Kernel + Memory Management

Recap

Mads Presfeldt

Agenda

- Kernel
  - What is a kernel?
  - Process Management
  - System Calls
- Memory Management
  - Virtual Memory
  - Key concepts
  - How Virtual Memory works
  - Page Replacement Algorithms
  - Allocation Strategies
  - Challenges and Considerations
  - TLB/Page table Old Exam Question Example

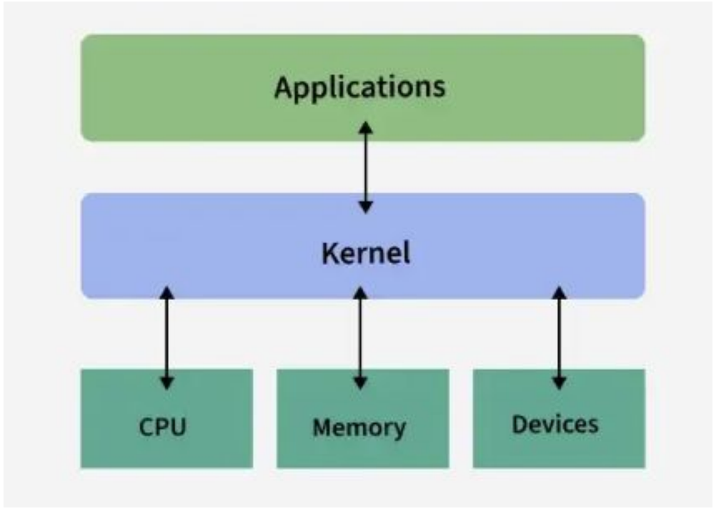
What is a Kernel?

A **Kernel** is the **core part of an operating system** and **is always resident in memory**.

Acts as a **bridge between software** applications and the **hardware** of a computer.

Key responsibilities:

- **Process Management**: creating, scheduling and terminating processes
- **Memory Management**: allocating RAM, handling virtual memory
- **Device Management**: controlling I/O devices through drivers
- **System Calls**: providing a safe interface for application to request OS services



Process Management

A Program: a file containing code. Stateless and dead.

A Process: a running instance of a program. The same program can be running in multiple instances. Stateful and alive

- **Switching** between multiple concurrent processes
- **Handling** process termination
- **Starting** new processes

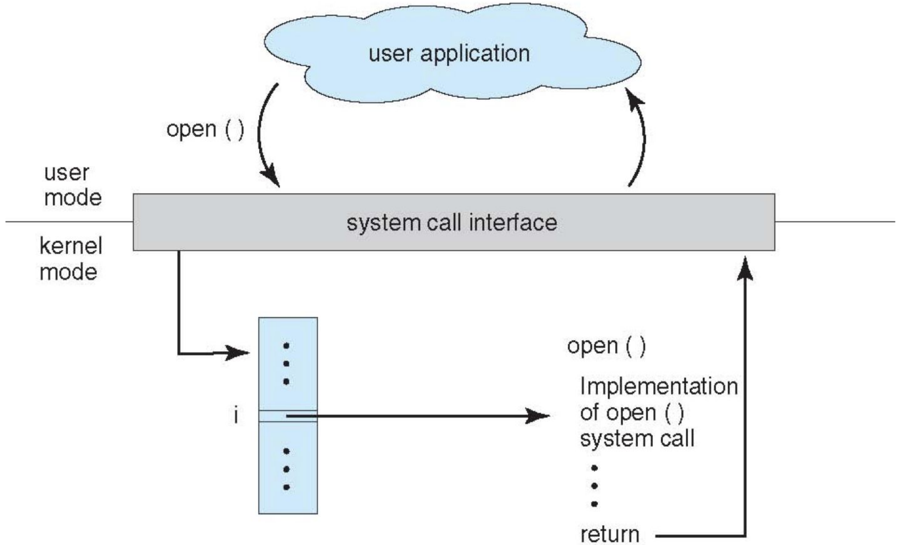
System Calls

System calls: a request by a process, that the kernel carries out some operation on its behalf.

Why system calls?: processes does not have direct access to hardware and system memory. To do IO we must thus use a system call.

