

# KVM on PowerPC

This time it's the server, baby

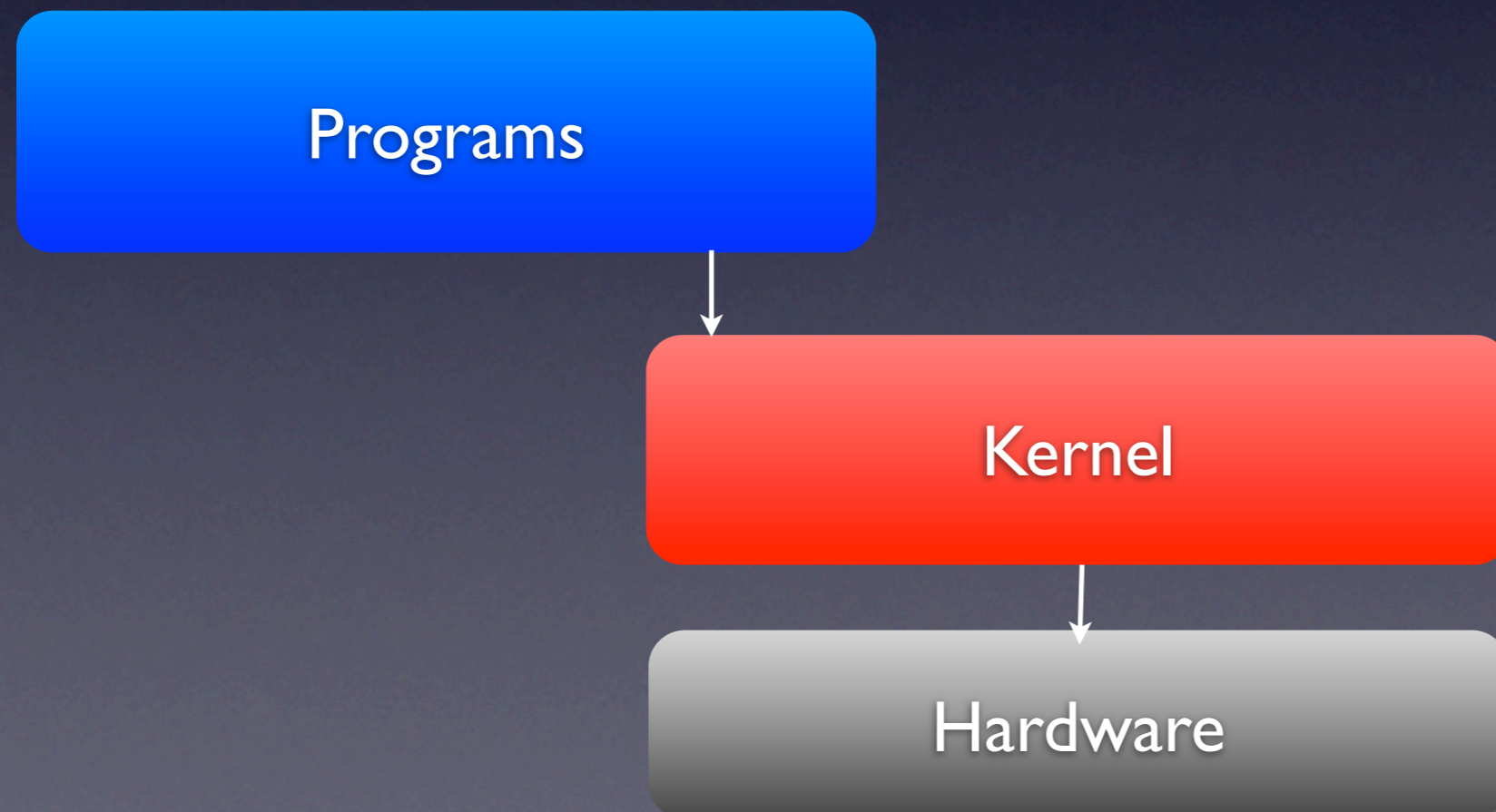
# About Me

- Alexander Graf
- SUSE Studio team
- KVM and Qemu developer
  - Server class PowerPC KVM port
  - S390x Qemu guest support
  - x86 Mac OS X in KVM
  - ...

