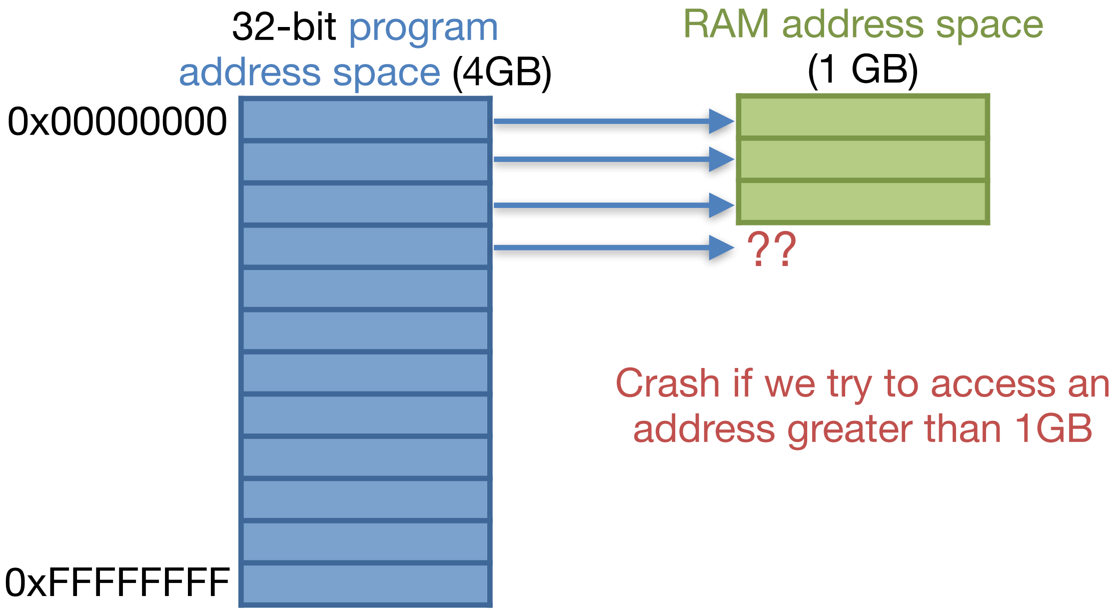
Virtual Memory

Special thanks to David Black-Schaffer

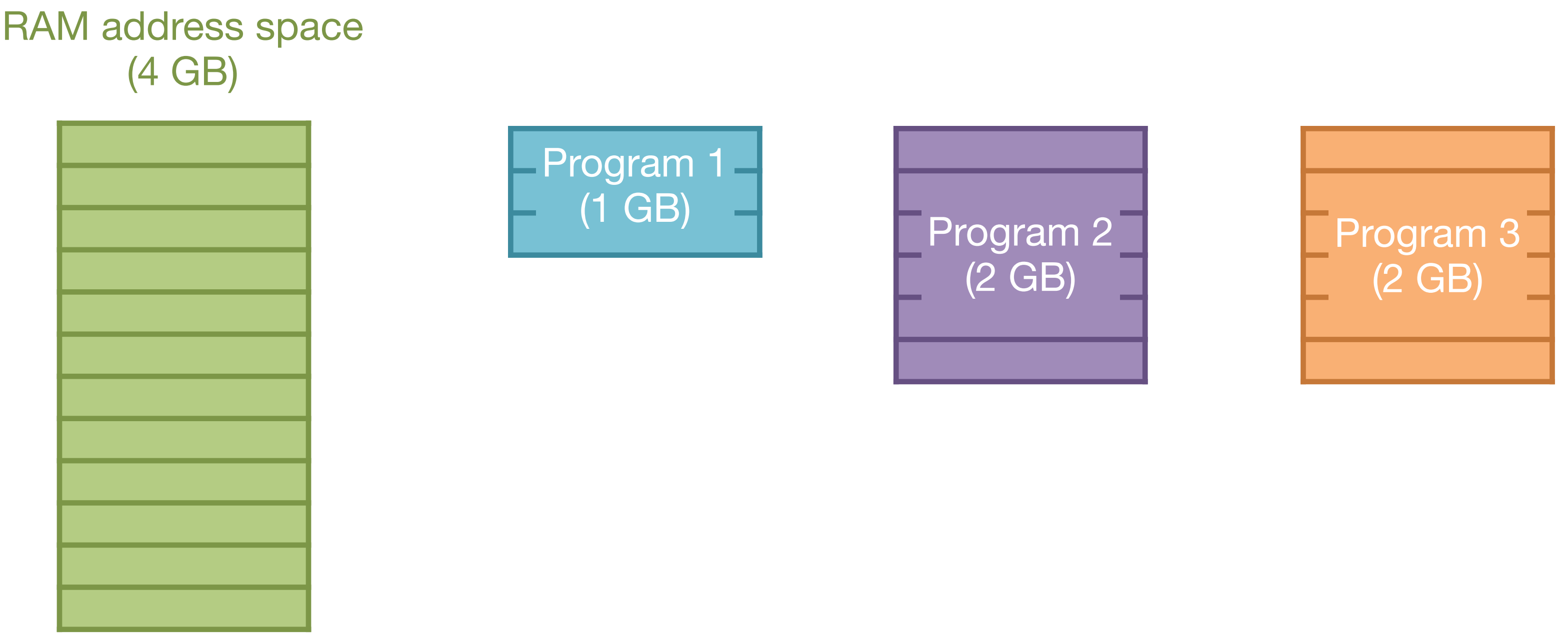
Three Problems with Memory

#1: Not Enough Space

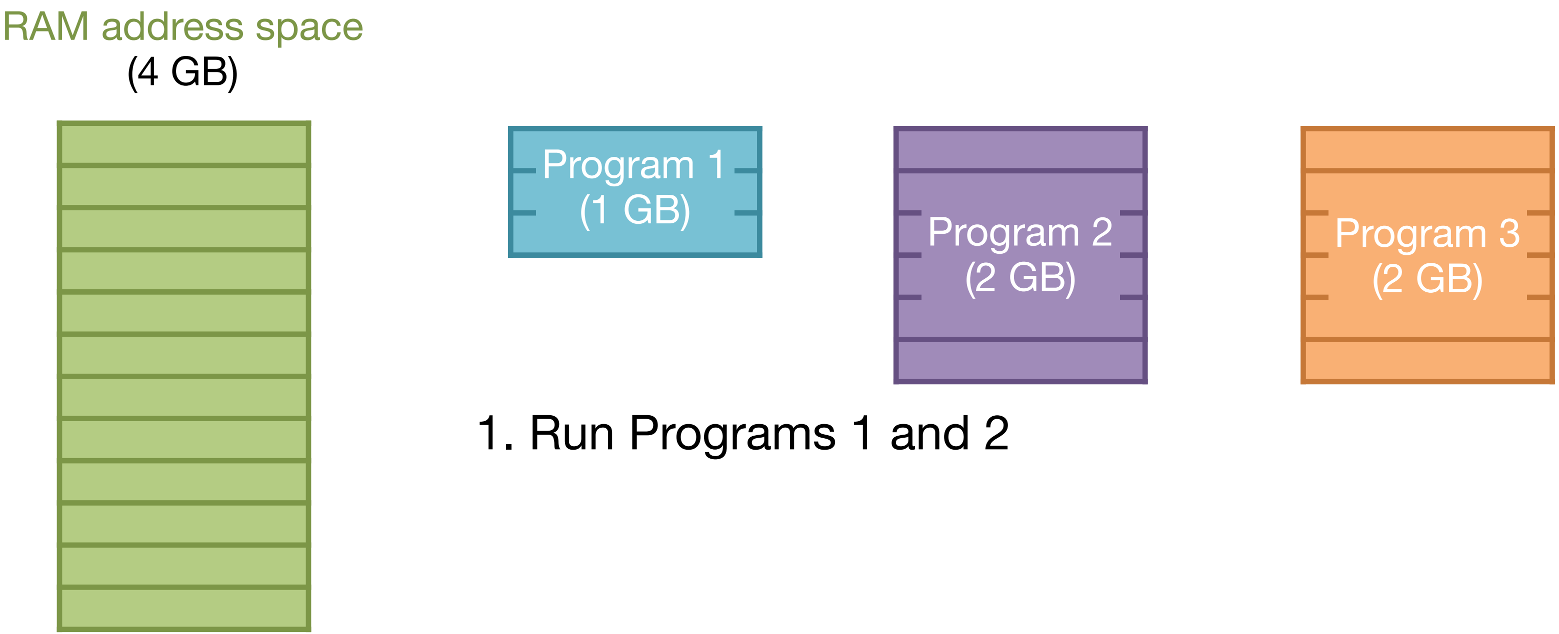
- RISC-V32 provides a 32-bit address space
  - Q: How much memory can I access with a 32-bit address?
    - 2^{32} bytes = 4GB