System calls are slow as CPU switches to kernel space, kernel executes the request, then return to user space. **But safe as only the kernel has access to hardware**

C System Calls examples: open(), read(), write()



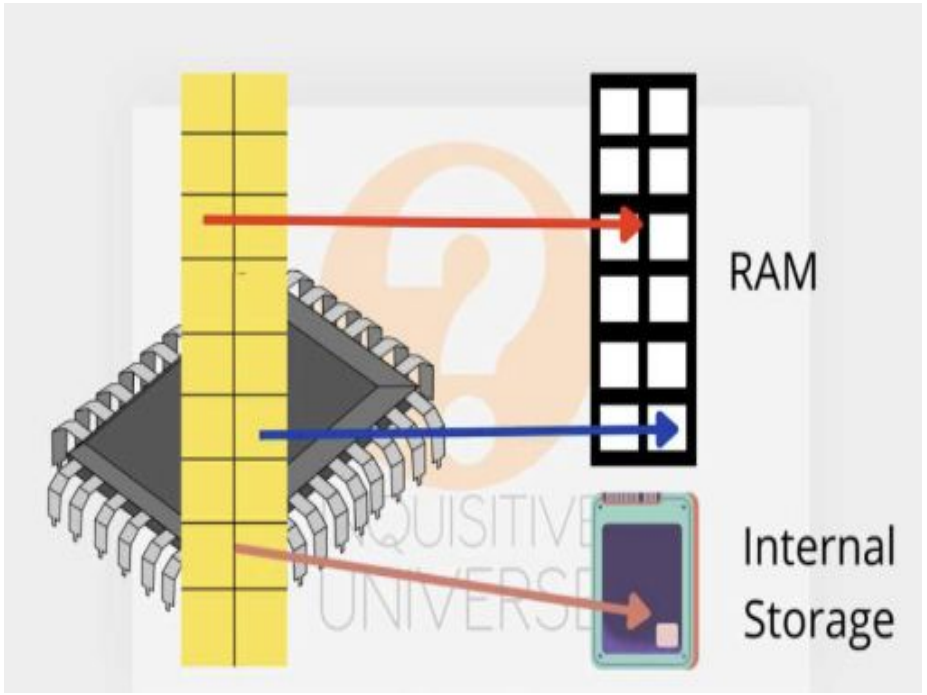
Virtual Memory

- **Definition**

Memory management technique that creates an **“idealized abstraction of the storage resources”** available on a large, contiguous memory space regardless of the underlying physical memory

- **Purpose**

It enables systems to run larger applications or multiple programs simultaneously by utilizing both physical memory (RAM) and secondary storage (disk).

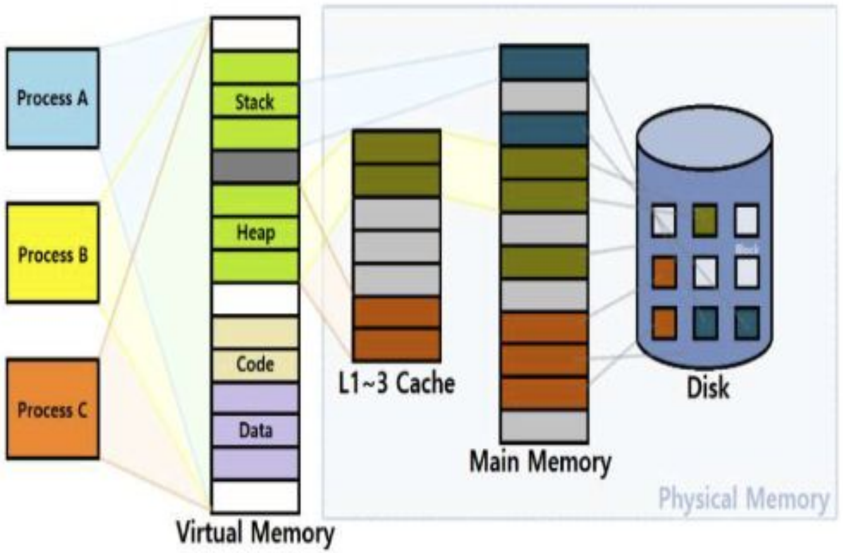


Key Concepts

- **Virtual Address space:** The range of addresses that an application can use, which the operating system maps to physical.

- **Paging:** Divides virtual memory into fixed-size blocks called pages, which correspond to blocks in physical memory known as frames.

- **Segmentation:** Divides virtual memory into variable-sized segments based on logical divisions within a program, such as functions or data structures.