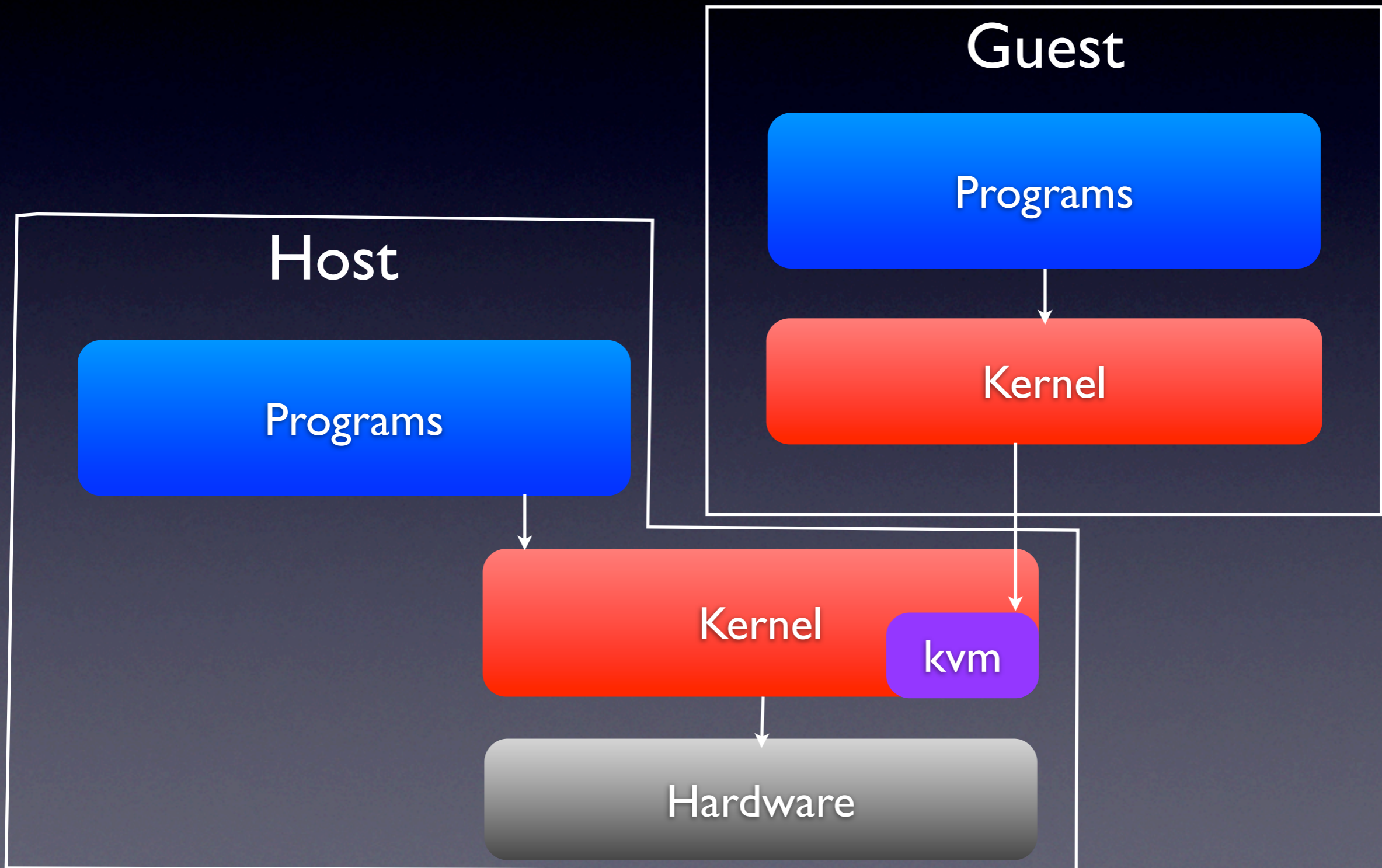
# PowerPC

- BookE alive in embedded market
- Book3S alive in server market & game consoles
- Desktop is dead

# What is KVM?



# What is KVM?



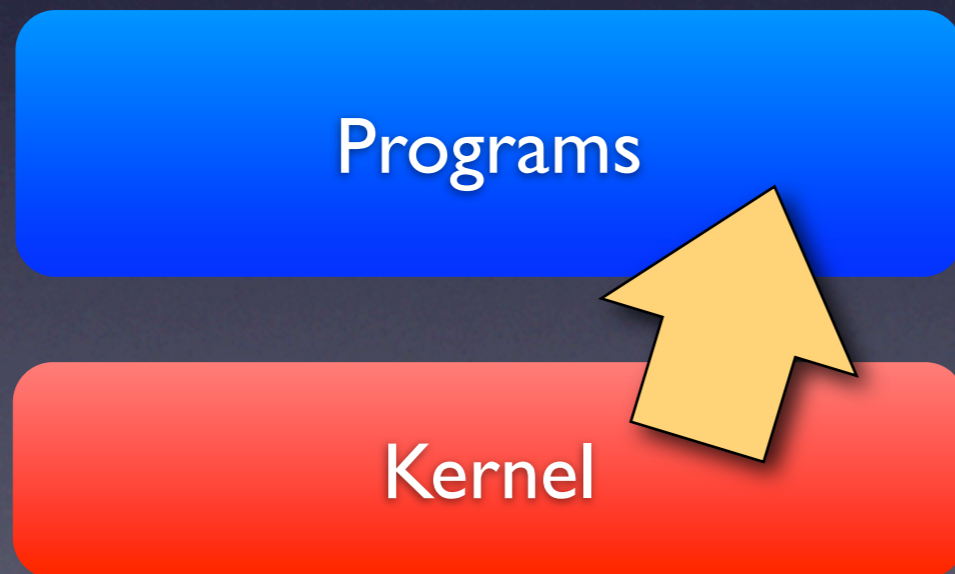
# Virtualization on x86



Programs

Kernel

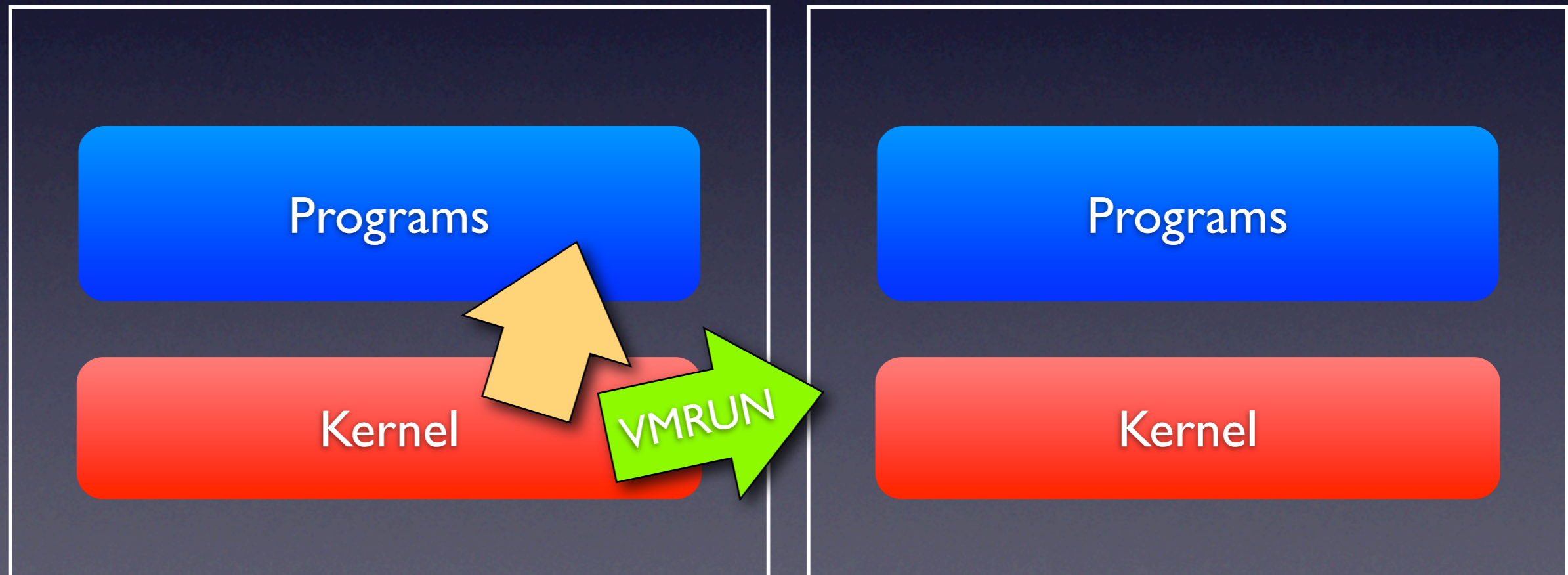
# Virtualization on x86



# Virtualization on x86

Host

Guest





# Virtualization on x86

- New instructions for world switch
- Defined conditions to get back to host

# KVM on PowerPC

# Book3S vs BookE

- Server market / Embedded market
- HTAB MMU / Soft TLB
- Instruction set compatible in PR=1
- Different PR=0 instructions

