Virtual Memory

EEL 3713C: Digital Computer Architecture

Quincy Flint

[Ionospheric Radio Lab in NEB]

Outline

1. Memory Problems
    - Not enough memory
    - Holes in address space
    - Programs overwriting
2. What is Virtual Memory?
    - Layer of indirection
    - How does indirection solve above
    - Page tables and translation
3. How do we implement VM?
    - Create and store page tables
    - Fast address translation
4. Virtual Memory and Caches
    - Prevent cache performance degradation when using VM

Quincy Flint

Memory Problems

Memory Problems

1. Not enough memory
2. Holes in address space
3. Programs writing to same address

What if we don't have enough memory?

- MIPS gives each program a 32-bit address space

What if we don't have enough memory?

- MIPS gives each program a 32-bit address space

Q: How much memory can we access in a 32-bit address space?

- 2^{30} Bytes [1 GB]
- 2^{32} Bytes [4 GB]
- 2^{32} Words [16 GB]
- Undetermined

What if we don't have enough memory?

- MIPS gives each program a 32-bit address space

Q: How much memory can we access in a 32-bit address space?

- 2^{30} Bytes [1 GB]
- 2^{32} Bytes [4 GB]
- 2^{32} Words [16 GB]
- Undetermined

4 GB but… really much of this is reserved for the operating system

What if we don't have enough memory?

- MIPS gives each program a 32-bit address space

Q: How much memory can we access in a 32-bit address space?
- 2^{30} Bytes [1 GB]
- 2^{32} Bytes [4 GB]
- 2^{32} Words [16 GB]
- Undetermined

4 GB but… really much of this is reserved for the operating system

Q: How much memory do you have on your DE0 board?
- 8 MB
- 128 MB
- 1 GB
- 4 GB

What if we don't have enough memory?

- MIPS gives each program a 32-bit address space

Q: How much memory can we access in a 32-bit address space?

- 2^{30} Bytes [1 GB]
- 2^{32} Bytes [4 GB]
- 2^{32} Words [16 GB]
- Undetermined

4 GB but… really much of this is reserved for the operating system

Q: How much memory do you have on your DE0 board?

- 8 MB
- 128 MB
- 1 GB
- 4 GB

8MB of onboard SDRAM plus what you can program in FPGA units. These devices do not typically support Virtual Memory.

What if we don't have enough memory?

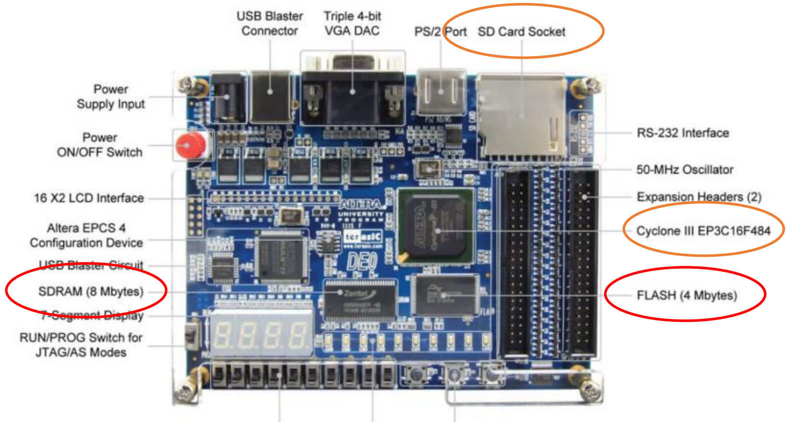
- MIPS gives each program a 32-bit address space

Q: How much memory can we access in a 32-bit address space?
- 2^{30} Bytes [1 GB]
- 2^{32} Bytes [4 GB]
- 2^{32} Words [16 GB]
- Undetermined

4 GB but… really much of this is reserved for the operating system

Q: How much memory do you have on your DE0 board?
- 8 MB
- 128 MB
- 1 GB
- 4 GB

8MB of onboard SDRAM plus what you can program in FPGA units. These devices do not typically support Virtual Memory.



What if we don't have enough memory?

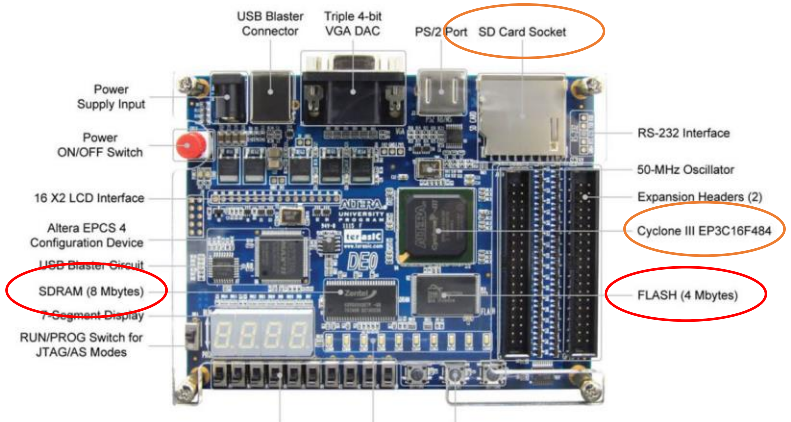
- MIPS gives each program a 32-bit address space
- What if we don't have 4 GB of memory?

Q: How much memory can we access in a 32-bit address space?
- 2^{30} Bytes [1 GB]
- 2^{32} Bytes [4 GB]
- 2^{32} Words [16 GB]
- Undetermined

4 GB but… really much of this is reserved for the operating system

Q: How much memory do you have on your DE0 board?
- 8 MB
- 128 MB
- 1 GB
- 4 GB

8MB of onboard SDRAM plus what you can program in FPGA units. These devices do not typically support Virtual Memory.



What if we don't have enough memory?

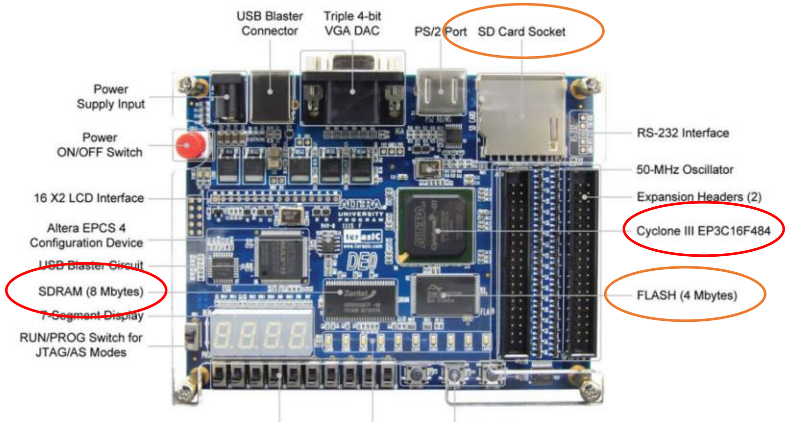
- MIPS gives each program a 32-bit address space
- What if we don't have 4 GB of memory?

4 GB [*32-bit*]
Program Address Space

0x FFFF FFFF

0x 0000 0000

1 GB [*30-bit*]
Physical Address Space



Q: How much memory can we access in a 32-bit address space?
- 2^{30} Bytes [1 GB]
- 2^{32} Bytes [4 GB]
- 2^{32} Words [16 GB]
- Undetermined

4 GB but… really much of this is reserved for the operating system

Q: How much memory do you have on your DE0 board?
- 8 MB
- 128 MB
- 1 GB
- 4 GB

8MB of onboard SDRAM plus what you can program in FPGA units. These devices do not typically support Virtual Memory.

What if we don't have enough memory?

- MIPS gives each program a 32-bit address space
- What if we don't have 4 GB of memory?

Q: How much memory can we access in a 32-bit address space?
- 2^{30} Bytes [1 GB]
- 2^{32} Bytes [4 GB]
- 2^{32} Words [16 GB]
- Undetermined

4 GB but… really much of this is reserved for the operating system

4 GB [*32-bit*]
Program Address Space

0x FFFF FFFF

Size is fixed

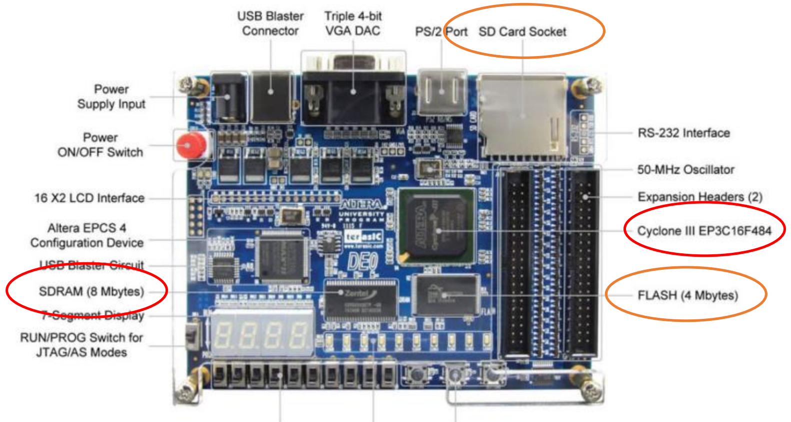
0x 0000 0000

1 GB [*30-bit*]
Physical Address Space

Size is variable

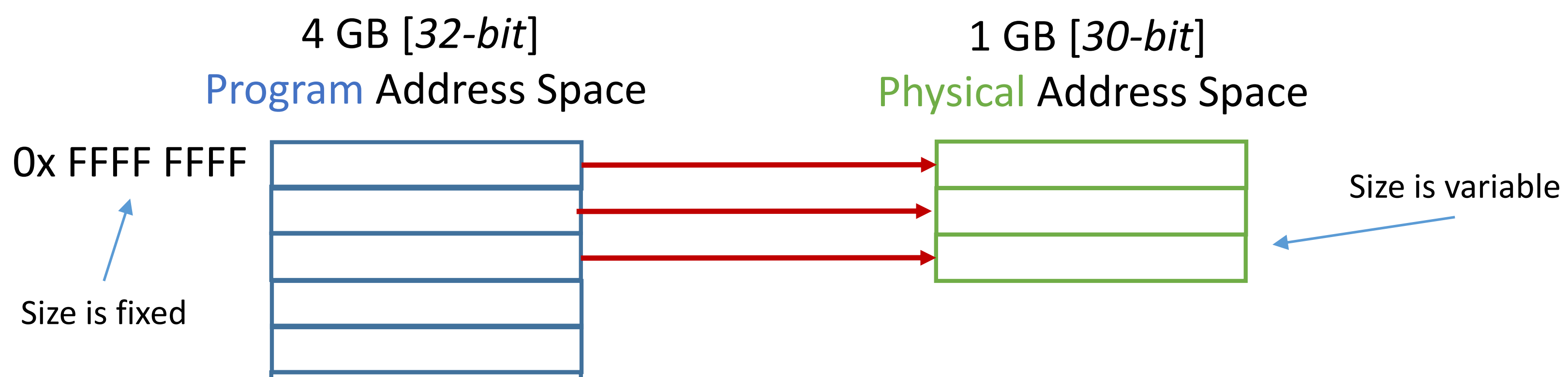
Q: How much memory do you have on your <u>DE0</u> board?
- 8 MB
- 128 MB
- 1 GB
- 4 GB

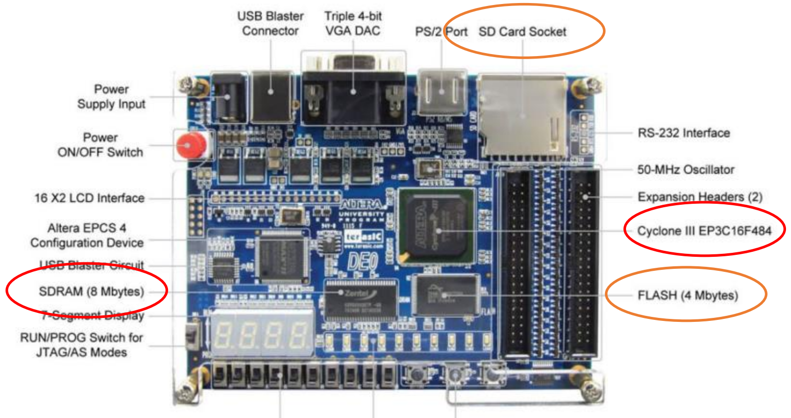
8MB of onboard SDRAM plus what you can program in FPGA units. These devices do not typically support Virtual Memory.



What if we don't have enough memory?

- MIPS gives each program a 32-bit address space
- What if we don't have 4 GB of memory?





Q: How much memory can we access in a 32-bit address space?

- 2^{30} Bytes [1 GB]
- 2^{32} Bytes [4 GB]
- 2^{32} Words [16 GB]
- Undetermined

4 GB but… really much of this is reserved for the operating system

Q: How much memory do you have on your DE0 board?

- 8 MB
- 128 MB
- 1 GB
- 4 GB

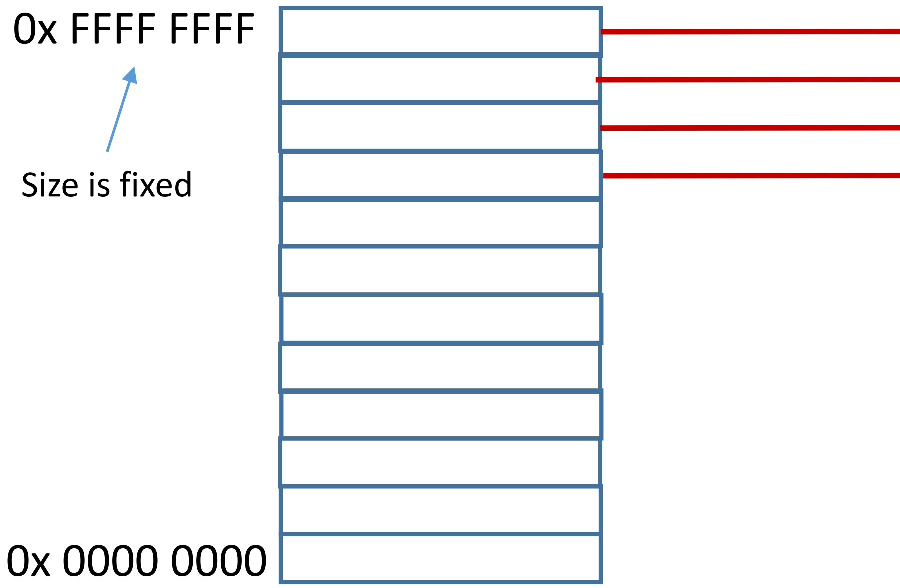
8MB of onboard SDRAM plus what you can program in FPGA units. These devices do not typically support Virtual Memory.

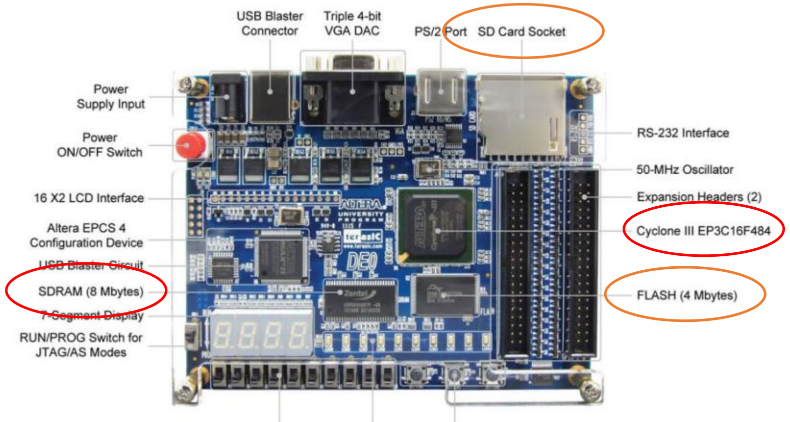
What if we don't have enough memory?

- MIPS gives each program a 32-bit address space
- What if we don't have 4 GB of memory?

4 GB [*32-bit*]
Program Address Space

1 GB [*30-bit*]
Physical Address Space





Q: How much memory can we access in a 32-bit address space?

- 2^{30} Bytes [1 GB]
- 2^{32} Bytes [4 GB]
- 2^{32} Words [16 GB]
- Undetermined

4 GB but… really much of this is reserved for the operating system

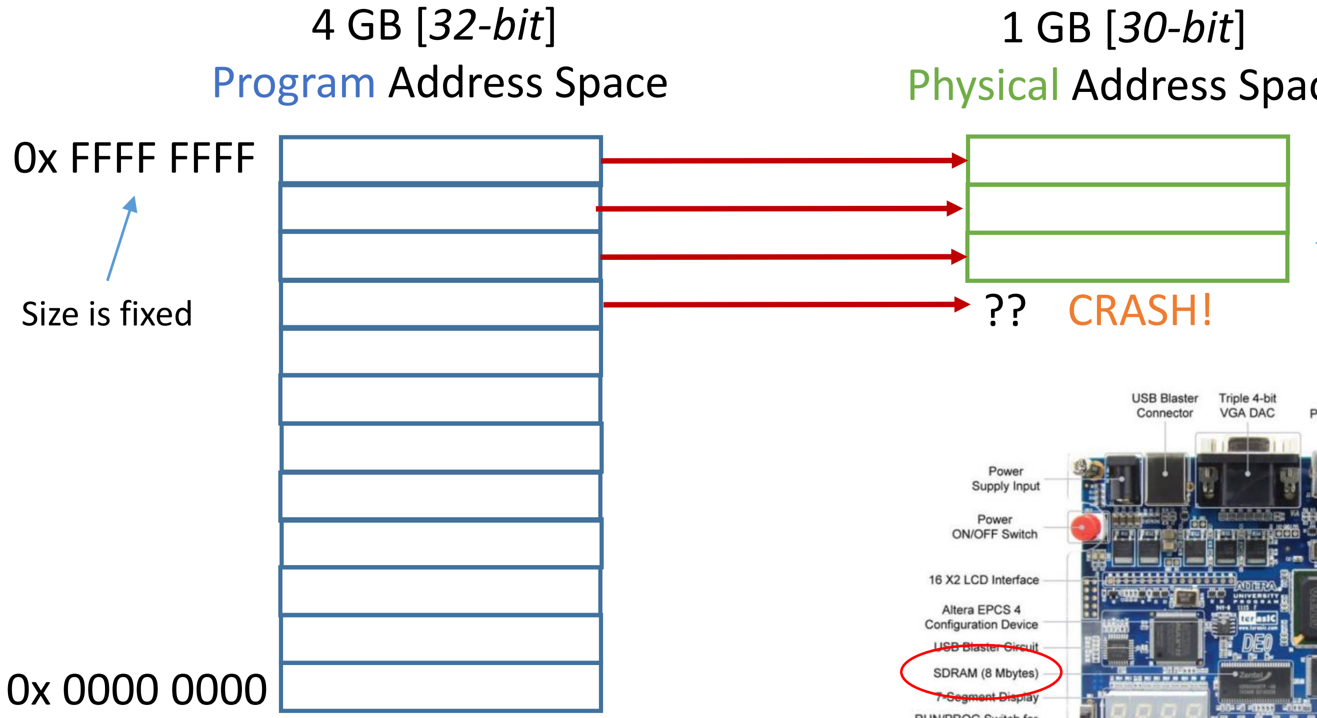
Q: How much memory do you have on your DE0 board?

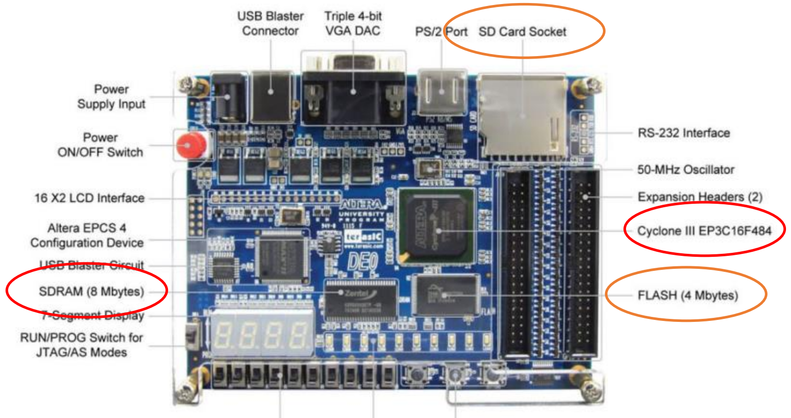
- 8 MB
- 128 MB
- 1 GB
- 4 GB

8MB of onboard SDRAM plus what you can program in FPGA units. These devices do not typically support Virtual Memory.

What if we don't have enough memory?

- MIPS gives each program a 32-bit address space
- What if we don't have 4 GB of memory?





Q: How much memory can we access in a 32-bit address space?

- 2^{30} Bytes [1 GB]
- 2^{32} Bytes [4 GB]
- 2^{32} Words [16 GB]
- Undetermined

4 GB but… really much of this is reserved for the operating system

Q: How much memory do you have on your DE0 board?

- 8 MB
- 128 MB
- 1 GB
- 4 GB

8MB of onboard SDRAM plus what you can program in FPGA units. These devices do not typically support Virtual Memory.

What if we don't have enough memory?

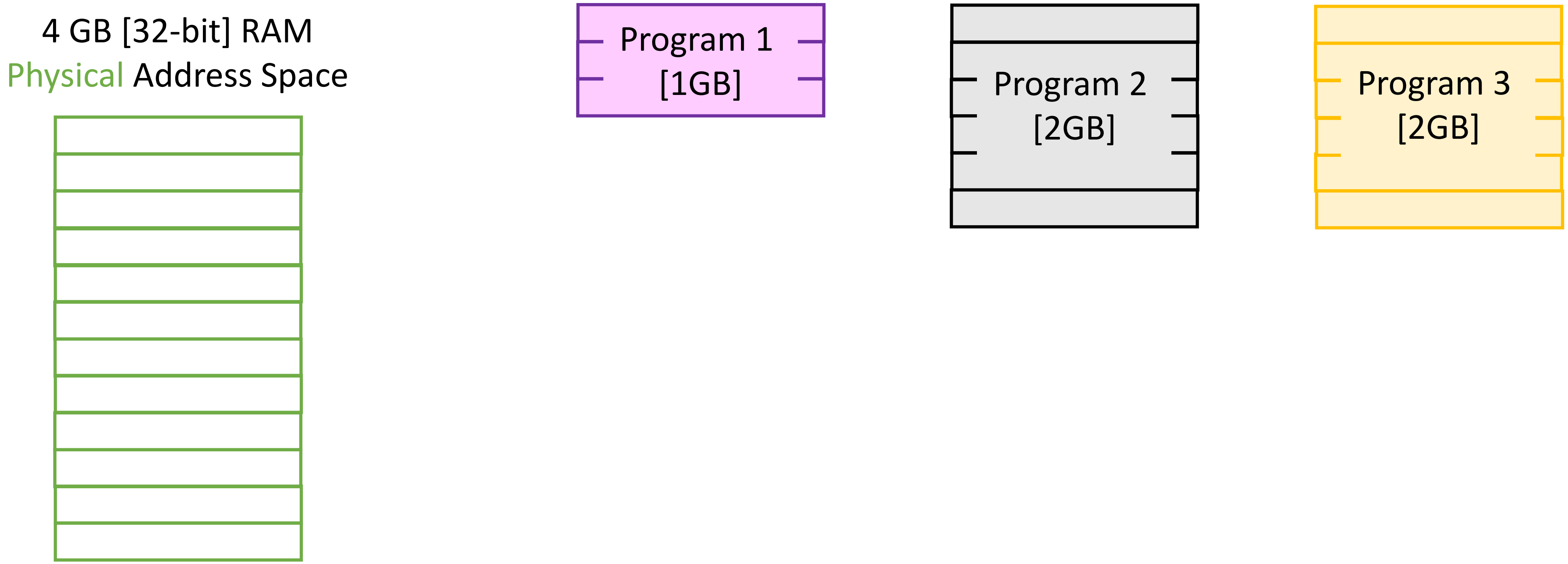
- Problem #1:
  - We promised each program a 32-bit address space, but the actual address space available depends on the amount of RAM installed.

What if we have holes in our address space?

- What happens if we are running multiple programs, then close one?
  - How do these programs share memory?

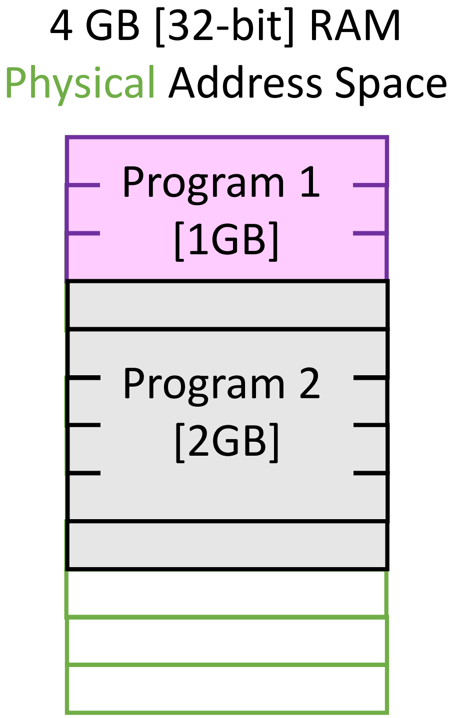
What if we have holes in our address space?

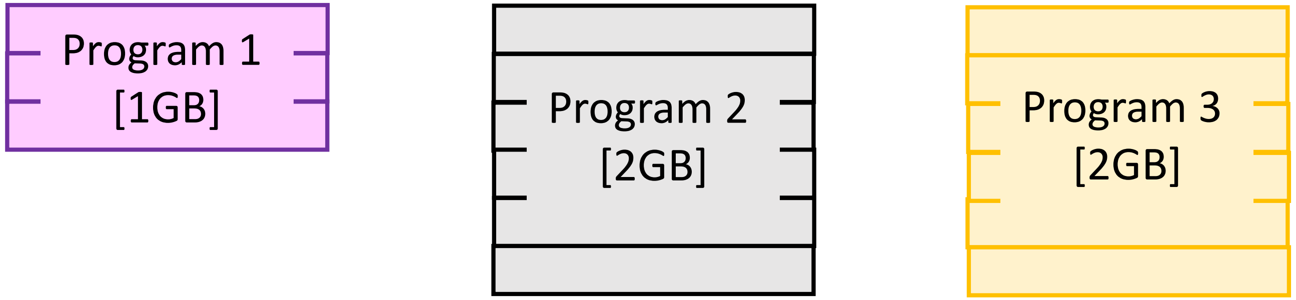
- What happens if we are running multiple programs, then close one?
  - How do these programs share memory?



What if we have holes in our address space?

- What happens if we are running multiple programs, then close one?
    - How do these programs share memory?





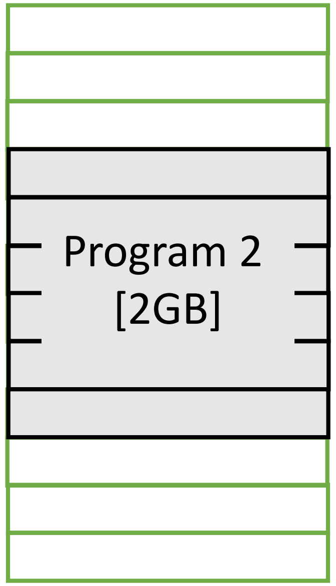
Program Sequence:

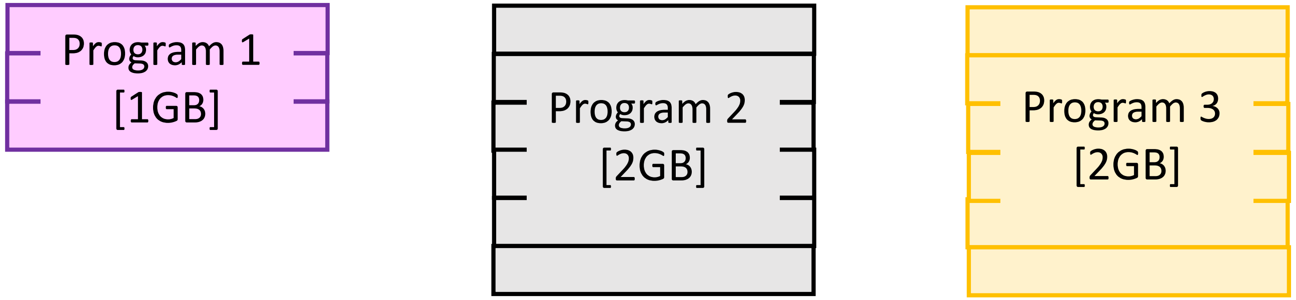
1. Run programs 1 and 2 [1 GB free]

What if we have holes in our address space?