#2: Holes in Address Space

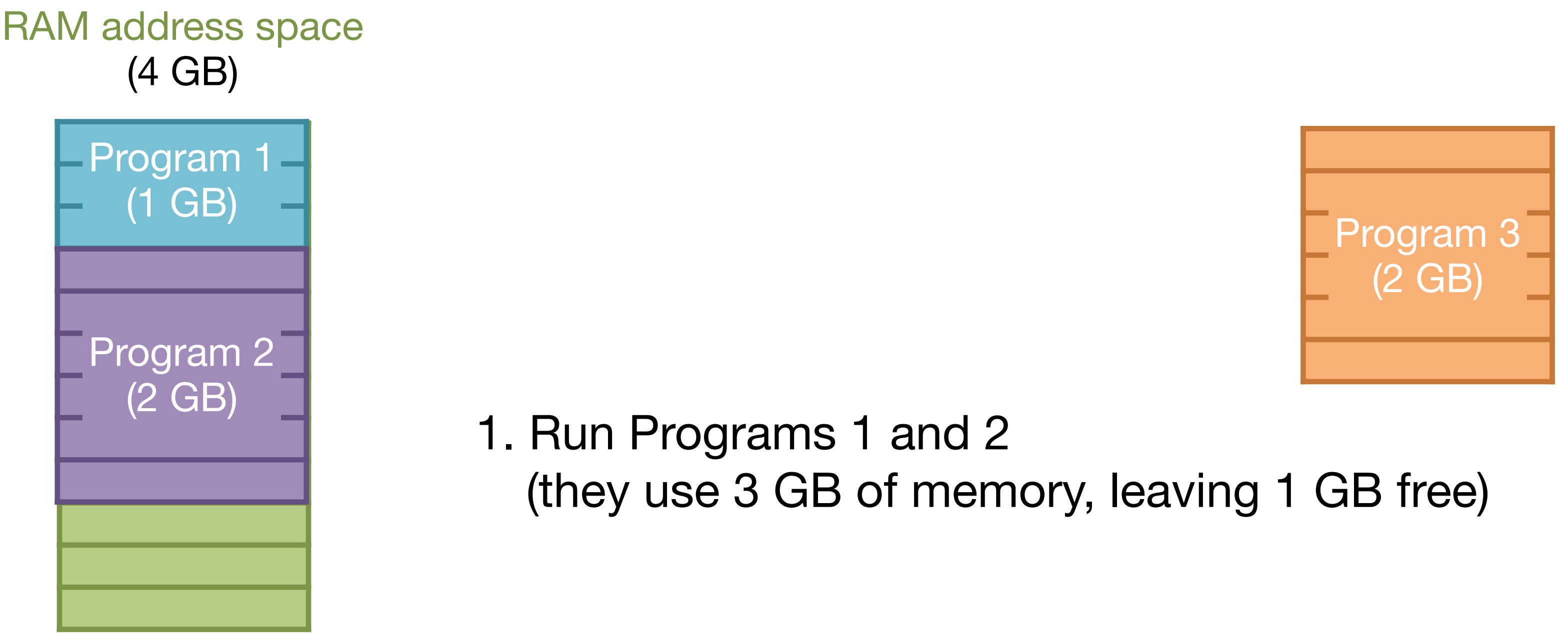


#2: Holes in Address Space



1. Run Programs 1 and 2

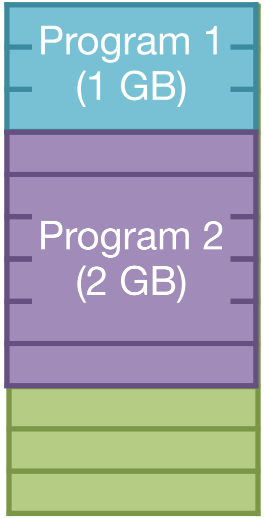
#2: Holes in Address Space

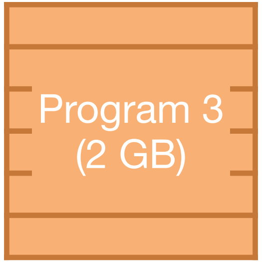


1. Run Programs 1 and 2
   (they use 3 GB of memory, leaving 1 GB free)

#2: Holes in Address Space

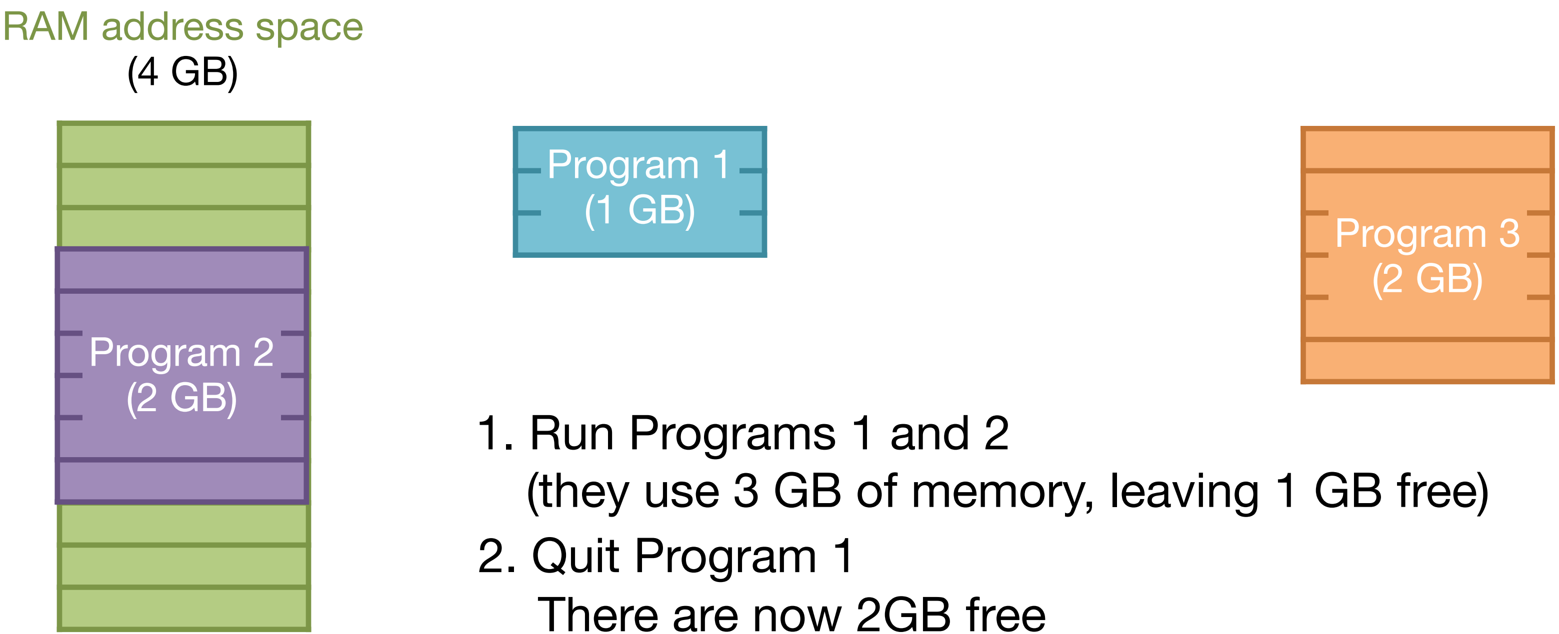
RAM address space
(4 GB)





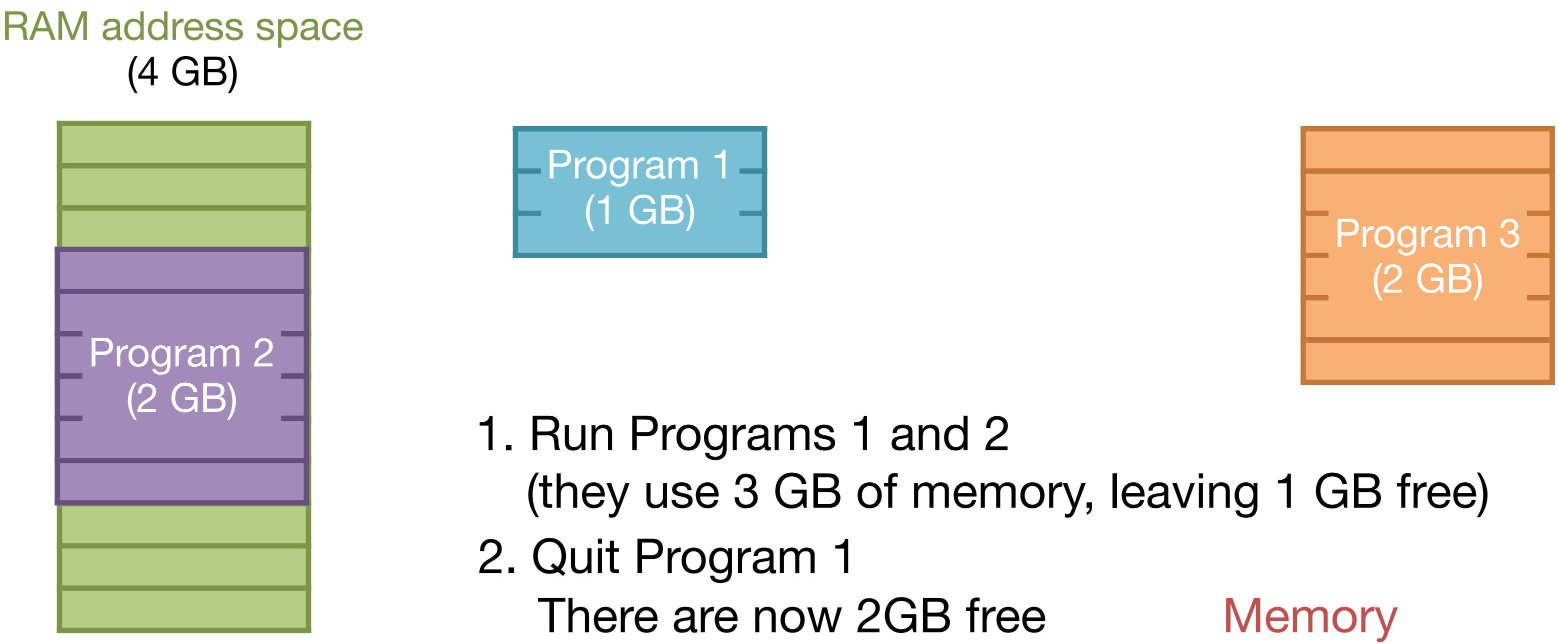
1. Run Programs 1 and 2
   (they use 3 GB of memory, leaving 1 GB free)
2. Quit Program 1

#2: Holes in Address Space



1. Run Programs 1 and 2
   (they use 3 GB of memory, leaving 1 GB free)
2. Quit Program 1
   There are now 2GB free

#2: Holes in Address Space



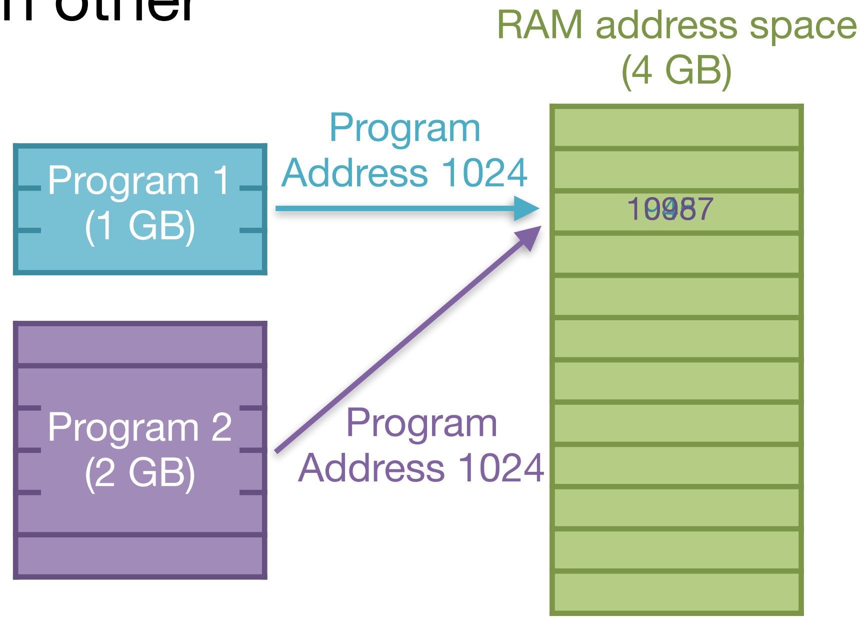
1. Run Programs 1 and 2
   (they use 3 GB of memory, leaving 1 GB free)
2. Quit Program 1
   There are now 2GB free
3. Try to run Program 3
   We can't, even though there is enough space!

Memory Fragmentation

#3: Ensuring Protection from Other Programs

- Each program can access any 32-bit memory address
- What if multiple programs access the same address?
- They can corrupt or crash each other

1. Program 1 stores your bank account balance at address 1024
2. Program 2 stores your video game score at address 1024



Problems with Memory

- If all programs have access to the same 32-bit address space
  - Can crash if there is less than 4GB of RAM
  - Can run out of space if we run multiple programs
  - Can corrupt other programs' data
- How do we solve this?
  - Give each program it's own virtual address space