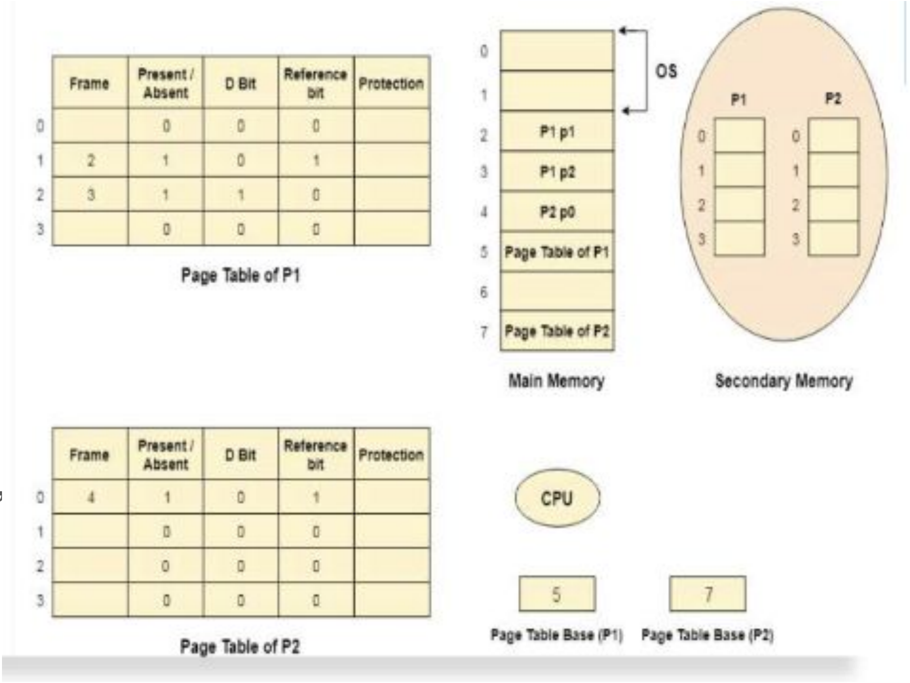


How Virtual Memory Works

- **Address Translation:** The Memory Management Unit (MMU) translates virtual addresses to physical addresses.

- **Page Tables:** Data structures that store the mapping between virtual pages and physical frames.

- **Demand Paging:** Pages are loaded into physical memory only when they are needed, reducing the initial load time and memory usage.



	Frame	Present / Absent	D Bit	Reference bit	Protection
0		0	0	0	
1	2	1	0	1	
2	3	1	1	0	
3		0	0	0	

Page Table of P1

	Frame	Present / Absent	D Bit	Reference bit	Protection
0	4	1	0	1	
1		0	0	0	
2		0	0	0	
3		0	0	0	

Page Table of P2

Benefits of Virtual Memory

- **Isolation:** Each process operates in its own virtual address space, enhancing security and stability.

- **Efficient Memory Use:** Allows for more processes to run concurrently by utilizing disk space to extend RAM.

- **Simplified Programming:** Programmers can write code without worrying about the physical memory limitations or allocation.

Page Replacement Algorithms

- **FIFO (First-In-First-Out):** Replacest the oldest page in memory.
- **LRU (Least Recently Used):** Replaces the page that hasen’t been used for the longest period.
- **Optimal:** Replaces the page that will not be used for the longest time in the future (Theoretical).

Page Replacement Algorithms

- **FIFO (First-In-First-Out):** Replacest the oldest page in memory.
- **LRU (Least Recently Used):** Replaces the page that hasen’t been used for the longest period.
- **Optimal:** Replaces the page that will not be used for the longest time in the future (Theoretical).

Allocation Strategies

- **Equal Allocation:** Each process receives an equal share of the available frames.
- **Proportional Allocation:** Frames are allocated based on the size or priority of each process.
- **Global vs Local Allocation:** Determines whether page replacement can occur across all processes (global) or is restricted to the process that caused the page fault (local).

Challenges and Considerations

- **Thrashing:** Occurs when excessive paging operations reduce overall system performance.
- **Overhead:** Maintaining page tables and performing address translations can introduce computational overhead.
- **Security:** Ensuring that processes cannot access each other’s memory spaces without permission.

Example Old Exam Question

Question 2.2.2: Consider a system with the following properties:

- Memory is byte-addressed.
- Virtual addresses are 15 bits wide.
- Physical addresses are 13 bits wide.
- The page size is 128 bytes.
- The TLB is 3-way set associative with 4 sets and 16 total entries. Its initial contents are:

Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	0F	10	0	11	15	1	1F	2E	1
1	0A	11	1	11	15	0	07	12	1
2	13	33	1	00	00	0	00	00	1
3	14	21	1	00	12	0	10	0A	1

- The page table has the following contents:

VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid
00	00	1	21	10	0	3F	23	0	12	34	1
01	33	1	0C	0D	0	08	17	1	13	15	1
A0	21	0	FA	00	1	A2	32	0	03	43	0