# The PR=1 Trick

Host

Guest

PR=1

Programs

Programs

PR=0

Kernel

Kernel



# The PR=1 Trick

Host

Guest

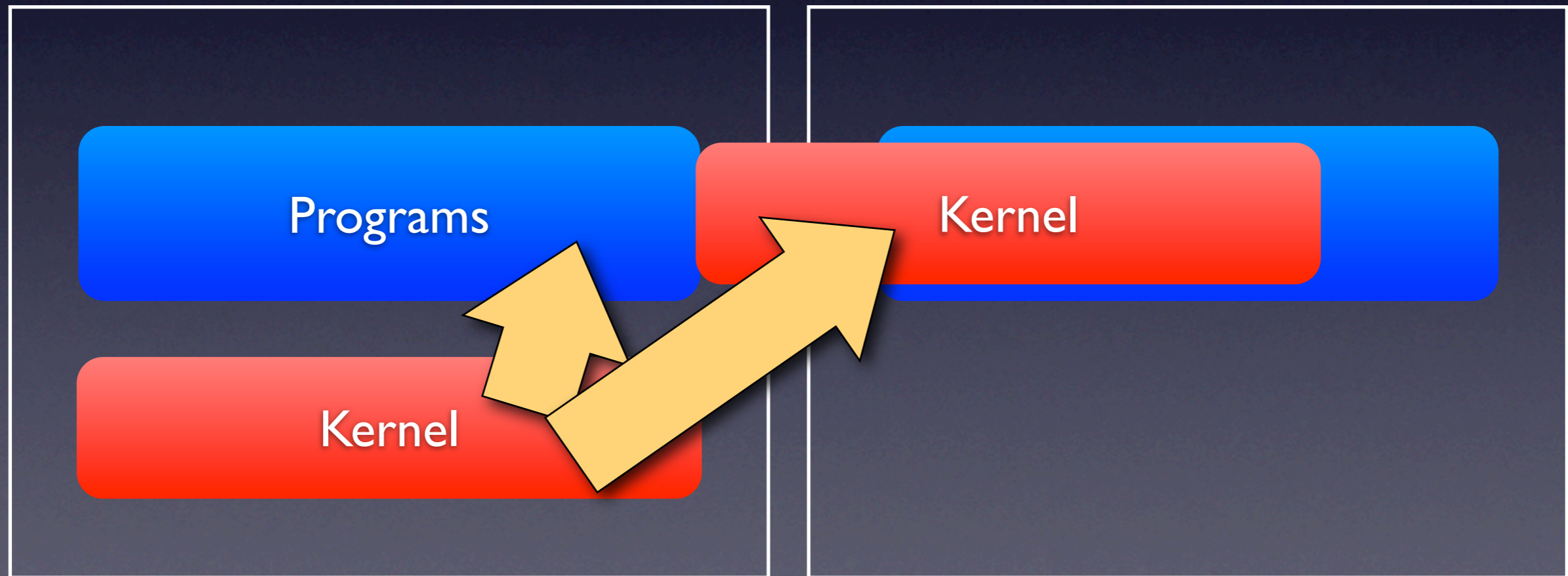
PR=1

Programs

Kernel

PR=0

Kernel



# The PR=I Trick

## Example Linux Kernel Code

```
c000000000000000380:      7c 42 13 78      mr      r2,r2
c000000000000000384:      7d b1 43 a6      mtsprg  1,r13
c000000000000000388:      7d b3 42 a6      mfsprg  r13,3
c00000000000000038c:      f8 6d 01 60      std     r3,352(r13)
c000000000000000390:      7c 73 02 a6      mfdar   r3
c000000000000000394:      f9 2d 01 20      std     r9,288(r13)
c000000000000000398:      7d 20 00 26      mfcr   r9
```

# The PR=I Trick

## Example Linux Kernel Code

```
c000000000000000380:      7c 42 13 78      mr      r2,r2
c000000000000000384:      7d b1 43 a6      mtsprg  1,r13
c000000000000000388:      7d b3 42 a6      mfsprg  r13,3
c00000000000000038c:      f8 6d 01 60      std     r3,352(r13)
c000000000000000390:      7c 73 02 a6      mfdar   r3
c000000000000000394:      f9 2d 01 20      std     r9,288(r13)
c000000000000000398:      7d 20 00 26      mfcr    r9
```

# The PR=I Trick

## Example Linux Kernel Code

```
c0000000000000380:      7c 42 13 78      mr      r2,r2
c0000000000000384:      7d b1 43 a6      mtsprg  1,r13
c0000000000000388:      7d b3 42 a6      mfsprg  r13,3
c000000000000038c:      f8 6d 01 60      std     r3,352(r13)
c0000000000000390:      7c 73 02 a6      mfdar   r3
c0000000000000394:      f9 2d 01 20      std     r9,288(r13)
c0000000000000398:      7d 20 00 26      mfcr    r9
```

kvm



# The PR=I Trick

## Example Linux Kernel Code

```
c0000000000000380:    7c 42 13 78    mr      r2,r2
c0000000000000384:    7d b1 43 a6    mtsprg 1,r13
c0000000000000388:    7d b3 42 a6    mfsprg  r13,3
c000000000000038c:    f8 6d 01 60    std    r3,352(r13)
c0000000000000390:    7c 73 02 a6    mfdar  r3
c0000000000000394:    f9 2d 01 20    std    r9,288(r13)
c0000000000000398:    7d 20 00 26    mfcr   r9
```



kvm

# The PR=I Trick

## Example Linux Kernel Code

```
c0000000000000380:      7c 42 13 78      mr      r2,r2
c0000000000000384:      7d b1 43 a6      mtsprg  1,r13
c0000000000000388:      7d b3 42 a6      mfsprg  r13,3
c000000000000038c:      f8 6d 01 60      std     r3,352(r13)
c0000000000000390:      7c 73 02 a6      mfdar   r3
c0000000000000394:      f9 2d 01 20      std     r9,288(r13)
c0000000000000398:      7d 20 00 26      mfcr    r9
```

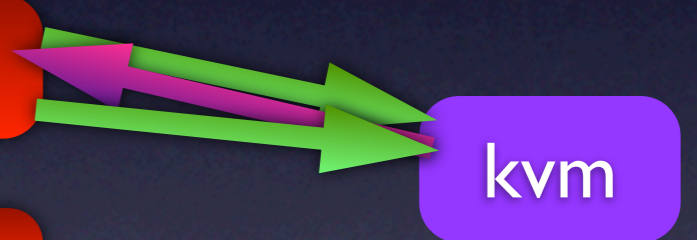


kvm

# The PR=I Trick

## Example Linux Kernel Code

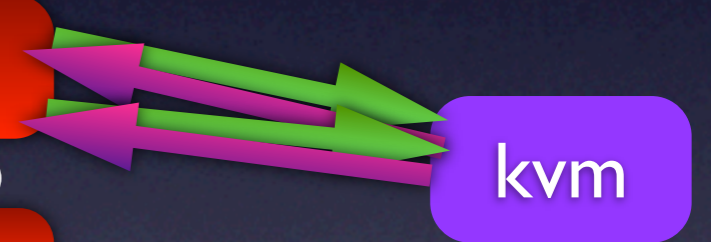
```
c0000000000000380:    7c 42 13 78    mr      r2,r2
c0000000000000384:    7d b1 43 a6    mtsprg 1,r13
c0000000000000388:    7d b3 42 a6    mfsprg  r13,3
c000000000000038c:    f8 6d 01 60    std     r3,352(r13)
c0000000000000390:    7c 73 02 a6    mfdar  r3
c0000000000000394:    f9 2d 01 20    std     r9,288(r13)
c0000000000000398:    7d 20 00 26    mfcr   r9
```