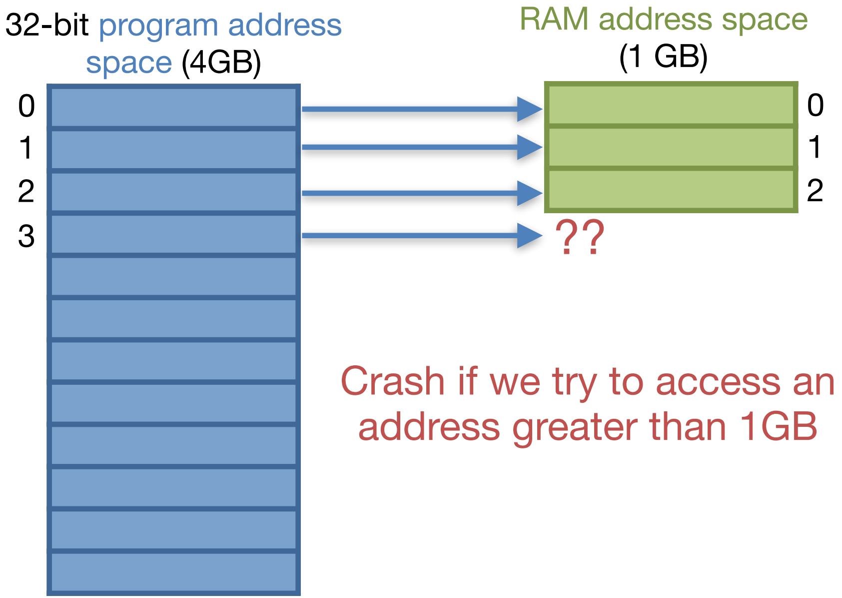
Virtual Memory

Virtual Memory: Indirection

Virtual memory takes program addresses and maps them to RAM addresses

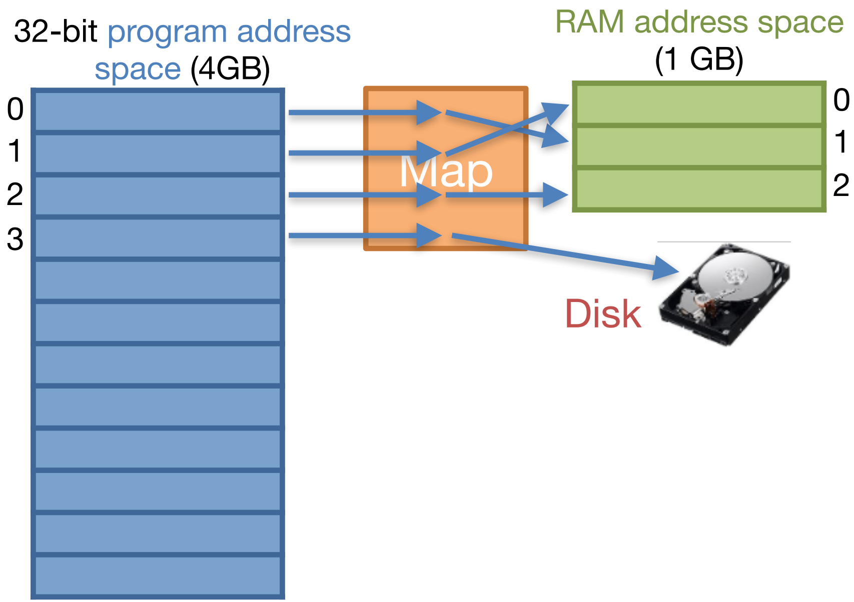
No Virtual Memory

program address = RAM address



Virtual Memory

program address maps to RAM address

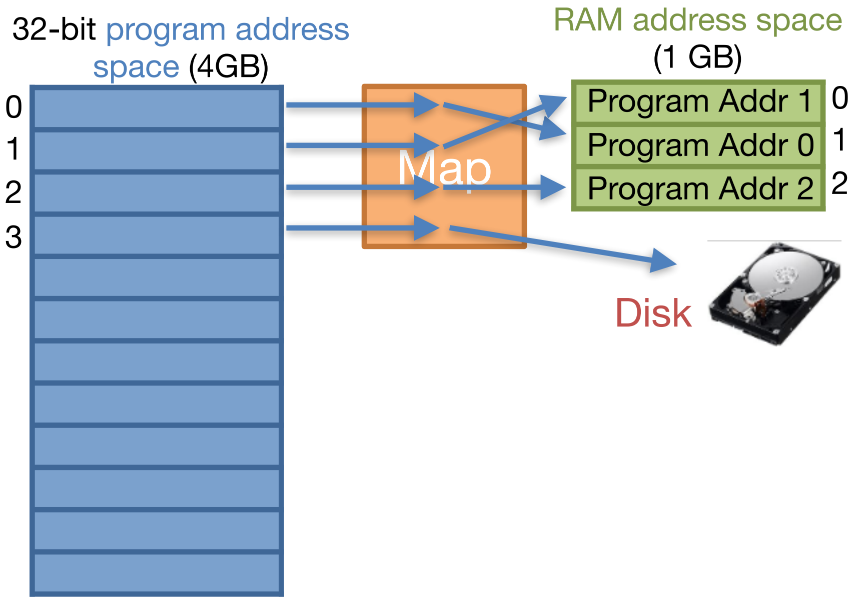


Solving Problem #1: Not Enough Memory

- Map some of the program's address space to disk
- When we need it, bring it into memory

1. Program accesses address 0
2. Program accesses address 1
3. Program accesses address 2
4. Program accesses address 3

Move out least recently accessed data
Bring in address 3 from disk

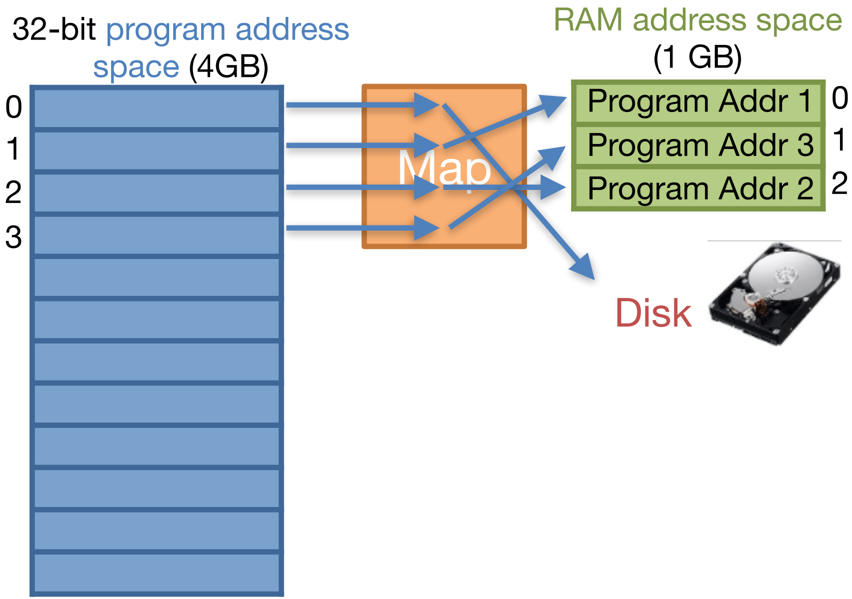


Solving Problem #1: Not Enough Memory

- Map some of the program's address space to disk
- When we need it, bring it into memory

1. Program accesses address 0
2. Program accesses address 1
3. Program accesses address 2
4. Program accesses address 3

Move out least recently accessed data
Bring in address 3 from disk

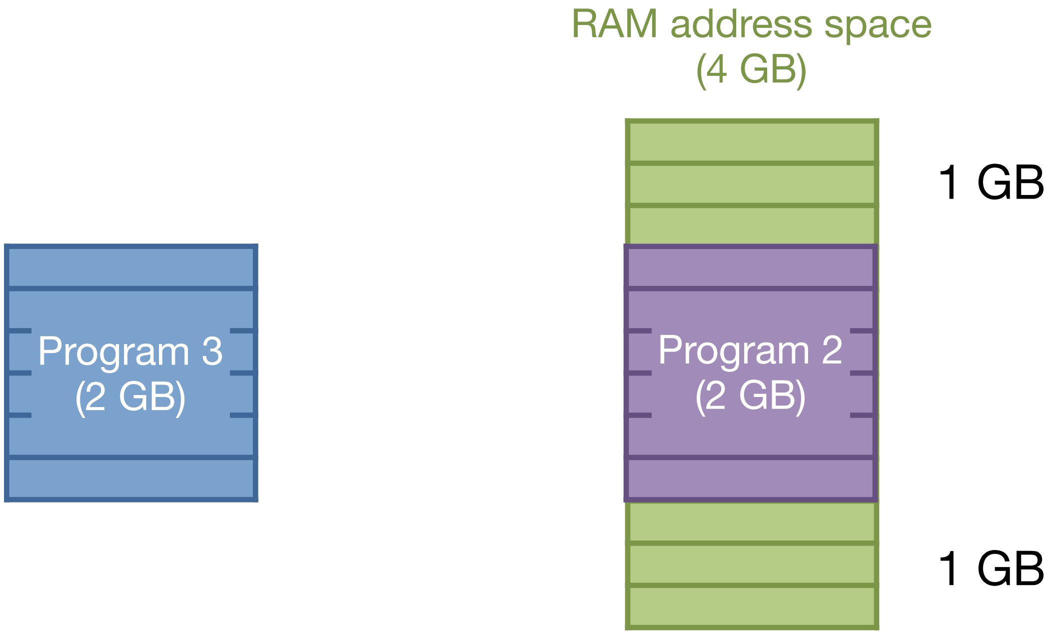


VM and Performance

- Q: What happens to the performance of the program when the data it needs is in the disk and not in main memory?
  - Performance decreases
  - Reading from disk is much slower than reading from RAM
  - Any time you can't fit your data in memory and have to go to disk you pay a HUGE performance penalty!
  - This is why buying more RAM makes your computer faster

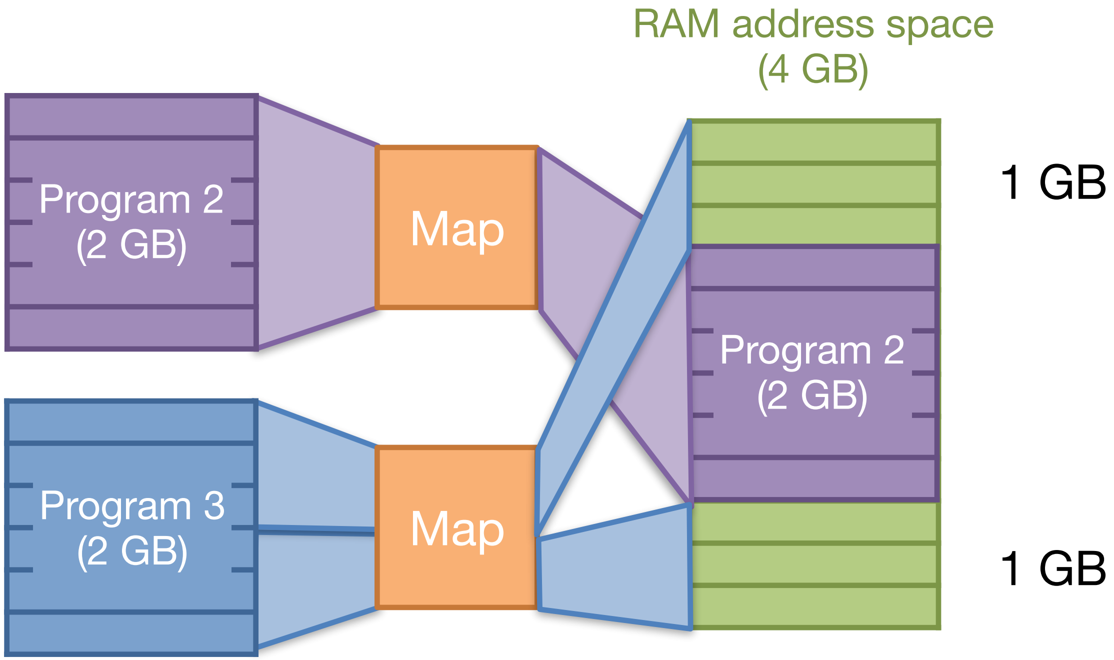
Solving Problem #2: Holes in Address Space

- How can we fit our program in the available space?



Solving Problem #2: Holes in Address Space

- How can we fit our program in the available space?

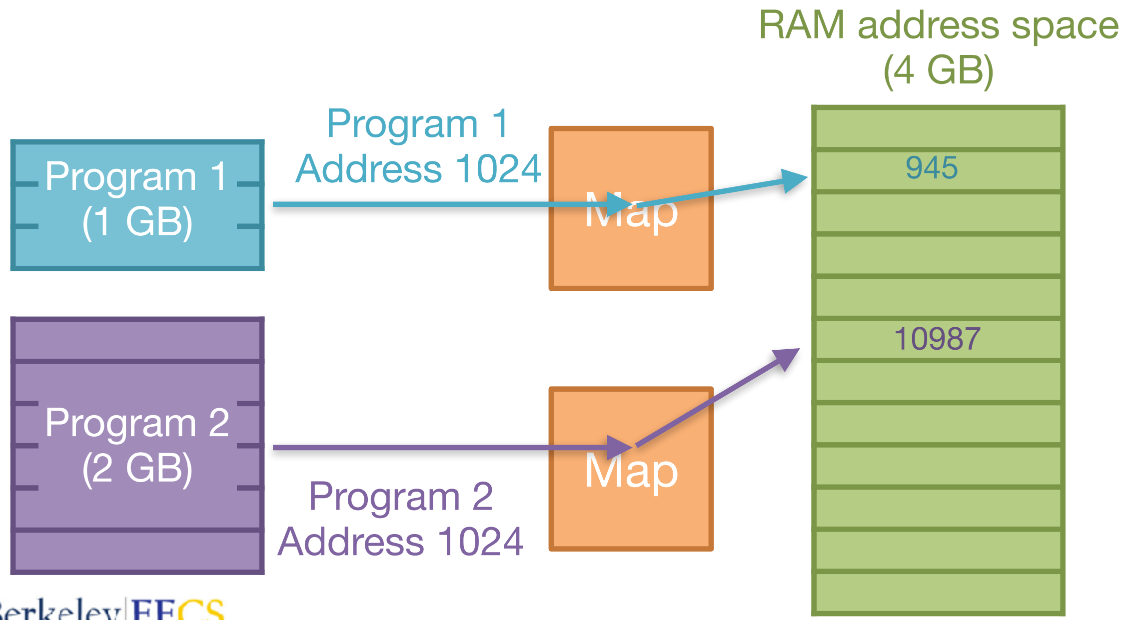


Each program has its own mapping

Mappings let us put our program data wherever we want in RAM

Solving Problem #3: Keeping Programs Secure

- Program 1's and Program 2's addresses map to different RAM addresses
- Because each program has its own address space, they cannot access each other's data



1. **Program 1** stores your bank balance at address 1024

VM maps it to **RAM address 1**

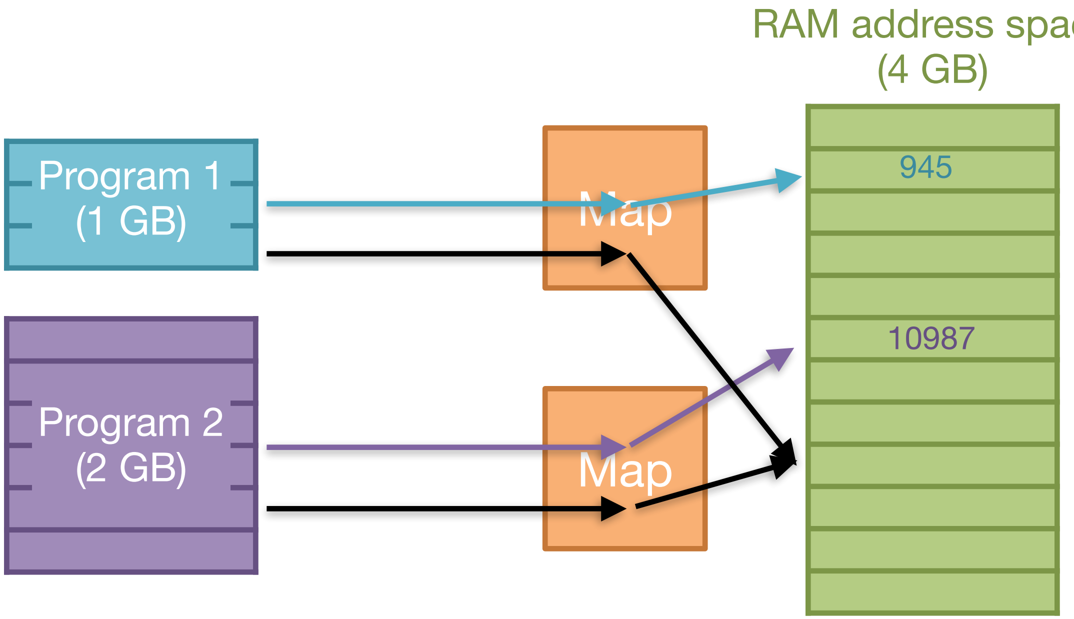
2. **Program 2** stores your video game score at address 1024

VM maps it to **RAM address 5**

Neither can touch the other's data!