Note that all addresses are given in hexadecimal. In the following questions, you are asked, for various virtual addresses, to show the translation from virtual to physical addresses in the memory system just described. *Hint: there is one TLB hit, one page table hit, and one page fault (not necessarily in that order). This should help you double-check your work.*

Virtual address: `0xef2`

1. Bits of virtual address

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

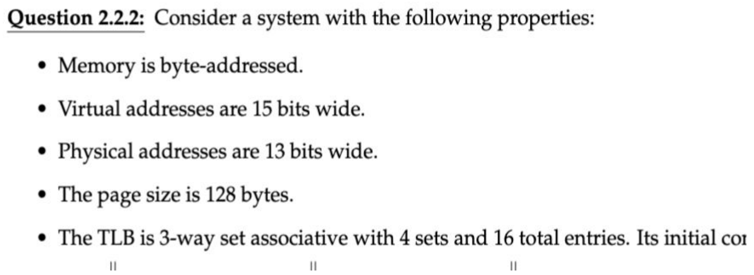
2. Address translation

Parameter	Value
VPN	
TLB index	
TLB tag	
TLB hit? (Y/N)	
Page fault? (Y/N)	
PPN	

3. Bits of phys. (if any)

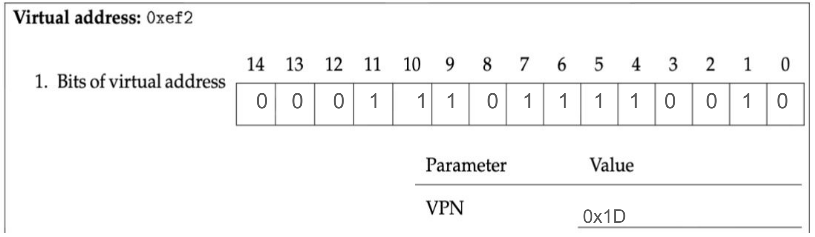
12	11	10	9	8	7	6	5	4	3	2	1	0

Example Old Exam Question



Question 2.2.2: Consider a system with the following properties:

- Memory is byte-addressed.
- Virtual addresses are 15 bits wide.
- Physical addresses are 13 bits wide.
- The page size is 128 bytes.
- The TLB is 3-way set associative with 4 sets and 16 total entries. Its initial co



Virtual address: `0xef2`

1. Bits of virtual address

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	1	1	1	1	0	0	1	0

Parameter	Value
VPN	0x1D

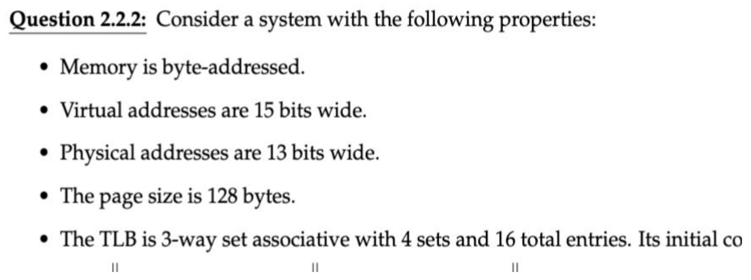
VA Bits: is found simply by converting the hex to binary.
0xef2 =1110 1111 0010.
As this is only 12 bit, add 0's for the higher bits to fill the 15 bits.

Page offset (1): Log_2(Page Size) = Log_2(128) = 7.
Then we know that the page offset is the 7 lowest bits of the **VA**

Page offset(2): 000 1110 1**111 0010**
111 0010 is the page offset

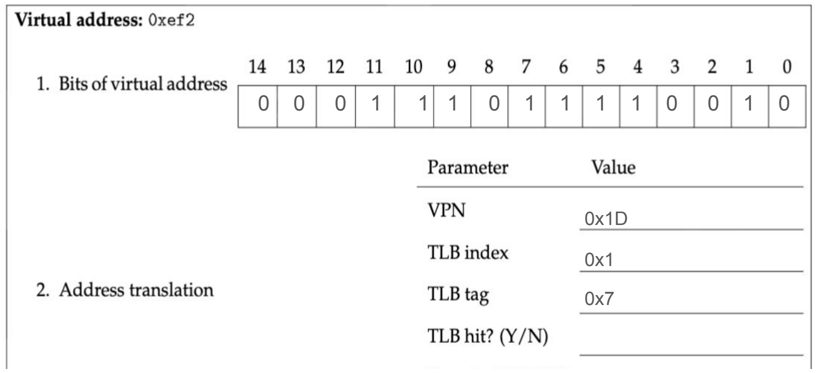
VPN: Remaining bits of **VA** (Not In Page offset) **000 1110 1**111 0010
0001 1101 is the VPN.
Convert to hex: **0x1D**