# The PR=I Trick

## Example Linux Kernel Code

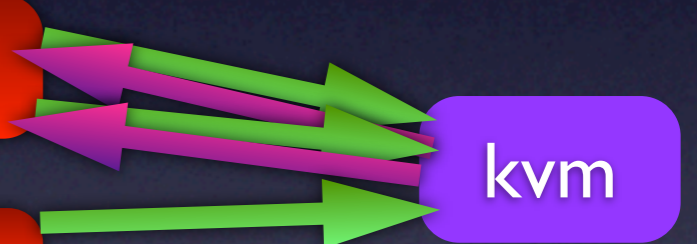
```
c0000000000000380:      7c 42 13 78      mr      r2,r2
c0000000000000384:      7d b1 43 a6      mtsprg  1,r13
c0000000000000388:      7d b3 42 a6      mfsprg  r13,3
c000000000000038c:      f8 6d 01 60      std     r3,352(r13)
c0000000000000390:      7c 73 02 a6      mfdar   r3
c0000000000000394:      f9 2d 01 20      std     r9,288(r13)
c0000000000000398:      7d 20 00 26      mfcr    r9
```



# The PR=I Trick

## Example Linux Kernel Code

```
c0000000000000380:    7c 42 13 78    mr      r2,r2
c0000000000000384:    7d b1 43 a6    mtsprg 1,r13
c0000000000000388:    7d b3 42 a6    mfsprg  r13,3
c000000000000038c:    f8 6d 01 60    std    r3,352(r13)
c0000000000000390:    7c 73 02 a6    mfdar  r3
c0000000000000394:    f9 2d 01 20    std    r9,288(r13)
c0000000000000398:    7d 20 00 26    mfcr   r9
```



# The PR=I Trick

## Example Linux Kernel Code

```
c0000000000000380:      7c 42 13 78      mr      r2,r2
c0000000000000384:      7d b1 43 a6      mtsprg  1,r13
c0000000000000388:      7d b3 42 a6      mfsprg  r13,3
c000000000000038c:      f8 6d 01 60      std     r3,352(r13)
c0000000000000390:      7c 73 02 a6      mfdar   r3
c0000000000000394:      f9 2d 01 20      std     r9,288(r13)
c0000000000000398:      7d 20 00 26      mfcr    r9
```



# The PR=I Trick

- All privileged instructions trap
- KVM can emulate them
- Runs all kernel code in user mode

# Virtual Memory

Virtual  
Memory



Physical  
Memory





# Virtual Memory

Virtual  
Memory



Physical  
Memory



# Virtual Memory

Virtual  
Memory



Physical  
Memory



# Virtual Memory

Effective Address

c00000012000380

Virtual Address

Real Address

# Virtual Memory

Effective Address

c0000001 2000380



Virtual Address

Real Address

# Segment Lookaside Buffer

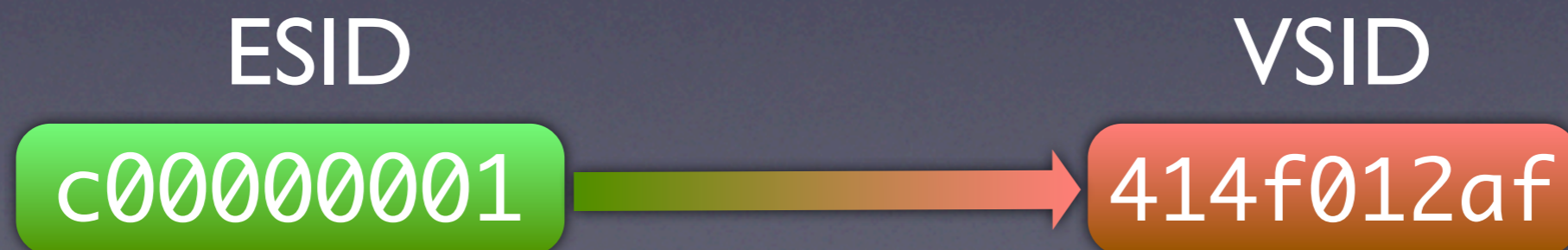
ESID	VSID
c00000000	408f92c94
d00000000	f09b89af5
c00000007	45cb97751
0	ef72a166e
f	faaa19203
c00000001	414f012af
...	...

# Segment Lookaside Buffer

ESID	VSID
c00000000	408f92c94
d00000000	f09b89af5
c00000007	45cb97751
0	ef72a166e
f	faaa19203
<b>c00000001</b>	<b>414f012af</b>
...	...

# Segment Lookaside Buffer

ESID	VSID
c00000000	408f92c94
d00000000	f09b89af5
c00000007	45cb97751
0	ef72a166e
f	faaa19203
<b>c00000001</b>	<b>414f012af</b>
...	...