Sharing Data Between Programs

- Programs share a lot of data (eg libraries)
- VM enables us to easily share data while protecting private data!



How Does VM Work?

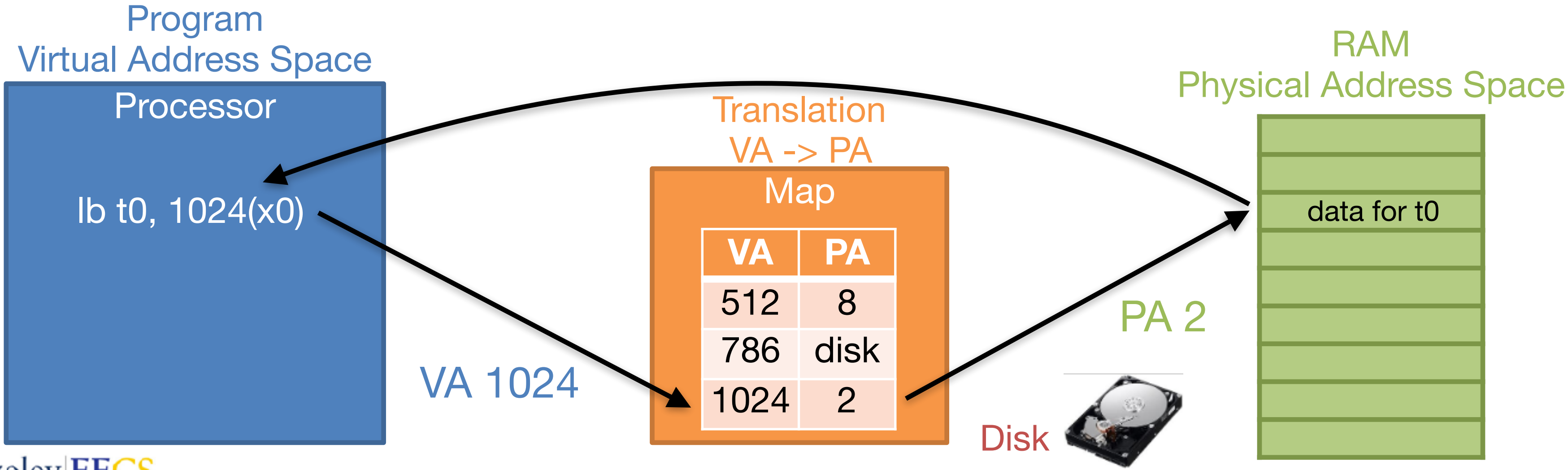
How Does VM Work?

- Virtual Address (VA)
  - What the program sees
  - e.g. lw t0, 1024(x0) # read virtual address 1024
  - In RV32I, we can access bytes 0 to $2^{32} - 1$
- Physical Address (PA)
  - The physical RAM in the computer
  - Address space is determined by how much RAM you have
  - If you have 2GB of RAM, you can access bytes 0 to $2^{31} - 1$

Making VM Work: Translation

How does a program access memory?

1. Program executes a load specifying a virtual address (VA)
2. Computer translates the address to the physical address (PA) in memory
3. If the PA is not in memory, the operating system loads it in from disk
4. The computer reads the RAM using the PA and returns the data to the program

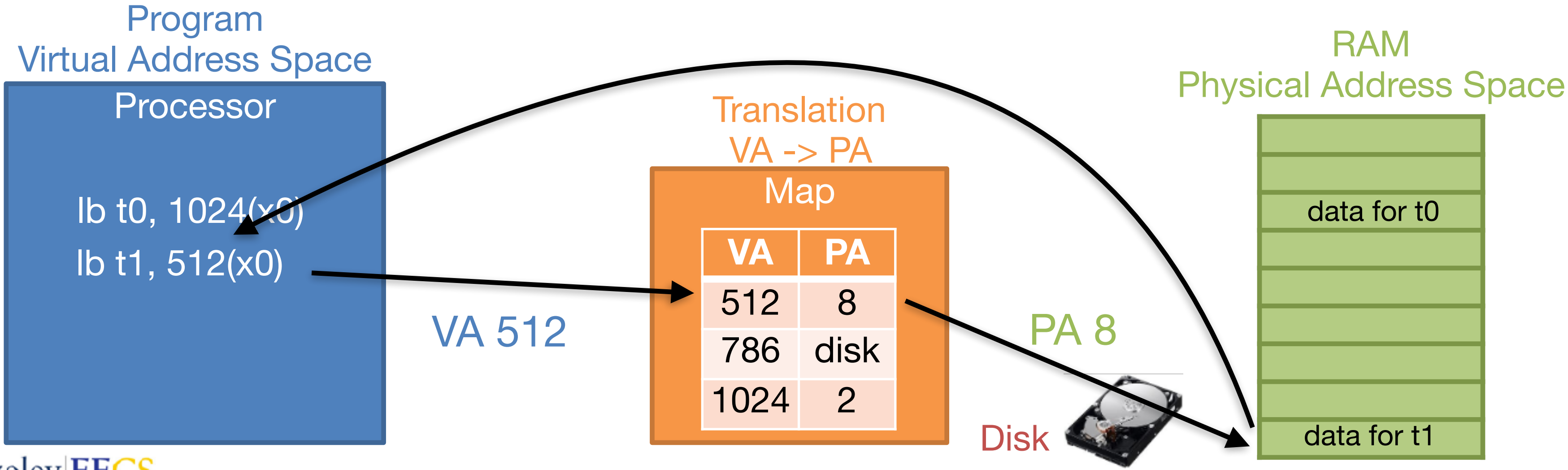


VA	PA
512	8
786	disk
1024	2

Making VM Work: Translation

How does a program access memory?

1. Program executes a load specifying a virtual address (VA)
2. Computer translates the address to the physical address (PA) in memory
3. If the PA is not in memory, the operating system loads it in from disk
4. The computer reads the RAM using the PA and returns the data to the program



Making VM Work: Translation

How does a program access memory?

1. Program executes a load specifying a virtual address (VA)
2. Computer translates the address to the physical address (PA) in memory
3. If the PA is not in memory, the operating system loads it in from disk
4. The computer reads the RAM using the PA and returns the data to the program

Program
Virtual Address Space

Processor

lb t0, 1024(x0)
lb t1, 512(x0)
add t3, t1, t2

No translation here!

Translation
VA -> PA

Map

VA	PA
512	8
786	disk
1024	2

Disk

RAM
Physical Address Space

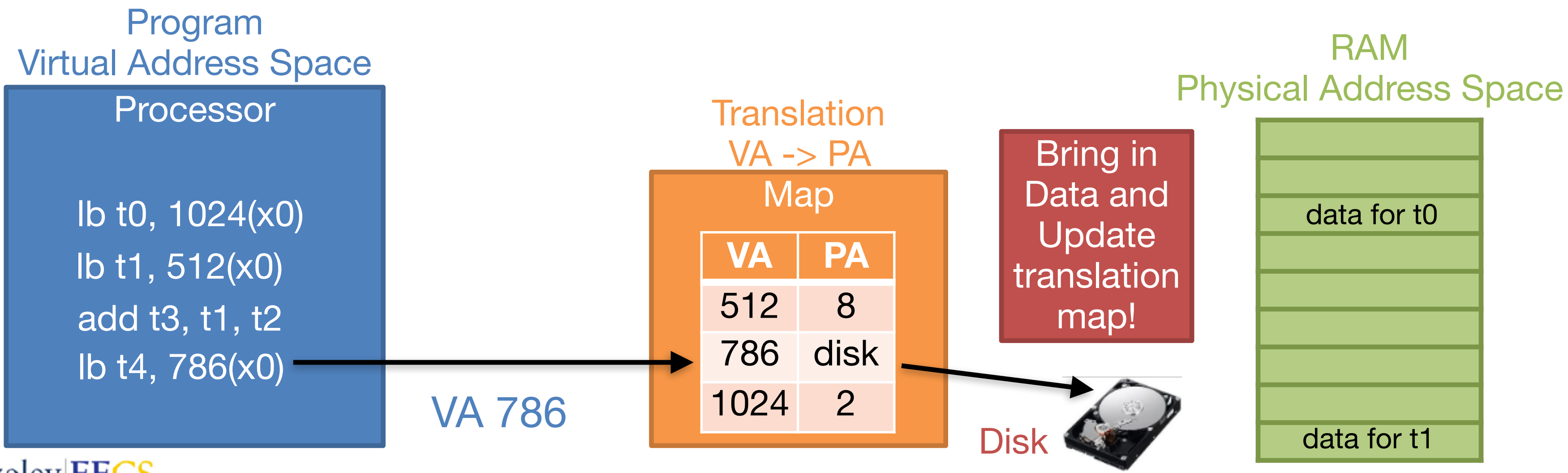
data for t0

data for t1

Making VM Work: Translation

How does a program access memory?

1. Program executes a load specifying a virtual address (VA)
2. Computer translates the address to the physical address (PA) in memory
3. If the PA is not in memory, the operating system loads it in from disk
4. The computer reads the RAM using the PA and returns the data to the program

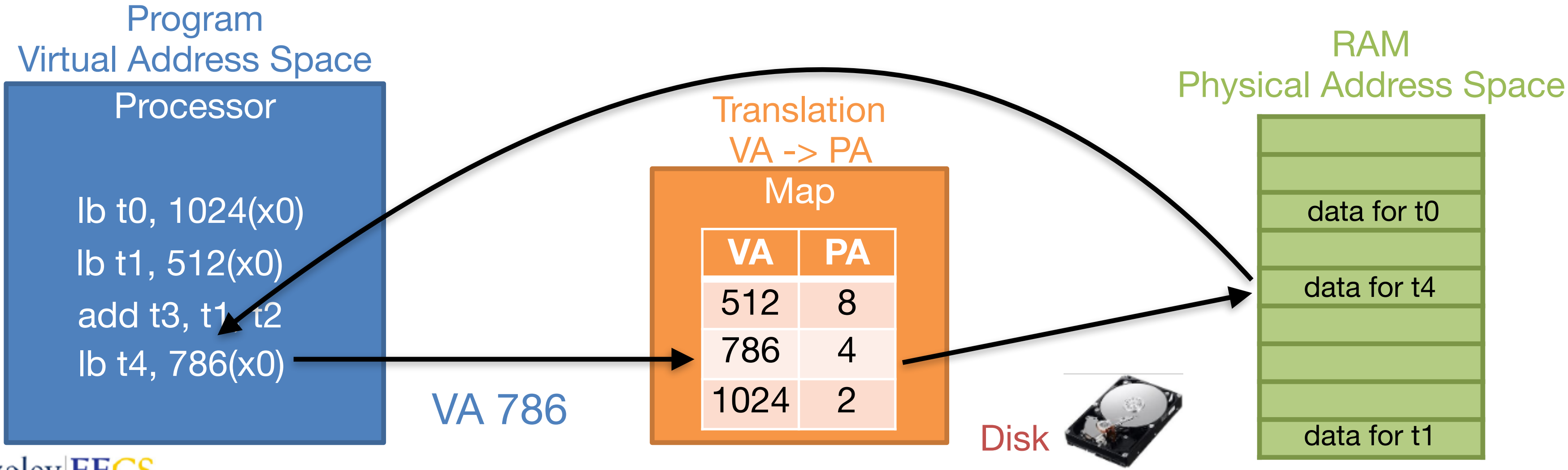


VA	PA
512	8
786	disk
1024	2

Making VM Work: Translation

How does a program access memory?