- What happens if we are running multiple programs, then close one?
  - How do these programs share memory?

4 GB [32-bit] RAM
Physical Address Space





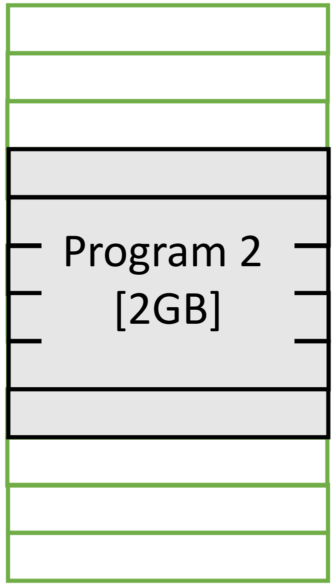
Program Sequence:

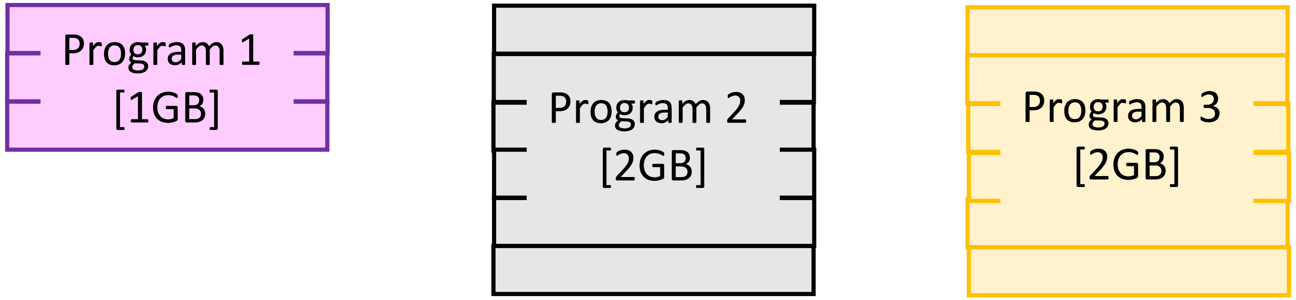
1. Run programs 1 and 2 [1 GB free]
2. Close program 1 [2 GB free]

What if we have holes in our address space?

- What happens if we are running multiple programs, then close one?
  - How do these programs share memory?

4 GB [32-bit] RAM
Physical Address Space





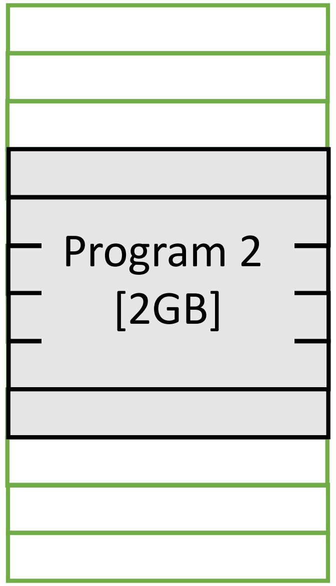
Program Sequence:

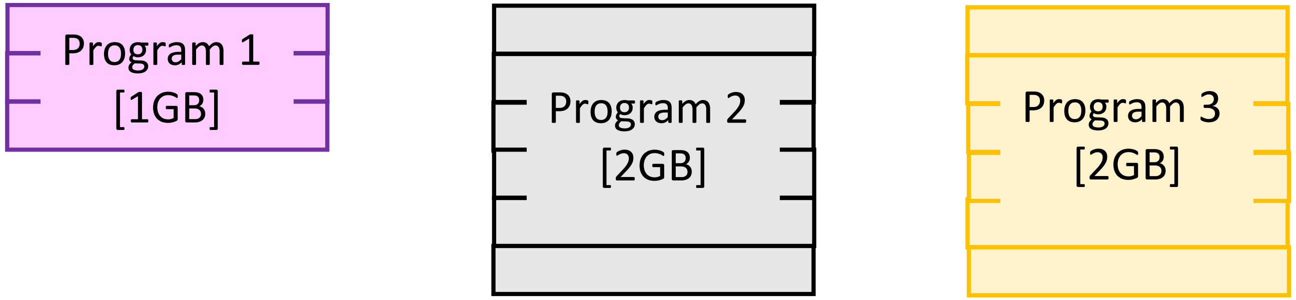
1. Run programs 1 and 2 [1 GB free]
2. Close program 1 [2 GB free]
3. Run program 3 [CANNOT!]

What if we have holes in our address space?

- What happens if we are running multiple programs, then close one?
  - How do these programs share memory?

4 GB [32-bit] RAM
Physical Address Space





Program Sequence:

1. Run programs 1 and 2 [1 GB free]
2. Close program 1 [2 GB free]
3. Run program 3 [CANNOT!]

Memory Fragmentation

What if we have holes in our address space?

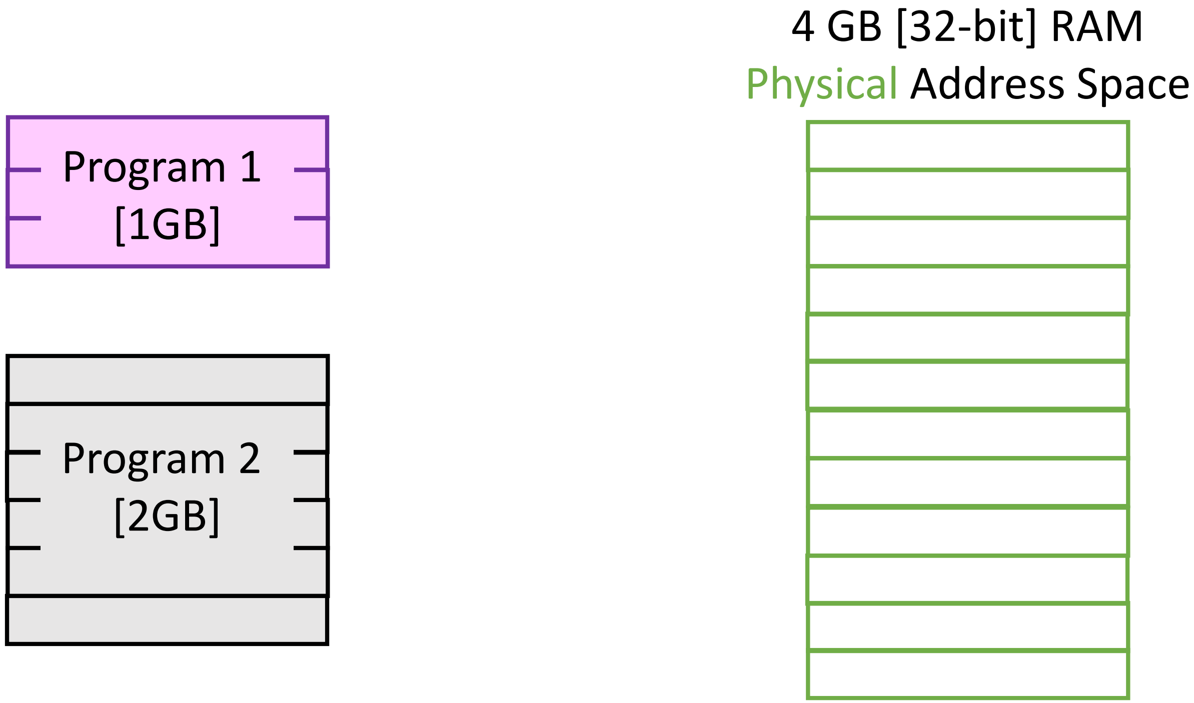
- Problem #2:
  - As applications execute and are terminated, non-sequential holes in the address space are left vacant [fragmented memory].

How do we keep programs isolated?

- What happens if multiple programs reference the same address?

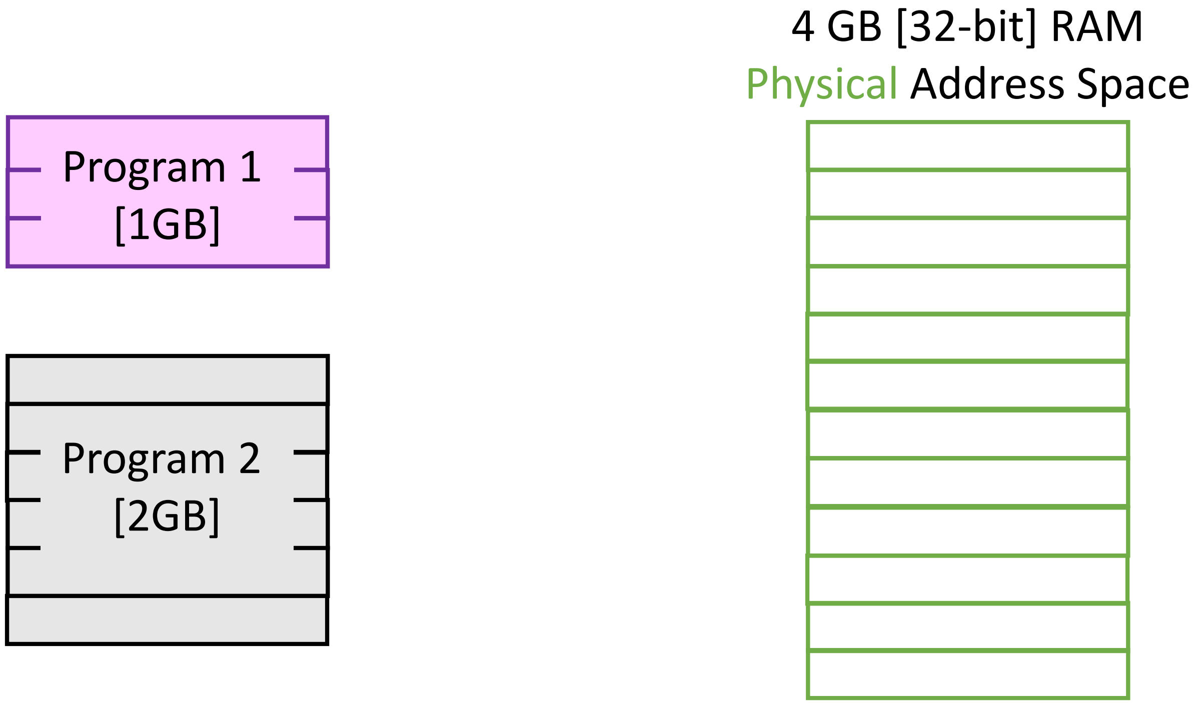
How do we keep programs isolated?

- What happens if multiple programs reference the same address?



How do we keep programs isolated?

- What happens if multiple programs reference the same address?

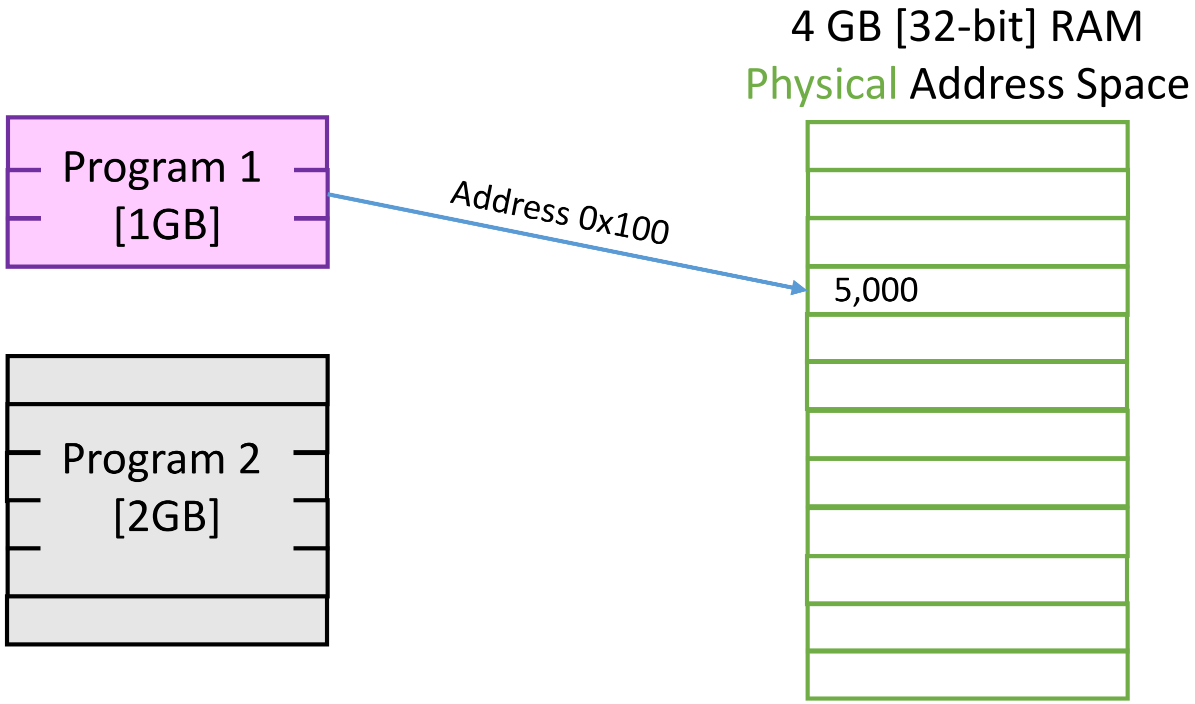


Code Segment:

```
P1: LW R2, 0x100(R0)
P2: LW R4, 0x100(R0)
```

How do we keep programs isolated?

- What happens if multiple programs reference the same address?

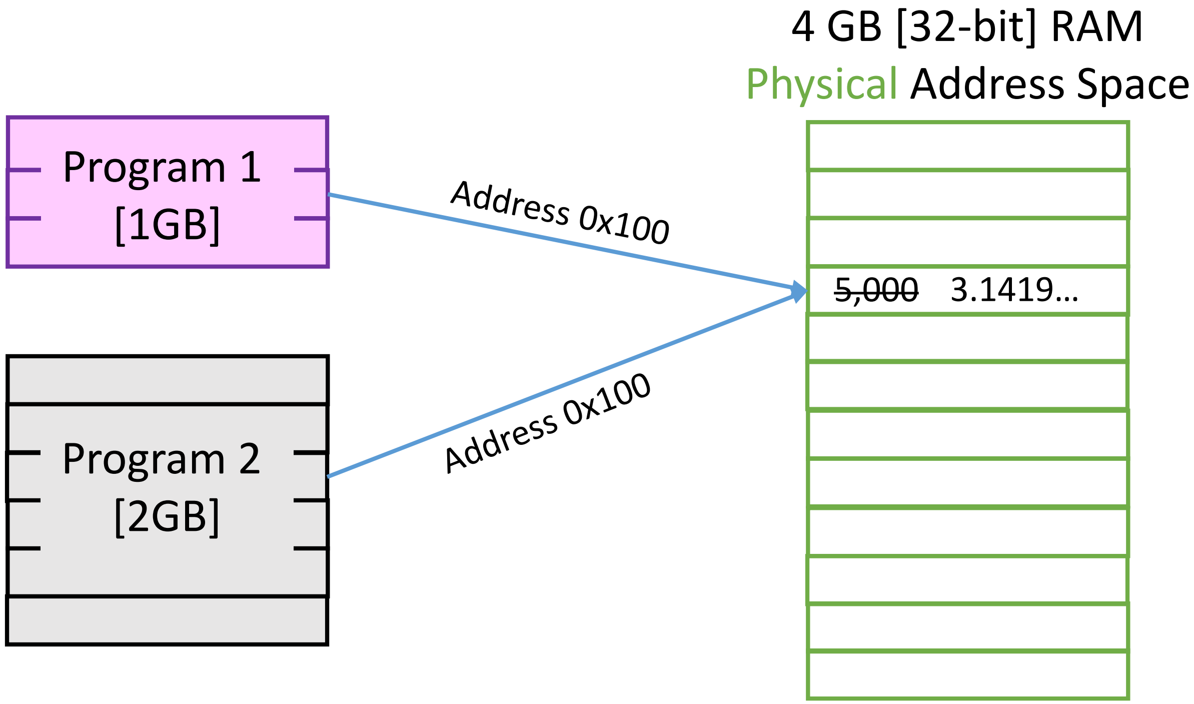


Code Segment:

```
P1: LW R2, 0x100(R0)
P2: LW R4, 0x100(R0)
```

How do we keep programs isolated?

- What happens if multiple programs reference the same address?

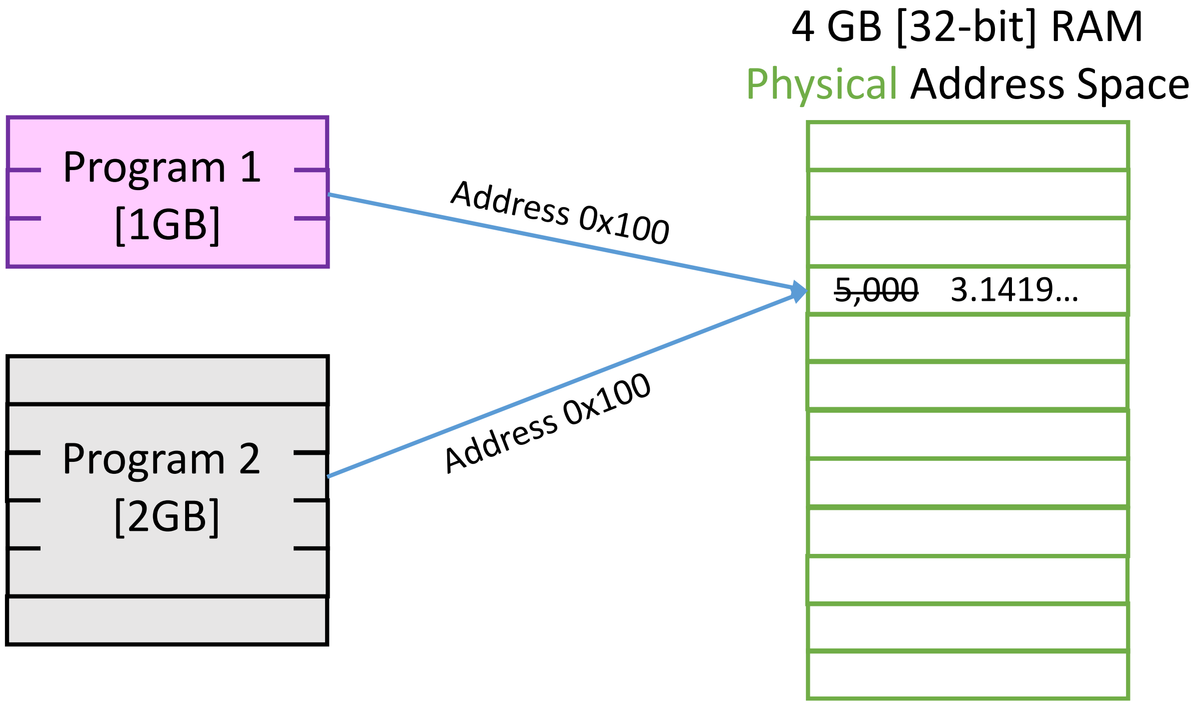


Code Segment:

```
P1: LW R2, 0x100(R0)
P2: LW R4, 0x100(R0)
```

How do we keep programs isolated?

- What happens if multiple programs reference the same address?



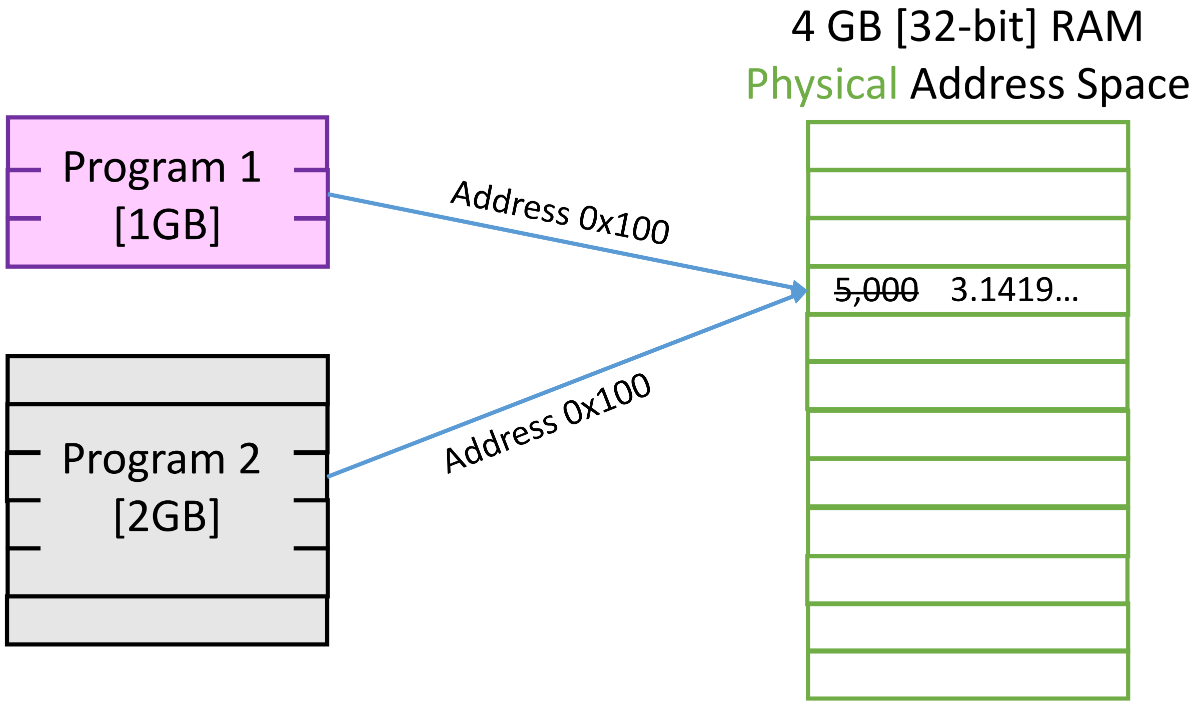
Code Segment:

```
P1: LW R2, 0x100(R0)
P2: LW R4, 0x100(R0)
```

Program 1: stores bank account balance
Program 2: stores pi

How do we keep programs isolated?

- What happens if multiple programs reference the same address?



Code Segment:

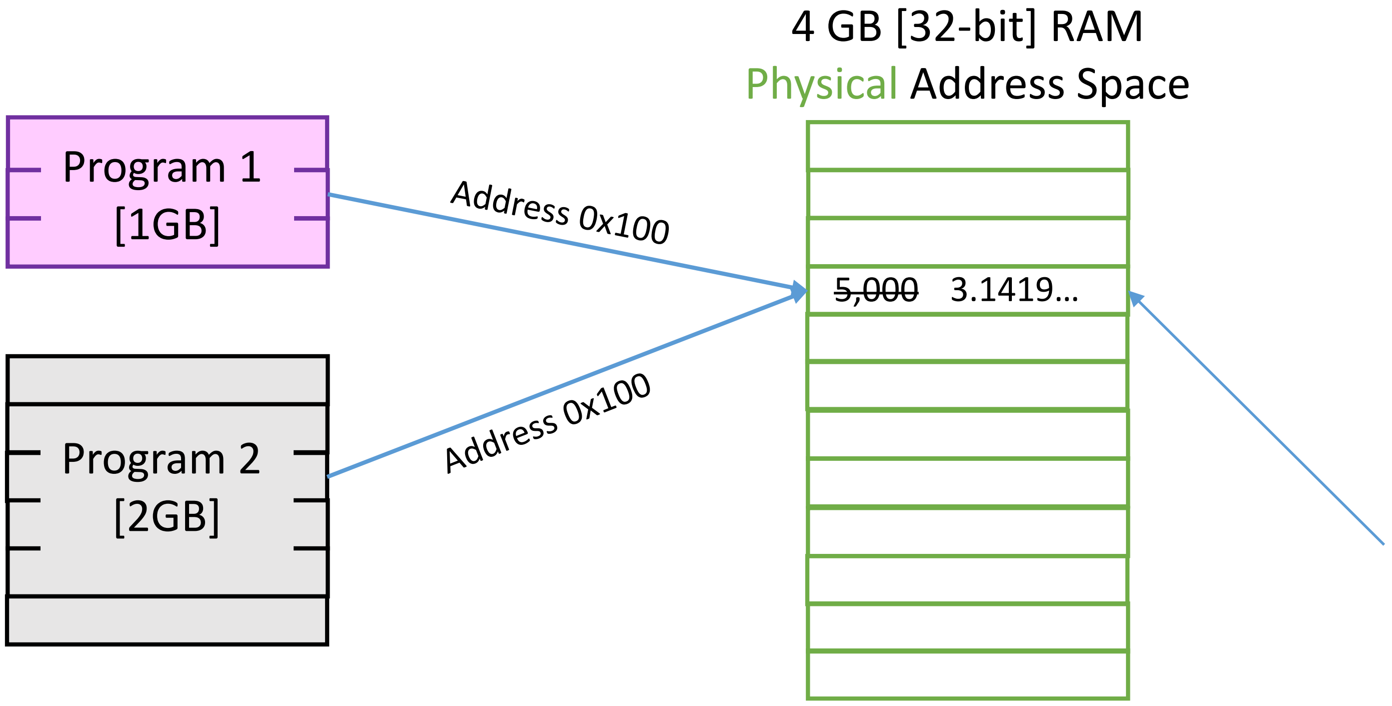
```
P1: LW R2, 0x100(R0)
P2: LW R4, 0x100(R0)
```

Program 1: stores bank account balance
Program 2: stores pi

This is legal…

How do we keep programs isolated?

- What happens if multiple programs reference the same address?



Code Segment:

```
P1: LW R2, 0x100(R0)
P2: LW R4, 0x100(R0)
```

Program 1: stores bank account balance
Program 2: stores pi

This is legal…

Data Corruption!

How do we keep programs isolated?

- Problem #3:
  - Programs with read/write access to the same memory space can over-write data from another process, causing data corruption.

Memory Problems: Out o

- If all programs can access the same memory space:
  - Will crash if we have less than 4 GB of RAM installed
  - Can run out of space if we run multiple applications
  - Can corrupt data on overwrite

Memory Problems: Out o

- If all programs can access the same memory space:
  - Will crash if we have less than 4 GB of RAM installed
  - Can run out of space if we run multiple applications
  - Can corrupt data on overwrite

- Solution:

Memory Problems: Out o

- If all programs can access the same memory space:
  - Will crash if we have less than 4 GB of RAM installed
  - Can run out of space if we run multiple applications
  - Can corrupt data on overwrite
- Solution:
  - Isolate memory spaces – assign “virtual memory space”

Memory Problems: Out o

- If all programs can access the same memory space:
  - Will crash if we have less than 4 GB of RAM installed
  - Can run out of space if we run multiple applications
  - Can corrupt data on overwrite

- Solution:
  - Isolate memory spaces – assign “virtual memory space”
  - Layer of indirection – map program memory space to physical resources

Memory Problems: Out o

Q: Which of the following is NOT a problem if programs share a 32-bit address space and we have less than 4GB of data available?

