Bài Tập Bộ Nhớ Ảo

1. Consider the following paging memory system: There are 4 page table entries (with values of 0xC, 0x2, 0x8, 0x5 for entries 0…3, respectively). The physical memory is 128 bytes, with frames of 8 bytes each.
   a. How large (the number of bits) is the physical address?
   b. How large is the virtual address?
   c. What is the physical address (in hex) that corresponds to virtual address 0x1D
   d. What is the physical address (in hex) that corresponds to virtual address 0x03?
2. Consider a single-level paging system with 12-bit virtual addresses, 24-bit physical addresses, and a 256 (2^8) byte page size.
   a. What is the maximum number of entries in a page table in this system?
   b. A process P1 has the following page table. Frame numbers are given in hexadecimal notation (recall that each hexadecimal digit represents 4 bits)

Page Number	Frame Number
0	0x1010
1	0x2034
2	0x43AC
3	0x1100
4	0xAC11
5	0x8000

   For each of the following physical addresses, indicate the virtual address to which it maps. If the physical address is not part of the physical memory assigned to P1, write NO TRANSLATION instead. Use hexadecimal notation for the virtual addresses.
   0x1100A0
   0xAC1100
   0xBA3424
   0x43ACA0
   c. Due to a bug in the OS161 as copy function, the following is the page table of P1's child process immediately after it returns from fork. Mark the entries in the page table that you are certain to be incorrect.

Page Number	Frame Number
0	0x2453
1	0x1010
2	0xEA35
3	0x3100
4	0x2034
5	0x9012

   d. Name one advantage and one disadvantage of having a virtual address space that is smaller than the physical address space
3. Consider a virtual memory system that uses paging. Virtual and physical addresses are both 32 bits long, and the page size is 4KB = 2^{12} bytes. A process P1 has the following page table. Frame numbers are given in hexadecimal notation (recall that each hexadecimal digit represents 4 bits)

	Frame Number
0	0x0014e
1	0x03b65
2	0x00351
3	0x00875
4	0x06a3f

a. For each of the following virtual addresses, indicate the physical address to which it maps. If the virtual address is not part of the address space of P1, write NO TRANSLATION instead. Use hexadecimal notation for the physical addresses.

0x00003b65

0x00006a3f
0x00000fe6

b. For each of the following physical addresses, indicate the virtual address that maps to it. If the physical address is not part of the physical memory assigned to P1, write NO TRANSLATION instead. Use hexadecimal notation for the virtual addresses
0x00351fff
0x03b65000
0x000e3000

4. Consider a machine with 48-bit virtual addresses and a page size of 16 MB. During a program execution the TLB contains the following valid entries (in hexidecimal).

Virtual Page Num	Physical Frame Num
0x 0	0x 1
0x BB	0x 2
0x BB9	0x 3
0x BB97	0x 4
0x BB972	0x 5
0x BB9720	0x 6

Translate the following virtual address into a 52-bit physical address (in hexidecimal). Show and explain how you derived your answer. Express the final physical address using all 52-bits. Virtual address = 0x 00BB 9720 AF05.

5. Consider a machine with 24-bit virtual addresses and a page size of 512 bytes. During a program execution the TLB contains the following valid entries (all in octal)

Virtual Page Num	Physical Frame Num
0	10
7	20
73	30
731	40
7310	50

Translate the following virtual address (in octal) into a 24-bit physical address (in octal). Show and explain how you derived your answer. Express the final physical address using all 24-bits. Virtual address = 07310525

6. Consider a machine with 27- bit virtual addresses and a page size of 4096 bytes. During a program execution the TLB contains the following entries (all in octal)

Virtual Page Num	Physical Frame Num	Valid	Dirty
6	20	1	1
6125	50	1	1
0	10	0	0
612	40	0	1
61	30	1	0

If possible, explain how the MMU will translate the following virtual address (in octal) into a 33-bit physical address (in octal). If it is not possible, explain what will happen and why. Show and explain how you derived your answer. Express the final physical address using all 33-bits. Load from virtual address = 612521276.

7. Consider a virtual memory system that uses segmentation combined with paging. Virtual addresses in this case are of the form (seg #, page #, offset). In this system, virtual and physical addresses are both 32 bits long, a process may have 16 segments, and the page size is 4KB (2^{12} bytes). A process P has three segments, and the following page tables are associated with its 3 segments. Frame numbers are given in hexadecimal:

Segment 0	
Page #	Frame #
0	0x00078
1	0x00024
2	0x00023

Segment 1	
Page #	Frame #
0	0x00088
1	0x00049
2	0x0003f
3	0x000ce
4	0x000cd

Segment 2	
Page #	Frame #
0	0x00079
1	0x00029
2	0x0002f
3	0x000ae

a. For each of the following virtual addresses (given in hexadecimal), indicate the physical address to which it maps. If the virtual address is not part of the address space of P, write NO TRANSLATION instead. Use hexadecimal notation for the physical addresses.
0x20001a60

0x000052ef

0x10004ab3
0xa00003c9

b. Explain how the virtual memory system would enable another process Q to share Segment 2 with process P.

c. Give one reason that would make it useful to share segments between processes as in part (b) of this question

8. Using the page references string shown below, fill in the frames and missing information to show how the OPT (optimal) page replacement algorithm would operate. Use a dash "_" to fill in blank locations. Note that when there is more than one page that is a possible victim, always choose the one with the lowest frame number.