1. Program executes a load specifying a virtual address (VA)
2. Computer translates the address to the physical address (PA) in memory
3. If the PA is not in memory, the operating system loads it in from disk
4. The computer reads the RAM using the PA and returns the data to the program

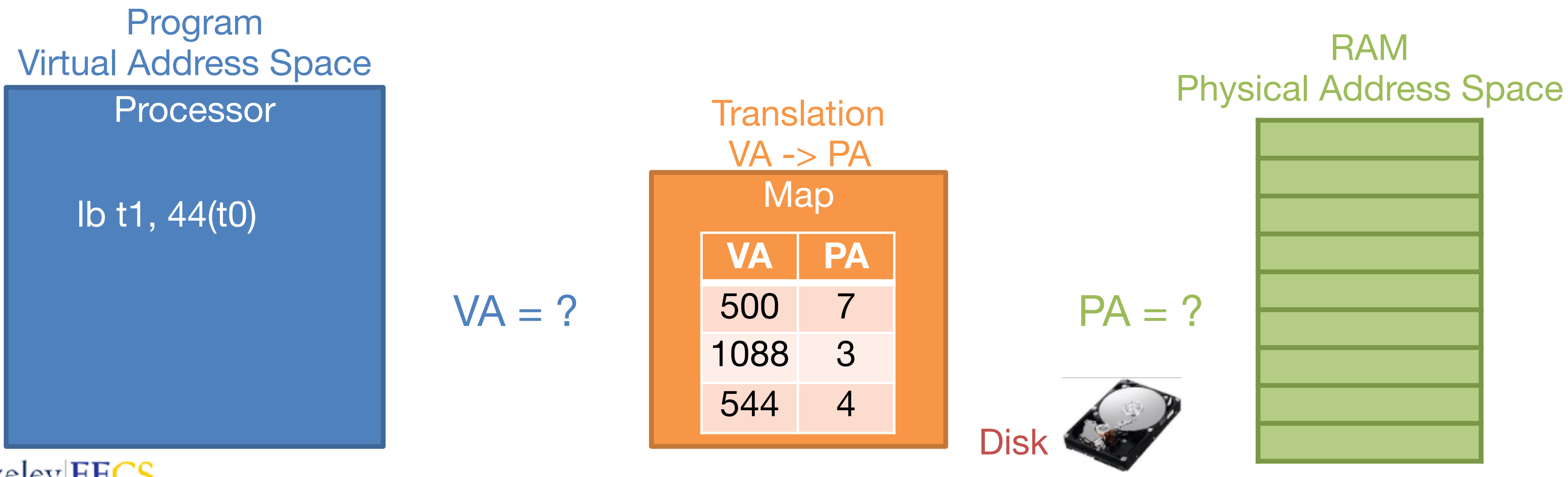


What Address is Loaded?

t0 hold address 500

The program executes **lb t1, 44(t0)**

What RAM address is accessed?



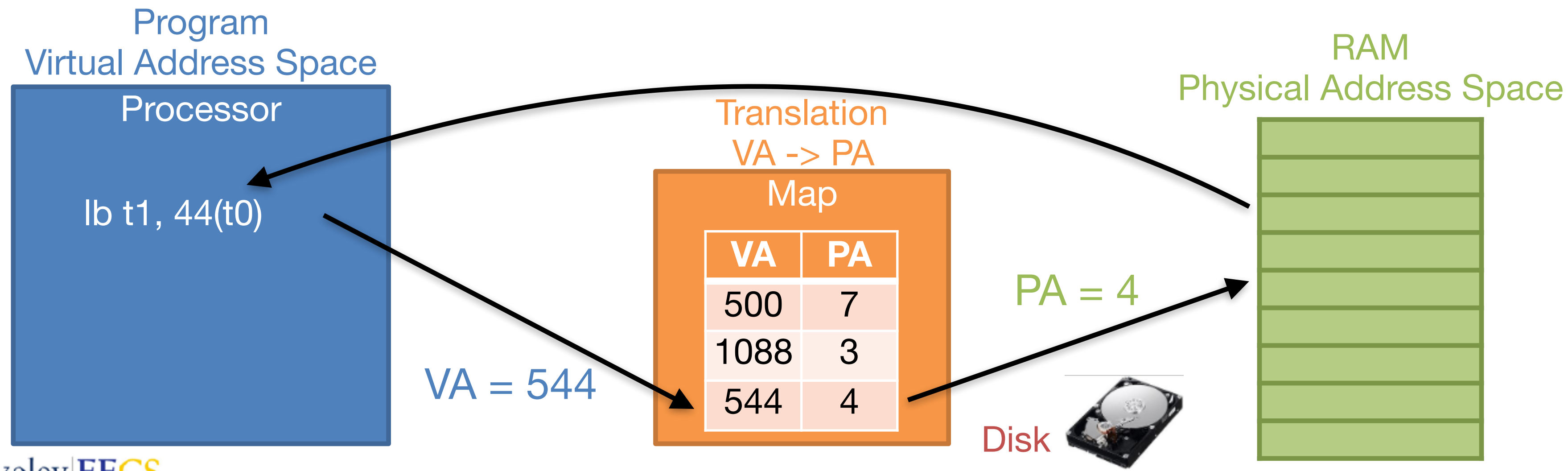
VA	PA
500	7
1088	3
544	4

What Address is Loaded?

t0 hold address 500

The program executes **lb t1, 44(t0)**

What RAM address is accessed?



Page Tables

Page Tables

- The map from Virtual Addresses (VA) to Physical Addresses (PA) is the Page Table
- So far we have had one Page Table Entry (PTE) for every Virtual Address
- If we have one entry for every word in our address space, how many entries would we have?

Page Tables

- The map from Virtual Addresses (VA) to Physical Addresses (PA) is the Page Table
- So far we have had one Page Table Entry (PTE) for every Virtual Address
- If we have one entry for every word in our address space, how many entries would we have?
  - 2^{30} = 1 billion!

How We Reduce the Number of Entries in the Page Table?

- We can divide the memory into chunks (**pages**) instead of words

Fine Grain

Map each word address

2^{30} entries

Page Table

VA -> PA

Map

VA	PA
500	32
1088	64
544	0

Coarse Grain

Map chunks of addresses

Fewer entries

Page Table

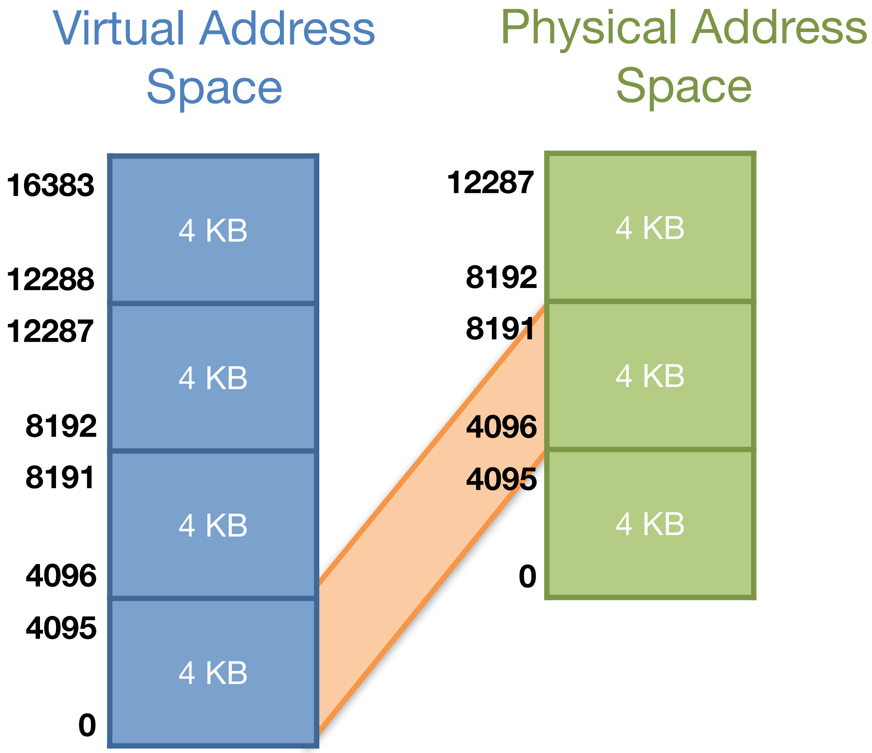
VA -> PA

Map

VA	PA
0-4095	4096-8191
…	…
…	…

Each entry now covers 4KB of data

How do we map addresses with pages?



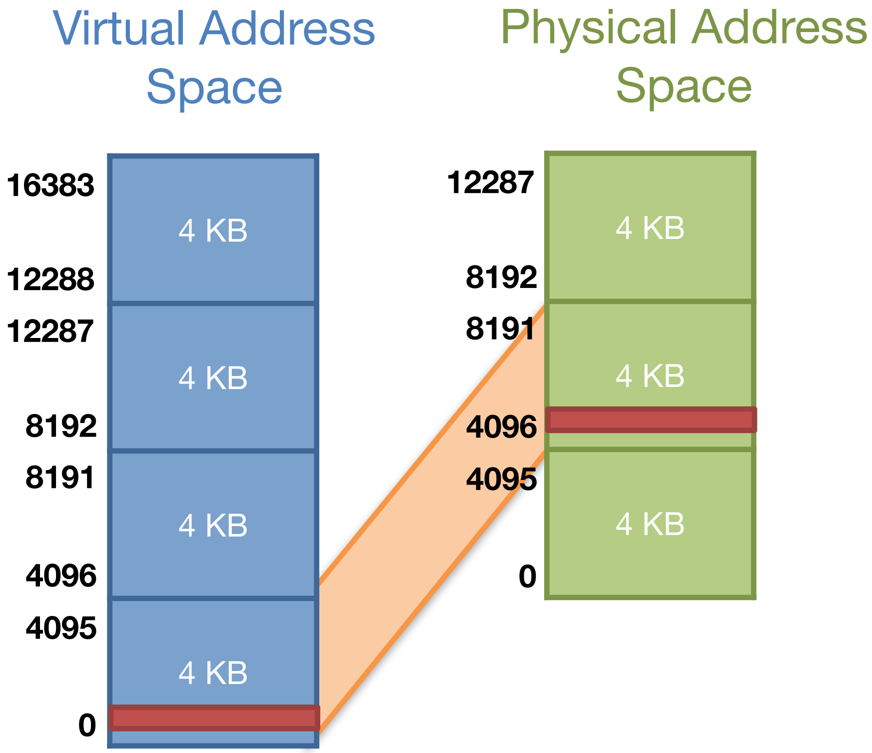
Page Table
VA -> PA

Map

VA	PA
0-4095	4096-8191
…	…
…	…

Q: What is the physical address of virtual address 4?

How do we map addresses with pages?



Page Table
VA -> PA

Map

VA	PA
0-4095	4096-8191
…	…
…	…

Q: What is the physical address of virtual address 4?

4096 + 4 = 4100

Coarse Grained Pages

- Now, when we bring in an entry from the disk, we must bring in a **whole page** instead of one word
- If I want to access a word stored at physical address 9120 and it's not in the RAM, the OS would need to bring in bytes 8192-12287
  - Remember, the program is not going to know the physical address, this is just an example

Coarse Grain

Map chunks of addresses

Fewer entries

Page Table

VA -> PA

Map

VA	PA
0-4095	4096-8191
…	…
…	…

Page Size

- Today, pages are usually 4KB (1024 words)
- Q: How many pages do we need in our page table with 4KB pages on a 32-bit machine?

Page Size

- Today, page tables are usually 4KB (1024 words)
- Q: How many pages do we need in our page table with 4KB pages on a 32-bit machine?
  - 2^{32} bytes / 2^{12} bytes = 2^{20} = 1 million

Number of bytes in memory

4 KB

- Q: How many entries do we have in our page table?