Example Old Exam Question



Question 2.2.2: Consider a system with the following properties:

- Memory is byte-addressed.
- Virtual addresses are 15 bits wide.
- Physical addresses are 13 bits wide.
- The page size is 128 bytes.
- The TLB is 3-way set associative with 4 sets and 16 total entries. Its initial co



Virtual address: `0xef2`

1. Bits of virtual address

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	1	1	1	1	0	0	1	0

2. Address translation

Parameter	Value
VPN	0x1D
TLB index	0x1
TLB tag	0x7
TLB hit? (Y/N)	

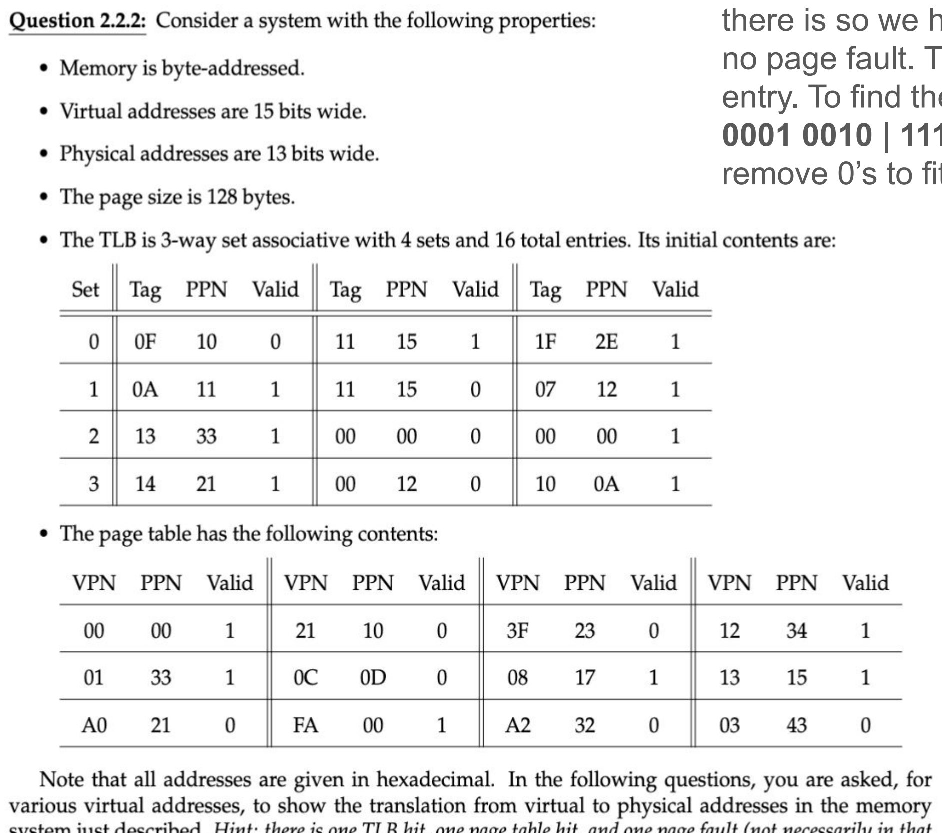
Page offset(2): 000 1110 1**111 0010**
111 0010 is the page offset

VPN: Remaining bits of **VA** (Not In Page offset) **000 1110 1**111 0010
0001 1101 is the VPN.
Convert to hex: **0x1D**

TLB Index (1): Log_2(sets). We have 4 sets. Log_2(4)=2. Now we know that the TLB index is the 2 lowest bit of the **VPN.**
TLB Index (2): 0001 11**01**
Convert to hex: **0x1**
TLB Tag: Remaining bits in the **VPN** (Not In TLB Index)
0001 1101 =00 0111
Convert to hex: **0x7**

Example Old Exam Question

TLB Check: Now we look at the TLB table first. Our TLB index is 1 so we check set 1, if there exists an entry for TLB Tag 0x7 with **valid = 1**. In this case there is so we have a TLB hit. If we have a valid TLB hit it follows that there is no page fault. The **Physical page number (PPN)** can be found in the TLB entry. To find the physical address we contacenate: **PA = PPN | Page Offset = 0001 0010 | 111 0010 = 000 1001 0111 0010.** If too large due to 0's simply remove 0's to fit.