- Reading some addresses will cause a crash
- Cannot address all of memory due to 16-bit MIPS immediates
- Programs can over-write data
- Programs may not fit in memory

Memory Problems: Out o

Q: Which of the following is NOT a problem if programs share a 32-bit address space and we have less than 4GB of data available?

- Reading some addresses will cause a crash
- Cannot address all of memory due to 16-bit MIPS immediates
- Programs can over-write data
- Programs may not fit in memory

A: Cannot address all of memory due to 16-bit MIPS immediates

We can reach full address by using 16-bit immediates to create a 32-bit immediate. Do a load then shift.

Quincy Flint

Virtual Memory Intro

What is Virtual Memory?

What is Virtual Memory?

"We can solve any problem (in computer science) by introducing an extra level of indirection."

– A. Koenig

What is Virtual Memory?

"We can solve any problem (in computer science) by introducing an extra level of indirection."

– A. Koenig

- Virtual Memory maps program addresses to RAM addresses

What is Virtual Memory?

"We can solve any problem (in computer science) by introducing an extra level of indirection."

– A. Koenig

- Virtual Memory maps program addresses to RAM addresses

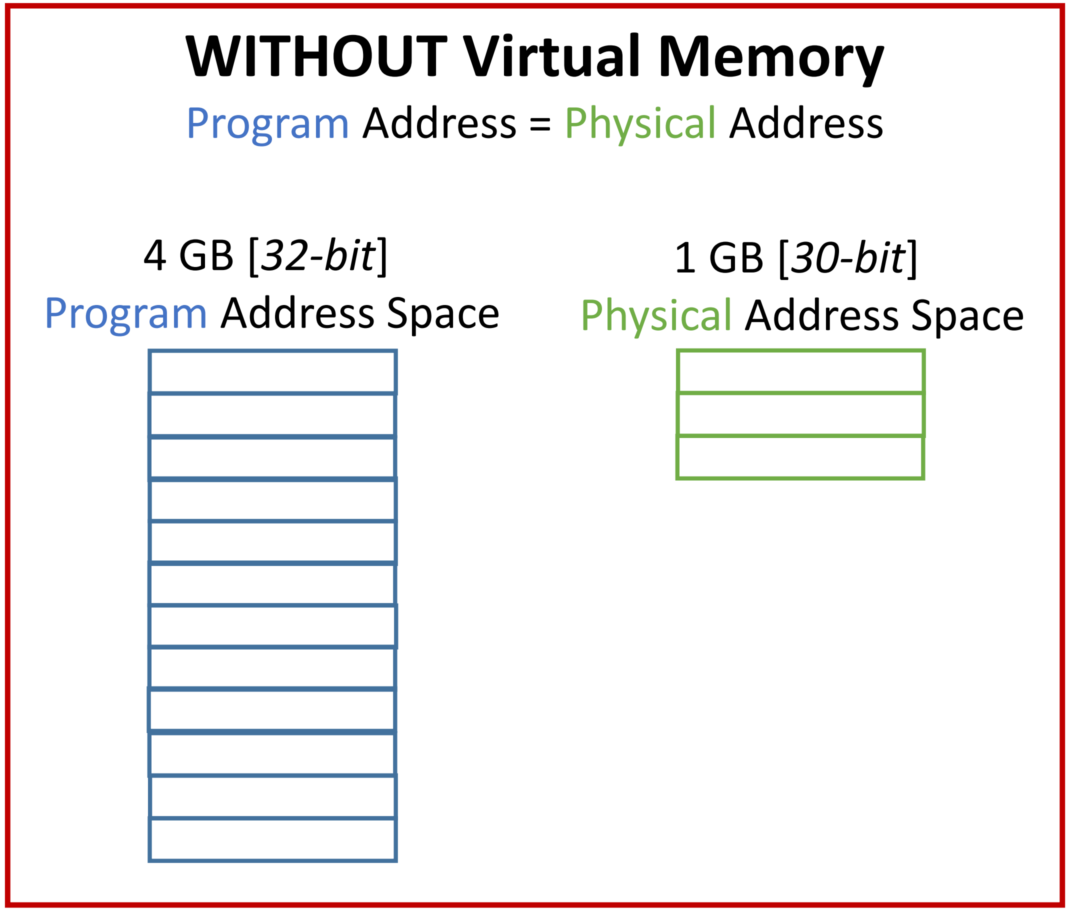
WITHOUT Virtual Memory

Program Address = Physical Address

What is Virtual Memory?

"We can solve any problem (in computer science) by introducing an extra level of indirection."
– A. Koenig

- Virtual Memory maps program addresses to RAM addresses



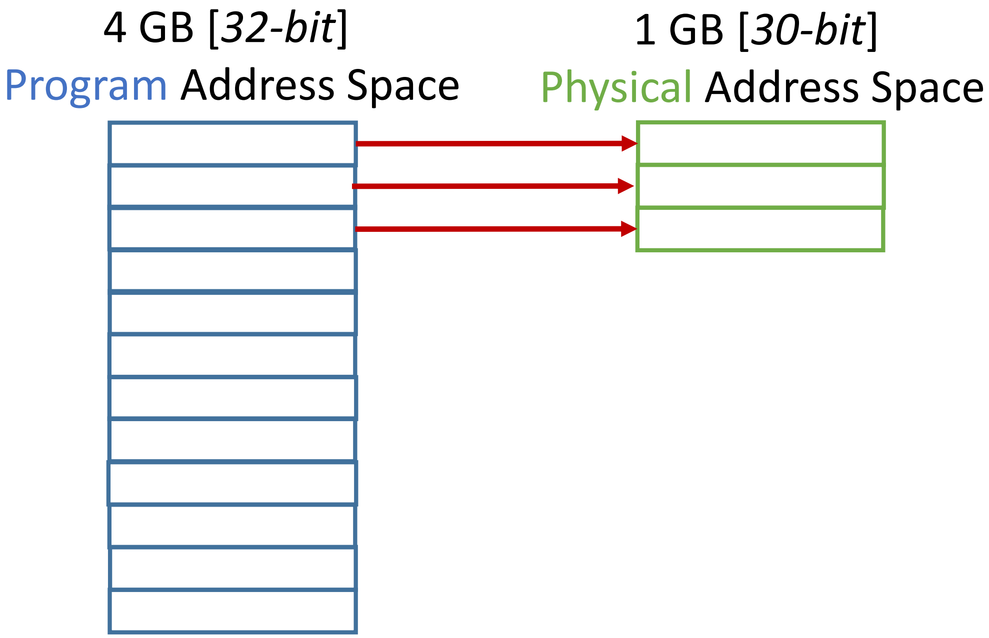
What is Virtual Memory?

"We can solve any problem (in computer science) by introducing an extra level of indirection."
– A. Koenig

- Virtual Memory maps program addresses to RAM addresses

WITHOUT Virtual Memory

Program Address = Physical Address

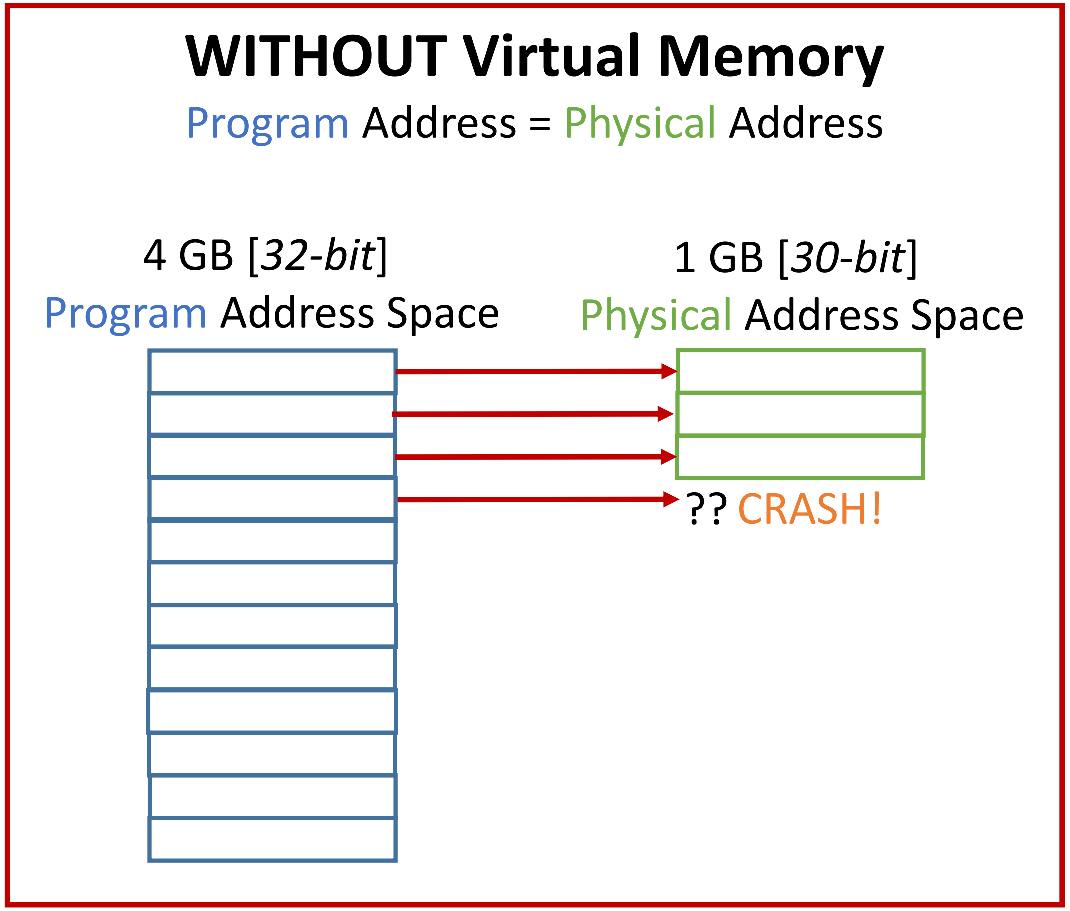


What is Virtual Memory?

"We can solve any problem (in computer science) by introducing an extra level of indirection."

– A. Koenig

- Virtual Memory maps program addresses to RAM addresses



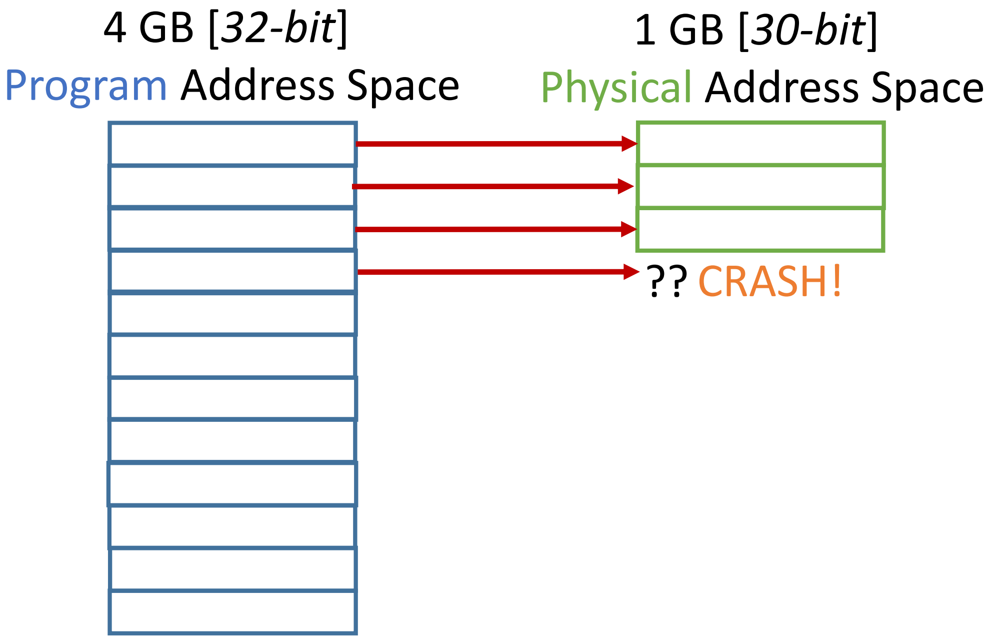
What is Virtual Memory?

"We can solve any problem (in computer science) by introducing an extra level of indirection."
– A. Koenig

- Virtual Memory maps program addresses to RAM addresses

WITHOUT Virtual Memory

Program Address = Physical Address



WITH Virtual Memory

Program Address is mapped to Physical Address

What is Virtual Memory?

"We can solve any problem (in computer science) by introducing an extra level of indirection."
– A. Koenig

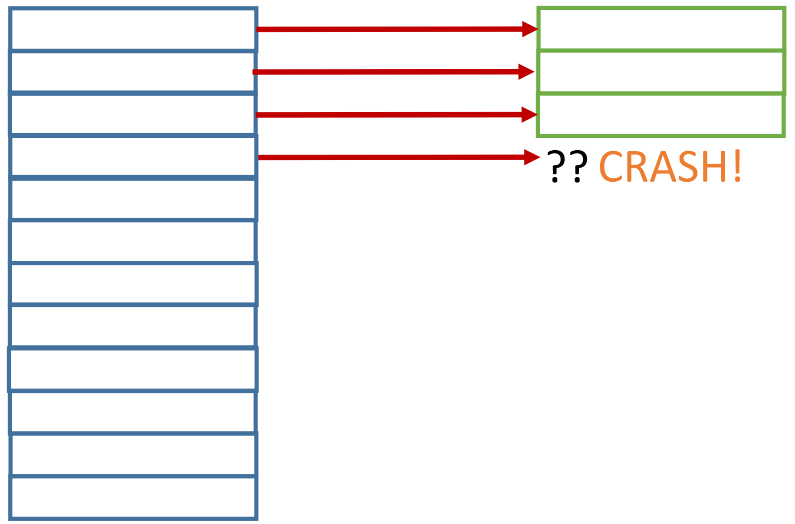
- Virtual Memory maps program addresses to RAM addresses

WITHOUT Virtual Memory

Program Address = Physical Address

4 GB [*32-bit*] Program Address Space

1 GB [*30-bit*] Physical Address Space

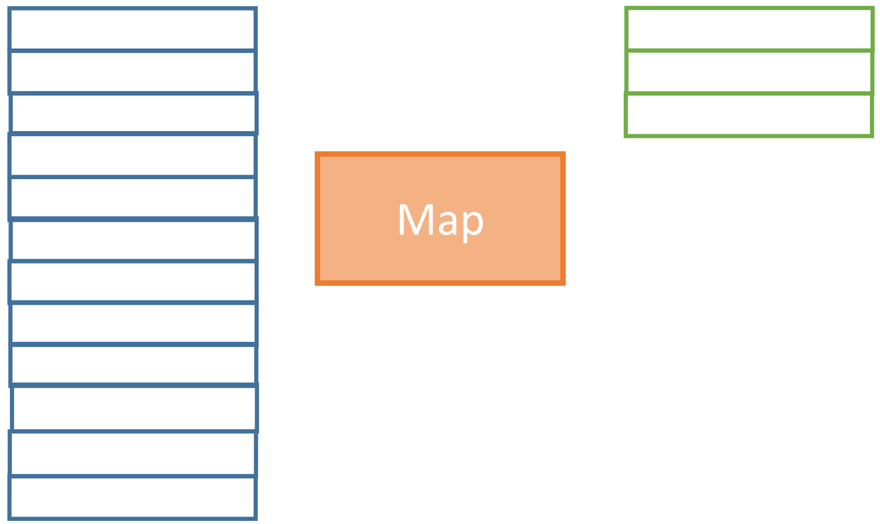


WITH Virtual Memory

Program Address is mapped to Physical Address

4 GB [*32-bit*] Program Address Space

1 GB [*30-bit*] Physical Address Space



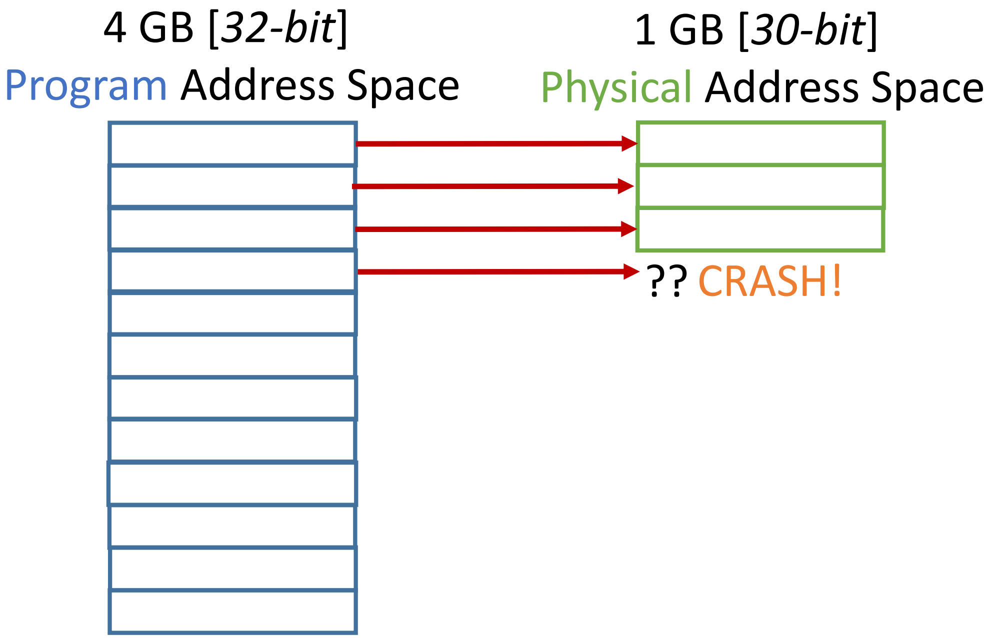
What is Virtual Memory?

"We can solve any problem (in computer science) by introducing an extra level of indirection."
– A. Koenig

- Virtual Memory maps program addresses to RAM addresses

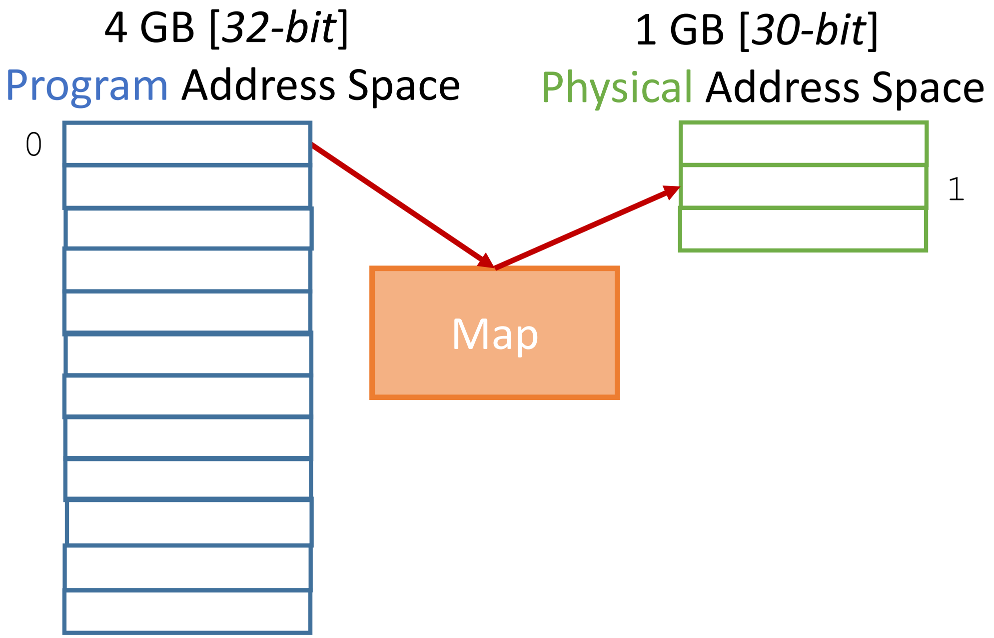
WITHOUT Virtual Memory

Program Address = Physical Address



WITH Virtual Memory

Program Address is mapped to Physical Address



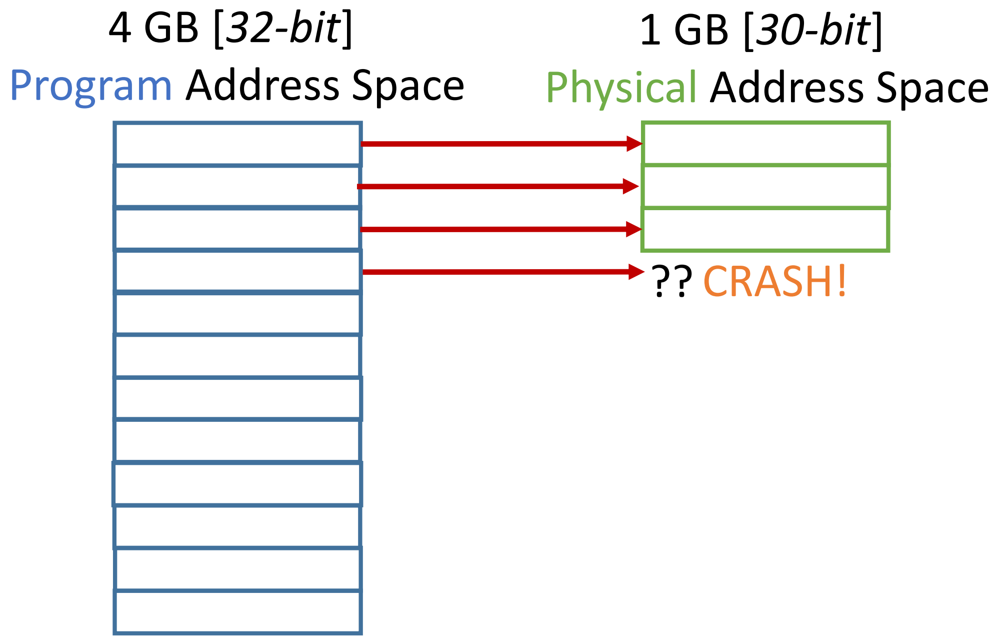
What is Virtual Memory?

"We can solve any problem (in computer science) by introducing an extra level of indirection."
– A. Koenig

- Virtual Memory maps program addresses to RAM addresses

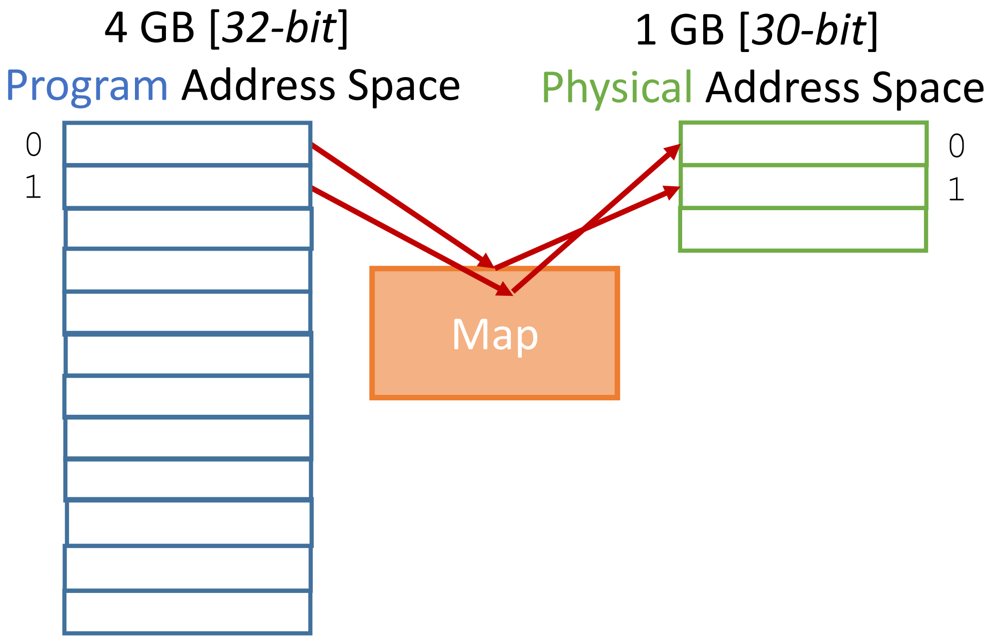
WITHOUT Virtual Memory

Program Address = Physical Address



WITH Virtual Memory

Program Address is mapped to Physical Address



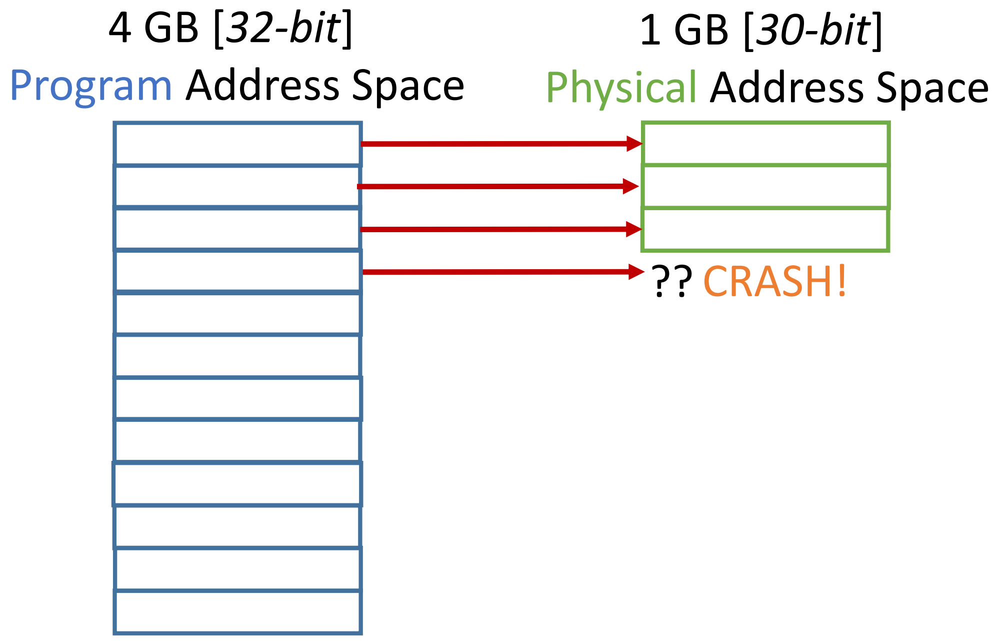
What is Virtual Memory?