Page Size

- Today, page tables are usually 4KB (1024 words)
- Q: How many pages do we need in our page table with 4KB pages on a 32-bit machine?
  - 2^{32} bytes / 2^{12} bytes = 2^{20} = 1 million

Number of bytes in memory (2^{32} bytes)

4 KB (2^{12} bytes)

- Q: How many entries do we have in our page table?
  - 1 million

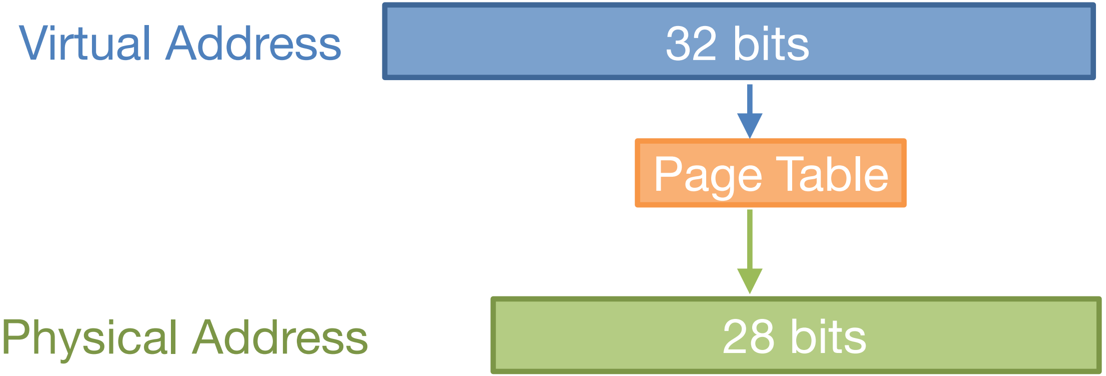
Address Translation

Address Translation

- What is the size of our virtual address and physical address on a 32 bit machine with 256 MB of RAM and 4KB pages?

Address Translation

- What is the size of our virtual address and physical address on a 32 bit machine with 256 MB of RAM and 4KB pages?
  - VA size = 32 bits
  - PA size = $\log_2(256\text{ MB}) = \log_2(2^8 * 2^{20}) = 28$ bits



Address Translation

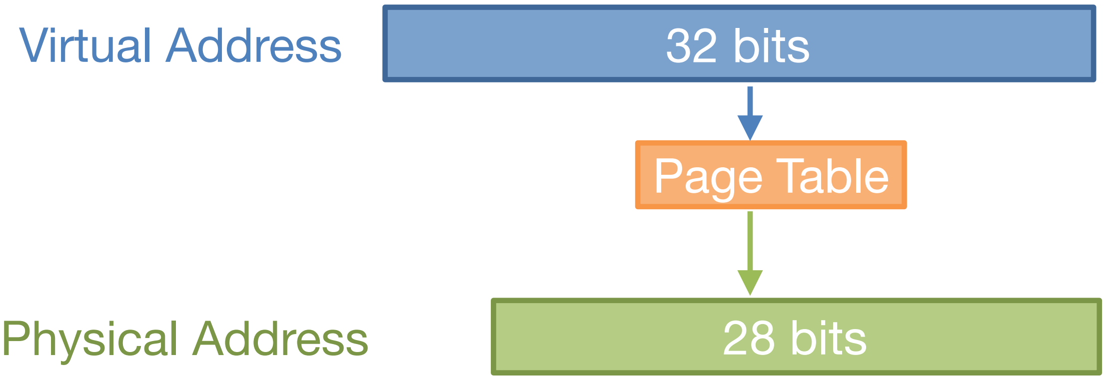
- Q: Why do we have more bits for the Virtual Address than the Physical Address?

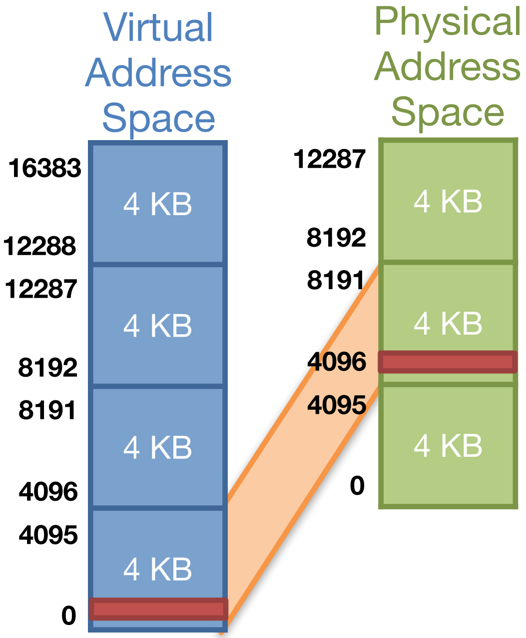
Address Translation

- Q: Why do we have more bits for the VPN than the PPN?
  - A: The virtual address space is larger than the physical address space

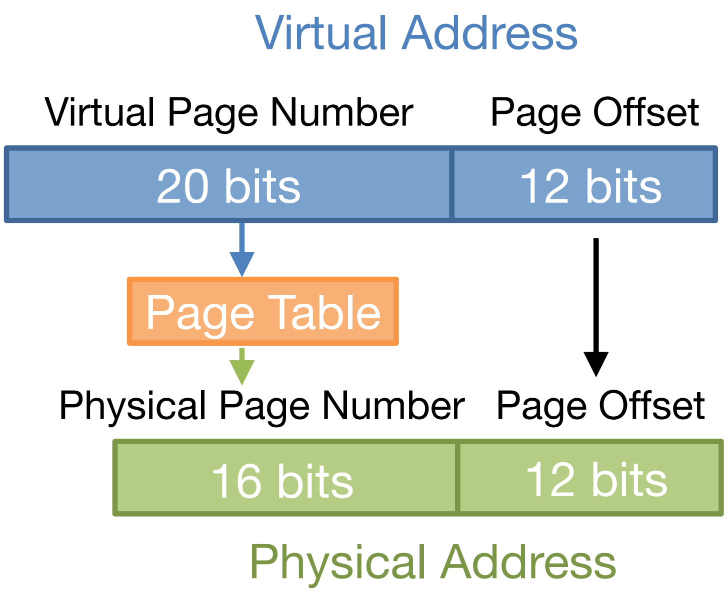
Address Translation

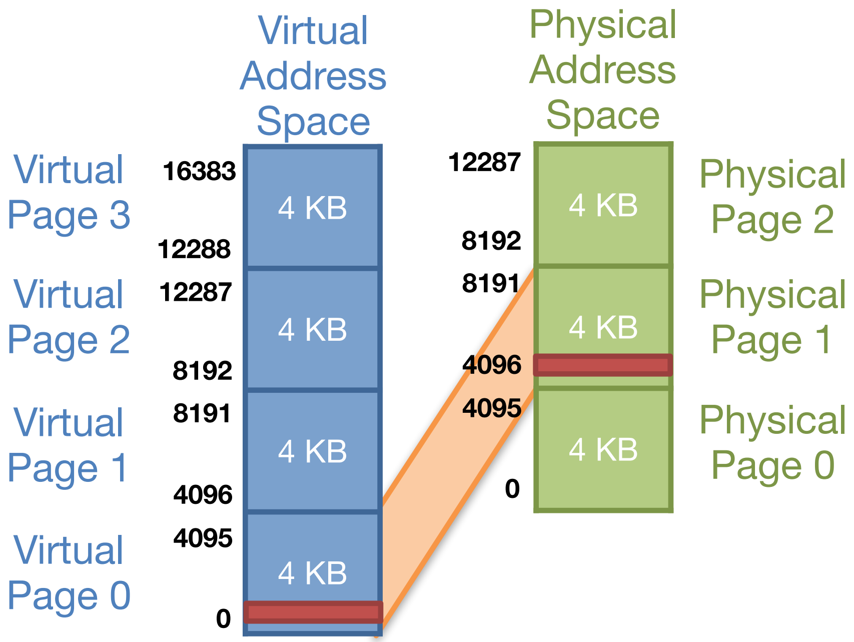
- What is the size of our virtual address and physical address on a 32 bit machine with 256 MB of RAM and 4KB pages?
  - VA size = 32 bits
  - PA size = $\log_2(256\ MB) = \log_2(2^8 * 2^{20}) = 28$ bits





Address Translation



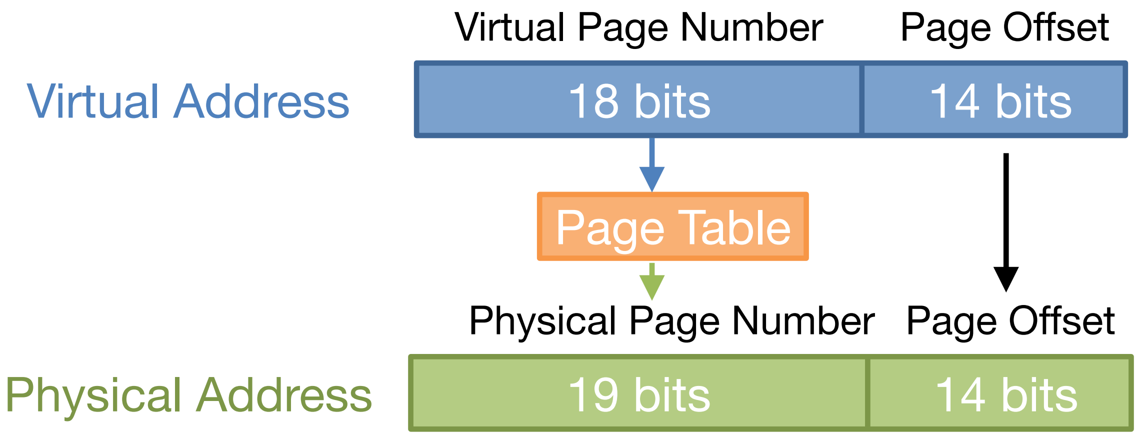


Question

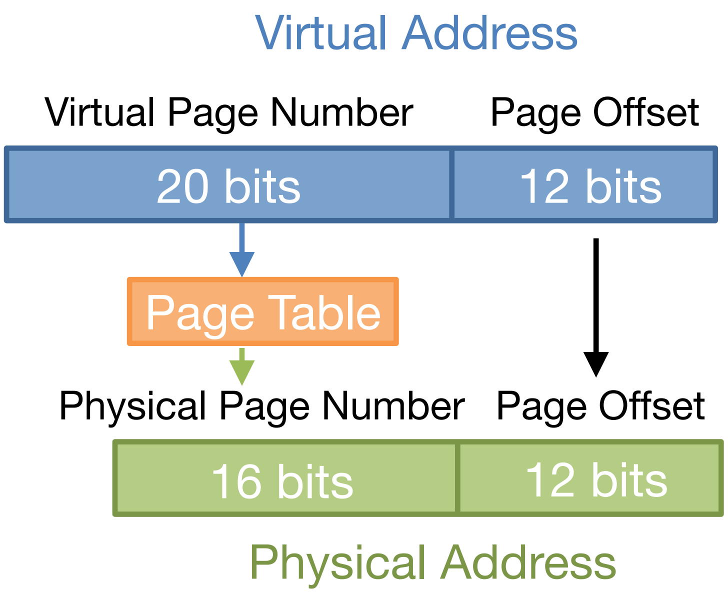
- Q: How many bits would there be for the VPN, PPN, and page offset on a 32-bit machine with 8GB of RAM and 16KB pages?

