100_{10} = page# 6 | page offset 4 → 128 64 32 16 8 4 2 1 → 0 1 1 0 | 0 1 0 0

TLB checked for Page 6 PTE, which is a hit, so frame 5 supplied

frame 5 | offset 4 → $101\,0100_2$

tag 101 | set 0 | 100

Cache set 0 checked for tag 101

1.75

3.5