"We can solve any problem (in computer science) by introducing an extra level of indirection."
– A. Koenig

- Virtual Memory maps program addresses to RAM addresses

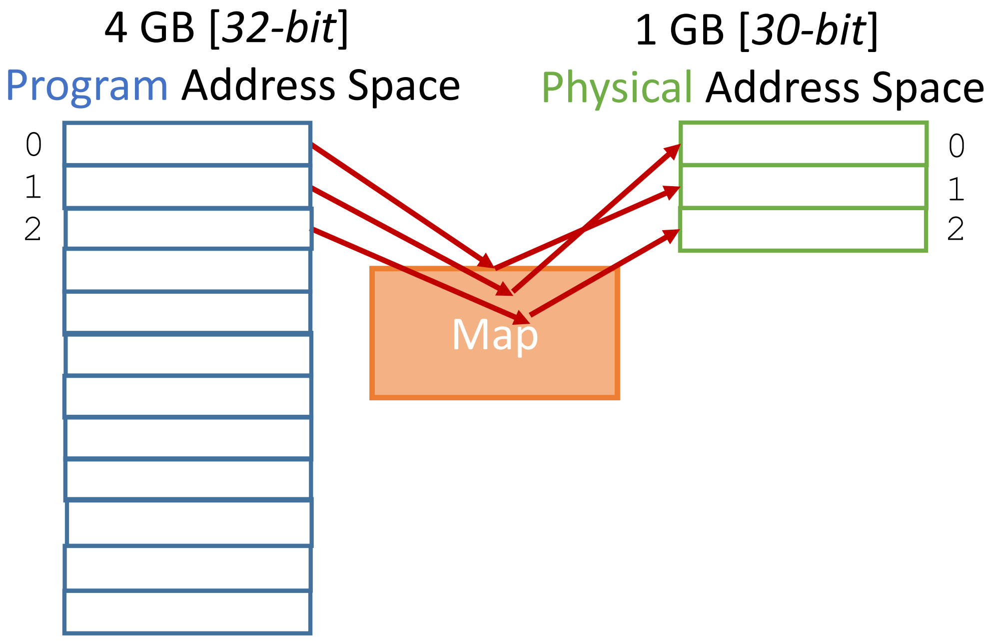
WITHOUT Virtual Memory

Program Address = Physical Address



WITH Virtual Memory

Program Address is mapped to Physical Address



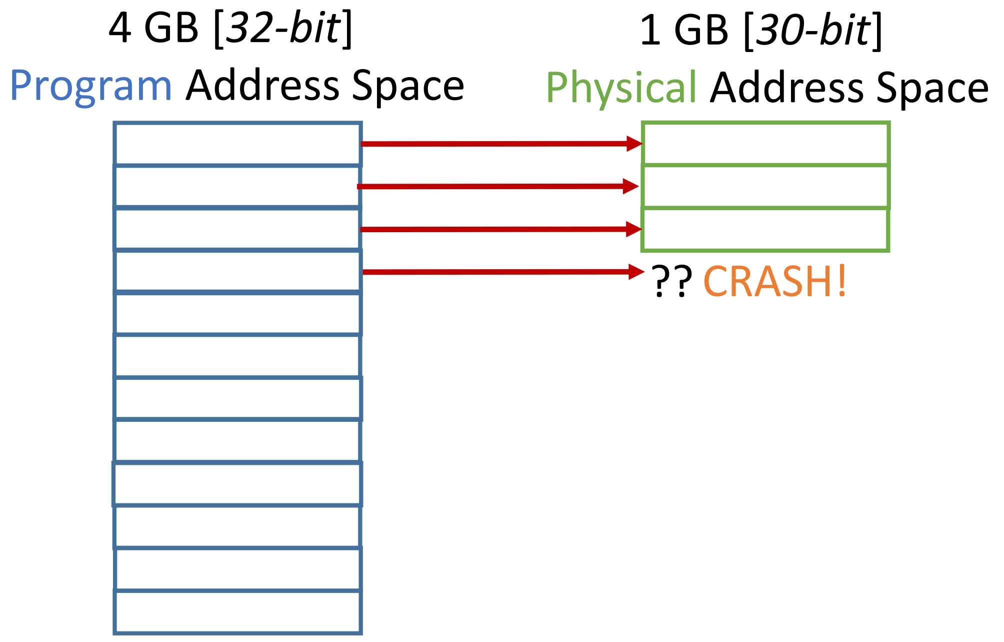
What is Virtual Memory?

"We can solve any problem (in computer science) by introducing an extra level of indirection."
– A. Koenig

- Virtual Memory maps program addresses to RAM addresses

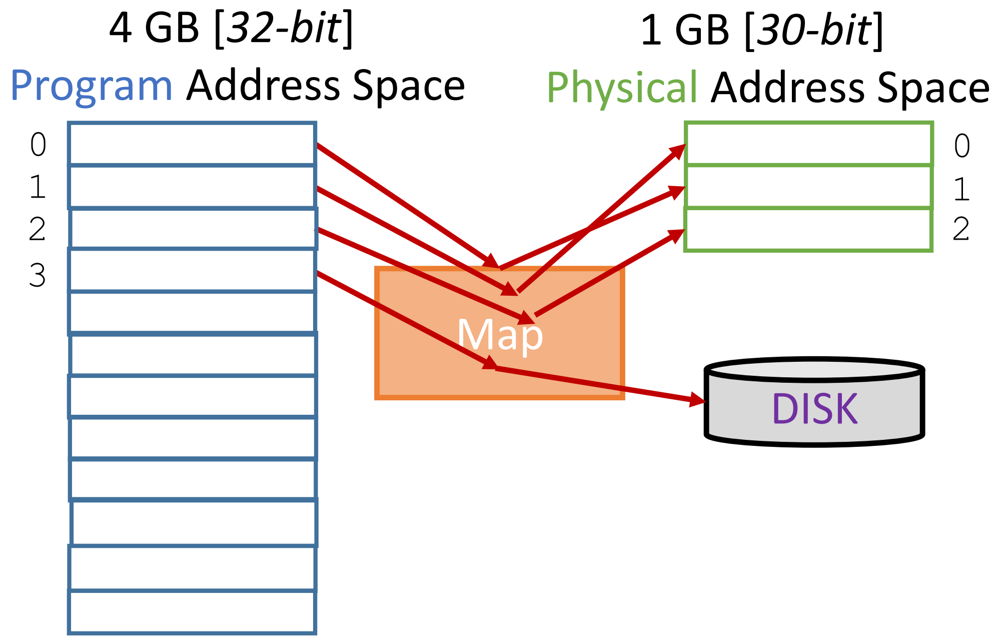
WITHOUT Virtual Memory

Program Address = Physical Address



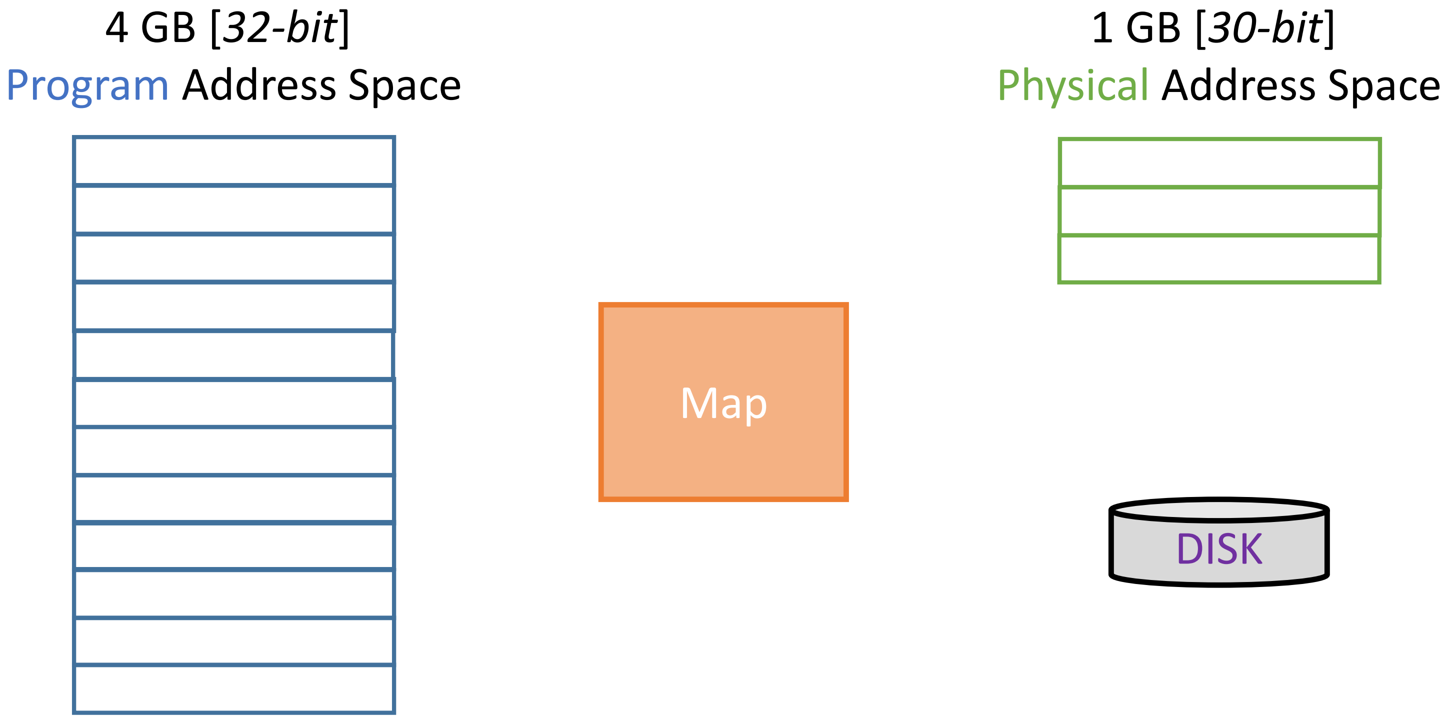
WITH Virtual Memory

Program Address is mapped to Physical Address



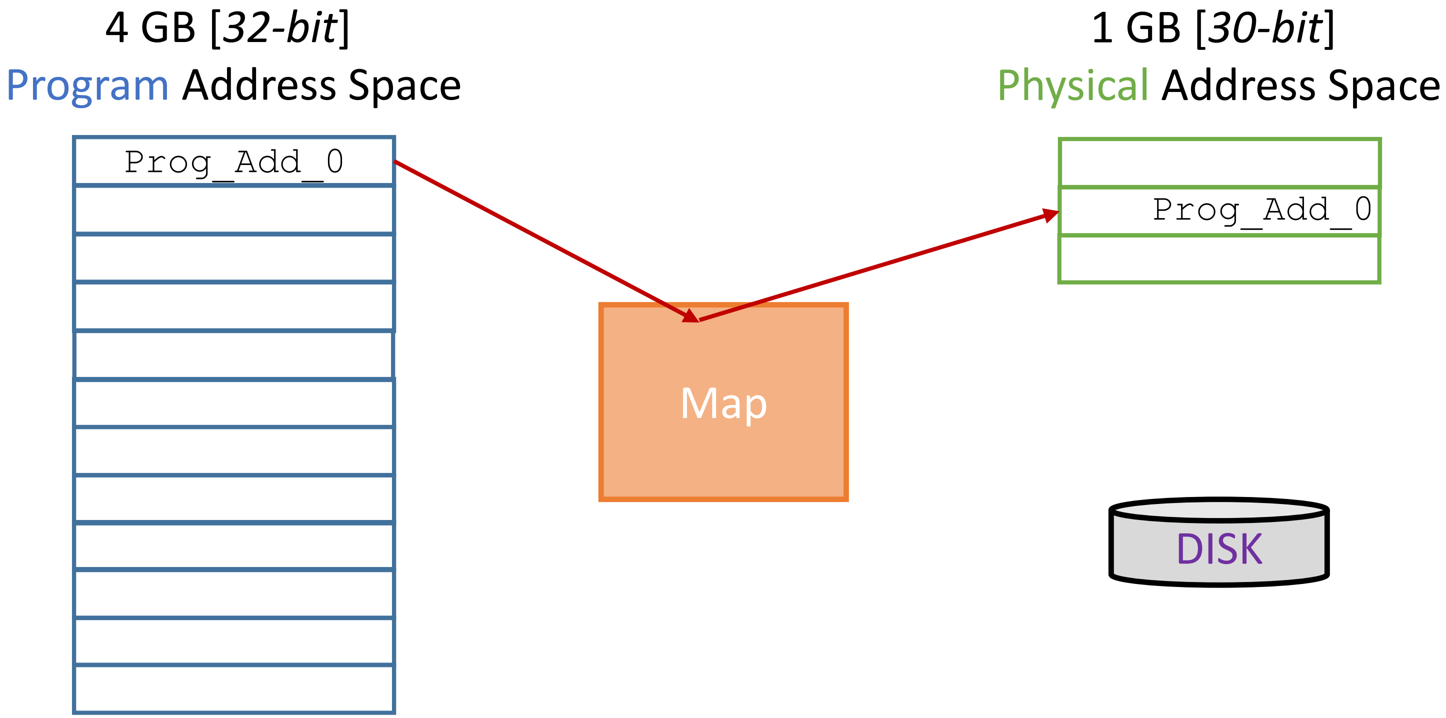
Solved: Problem #1 *(Not Enough Memory)*

- Map some program addresses to disk



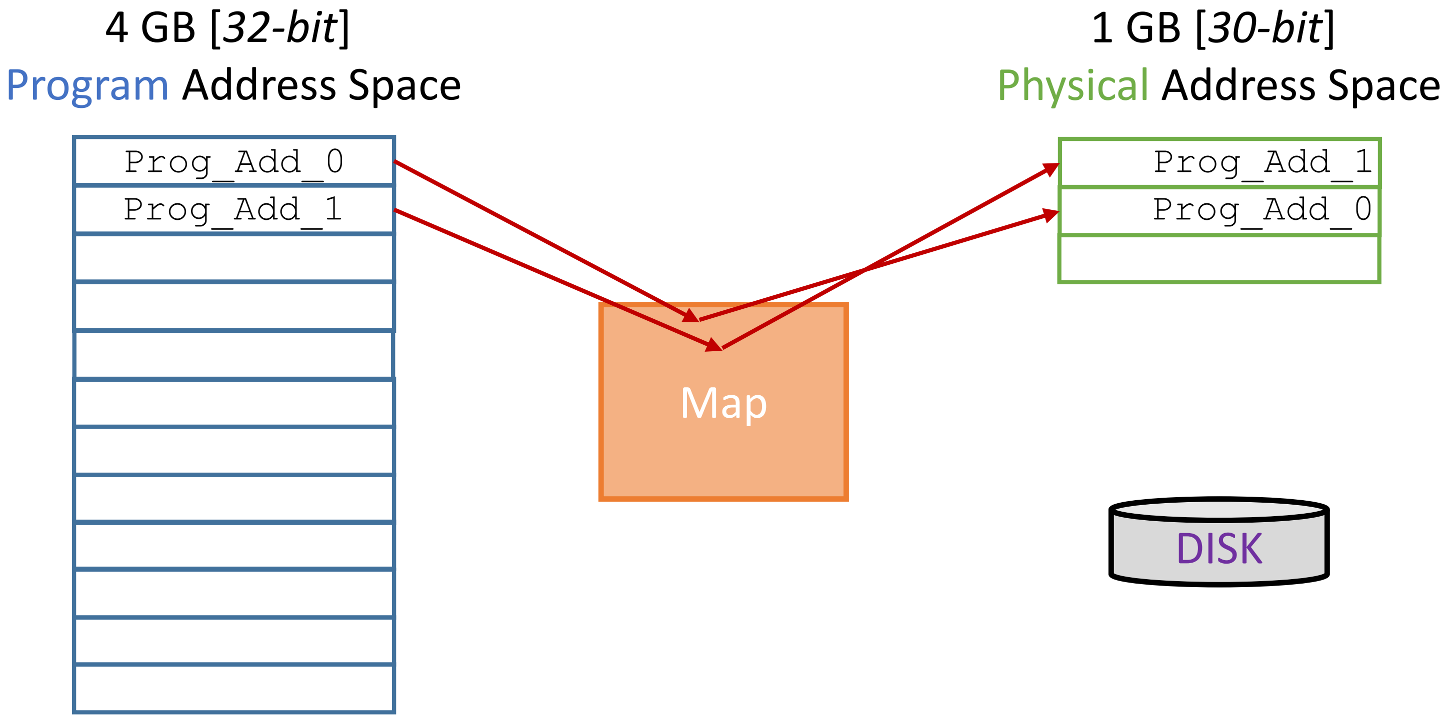
Solved: Problem #1 *(Not Enough Memory)*

- Map some program addresses to disk



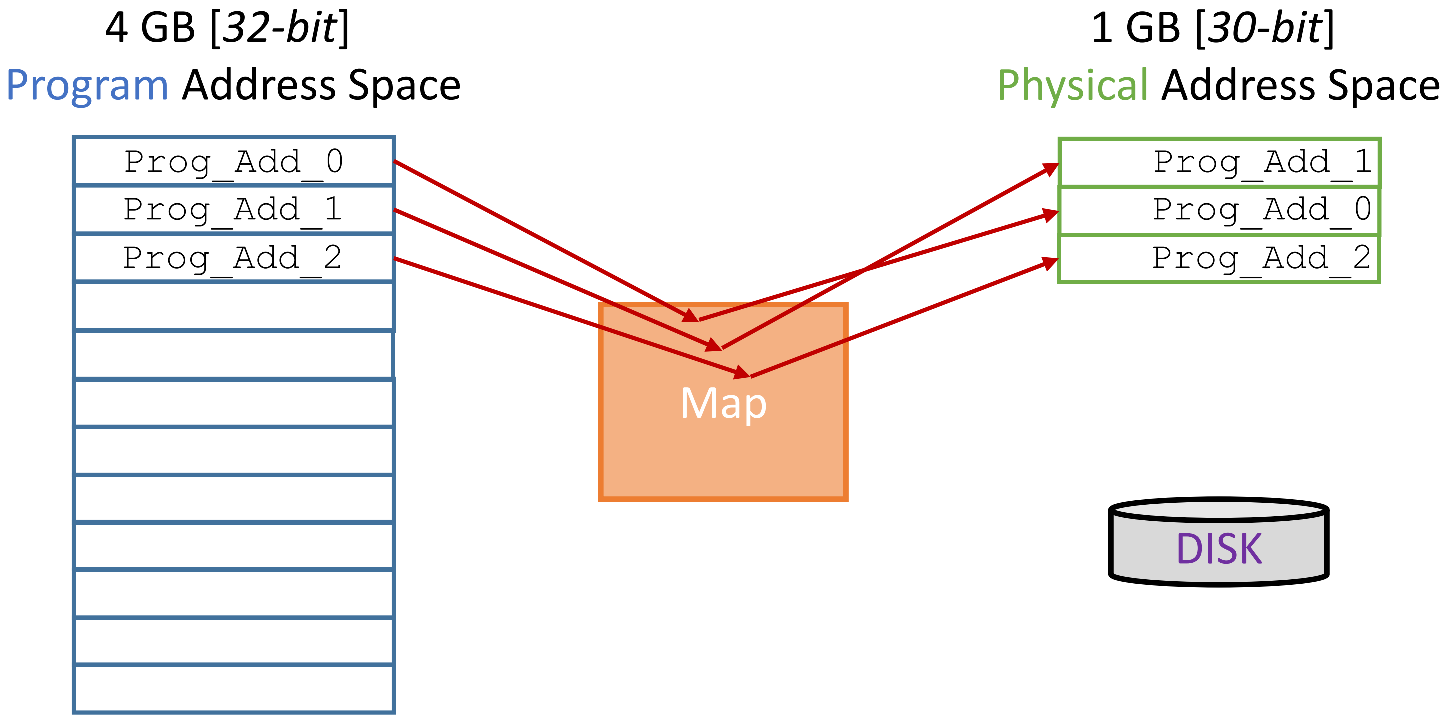
Solved: Problem #1 *(Not Enough Memory)*

- Map some program addresses to disk



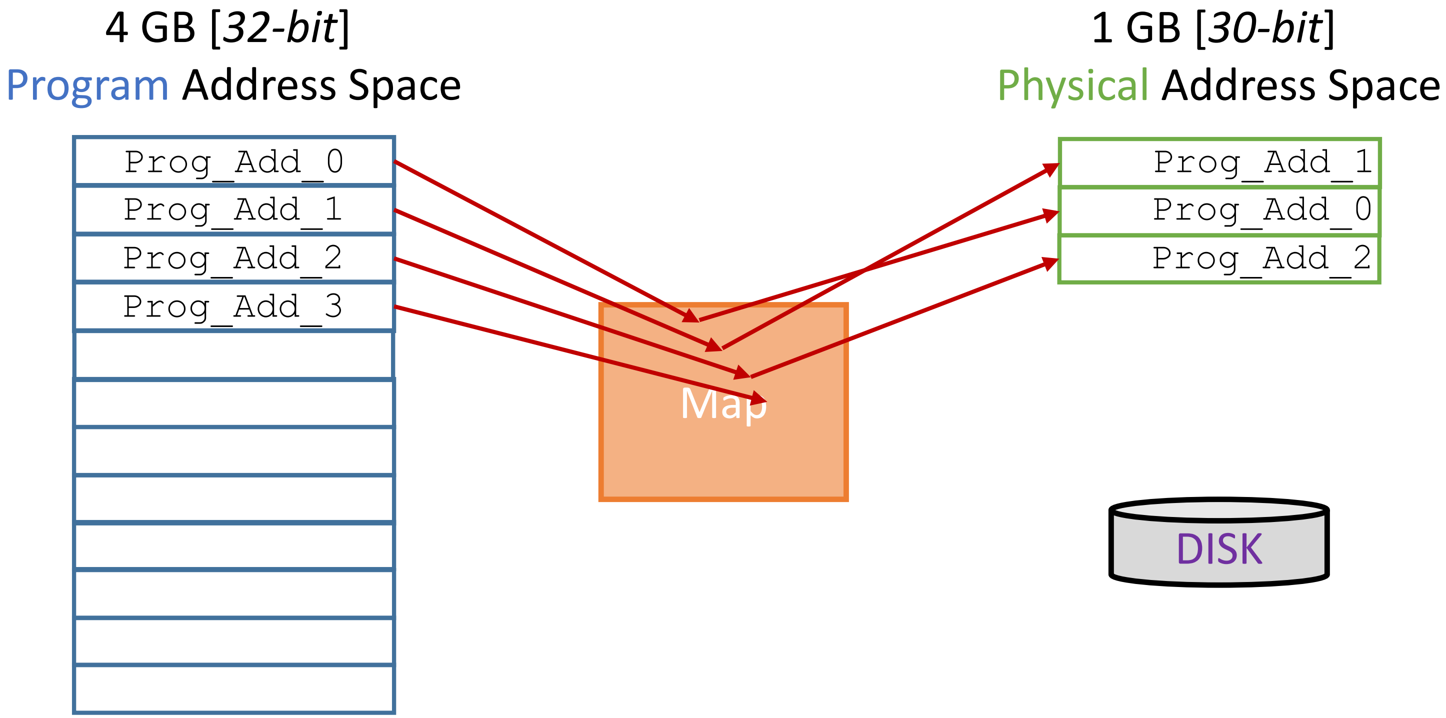
Solved: Problem #1 *(Not Enough Memory)*

- Map some program addresses to disk



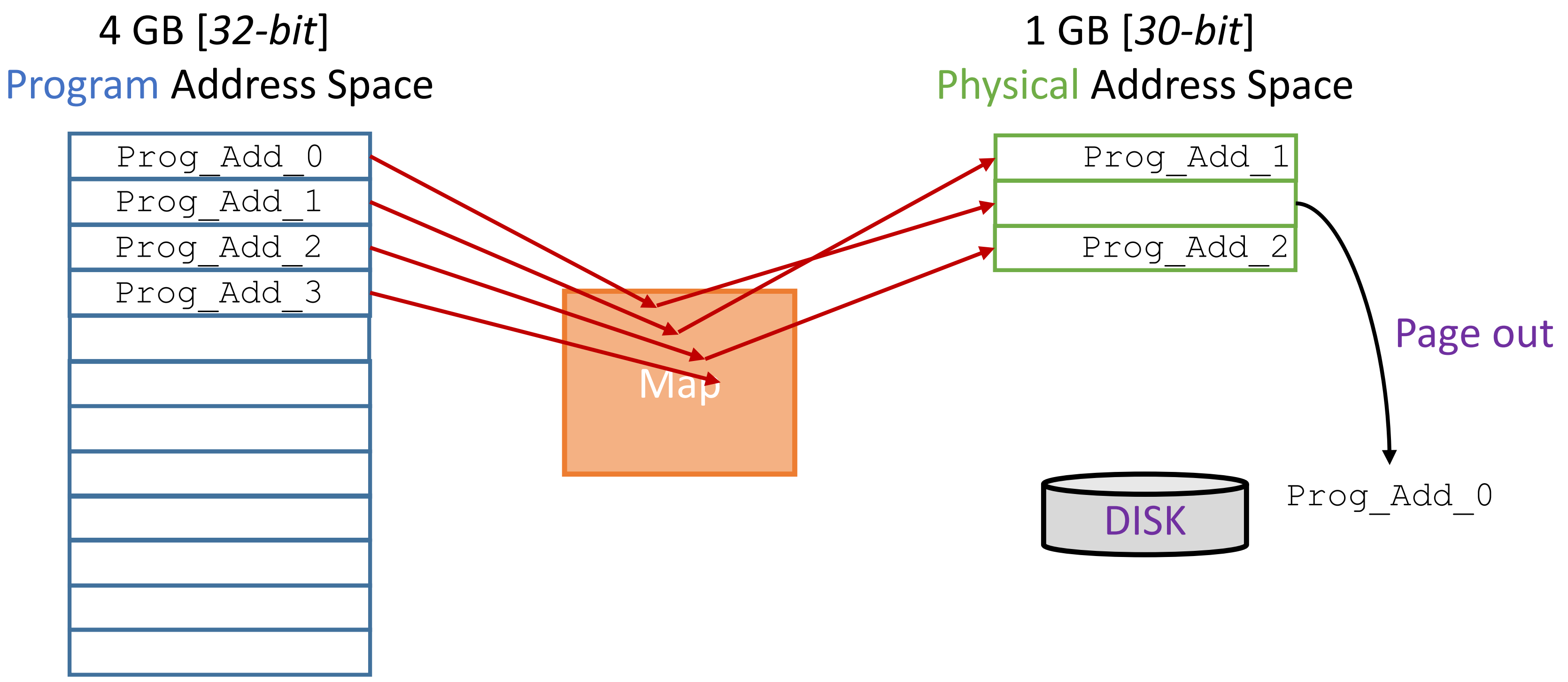
Solved: Problem #1 *(Not Enough Memory)*

- Map some program addresses to disk



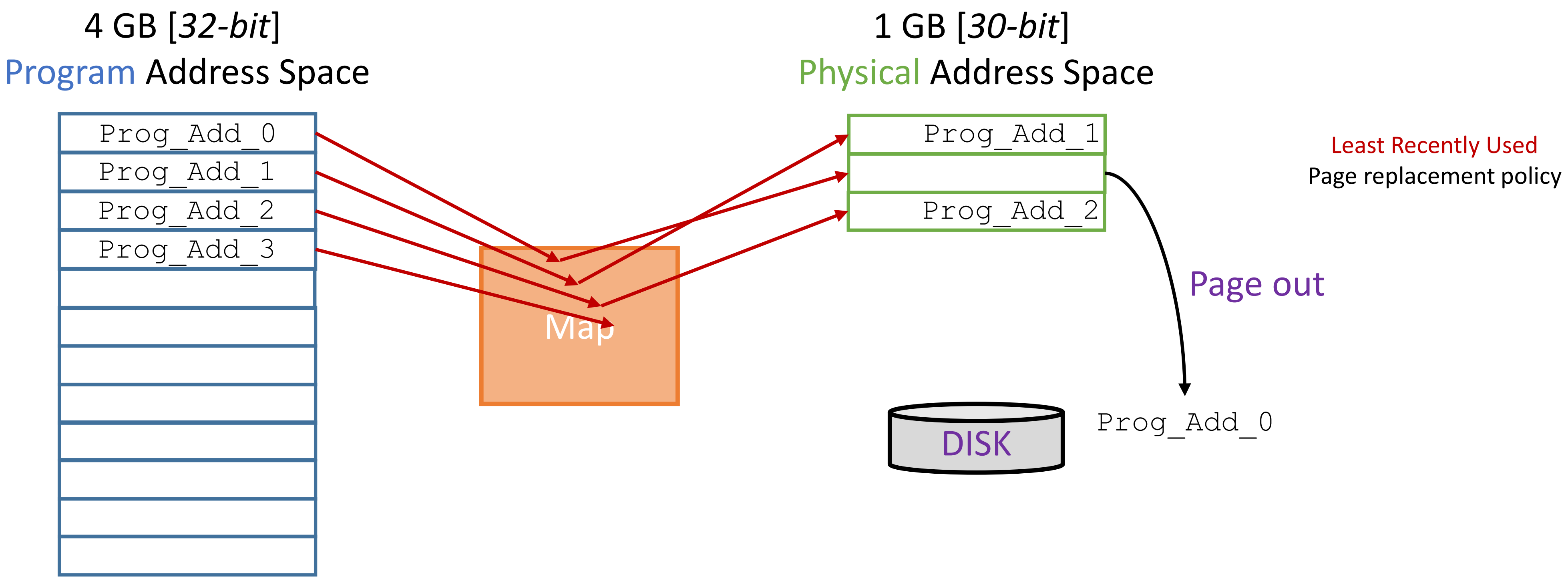
Solved: Problem #1 *(Not Enough Memory)*