Question 2.2.2: Consider a system with the following properties:

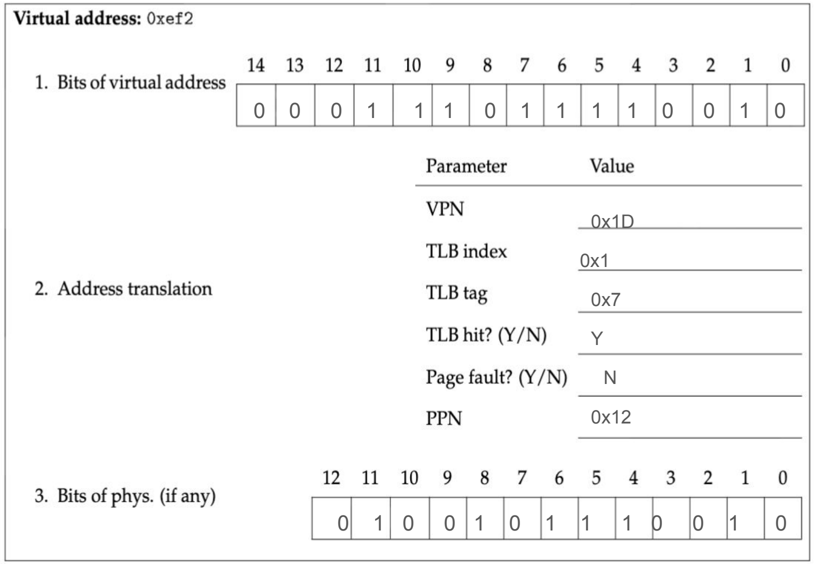
- Memory is byte-addressed.
- Virtual addresses are 15 bits wide.
- Physical addresses are 13 bits wide.
- The page size is 128 bytes.
- The TLB is 3-way set associative with 4 sets and 16 total entries. Its initial contents are:

Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	0F	10	0	11	15	1	1F	2E	1
1	0A	11	1	11	15	0	07	12	1
2	13	33	1	00	00	0	00	00	1
3	14	21	1	00	12	0	10	0A	1

- The page table has the following contents:

VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid
00	00	1	21	10	0	3F	23	0	12	34	1
01	33	1	0C	0D	0	08	17	1	13	15	1
A0	21	0	FA	00	1	A2	32	0	03	43	0

Note that all addresses are given in hexadecimal. In the following questions, you are asked, for various virtual addresses, to show the translation from virtual to physical addresses in the memory system just described. *Hint: there is one TLB hit, one page table hit, and one page fault (not necessarily in that*



Virtual address: `0xef2`

1. Bits of virtual address

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	1	1	1	1	0	0	1	0

2. Address translation

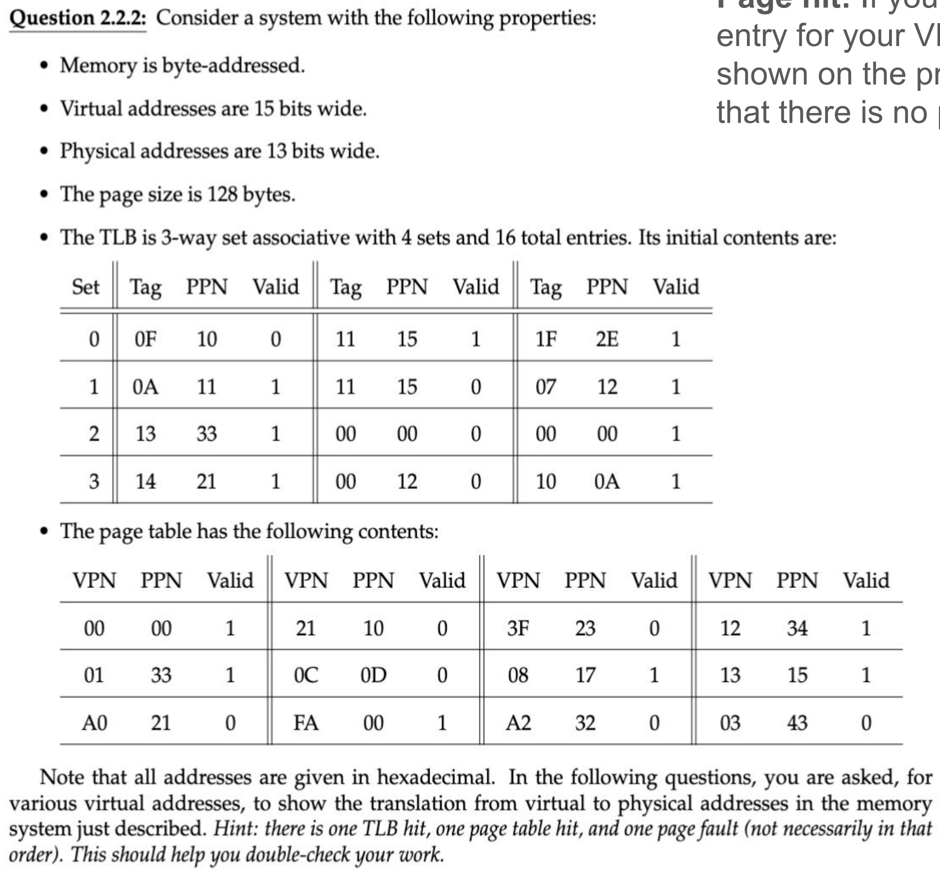
Parameter	Value
VPN	0x1D
TLB index	0x1
TLB tag	0x7
TLB hit? (Y/N)	Y
Page fault? (Y/N)	N
PPN	0x12