# Virtual Memory

Effective Address

c00000012000380



Virtual Address

Real Address



# Virtual Memory

Effective Address

c0000001 2000380



Virtual Address

414f012af 2000380

Real Address

# Virtual Memory

Effective Address

c00000012000380

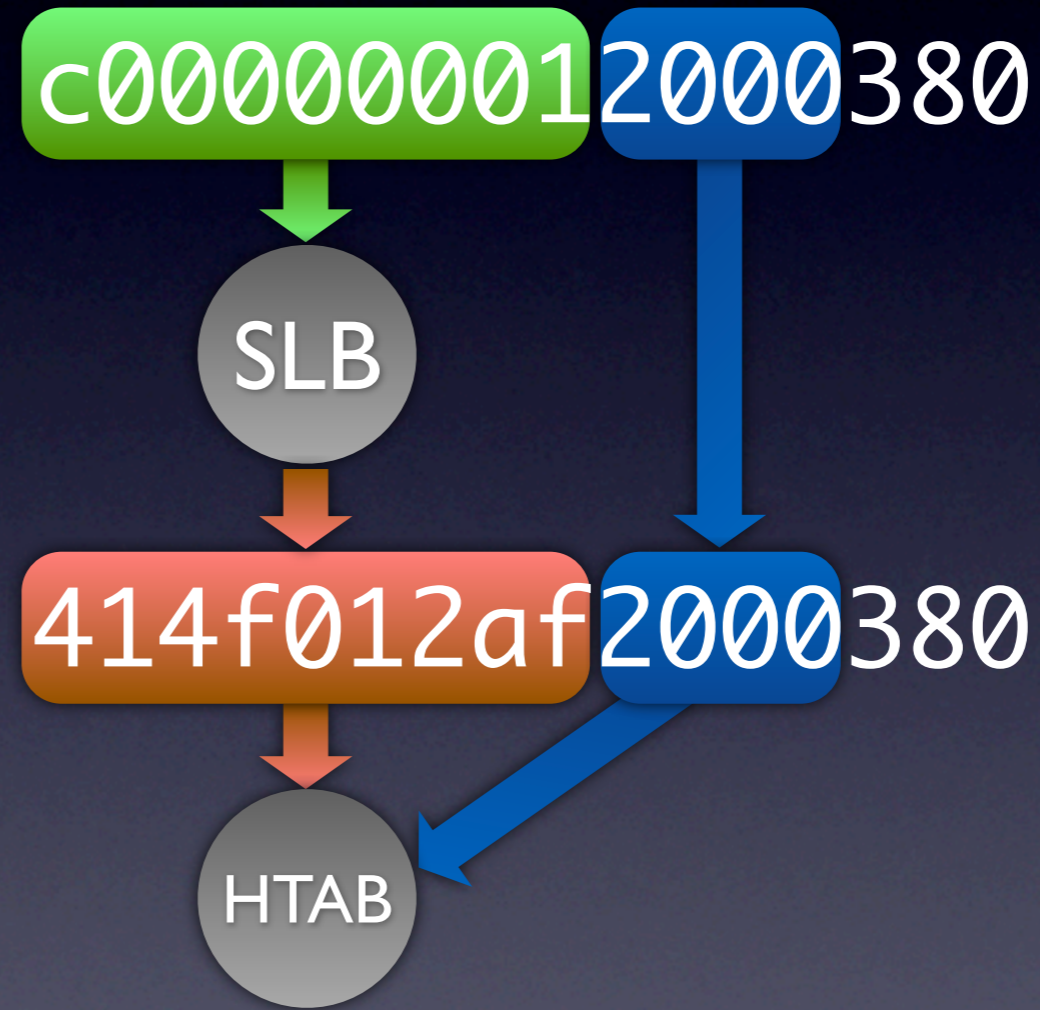
SLB

Virtual Address

414f012af2000380

HTAB

Real Address



# HTAB

414f012af

2000

380



Index into HTAB

HTAB



# HTAB

414f012af 2000 380

PTEG

VSID	Real Page
408f92c94	5000
414f012af	12000
...	...

# HTAB

414f012af 2000 380

PTEG

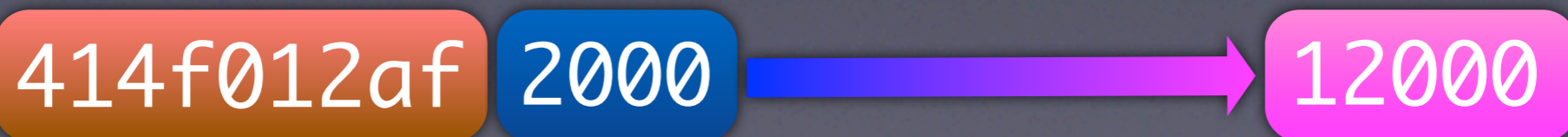
VSID	Real Page
408f92c94	5000
414f012af	12000
...	...

# HTAB

414f012af 2000 380

PTEG

VSID	Real Page
408f92c94	5000
414f012af	12000
...	...



# Virtual Memory

Effective Address

c00000012000380

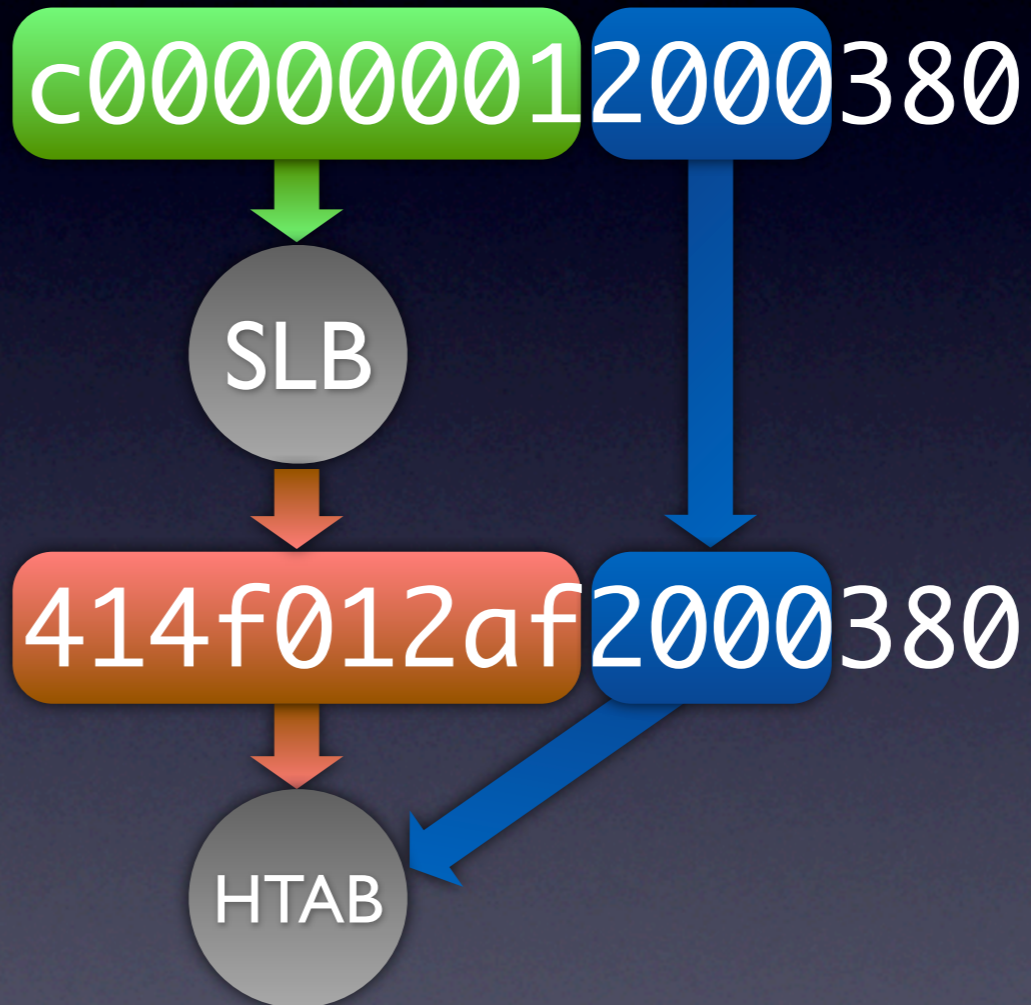
SLB

Virtual Address

414f012af2000380

HTAB

Real Address



# Virtual Memory

Effective Address

c000000012000380

SLB

Virtual Address

414f012af2000380

HTAB

Real Address

0000000012000380



# Shadow Paging (Segments)

- Each process gets 65535 VSIDs allocated
- Allocate a new process context



# Shadow Paging (Segments)

- Then map guest VSIDs to process VSIDs

Guest VSID	Host VSID
414f012af	123450001
408f92c94	123450002
adc7d4c2d	123450003

# Shadow Paging (Pages)

- Guest accesses an unmapped page
- Read guest SLB & HTAB
- Resolve Host Real Address
- Convert VSID to Shadow VSID
- Call Linux MMU helper to add host PTE