- Map some program addresses to disk



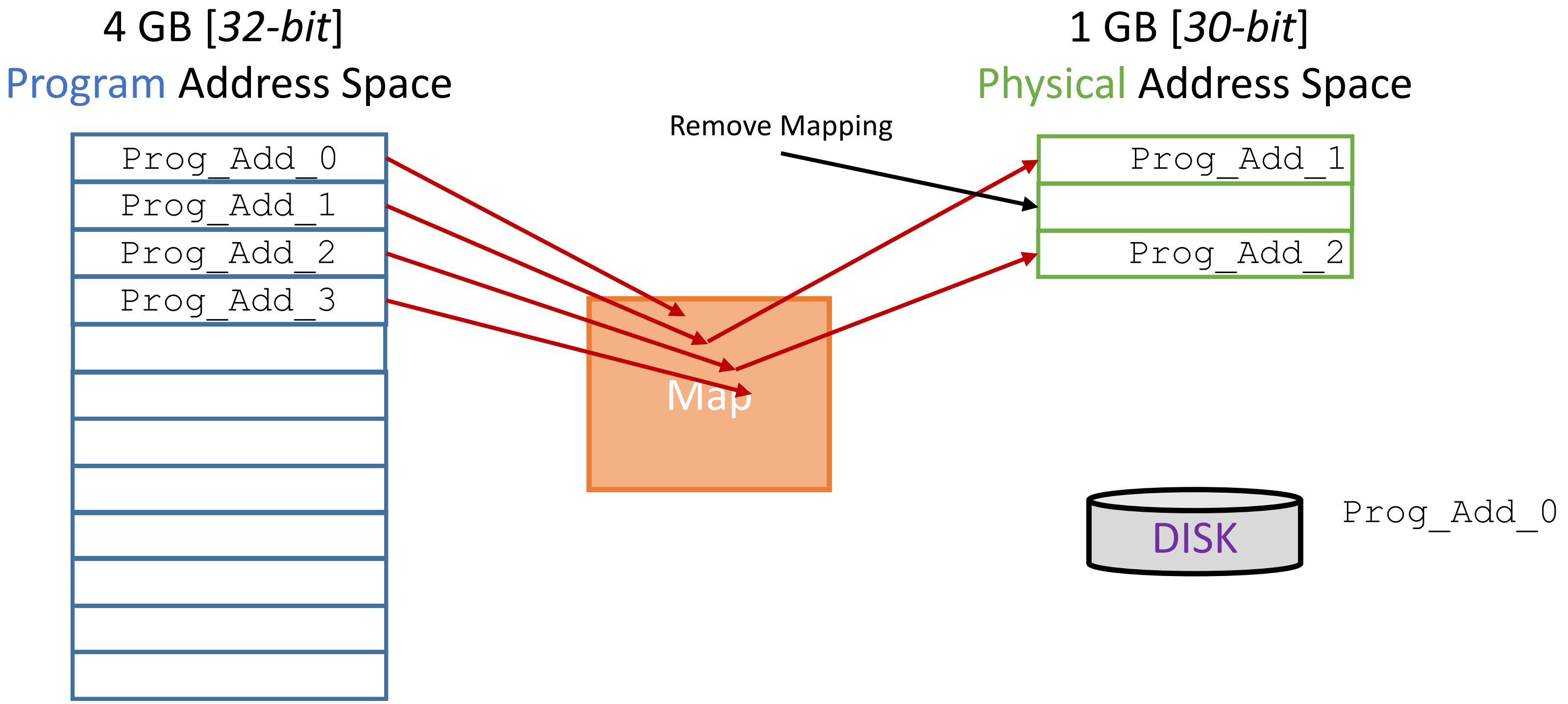
Solved: Problem #1 *(Not Enough Memory)*

- Map some program addresses to disk



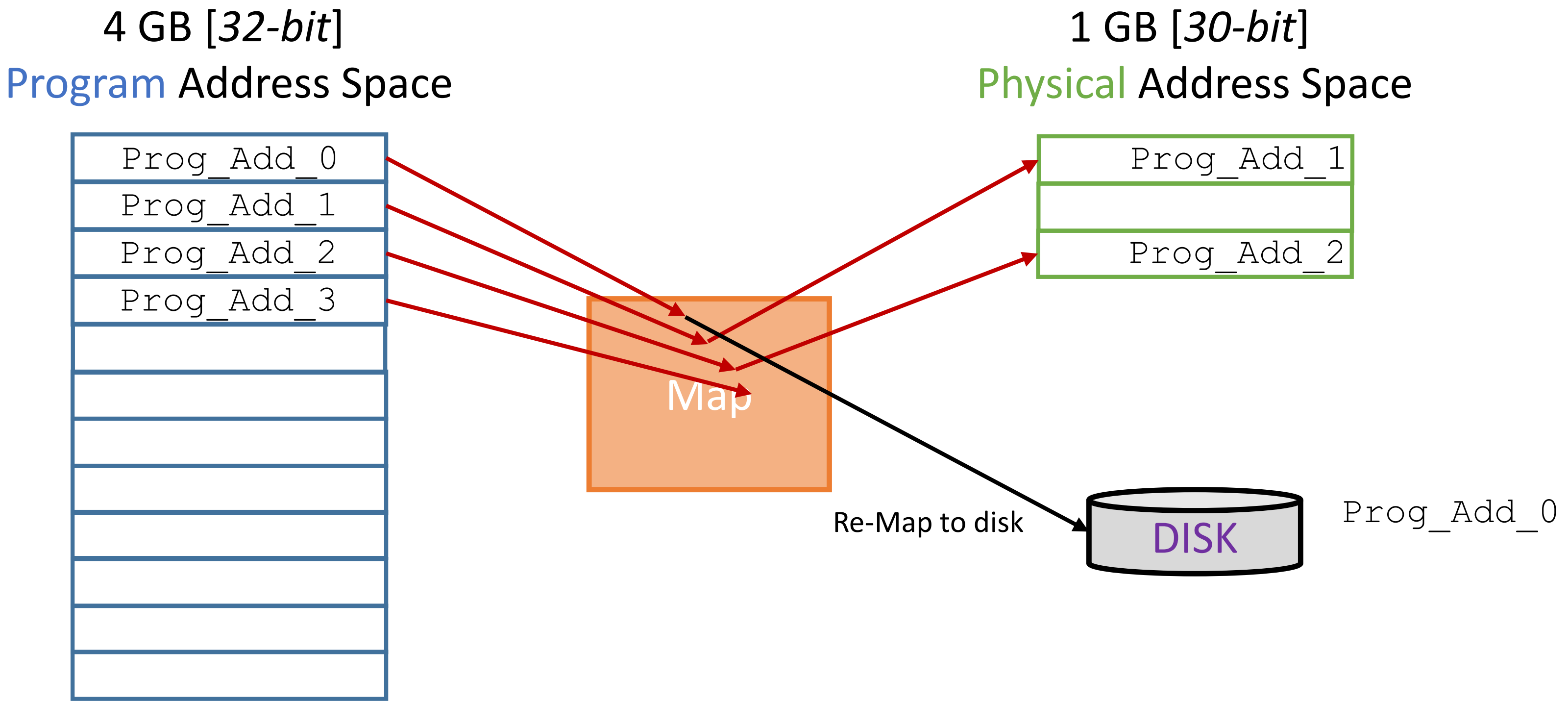
Solved: Problem #1 *(Not Enough Memory)*

- Map some program addresses to disk



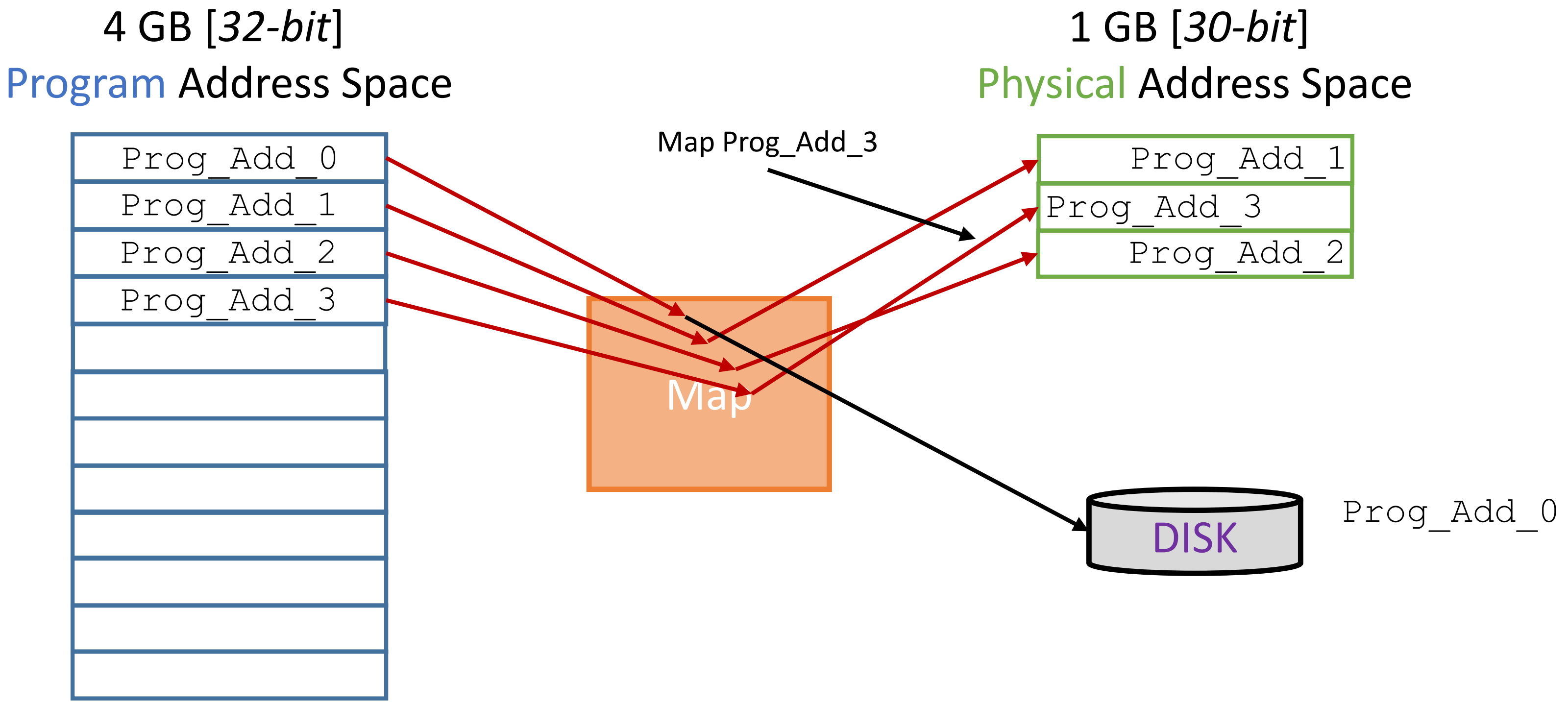
Solved: Problem #1 *(Not Enough Memory)*

- Map some program addresses to disk



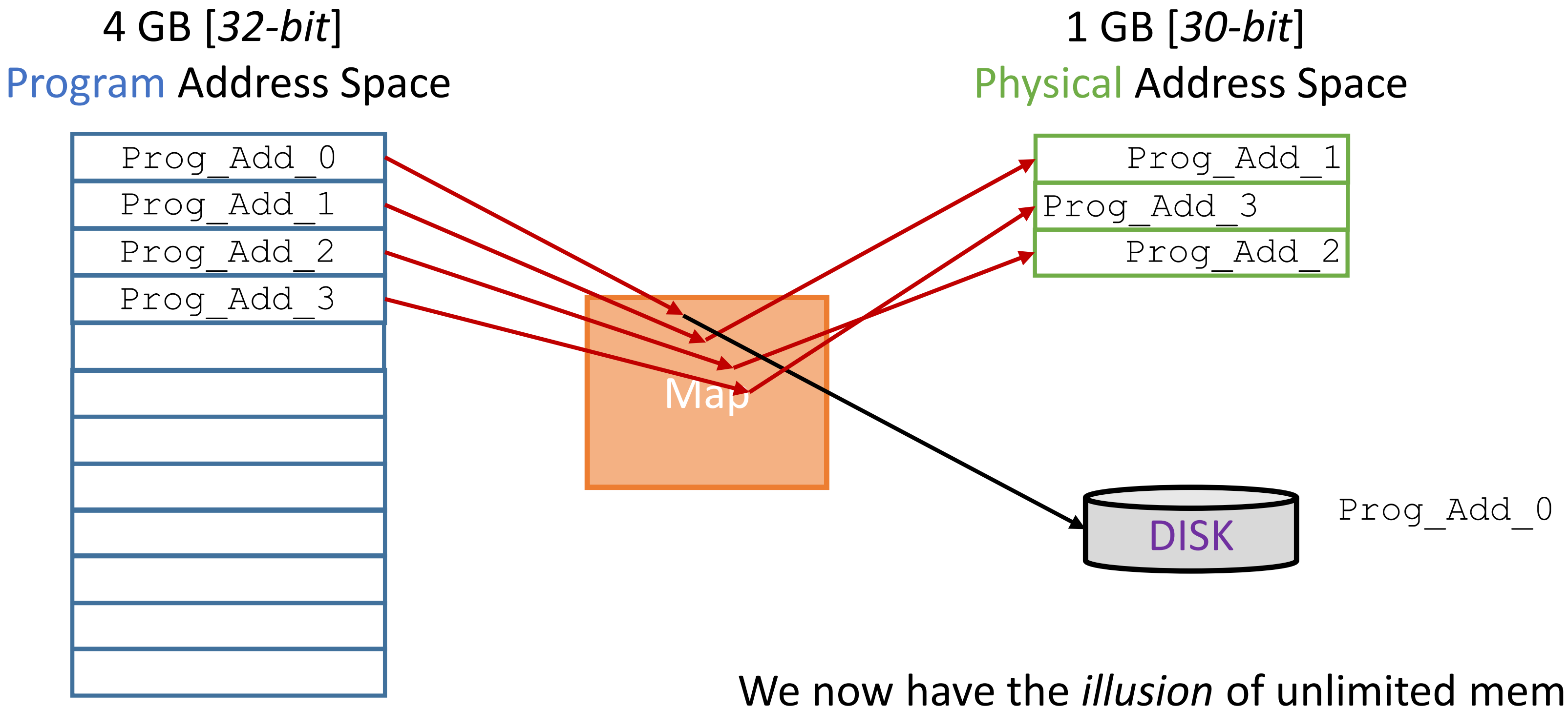
Solved: Problem #1 *(Not Enough Memory)*

- Map some program addresses to disk



Solved: Problem #1 *(Not Enough Memory)*

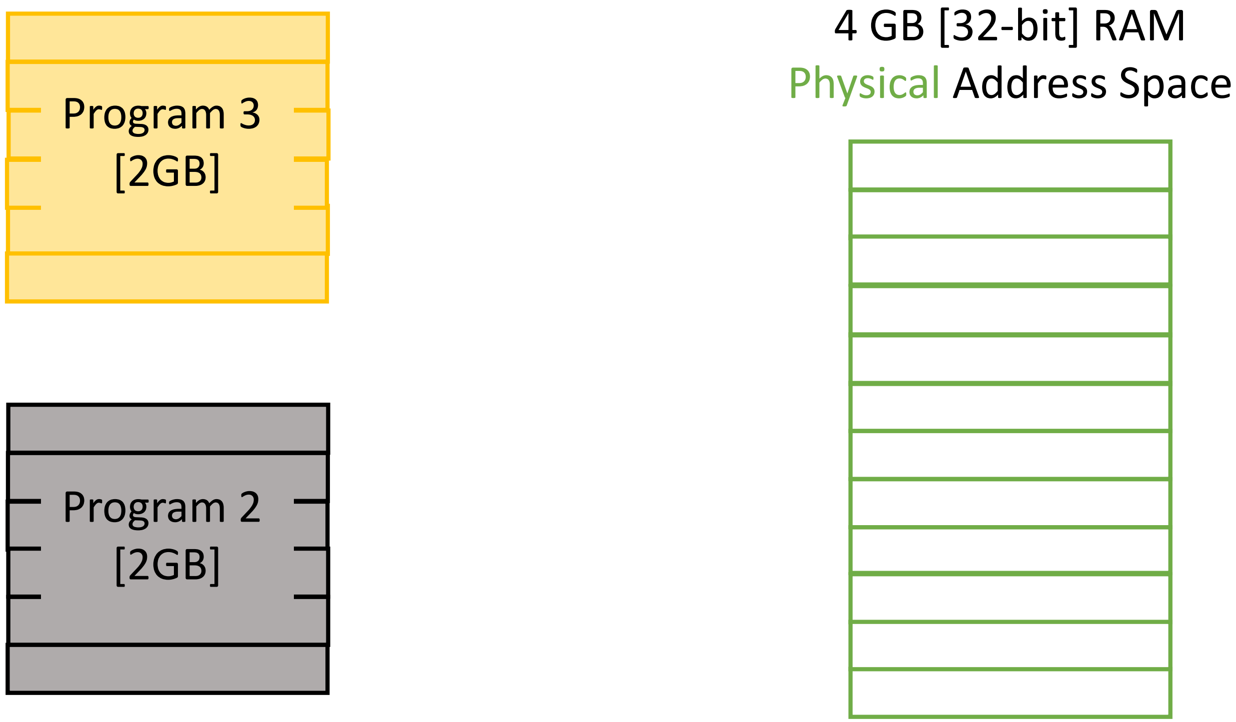
- Map some program addresses to disk



We now have the *illusion* of unlimited memory!

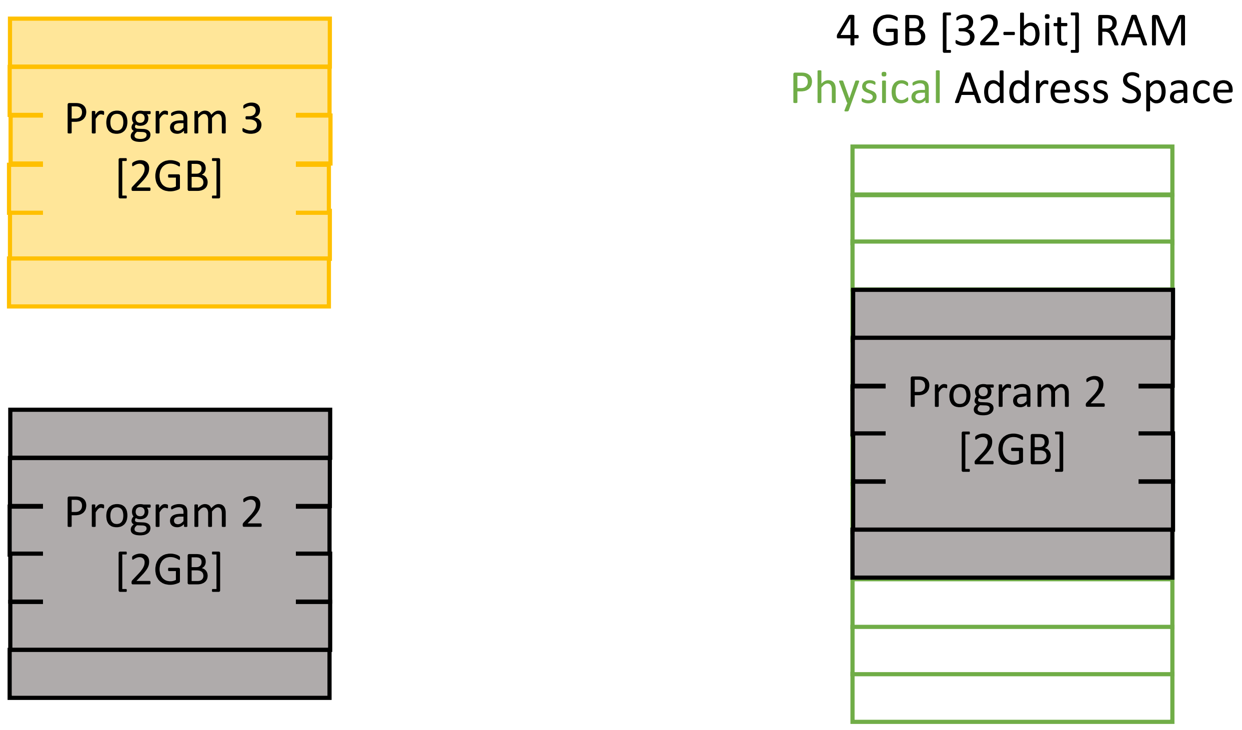
Solved: Problem #2 *(Holes in Memory)*

- We can map program addresses to non-sequential RAM addresses



Solved: Problem #2 *(Holes in Memory)*

- We can map program addresses to non-sequential RAM addresses

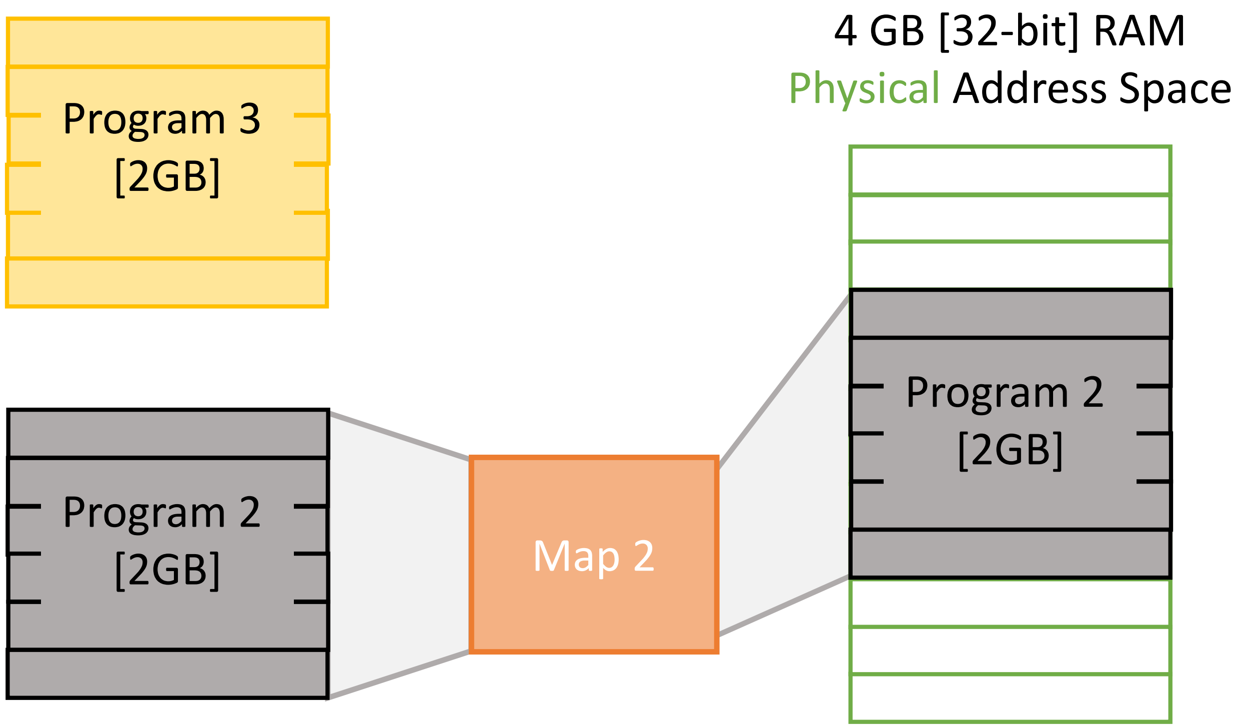


Program Sequence:

1. Run programs 1 and 2 [1 GB free]
2. Close program 1 [2 GB free]
3.

Solved: Problem #2 *(Holes in Memory)*

- We can map program addresses to non-sequential RAM addresses

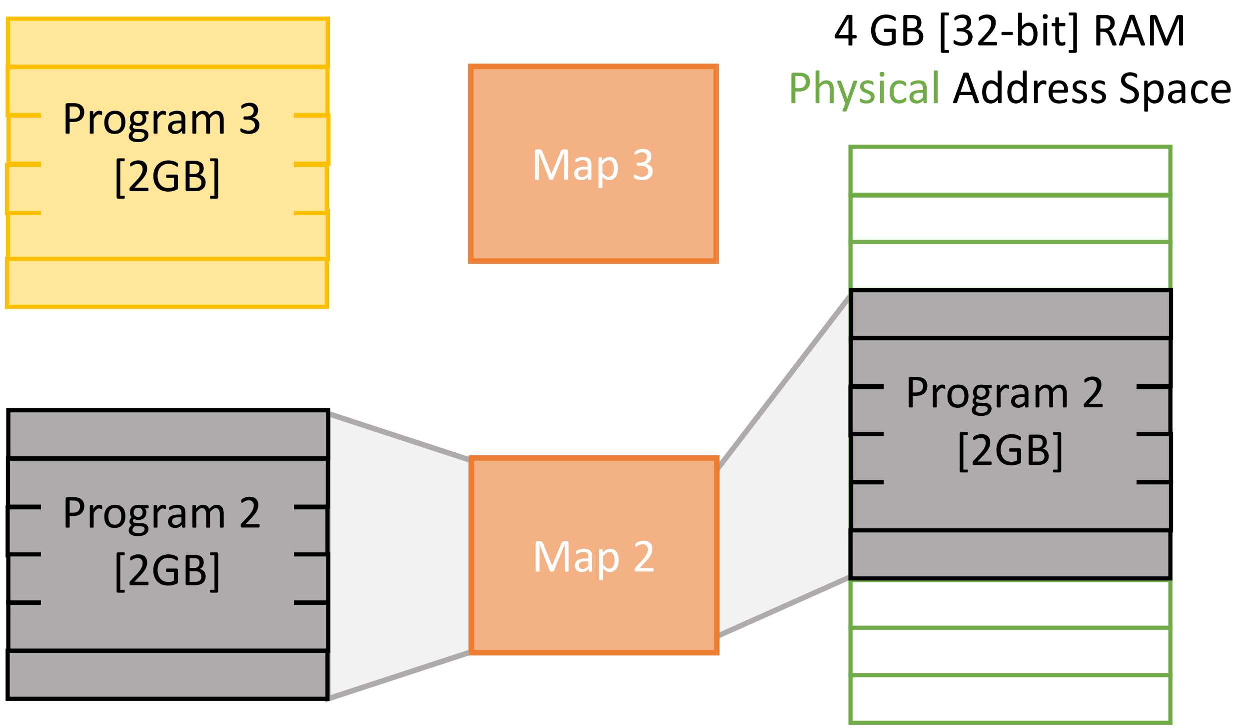


Program Sequence:

1. Run programs 1 and 2 [1 GB free]
2. Close program 1 [2 GB free]
3.

Solved: Problem #2 *(Holes in Memory)*

- We can map program addresses to non-sequential RAM addresses

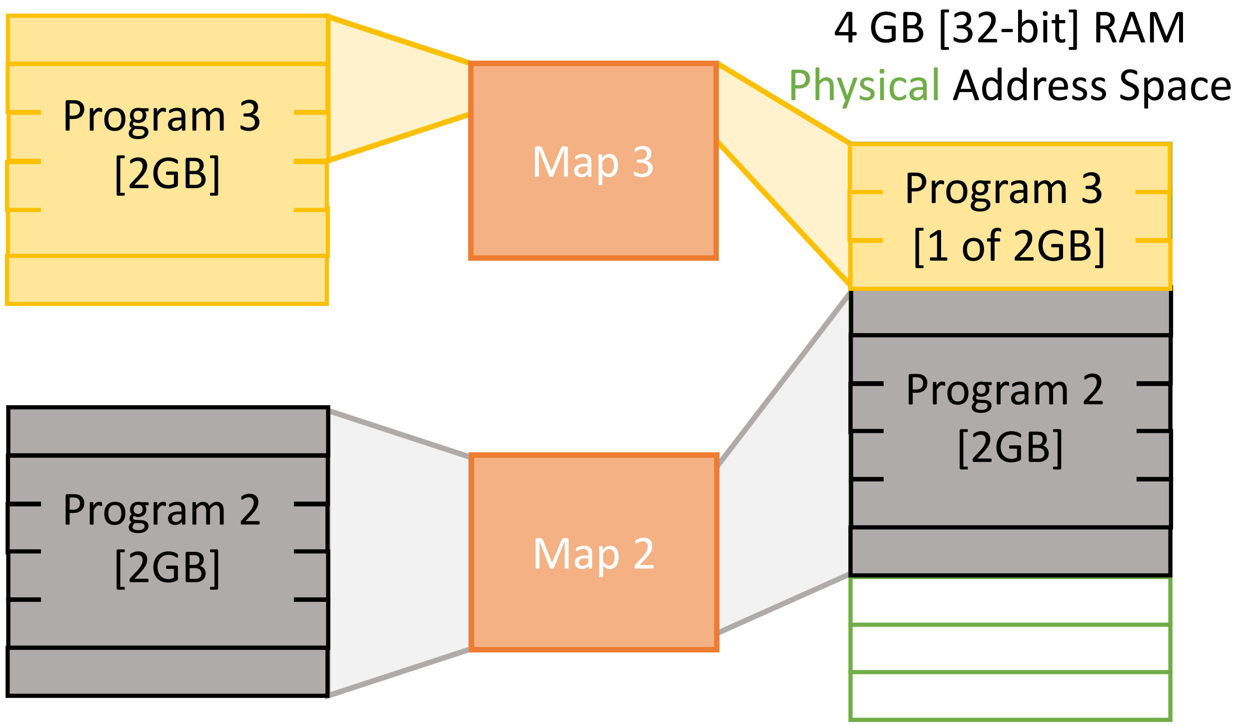


Program Sequence:

1. Run programs 1 and 2 [1 GB free]
2. Close program 1 [2 GB free]
3. Run program 3

Solved: Problem #2 *(Holes in Memory)*

- We can map program addresses to non-sequential RAM addresses

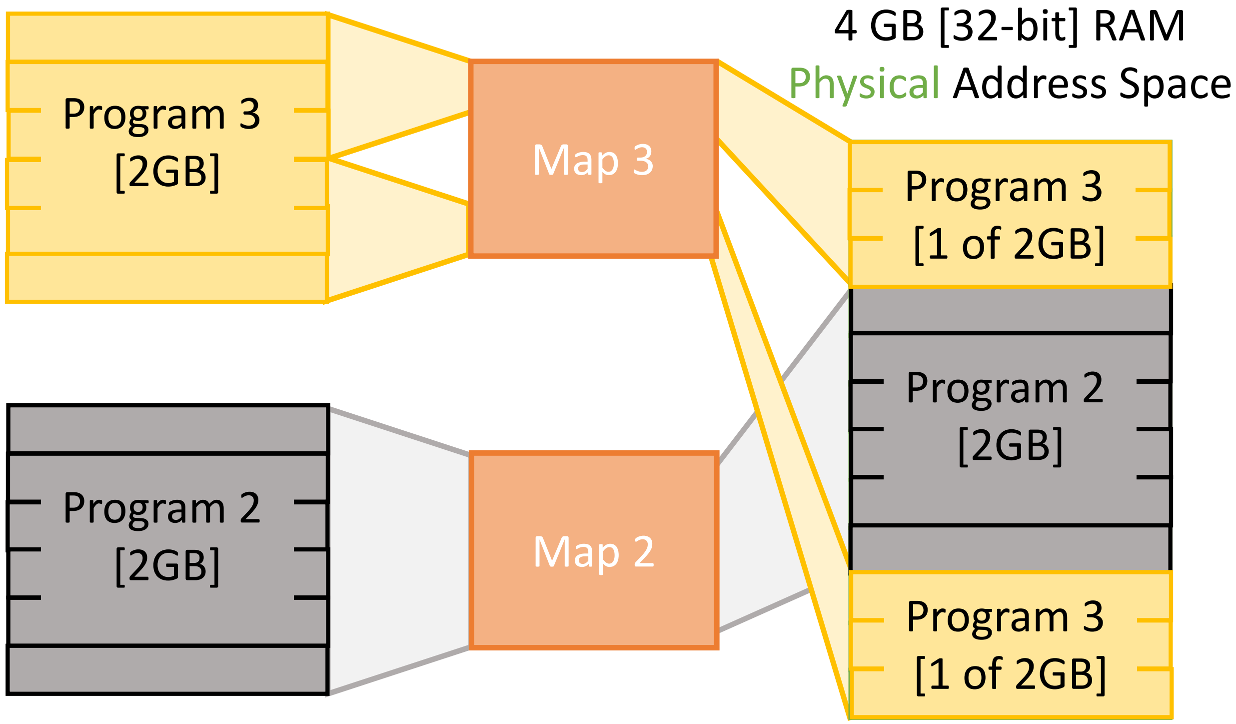


Program Sequence:

1. Run programs 1 and 2 [1 GB free]
2. Close program 1 [2 GB free]
3. Run program 3

Solved: Problem #2 *(Holes in Memory)*

- We can map program addresses to non-sequential RAM addresses

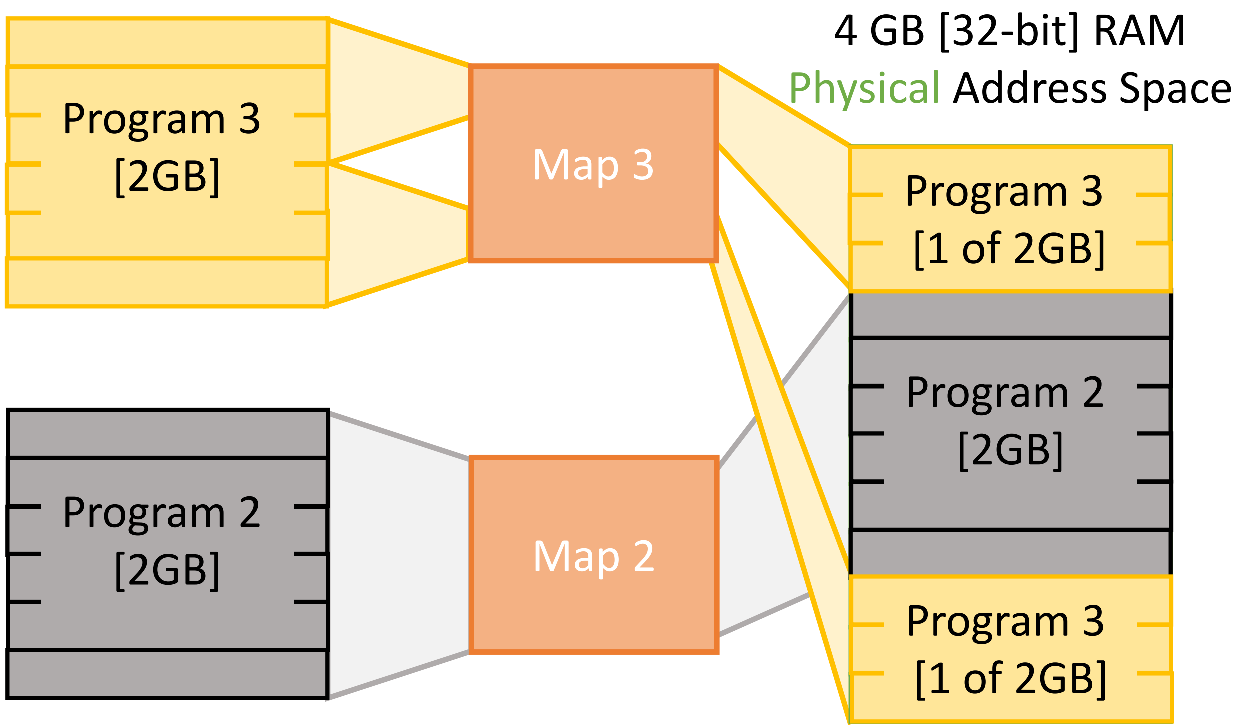


Program Sequence:

1. Run programs 1 and 2 [1 GB free]
2. Close program 1 [2 GB free]
3. Run program 3 [CAN DO!]

Solved: Problem #2 *(Holes in Memory)*

- We can map program addresses to non-sequential RAM addresses



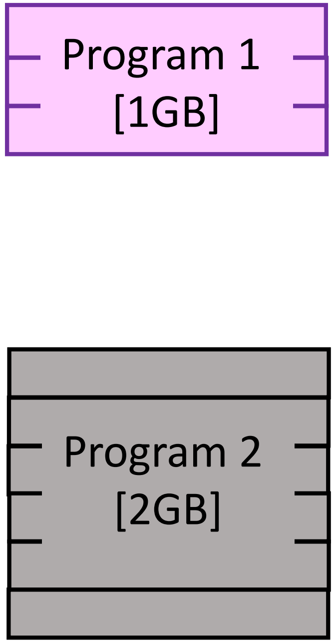
Program Sequence:

1. Run programs 1 and 2 [1 GB free]
2. Close program 1 [2 GB free]
3. Run program 3 [CAN DO!]

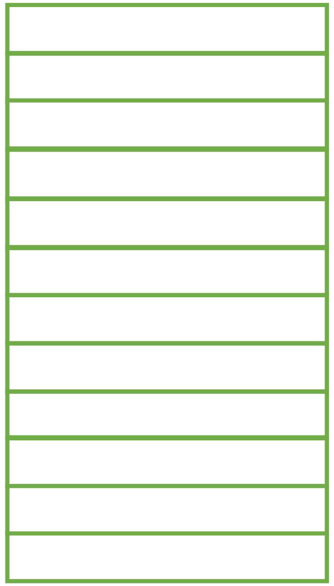
We can now put programs anywhere
We want to in memory!

Solved: Problem #3 *(Data Corruption)*

- We can map a program address to a distinct RAM address



4 GB [32-bit] RAM
Physical Address Space



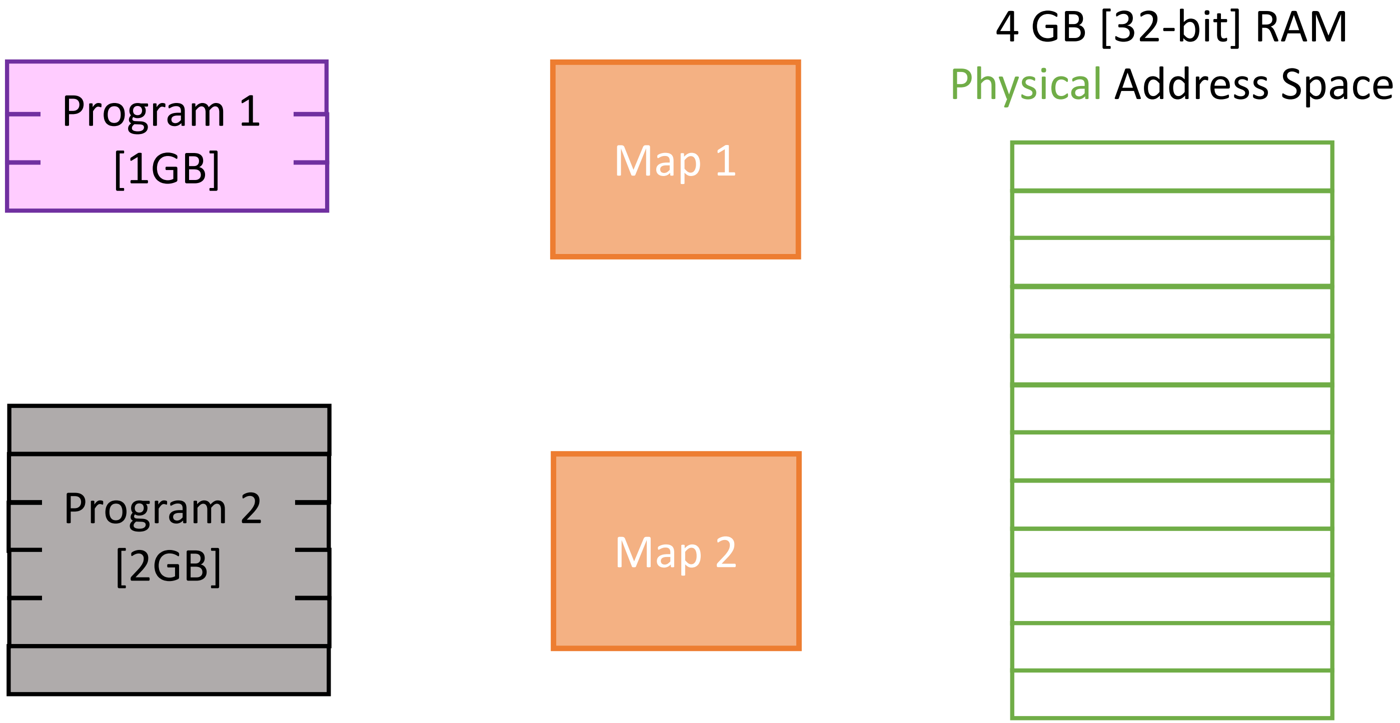
Code Segment:

```
P1: LW R2, 0x100(R0)
P2: LW R4, 0x100(R0)
```

Program 1: stores bank account balance
Program 2: stores pi

Solved: Problem #3 *(Data Corruption)*

- We can map a program address to a distinct RAM address

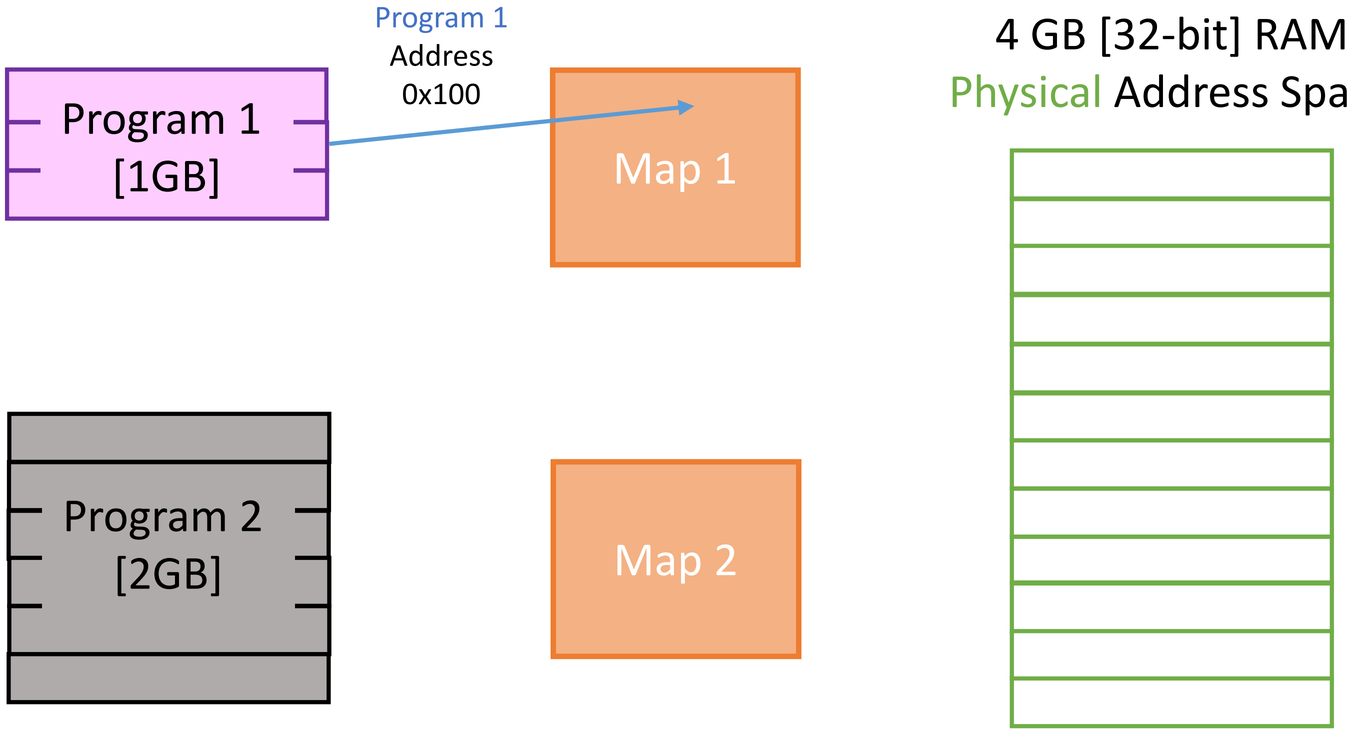



```
Code Segment:
P1: LW R2, 0x100(R0)
P2: LW R4, 0x100(R0)
```

Program 1: stores bank account balance
Program 2: stores pi

Solved: Problem #3 *(Data Corruption)*

- We can map a program address to a distinct RAM address



Code Segment:

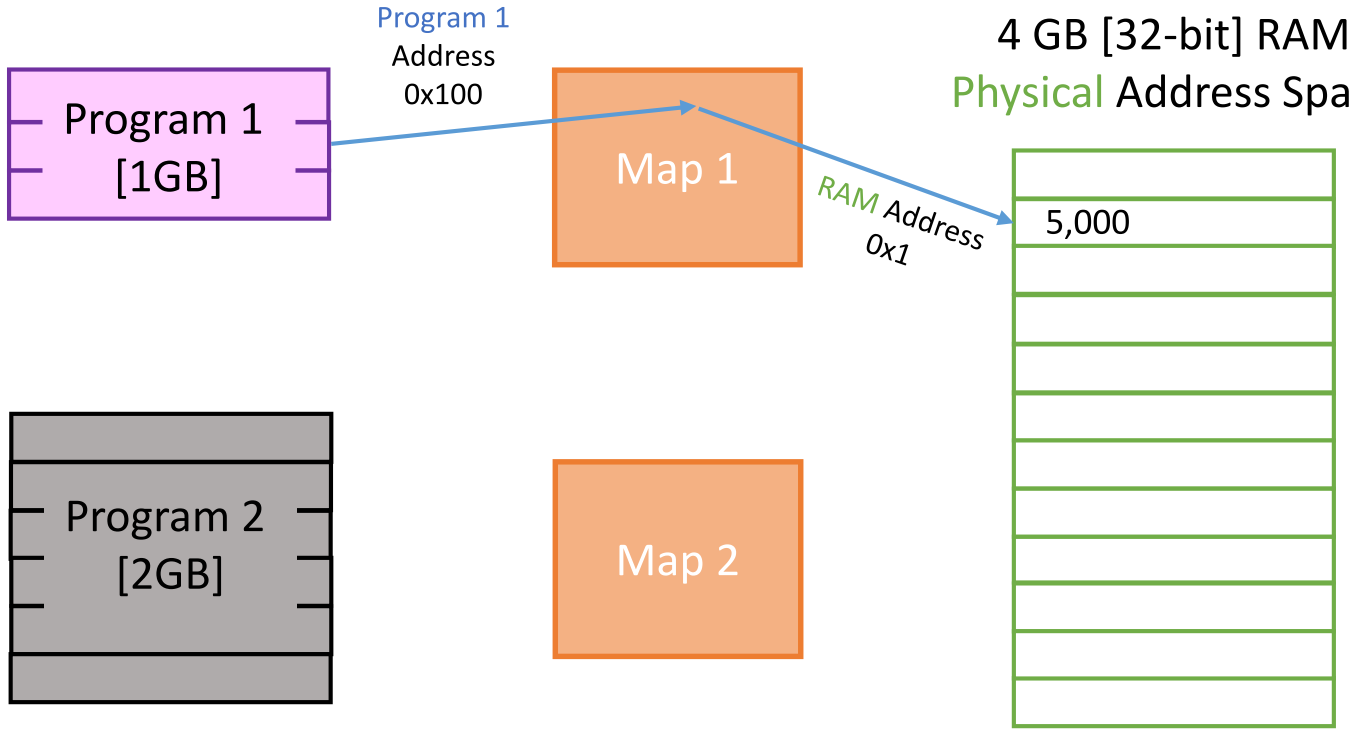
```
P1: LW R2, 0x100(R0)
P2: LW R4, 0x100(R0)
```

Program 1: stores bank account balance

Program 2: stores pi

Solved: Problem #3 *(Data Corruption)*

- We can map a program address to a distinct RAM address



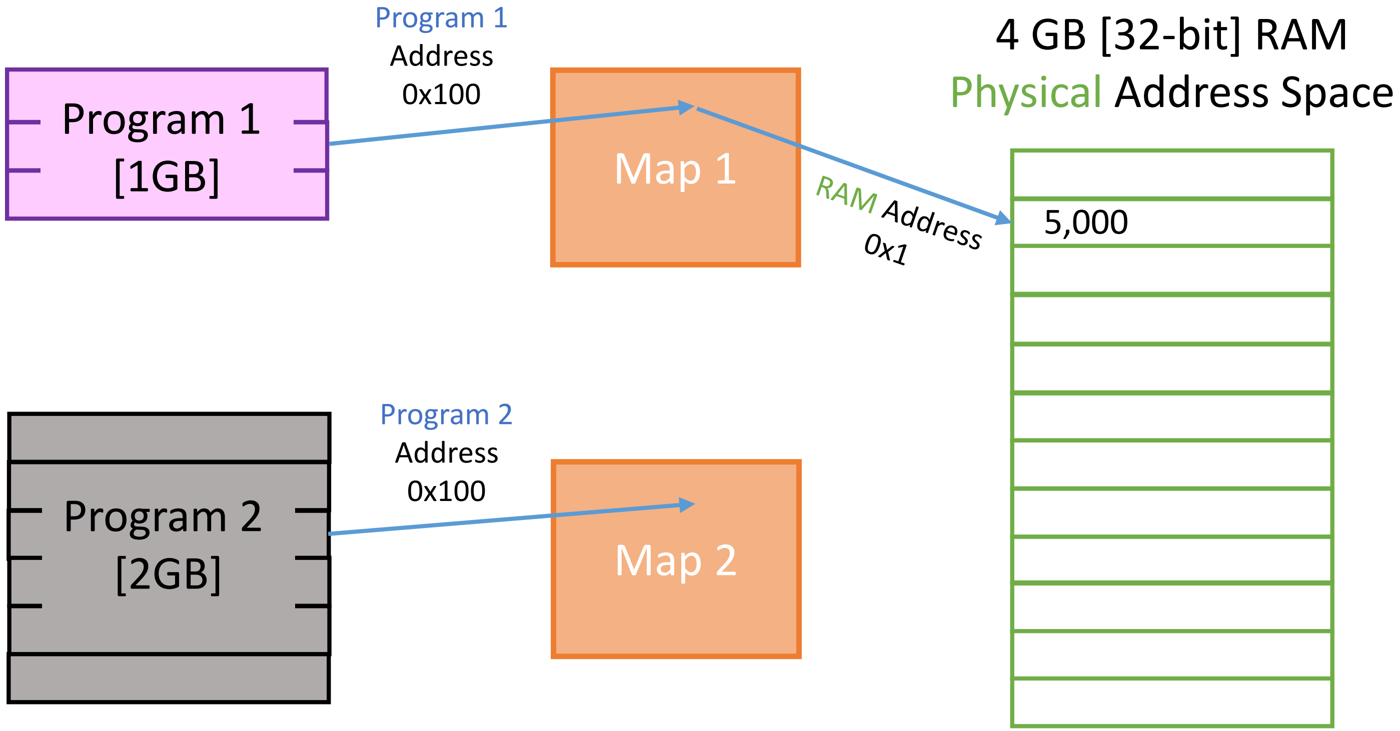
Code Segment:

```
P1: LW R2, 0x100(R0)
P2: LW R4, 0x100(R0)
```

Program 1: stores bank account balance
Program 2: stores pi

Solved: Problem #3 *(Data Corruption)*

- We can map a program address to a distinct RAM address



Code Segment:

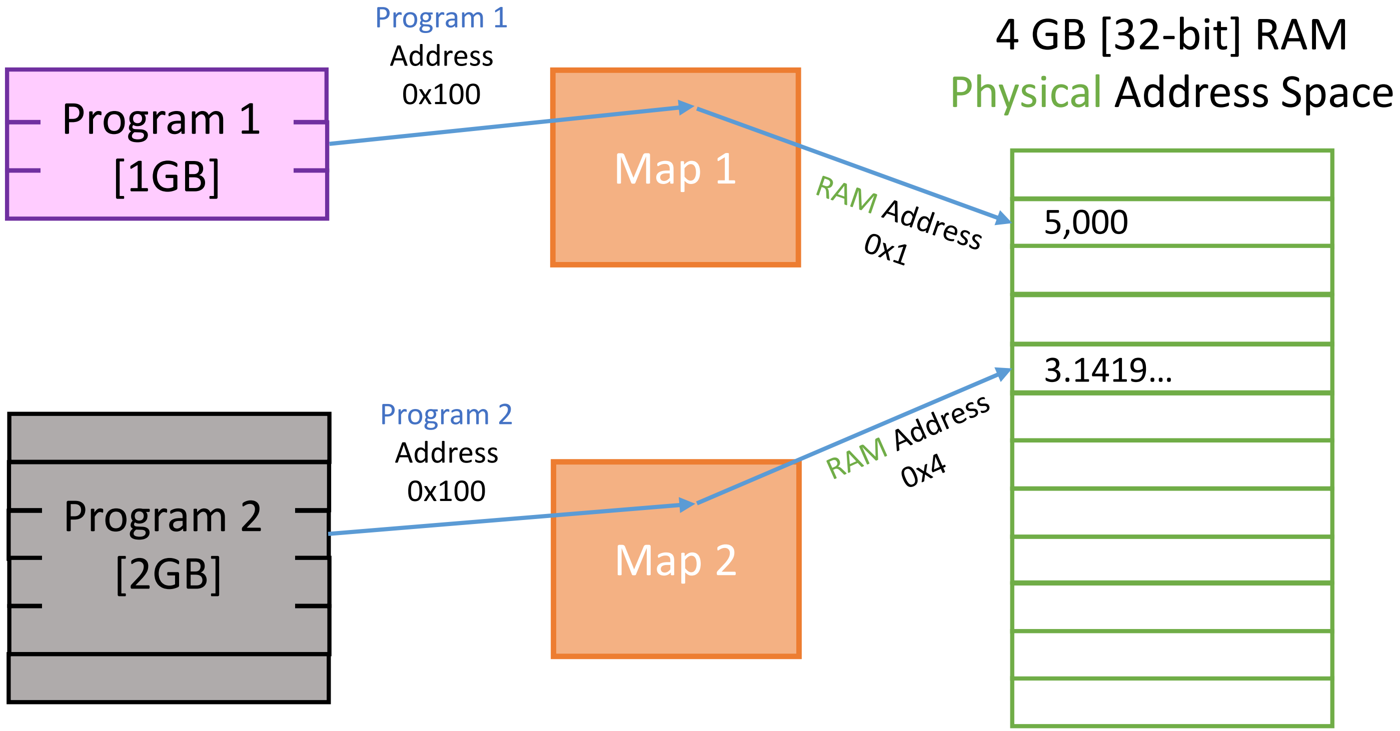
```
P1: LW R2, 0x100(R0)
P2: LW R4, 0x100(R0)
```