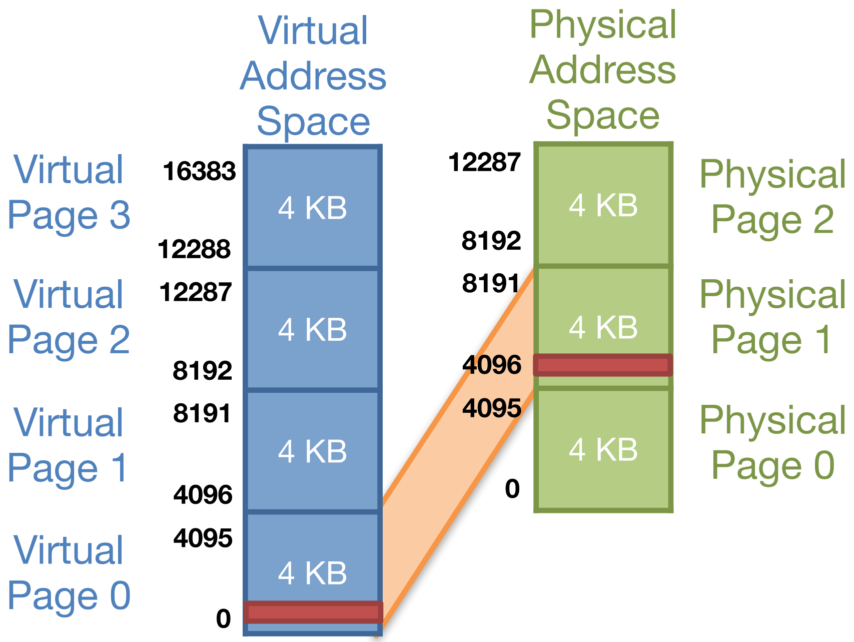
Question

- Q: How many bits would there be for the VPN, PPN, and page offset on a 32-bit machine with 8GB of RAM and 16KB pages?
  - Number of page offset bits = $\log_2(16\ \text{KB}) = \log_2(2^4 * 2^{10}) = 14$
  - Number of VPN bits = $32 - 14 = 18$
  - Number of PPN bits = $\log_2(8\text{GB}) - 14 = \log_2(2^3 * 2^{30}) - 14 = 33 - 14 = 19$



Translation Walk Through





Translation Walk-Through

Page Table

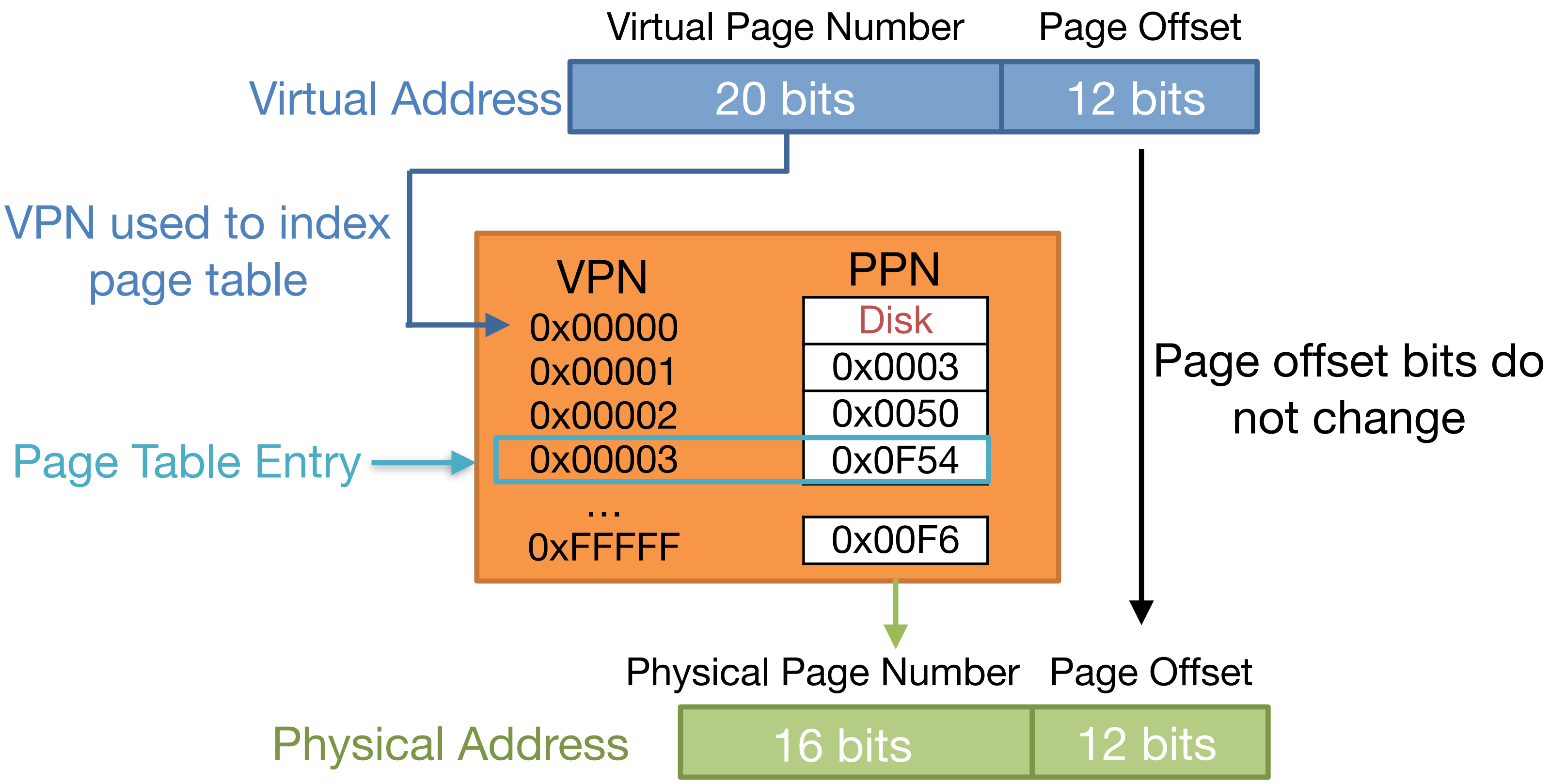
VPN	PPN
0x00000	Disk
0x00001	0x0003
0x00002	0x0050
0x00003	0x0F54
...	
0xFFFFF	0x00F6

Page Table Entry → (row 0x00003 / 0x0F54)

Page table contains mapping of every VPN to PPN

Each process has it's own page table

Translation Walk-Through



Example Translation #1

Virtual Address

Virtual Page Number	Page Offset
20 bits	12 bits

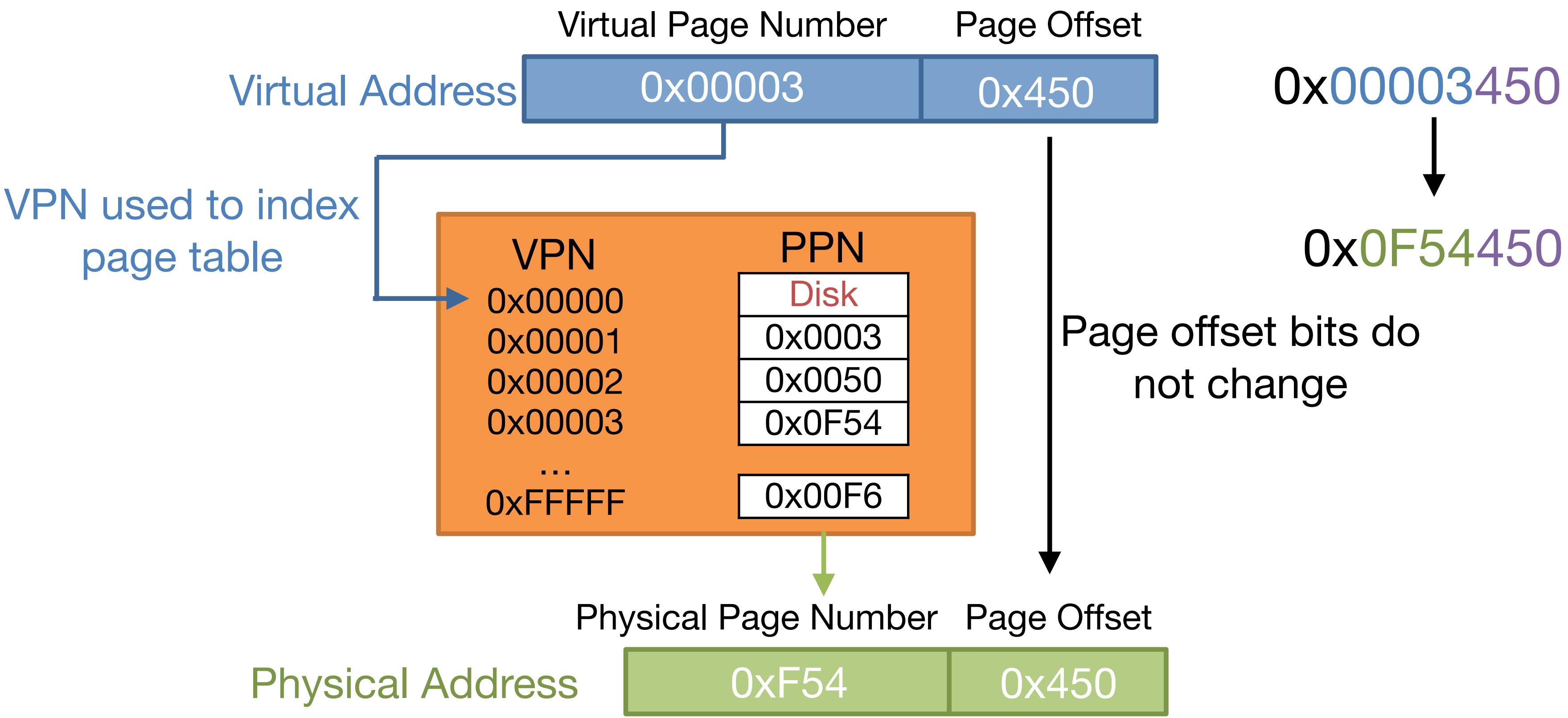
0x00003450

VPN	PPN
0x00000	Disk
0x00001	0x0003
0x00002	0x0050
0x00003	0x0F54
…	
0xFFFFF	0x00F6

Physical Address

Physical Page Number	Page Offset
16 bits	12 bits

Example Translation #1



Example Translation #2

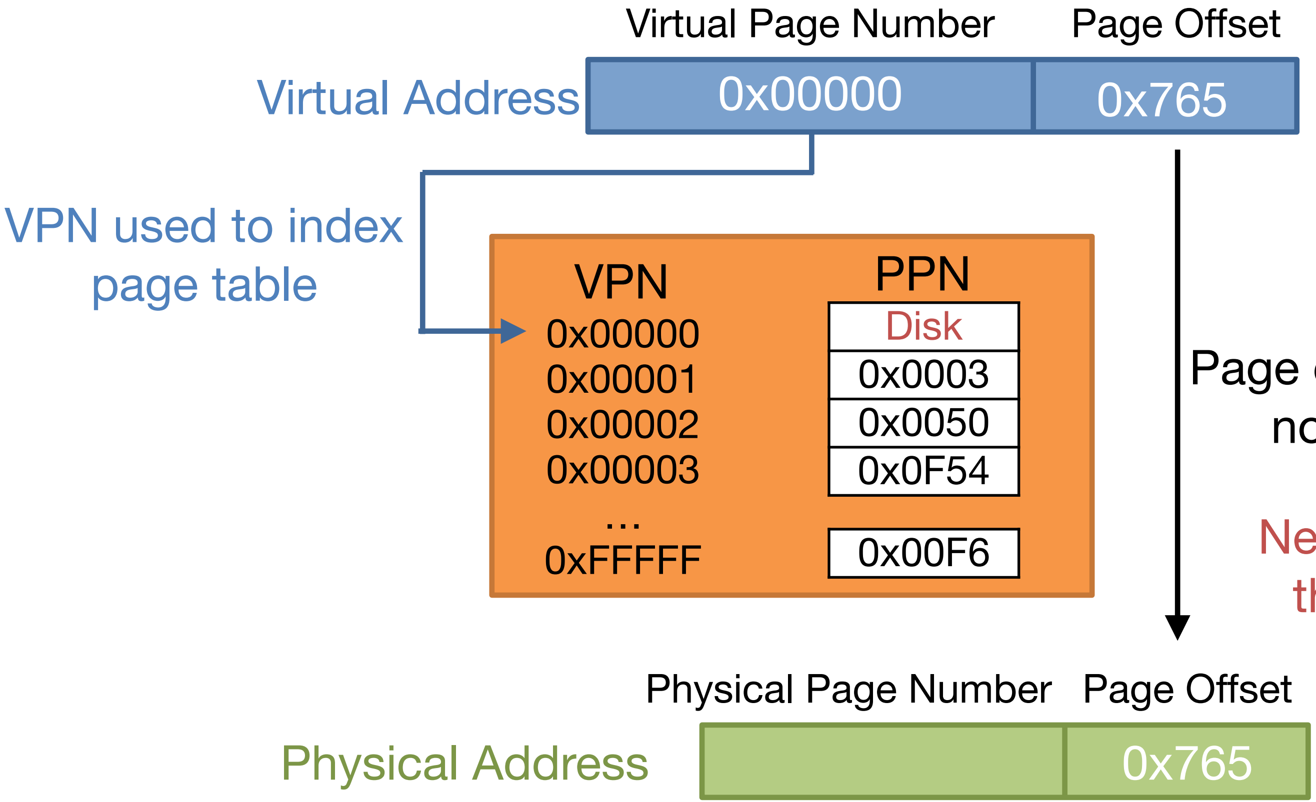
Virtual Address: Virtual Page Number (20 bits) | Page Offset (12 bits)