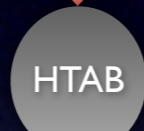
Guest Effective  
Address

c000000012000380



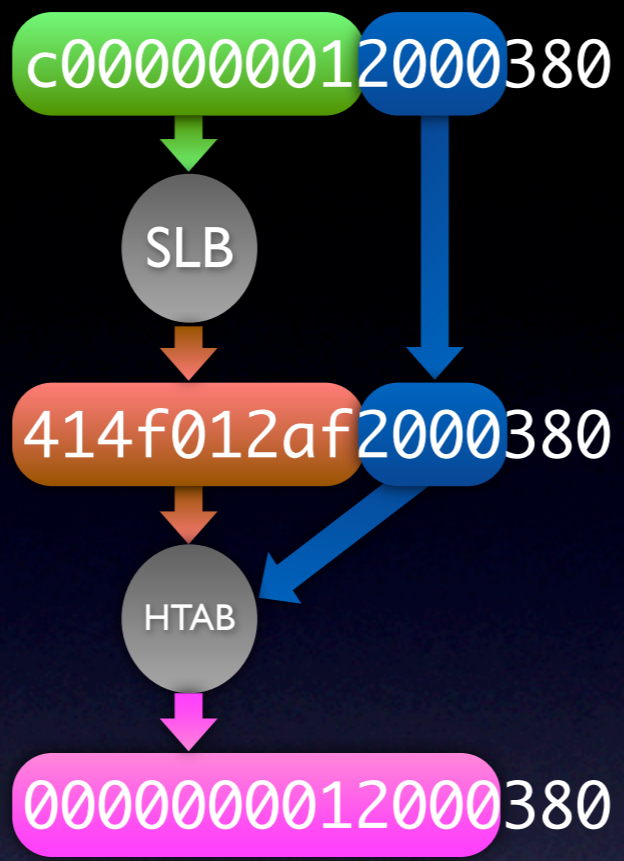
Guest Virtual  
Address

414f012af2000380



Guest Real  
Address

0000000012000380



Guest Effective Address

c000000012000380



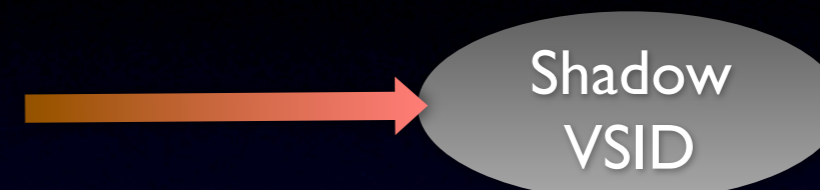
Guest Virtual Address

414f012af2000380



Guest Real Address

0000000012000380



1234500012000380

Guest Effective Address

c000000012000380

Guest Virtual Address

414f012af2000380

Guest Real Address

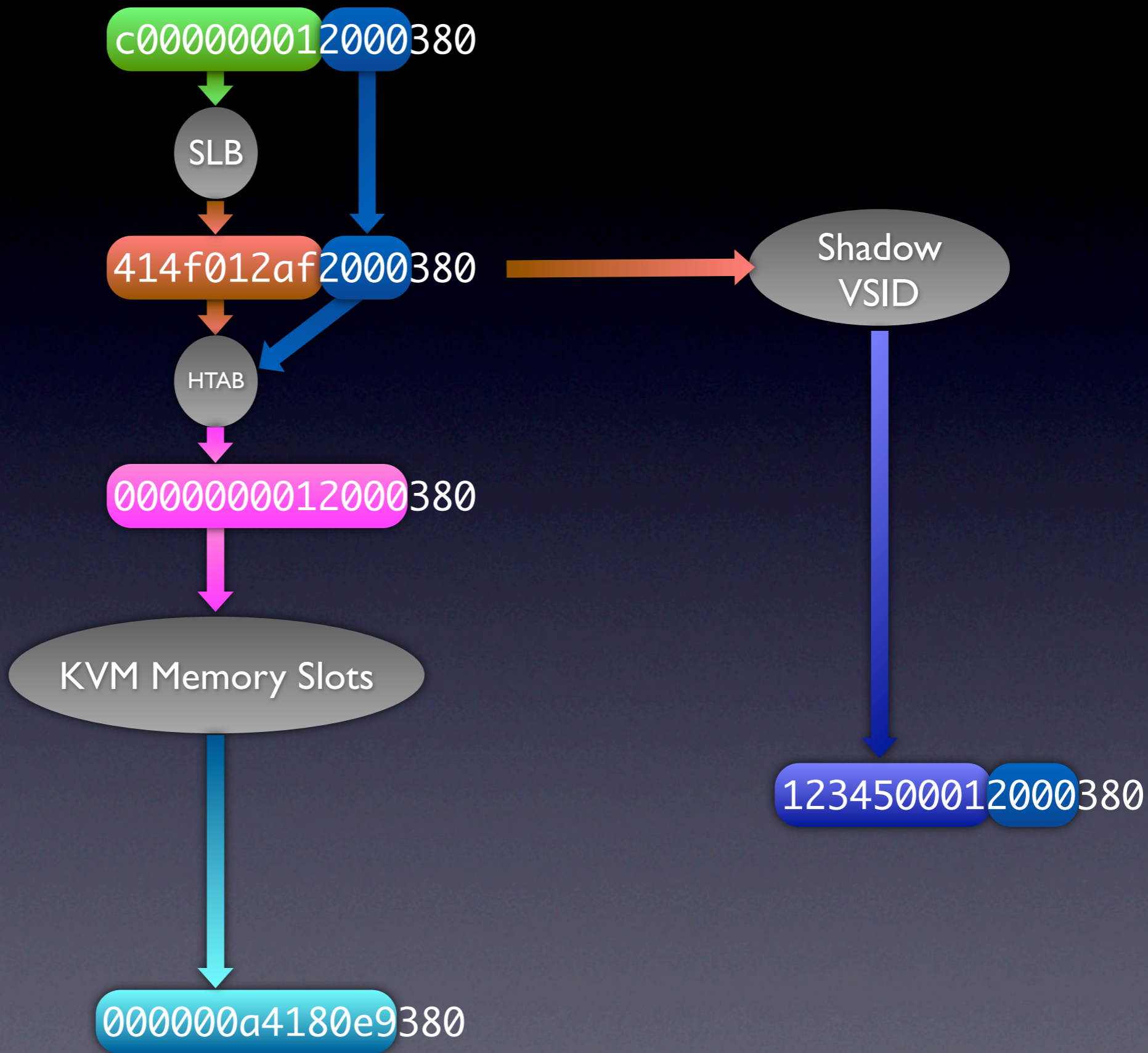
0000000012000380

Shadow Virtual Address

1234500012000380

Host Real Address

000000a4180e9380



Guest Effective Address