Program 1: stores bank account balance

Program 2: stores pi

Solved: Problem #3 *(Data Corruption)*

- We can map a program address to a distinct RAM address



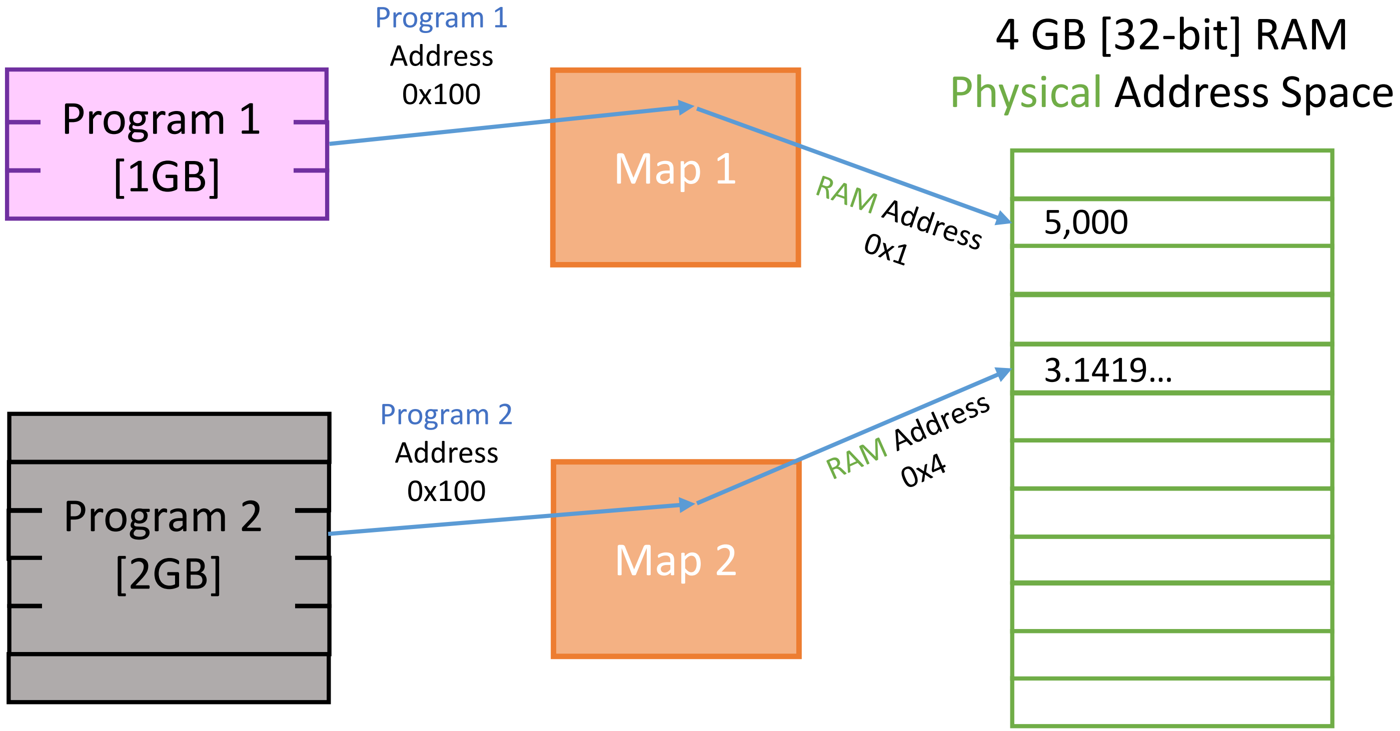
Code Segment:

```
P1: LW R2, 0x100(R0)
P2: LW R4, 0x100(R0)
```

Program 1: stores bank account balance
Program 2: stores pi

Solved: Problem #3 *(Data Corruption)*

- We can map a program address to a distinct RAM address



Code Segment:

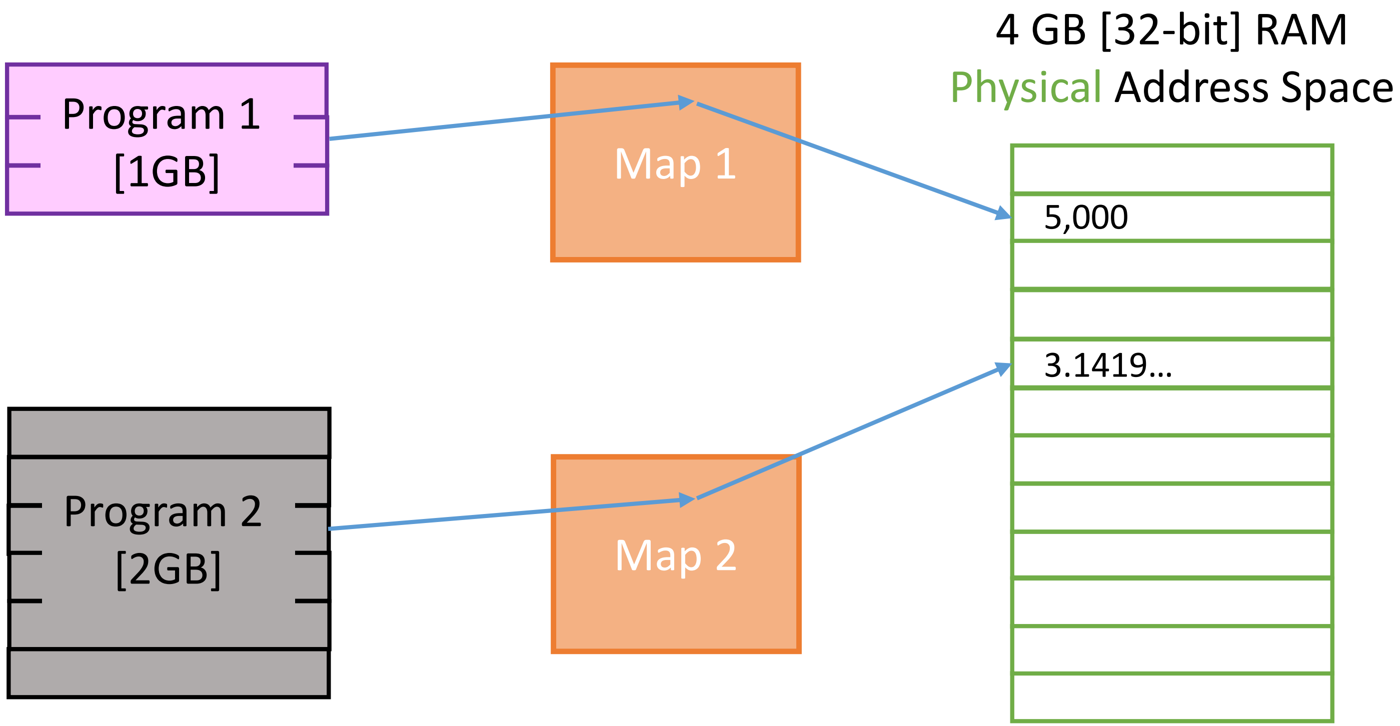
```
P1: LW R2, 0x100(R0)
P2: LW R4, 0x100(R0)
```

Program 1: stores bank account balance
Program 2: stores pi

Applications with the same program address no longer map to the same hardware address!

Solved: Problem #3 – Sharing Data

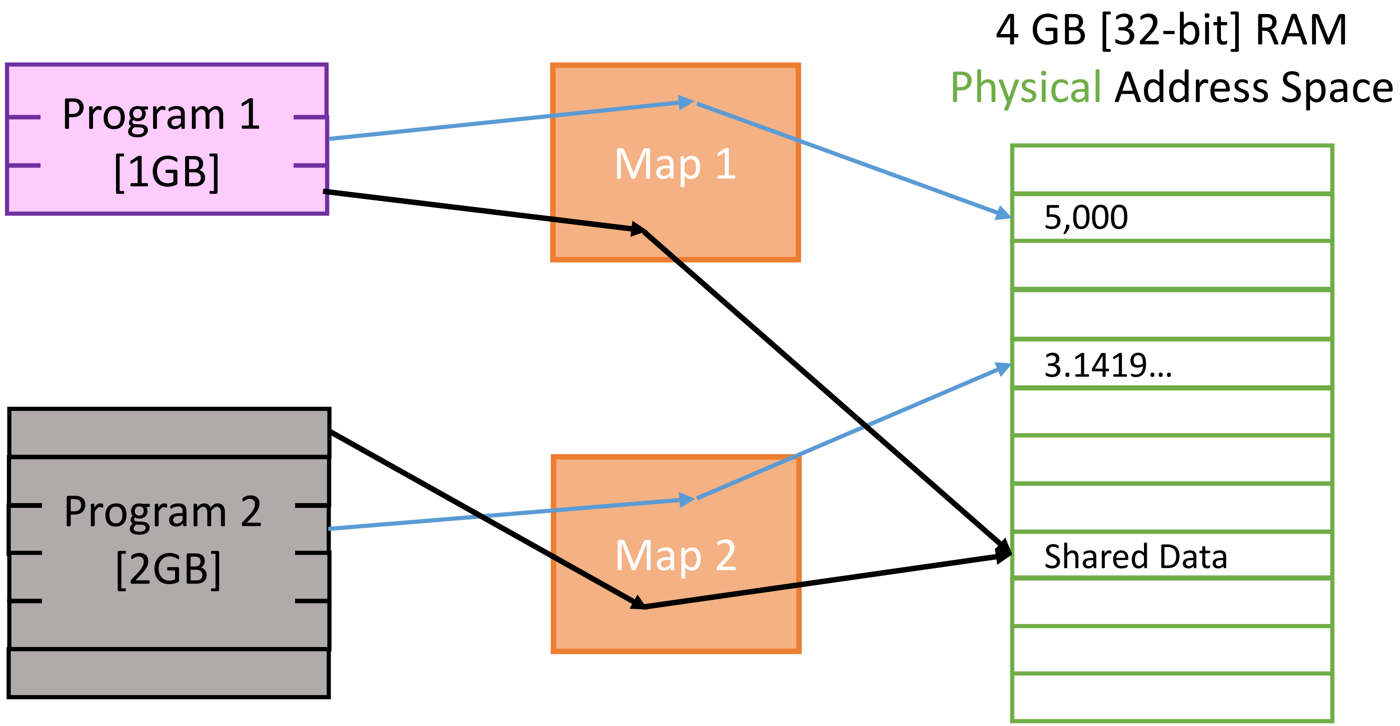
- What if I *want* to share data?



Solved: Problem #3 – Sharing Data

- What if I *want* to share data?

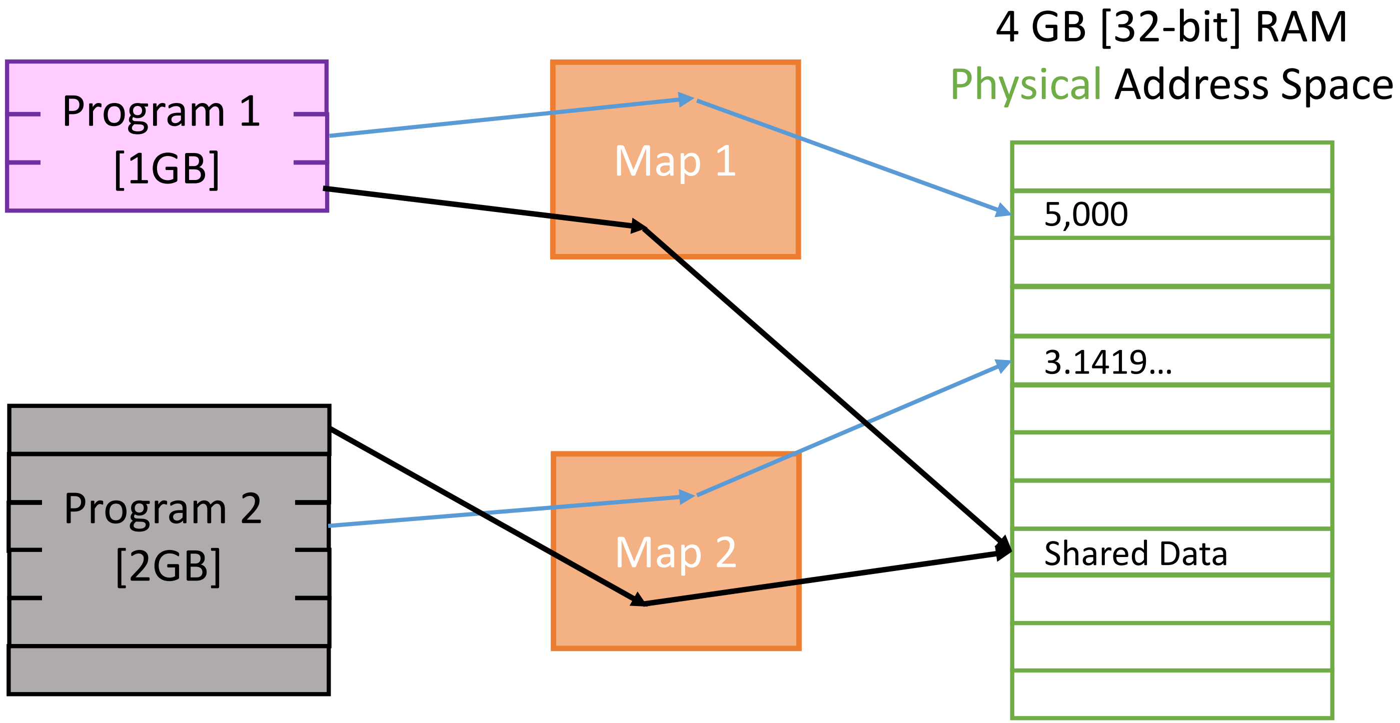
We can do this too!

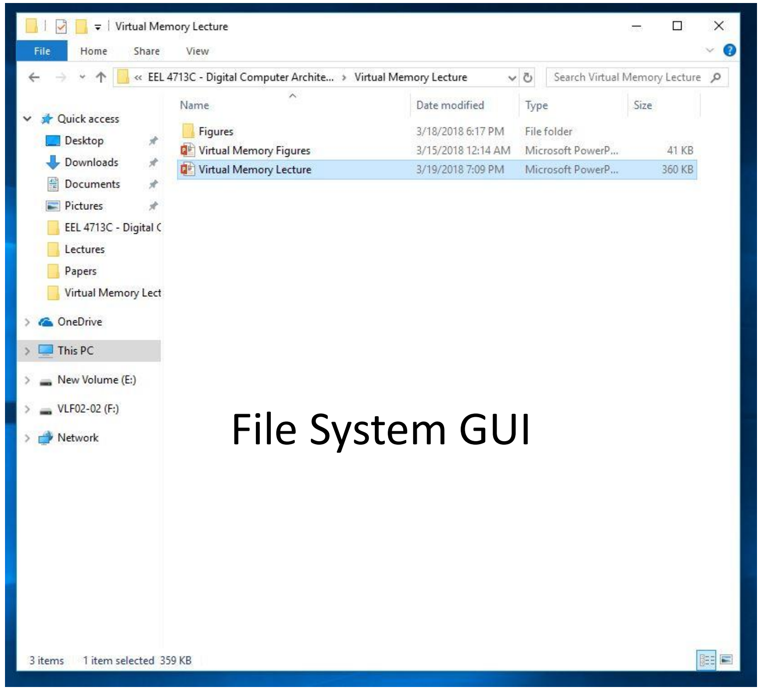


Solved: Problem #3 – Sharing Data

- What if I *want* to share data?

We can do this too!





Quincy Flint

How Does VM Work?

How does Virtual Memory Work?

How does Virtual Memory Work?

- Separate memory spaces:
  - Virtual Memory
    - What the program sees
  - Physical Memory
    - The physical RAM installed in machine

How does Virtual Memory Work?

- Separate memory spaces:
  - Virtual Memory
    - What the program sees
  - Physical Memory
    - The physical RAM installed in machine
- Virtual Address [VA]
  - What the program uses
  - In MIPS we have a 32-bit address space, 0 to 2^{32}-1

How does Virtual Memory Work?

- Separate memory spaces:
  - Virtual Memory
    - What the program sees
  - Physical Memory
    - The physical RAM installed in machine

- Virtual Address [VA]
  - What the program uses
  - In MIPS we have a 32-bit address space, 0 to 2^{32}-1

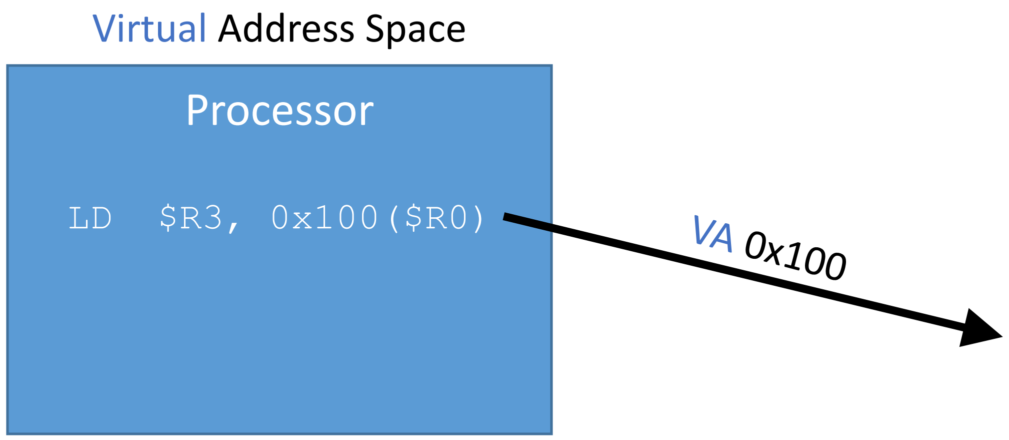
- Physical Address [PA]
  - What the hardware uses
  - Address space determined by RAM, if 1GB RAM then 0 to 2^{30}-1

How does Virtual Memory Work?

- How does a program access memory?

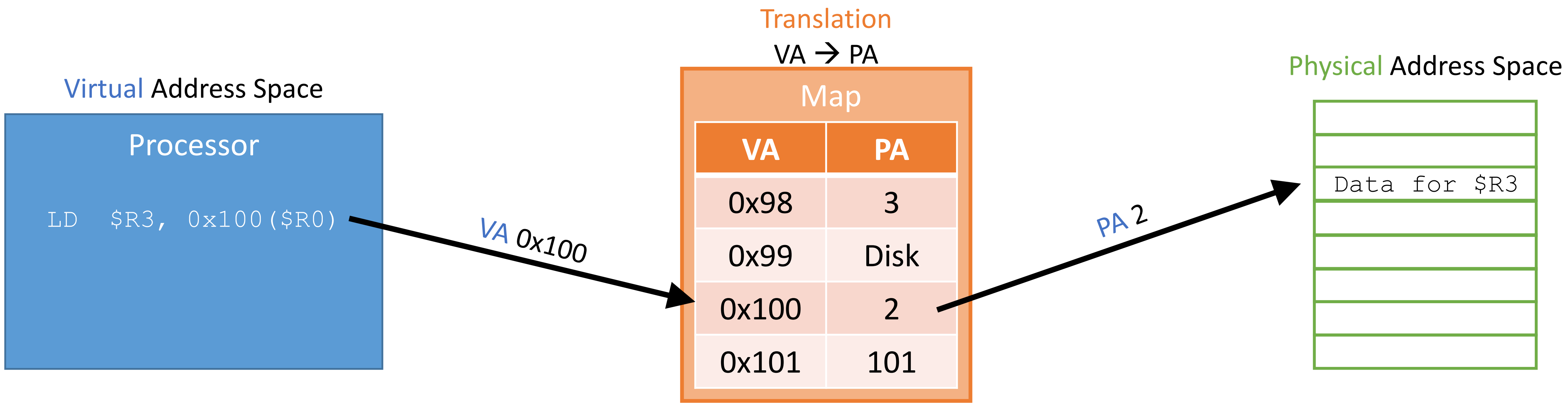
How does Virtual Memory Work?

- How does a program access memory?
    1. Program executes a load with a virtual address



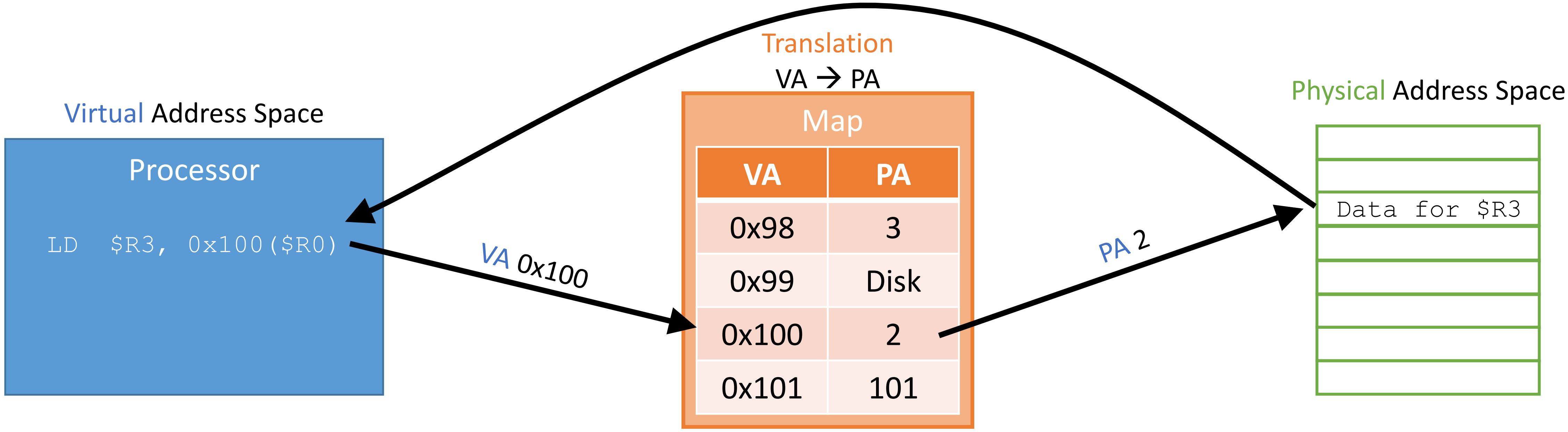
How does Virtual Memory Work?

- How does a program access memory?
  1. Program executes a load with a virtual address
  2. Computer translates virtual address to a physical address



How does Virtual Memory Work?

- How does a program access memory?
  1. Program executes a load with a virtual address
  2. Computer translates virtual address to a physical address
  3. Computer reads data from RAM and returns to the program



How does Virtual Memory Work?

- How does a program access memory?
  1. Program executes a load with a virtual address
  2. Computer translates virtual address to a physical address
  3. Computer reads data from RAM and returns to the program

Virtual Address Space

Processor

```
LD  $R3, 0x100($R0)
LD  $R2,  0x98($R0)
```

Translation

VA → PA

Map

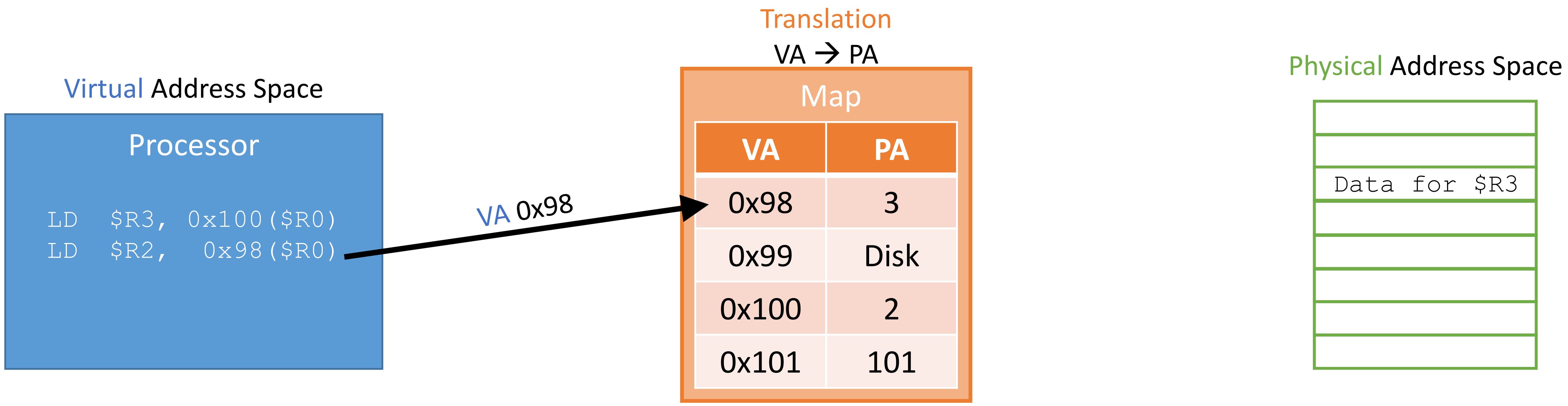
VA	PA
0x98	3
0x99	Disk
0x100	2
0x101	101

Physical Address Space

Data for $R3

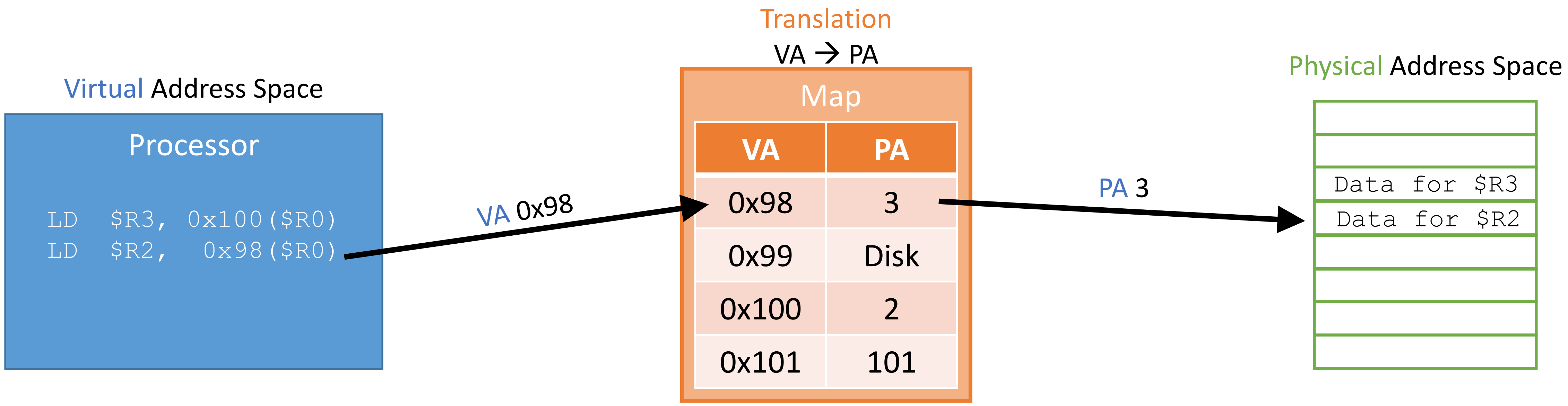
How does Virtual Memory Work?