3. Bits of phys. (if any)

12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	1	1	1	0	0	1	0

Example Old Exam Question

Page hit: If you don't have a valid TLB hit, you check the page table for a **valid** entry for your VPN. If there is a valid entry you find the **PA** in the same way as shown on the previous slide - PPN | Page Offset. This of course also means that there is no page fault.



Question 2.2.2: Consider a system with the following properties:

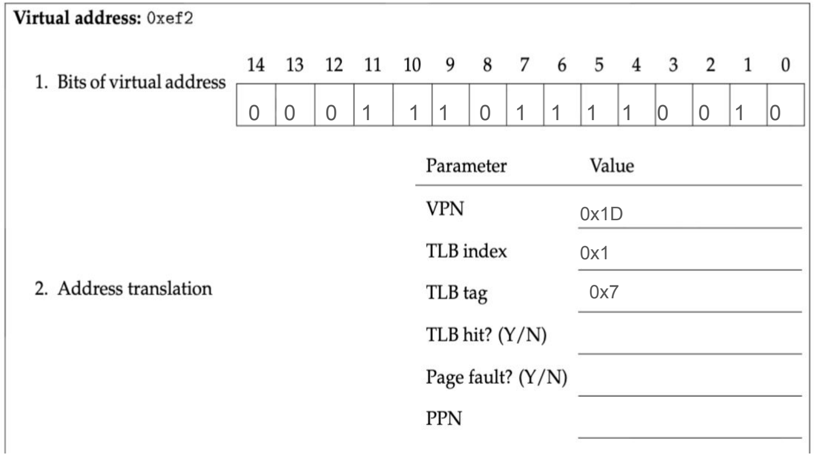
- Memory is byte-addressed.
- Virtual addresses are 15 bits wide.
- Physical addresses are 13 bits wide.
- The page size is 128 bytes.
- The TLB is 3-way set associative with 4 sets and 16 total entries. Its initial contents are:

Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	0F	10	0	11	15	1	1F	2E	1
1	0A	11	1	11	15	0	07	12	1
2	13	33	1	00	00	0	00	00	1
3	14	21	1	00	12	0	10	0A	1

- The page table has the following contents:

VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid
00	00	1	21	10	0	3F	23	0	12	34	1
01	33	1	0C	0D	0	08	17	1	13	15	1
A0	21	0	FA	00	1	A2	32	0	03	43	0

Note that all addresses are given in hexadecimal. In the following questions, you are asked, for various virtual addresses, to show the translation from virtual to physical addresses in the memory system just described. *Hint: there is one TLB hit, one page table hit, and one page fault (not necessarily in that order). This should help you double-check your work.*