c000000012000380

Guest Virtual Address

414f012af2000380

Guest Real Address

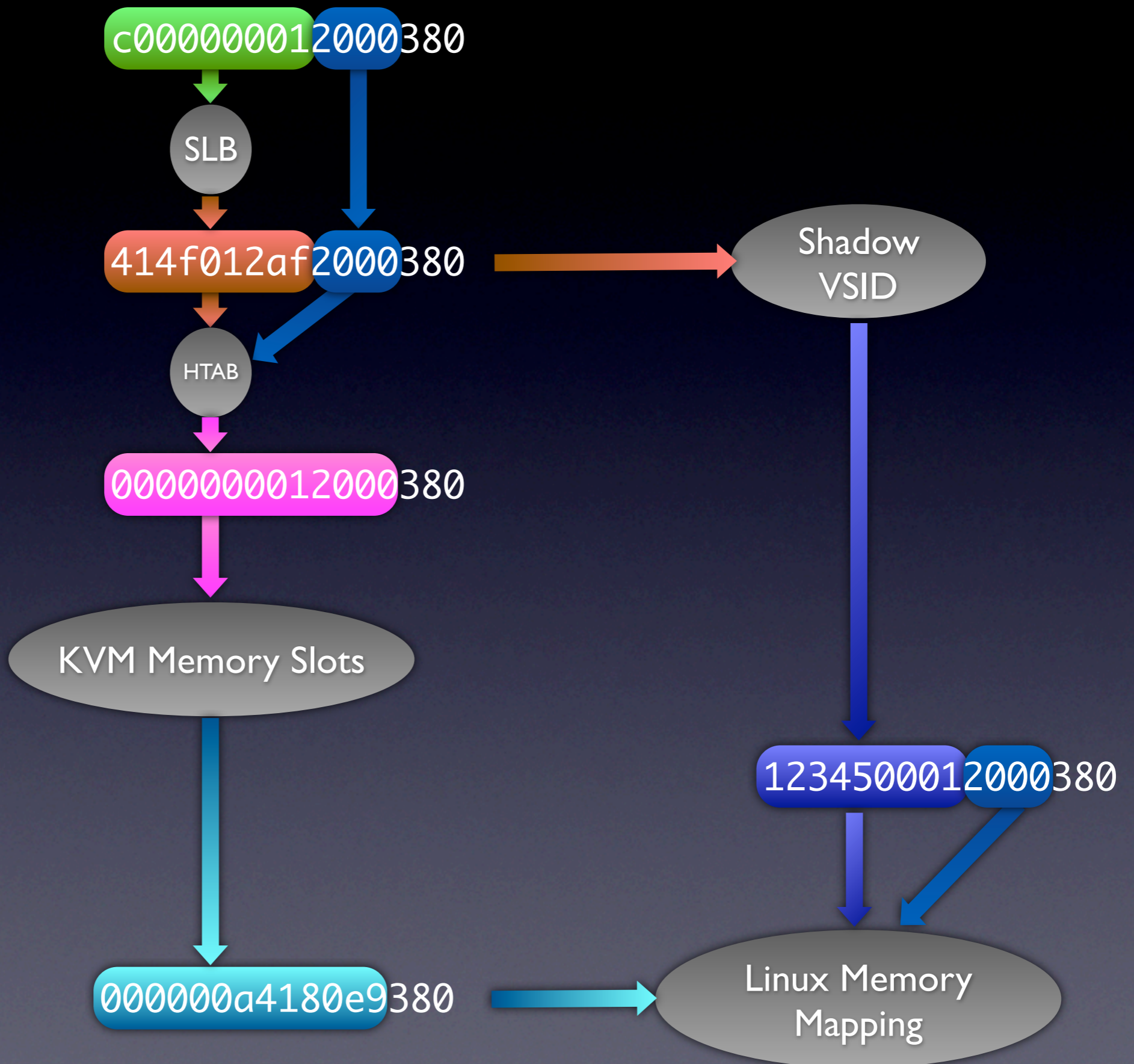
0000000012000380

Shadow Virtual Address

1234500012000380

Host Real Address

000000a4180e9380



# Guest Entry

## Virtual CPU

### CPU Registers

r0 - r31

lr

cr

...

### SLB Entries

ESID	VSID
c00000000	123450042
d00000000	123450002
c00000007	123450003
c00000001	123450001
...	...

## CPU



# Guest Entry

## Virtual CPU

### CPU Registers

r0 - r31

lr

cr

...

### SLB Entries

ESID	VSID
c00000000	123450042
d00000000	123450002
c00000007	123450003
c00000001	123450001
...	...

## CPU

### CPU Registers

r0 - r31

lr

cr

...