- How does a program access memory?
  1. Program executes a load with a virtual address
  2. Computer translates virtual address to a physical address
  3. Computer reads data from RAM and returns to the program



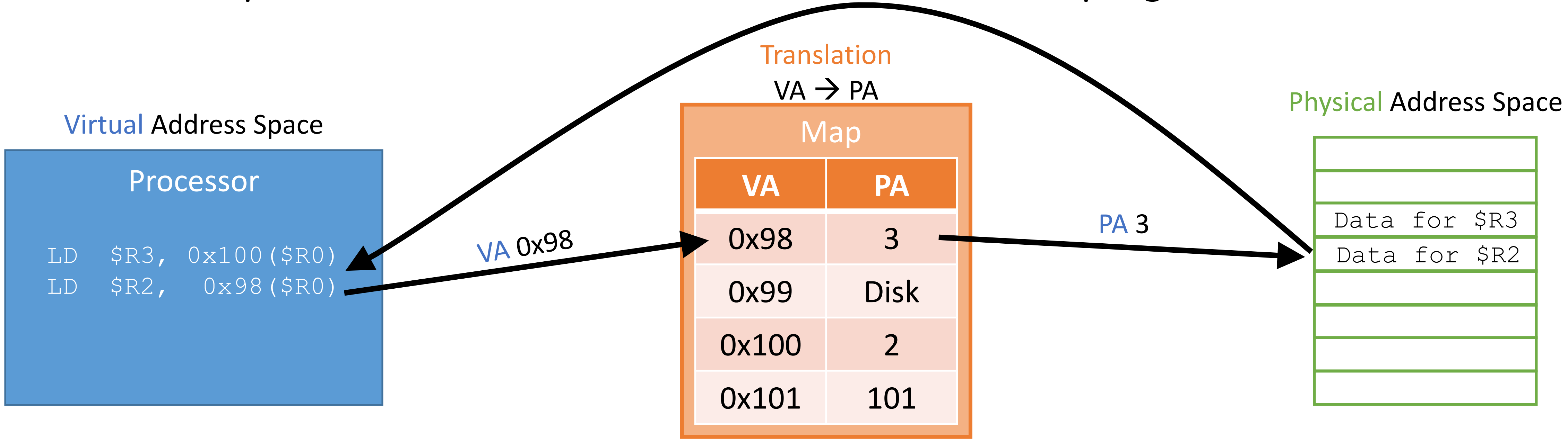
How does Virtual Memory Work?

- How does a program access memory?
  1. Program executes a load with a virtual address
  2. Computer translates virtual address to a physical address
  3. Computer reads data from RAM and returns to the program



How does Virtual Memory Work?

- How does a program access memory?
  1. Program executes a load with a virtual address
  2. Computer translates virtual address to a physical address
  3. Computer reads data from RAM and returns to the program



How does Virtual Memory Work?

- How does a program access memory?
  1. Program executes a load with a virtual address
  2. Computer translates virtual address to a physical address
  3. Computer reads data from RAM and returns to the program

Virtual Address Space

Processor

```
LD  $R3, 0x100($R0)
LD  $R2,  0x98($R0)
ADD $R4, $R2, $R3 .
```

Translation

VA → PA

Map

VA	PA
0x98	3
0x99	Disk
0x100	2
0x101	101

Physical Address Space

Data for $R3
Data for $R2

How does Virtual Memory Work?

- How does a program access memory?
  1. Program executes a load with a virtual address
  2. Computer translates virtual address to a physical address
  3. Computer reads data from RAM and returns to the program

Virtual Address Space

Processor

```
LD  $R3, 0x100($R0)
LD  $R2,  0x98($R0)
ADD $R4, $R2, $R3 .
LD  $R5,  0x99($R0)
```

Translation

VA → PA

Map

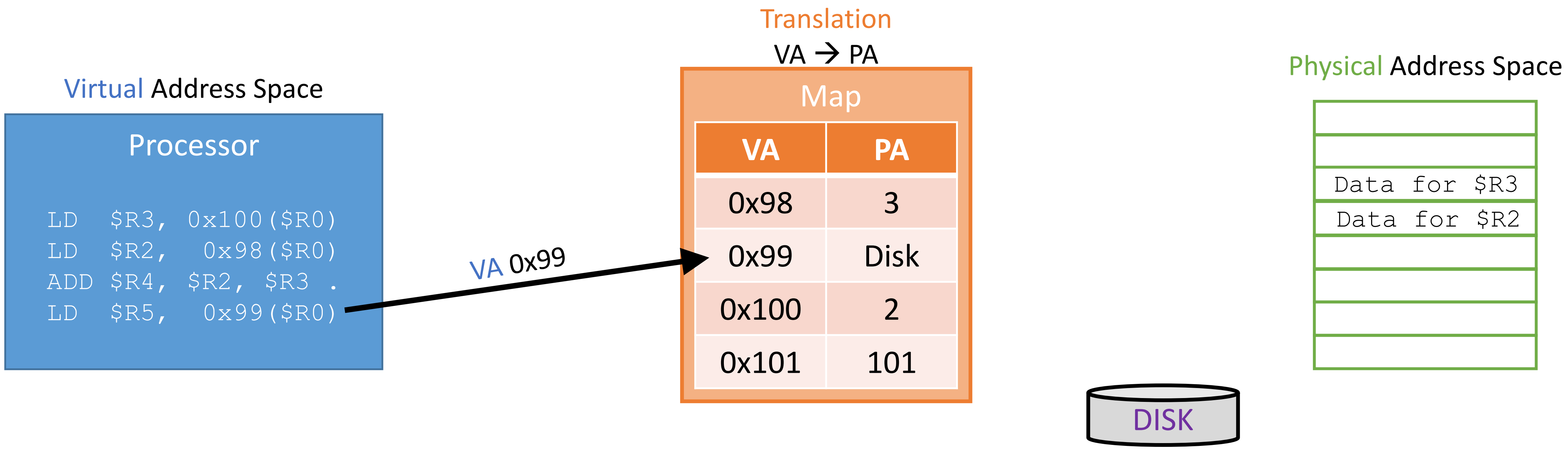
VA	PA
0x98	3
0x99	Disk
0x100	2
0x101	101

Physical Address Space

Data for $R3
Data for $R2

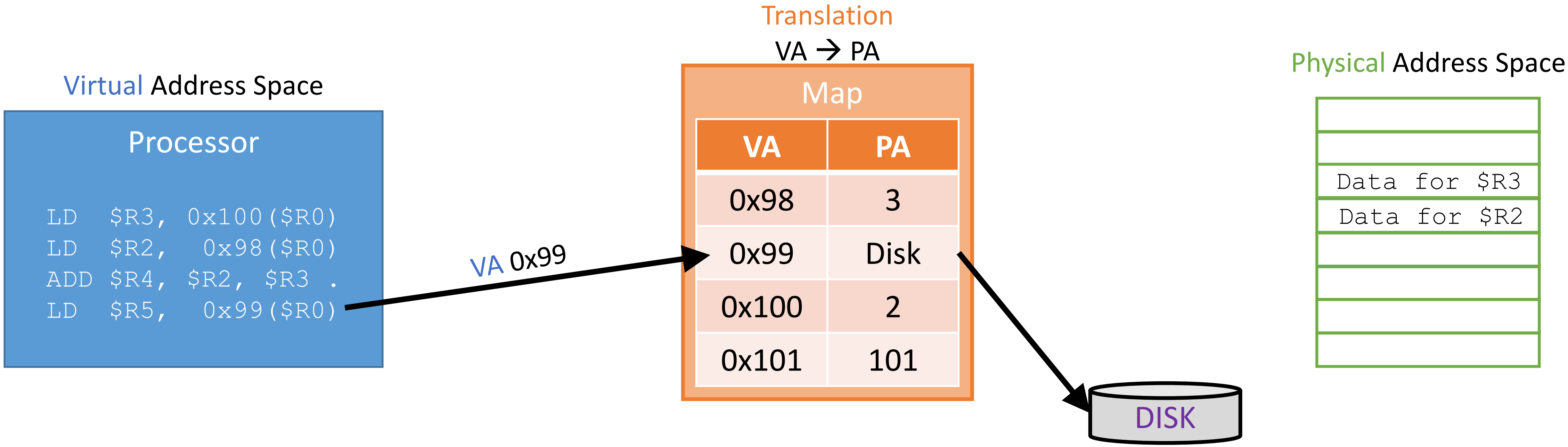
How does Virtual Memory Work?

- How does a program access memory?
  1. Program executes a load with a virtual address
  2. Computer translates virtual address to a physical address
  3. Computer reads data from RAM and returns to the program



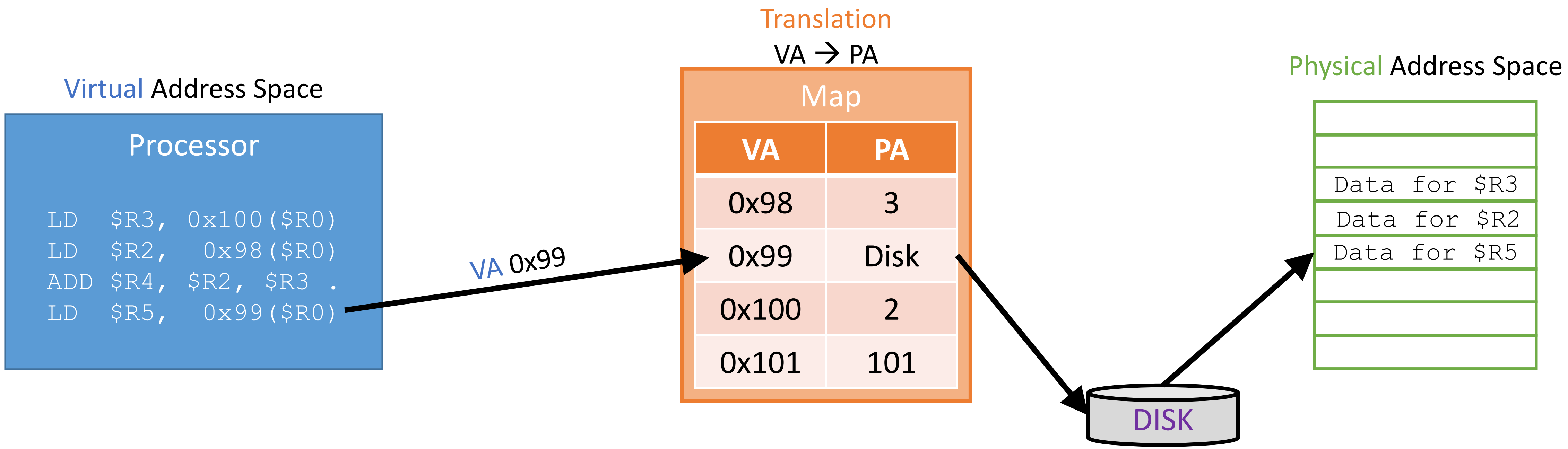
How does Virtual Memory Work?

- How does a program access memory?
    1. Program executes a load with a virtual address
    2. Computer translates virtual address to a physical address
    3. Computer reads data from RAM and returns to the program



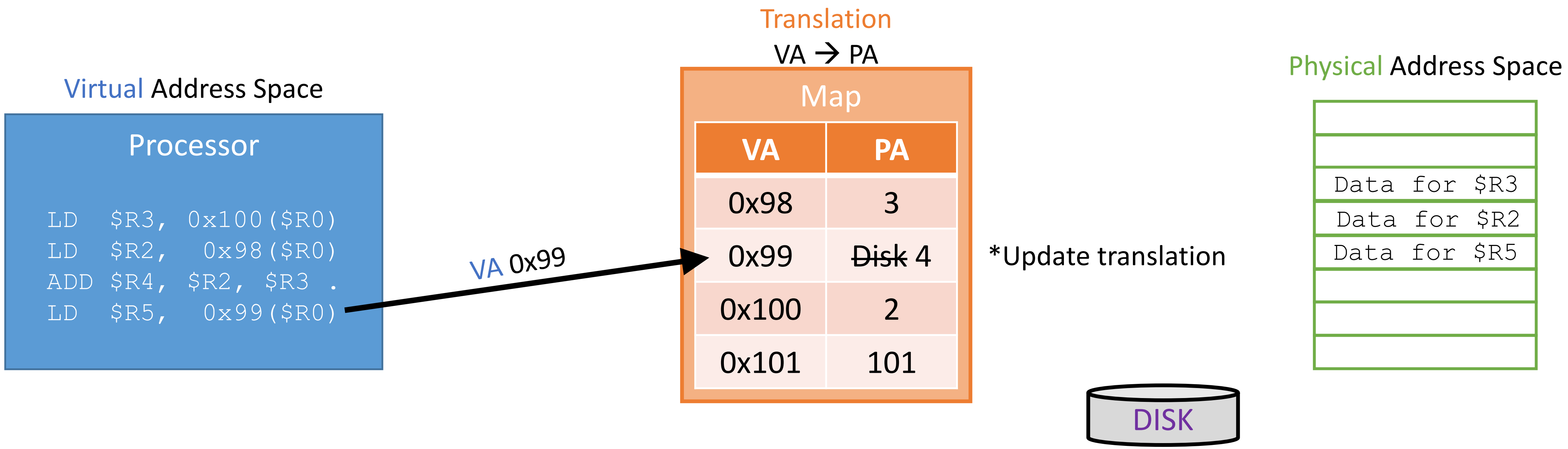
How does Virtual Memory Work?

- How does a program access memory?
  1. Program executes a load with a virtual address
  2. Computer translates virtual address to a physical address
  3. Computer reads data from RAM and returns to the program



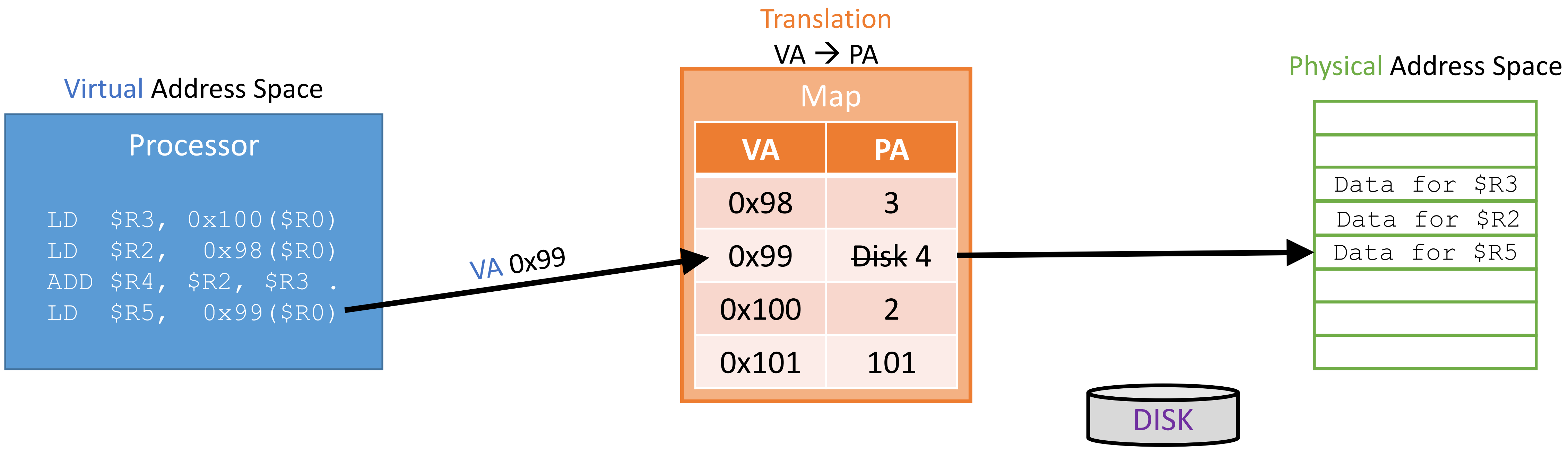
How does Virtual Memory Work?

- How does a program access memory?
  1. Program executes a load with a virtual address
  2. Computer translates virtual address to a physical address
  3. Computer reads data from RAM and returns to the program



How does Virtual Memory Work?

- How does a program access memory?
  1. Program executes a load with a virtual address
  2. Computer translates virtual address to a physical address
  3. Computer reads data from RAM and returns to the program



How does Virtual Memory Work?

- How does a program access memory?
  1. Program executes a load with a virtual address
  2. Computer translates virtual address to a physical address
  3. Computer reads data from RAM and returns to the program

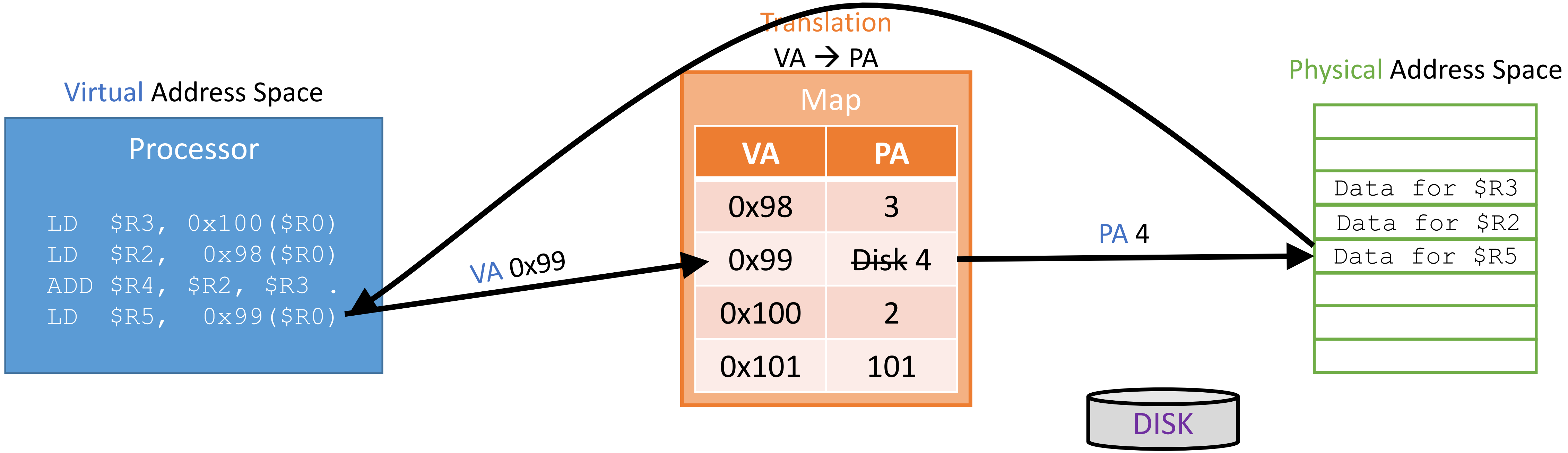


Illustration from the textbook

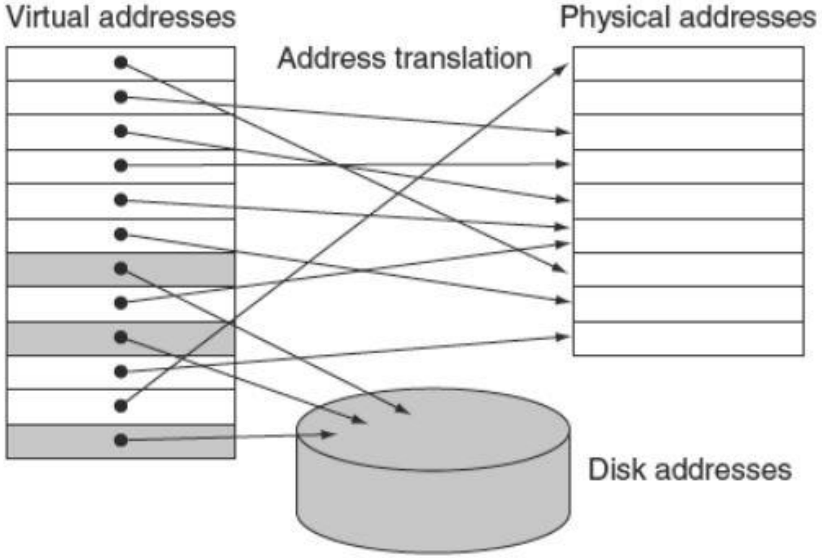


FIGURE 5.25 In virtual memory, blocks of memory (called *pages*) are mapped from one set of addresses (called *virtual addresses*) to another set (called *physical addresses*). The processor generates virtual addresses while the memory is accessed using physical addresses. Both the virtual memory and the physical memory are broken into pages, so that a virtual page is mapped to a physical page. Of course, it is also possible for a virtual page to be absent from main memory and not be mapped to a physical address; in that case, the page resides on disk. Physical pages can be shared by having two virtual addresses point to the same physical address. This capability is used to allow two different programs to share data or code.

Quiz: What address is loaded?

Q: A program issues `LD $R3, 0($R12)` where $R12 holds the value 0x102. What location in RAM is accessed?

- Physical address 0
- Physical address 102
- Not enough information

Quiz: What address is loaded?

Q: A program issues `LD $R3, 0($R12)` where $R12 holds the value 0x102. What location in RAM is accessed?

- Physical address 0
- Physical address 102
- Not enough information

We don't have enough information.

The program wants to access location 0x102 but we need to know the VA to PA mapping.

Quiz: What address is loaded?

Q: A program issues `LD $R3, 0($R12)` where $R12 holds the value 0x102. What location in RAM is accessed?

- Physical address 0
- Physical address 102
- Not enough information

We don't have enough information.

The program wants to access location 0x102 but we need to know the VA to PA mapping.

Virtual Address Space

Processor

```
LD $R3, $R12($R0)
```

Translation

VA → PA

Map

VA	PA
0x98	3
0x99	Disk
0x100	2
0x101	101
0x102	

Physical Address Space

DISK

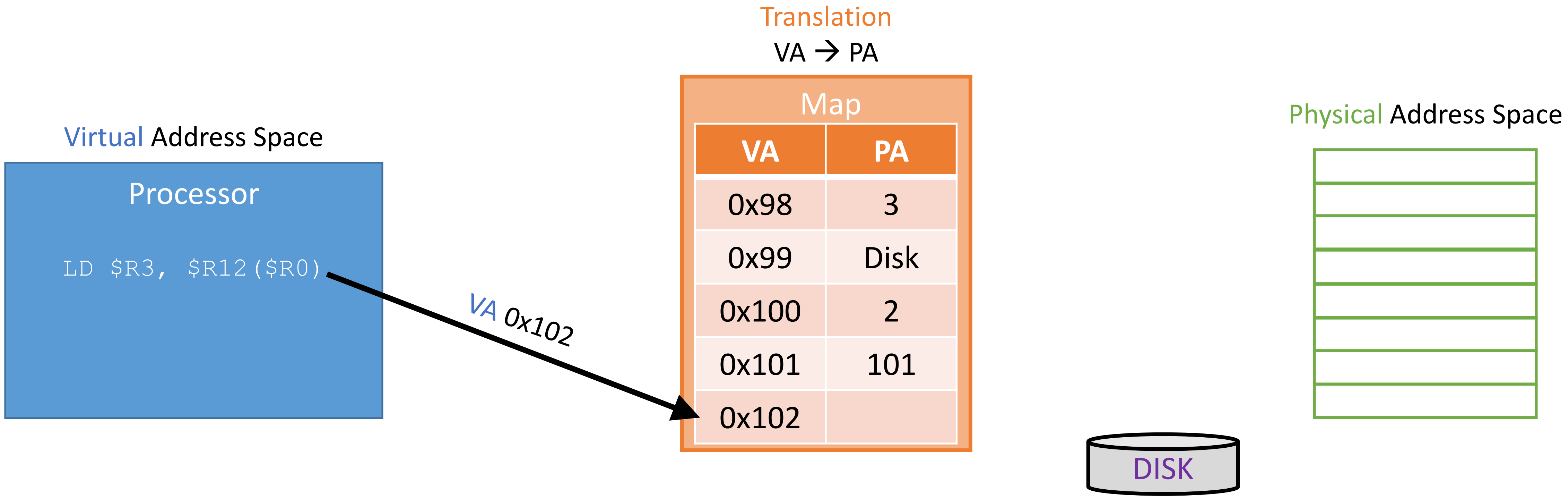
Quiz: What address is loaded?

Q: A program issues `LD $R3, 0($R12)` where $R12 holds the value 0x102. What location in RAM is accessed?

- Physical address 0
- Physical address 102
- Not enough information

We don't have enough information.

The program wants to access location 0x102 but we need to know the VA to PA mapping.



Map

VA	PA
0x98	3
0x99	Disk
0x100	2
0x101	101
0x102	

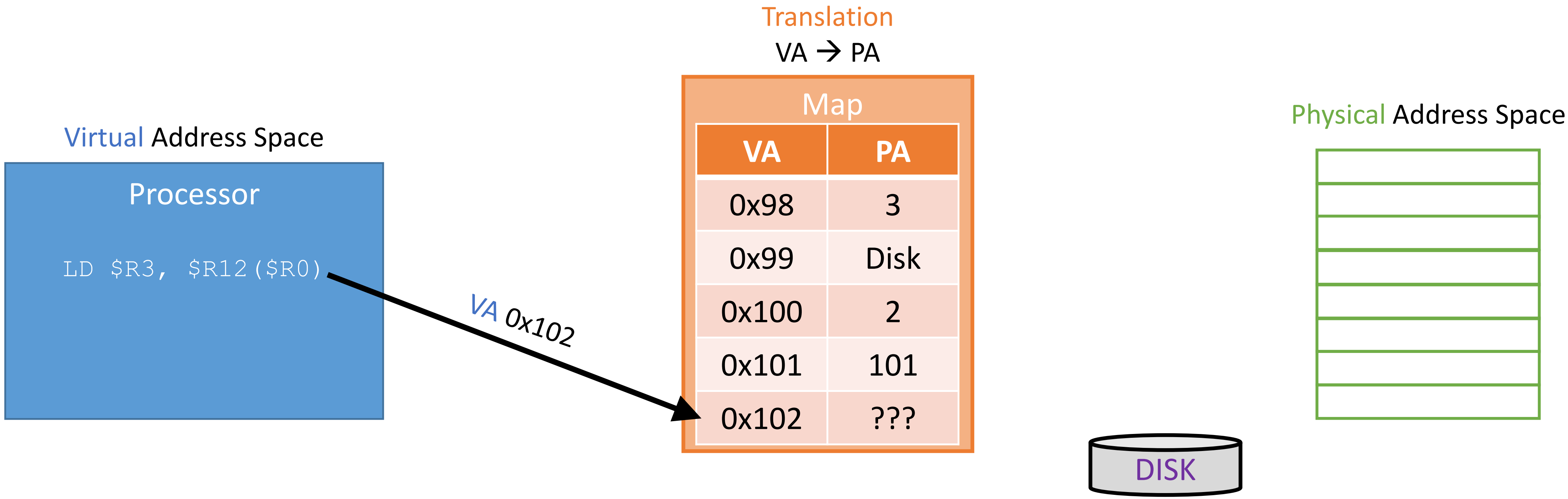
Quiz: What address is loaded?

Q: A program issues `LD $R3, 0($R12)` where $R12 holds the value 0x102. What location in RAM is accessed?

- Physical address 0
- Physical address 102
- Not enough information

We don't have enough information.

The program wants to access location 0x102 but we need to know the VA to PA mapping.



Map

VA	PA
0x98	3
0x99	Disk
0x100	2
0x101	101
0x102	???

Quiz: What address is loaded?

Q: A program issues `LD $R3, 0($R12)` where $R12 holds the value 0x102. What location in RAM is accessed?

- Physical address 0
- Physical address 102
- Not enough information

We don't have enough information.

The program wants to access location 0x102 but we need to know the VA to PA mapping.

Translation

VA → PA

Virtual Address Space

Processor

`LD $R3, $R12($R0)`

VA 0x102

Map

VA	PA
0x98	3
0x99	Disk
0x100	2
0x101	101
0x102	???

PA ???

Physical Address Space

DISK

References

- David Black-Schaffer: Lecture Series on Virtual Memory
- Patterson, Hennessy: Computer Organization and Design: the Hardware/Software Interface