0x00000765

VPN	PPN
0x00000	Disk
0x00001	0x0003
0x00002	0x0050
0x00003	0x0F54
…	
0xFFFFF	0x00F6

Physical Address: Physical Page Number (16 bits) | Page Offset (12 bits)

Example Translation #2

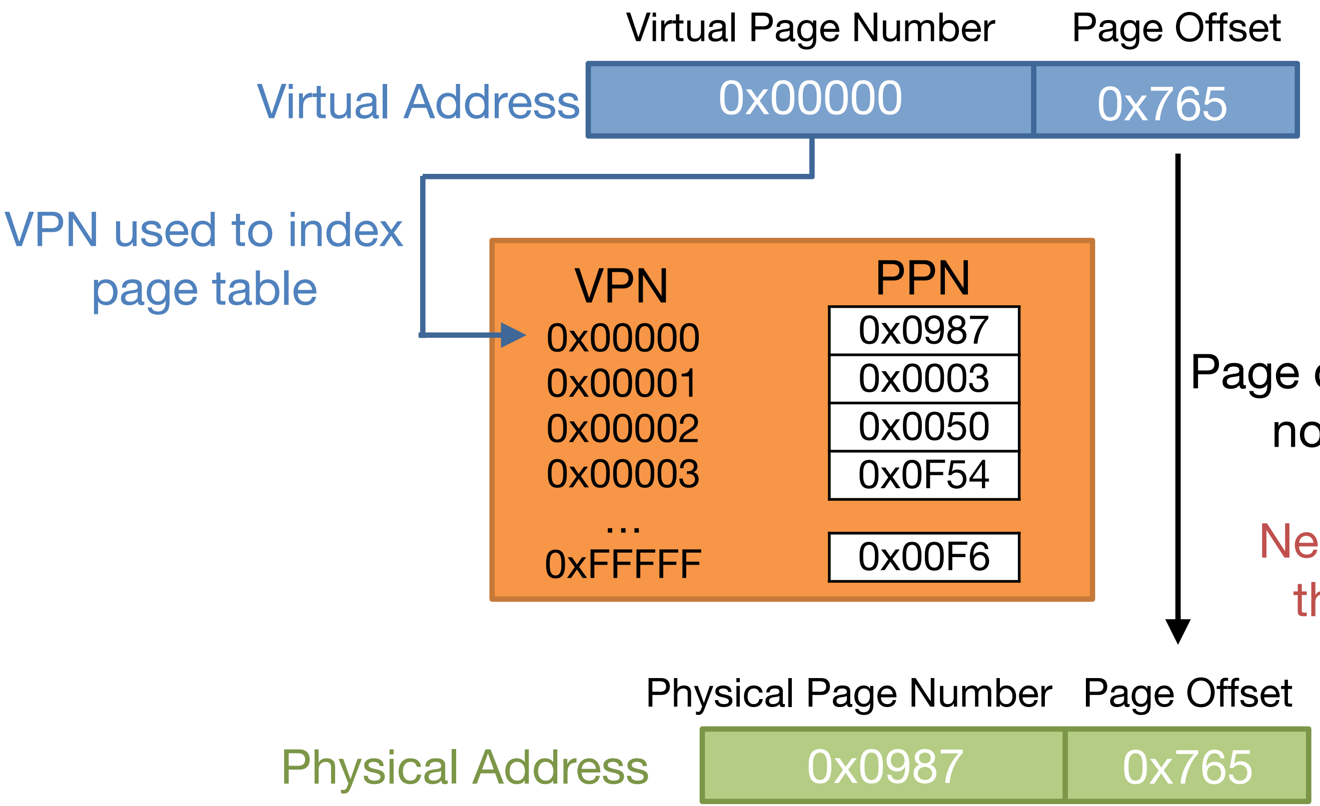


0x00000765

Page offset bits do not change

Need to go to disk to bring in the data and assign it to a PPN!

Example Translation #2



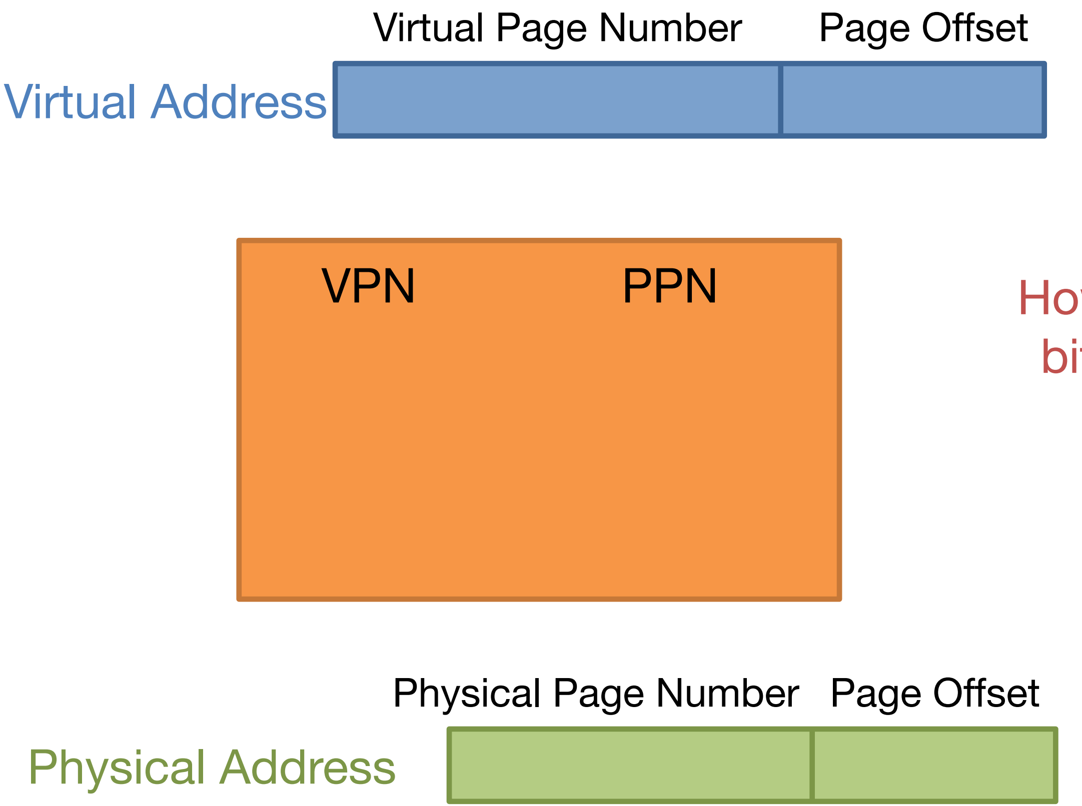
0x00000765

0x0987765

Page offset bits do not change

Need to go to disk to bring in the data and assign it to a PPN!

Increasing Page Size to 64KB



How many page offset bits would there be?

Increasing Page Size to 64KB

Virtual Address

Virtual Page Number	Page Offset
16 bits	16 bits

VPN	PPN
0x0000	0x987
0x0001	0x003
0x0002	0x050
0x0003	0xF54
…	
0xFFFF	0x0F6

How many page offset bits would there be?

$\log_2(64KB) = \log_2(2^6 * 2^{10}) = 16$

Physical Address

Physical Page Number	Page Offset
12 bits	16 bits

Page Faults

What happens when a page is not in RAM?

- Q: How do I know if my page is not in RAM?

What happens when a page is not in RAM?

- Q: How do I know if my page is not in RAM?
  - The page table entry points to the disk

What happens when a page is not in RAM?

1. Hardware (CPU) generates a **page fault exception**
2. The hardware jumps to the OS page fault handler to fix it
3. The OS chooses a page to evict from RAM (if no more free space)
4. If the page is **dirty**, it needs to be written back to disk first
5. The OS updates the corresponding PTE to point to disk
6. The OS brings in the page we wanted disk and puts it in RAM
7. The OS updates the PTE of the new page
8. The OS jumps back to the instruction that caused the page fault (This time it won't cause a page fault since the page has been loaded.)

How long does handling a page fault take?

- A really really really long time
- One of the slowest things that could happen
- This is why having more RAM will make your computer faster

Page Replacement Policies

First in, First out (FIFO)

- Evict the oldest page in the page table
- Only need to update the table when we access a page that is not in RAM

Page access pattern:	0	1	2	0	3	0	1	2	4	
Time	0	1	2	3	4	5	6	7	8	
Space for 4 pages in RAM	0	1	2	2	3	3	3	3	4	Last in (newest)
		0	1	1	2	2	2	2	3	
			0	0	1	1	1	1	2	
					0	0	0	0	1	First in (oldest)

Least Recently Used (LRU)

- Evict the page that has not been used for the longest time
- Update the table on **every** access

Page access pattern:	0	1	2	0	3	0	1	2	4
Time	0	1	2	3	4	5	6	7	8
MRU	**0**	**1**	**2**	**0**	**3**	**0**	**1**	**2**	**4**
Space for 4 pages in RAM		0	1	2	0	3	0	1	2
			0	1	2	2	3	0	1
LRU					1	1	2	3	0

Random

- Selects a page at random to replace

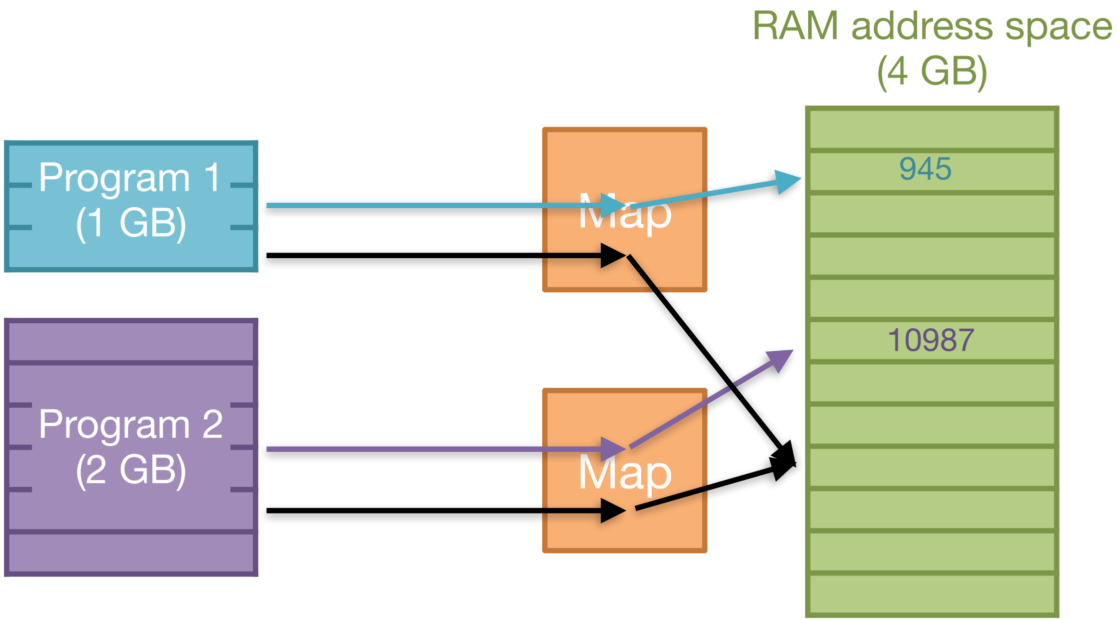
Which is better?

- LRU tends to be better, but there are exceptions
- Also, LRU is expensive
  - Need to update on each access
  - Lots of metadata
  - We usually use an approximation
- Random would be better for the following access pattern if we can only store 4 pages in memory
  - 0, 1, 2, 3, 4, 0, 1, 2, 3

Memory Protection

Memory Protection

- If each process has its own Page Table, then we can map each program's virtual addresses to unique physical addresses
- Prevents programs from accessing each other's data



Memory Protection

- Q: Which page is shared between these two programs?

Program 1 Page Table

VPN	PPN
0x00000	0x0040
0x00001	0x0003
0x00002	0x0050
0x00003	0x0F54
…	
0xFFFFF	0x00F6

Program 2 Page Table

VPN	PPN
0x00000	0x0010
0x00001	0x0050
0x00002	0x042A
0x00003	0x0000
…	
0xFFFFF	0x6743

Memory Protection

- Q: Which page is shared between these two programs?

Program 1 Page Table

VPN	PPN
0x00000	0x0040
0x00001	0x0003
0x00002	0x0050
0x00003	0x0F54
…	
0xFFFFF	0x00F6

Program 2 Page Table

VPN	PPN
0x00000	0x0010
0x00001	0x0050
0x00002	0x042A
0x00003	0x0000
…	
0xFFFFF	0x6743