Virtual address: `0xef2`

1. Bits of virtual address

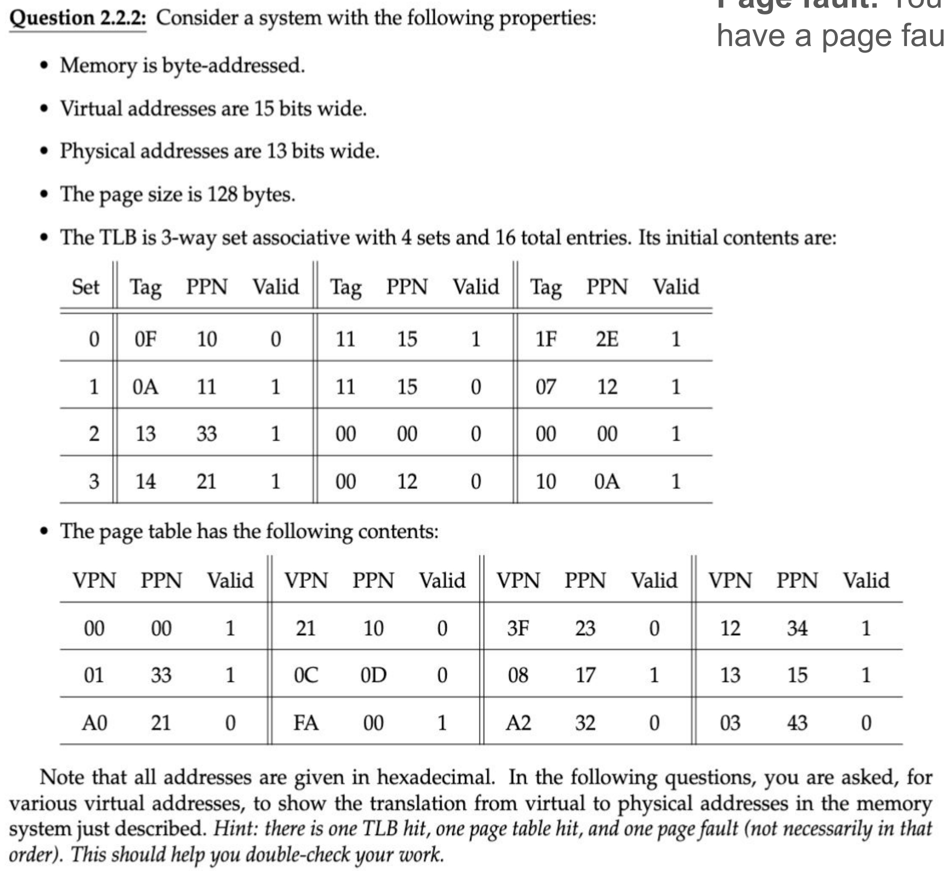
14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	1	1	1	1	0	0	1	0

2. Address translation

Parameter	Value
VPN	0x1D
TLB index	0x1
TLB tag	0x7
TLB hit? (Y/N)	
Page fault? (Y/N)	
PPN	

Example Old Exam Question

Page fault: You neither have a valid TLB hit or valid Page table hit, then you have a page fault. Here there is no physical address.



Question 2.2.2: Consider a system with the following properties:

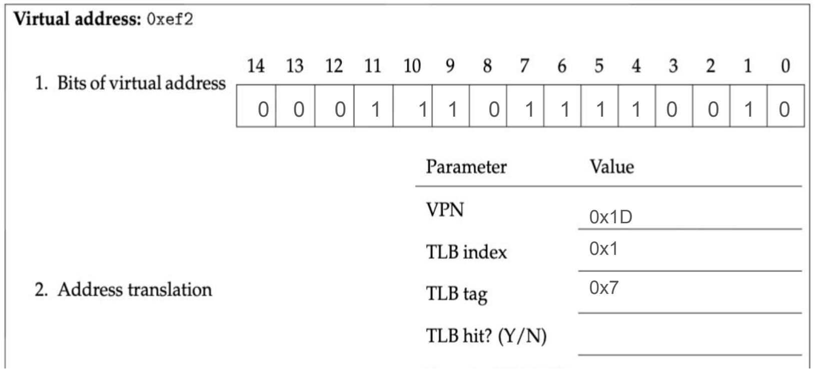
- Memory is byte-addressed.
- Virtual addresses are 15 bits wide.
- Physical addresses are 13 bits wide.
- The page size is 128 bytes.
- The TLB is 3-way set associative with 4 sets and 16 total entries. Its initial contents are:

Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	0F	10	0	11	15	1	1F	2E	1
1	0A	11	1	11	15	0	07	12	1
2	13	33	1	00	00	0	00	00	1
3	14	21	1	00	12	0	10	0A	1

- The page table has the following contents:

VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid
00	00	1	21	10	0	3F	23	0	12	34	1
01	33	1	0C	0D	0	08	17	1	13	15	1
A0	21	0	FA	00	1	A2	32	0	03	43	0

Note that all addresses are given in hexadecimal. In the following questions, you are asked, for various virtual addresses, to show the translation from virtual to physical addresses in the memory system just described. *Hint: there is one TLB hit, one page table hit, and one page fault (not necessarily in that order). This should help you double-check your work.*



Virtual address: `0xef2`

1. Bits of virtual address

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	1	1	1	1	0	0	1	0

2. Address translation

Parameter	Value
VPN	0x1D
TLB index	0x1
TLB tag	0x7
TLB hit? (Y/N)	

Solving old Exams

The example is from 22/23 exam, so maybe try another one, or try and solve all 3 cases yourself.