Num	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Refs	A	B	C	D	B	E	C	G	D	A	G	D	B	E	C
Frame 1	A														
Frame 2	–														
Frame 3	–														
Frame 4	–														
Fault ?	X														

9. Fill in the frames and missing information below to show how the LRU (least recently used) page replacement algorithm would operate. Use a dash "_" to fill in blank locations. Note that when there is more than one page that is a possible victim, always choose the one with the lowest frame number.

Num	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Refs	A	B	C	D	B	E	C	G	D	A	G	D	B	E	C
Frame 1	A														
Frame 2	–														
Frame 3	–														
Frame 4	–														
Fault ?	X														

10. Assume that there are 5 pages, A, B, C, D, and E. Fill in the page reference string and complete the rest of the information in the table below so that LRU is the worst page replacement algorithm (i.e., it results in the maximum number of page faults). Use a dash "_" to fill in blank locations. Note that when there is more than one page that is a possible victim, always choose the one with the lowest frame number.

Num	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Refs	A	B	C	D	E										
Frame 1	A														
Frame 2	–														
Frame 3	–														
Frame 4	–														
Fault ?	X														

11. Consider a virtual memory system that uses 2-level paging. The page size in this system is 256 (2^8) bytes. Each individual page table fits exactly into one memory frame, and the size of each page table entry (PTE) is 8 bytes.

a. What is the maximum size (in bytes) of a virtual address space in this system?

b. Suppose that there is a process with a virtual address space of the maximum size. How many bytes of memory are occupied by the page tables for this process?

c. Suppose that there is a process with a virtual address space of size 10240 ($10x2^{10}$) bytes. Also suppose that the entire address space is in memory. How many valid PTEs will exist in the page tables for this process?

12. Consider the contents of the following TLB and Page Table (snippet):

TLB

Page Number	Frame Number	Valid Bit
0x4019d	0x21f	1
0x4212f	0x026	0
0x0811d	0x11d	0
0x40015	0xf3e	1
0x0842f	0x20e	0
0x4002a	0x72d	0
0x40016	0x154	0
0x404ec	0x84b	1

Page Table (Snippet)

Page Number	Frame Number	Valid Bit
...	...	...
0x40013	0x542	1
0x40014	0xb6c	1
0x40015	0xf3e	1
0x40016	0x532	1
0x40017	0x68b	0
0x40018	0xf58	1
0x40019	0xc14	1
0x40020	0x49c	0
...	...	...

Given a page size of 4K (2^{12}), translate each of the following virtual addresses into physical addresses. For each, report whether it is a TLB hit or miss, a page table hit or miss and the corresponding physical address or identify if the translation results in a page fault.

a. 0x40015b0c
b. 0x40016c70
c. 0x4001775a
d. 0x40019b40

13. Consider a small computer system with 16-bit virtual addresses and 16-bit physical addresses. The page size in this system is 256 (2^8) bytes. Suppose that there are two processes, P1 and P2, running in this system, with page tables as shown below. For the purposes of this question, assume that all of the page table entries shown in the tables are valid.

P_1's page table

Page Number	Frame Number
0	0x10
1	0x14
2	0x4a
3	0x05
4	0x22
5	0x21
6	0x1a
7	0x58

P_2's page table

Page Number	Frame Number
0	0x3b
1	0x01
2	0x0b
3	0x38
4	0x30
5	0x2c

a. For each of the following virtual addresses from P1's virtual address space, indicate the physical address to which it corresponds. Give your answers in hexadecimal. If the specified virtual address is not part of the virtual address space of P1, write "NO TRANSLATION" instead.
   0x034a
   0x1004
   0x0022
b. For each of the following physical addresses, indicate which process's virtual address space maps to that physical address, and indicate which specific virtual address maps

there. If neither process has a virtual address space that maps to the given physical address, write “NO MAPPING" instead
0x2222
0x38ff
0x0123

c. What is the size (in bytes) of P1’s virtual address space?

14. Consider a virtual memory system that uses paging. Virtual and physical addresses are both 32 bits long, and the page size is 4KB = 2^{12} bytes. We have a TLB that can hold eight entries and during execution of a process P1, it has the following entries.

Virtual Page#	Physical Frame#	Valid	Dirty
AC	2CD	1	1
DA	5DF	1	1
0	1F	1	0
29	6C1	0	1
21	32	0	0

Notes: All numbers in this question are in hexadecimal. Also consider the Dirty bit has the same meaning as in an OS/161 TLB entry.

a. If possible, explain how the MMU will translate the following virtual addresses into physical address. If it is not possible, explain what will happen and why.
Load from virtual address = 0x000A CF91
Store at virtual address = 0x0000 03F8

b. Now consider we have the same entries in the TLB and the following is part of the page table for process P1.

	Physical Frame#	Read/Write
...	...	...
1	3D2	1
...	...	...
13	1F6	1
...	...	...
132	259	1
...	...	...
1328	981	1
...	...	...
13289	FFC	1
...	...	...

Note: Consider the Read/Write flag in page table entry has the same meaning as the Dirty flag in the TLB entry. If possible, explain how the MMU will translate the following virtual addresses into physical address. If it is not possible, explain what will happen and why. Show and explain how you derived your answer. Load from virtual address = 0x0001 3289.