### SLB Entries

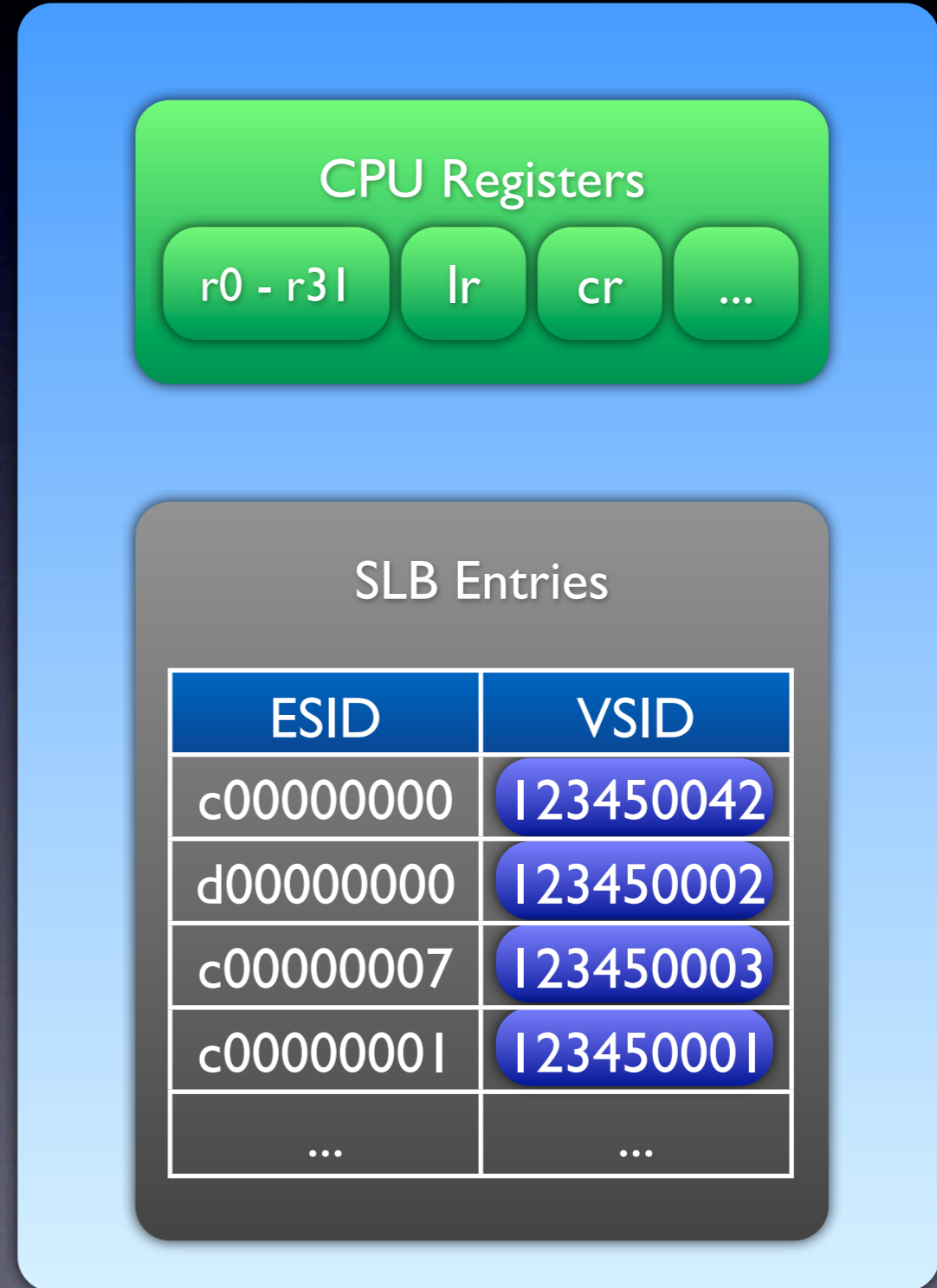
ESID	VSID
c00000000	123450042
d00000000	123450002
c00000007	123450003
c00000001	123450001
...	...

# Guest Entry

Virtual CPU



CPU



# Guest Entry

## Virtual CPU

### CPU Registers

r0 - r31

lr

cr

...

## CPU

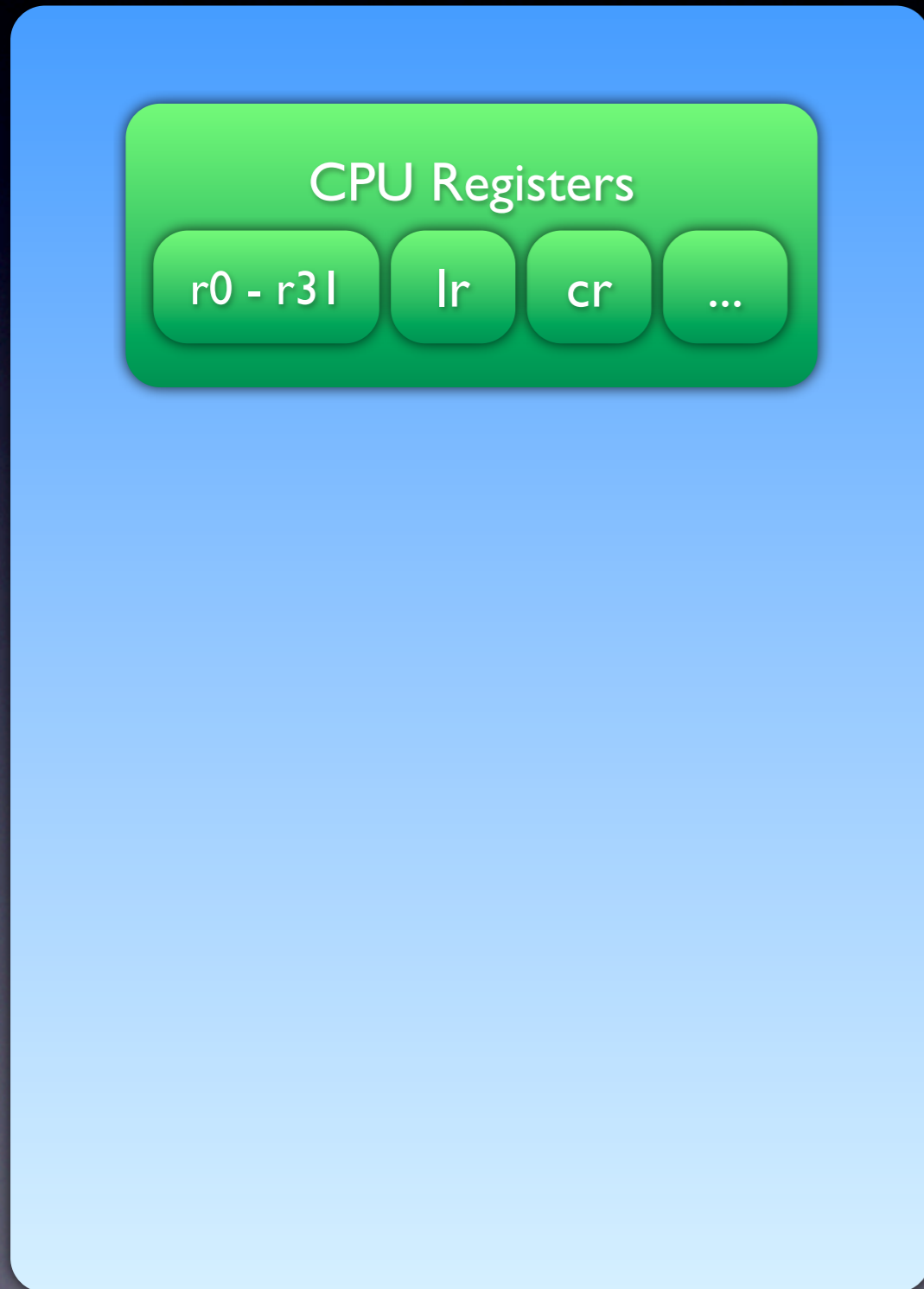
### SLB Entries

ESID	VSID
c00000000	123450042
d00000000	123450002
c00000007	123450003
c00000001	123450001
...	...

# Guest Entry

Virtual CPU

CPU



# Mac-on-Linux

- Used to run Mac OS on PPC back in the day
- PPC32 hosts only
- Now has KVM interface

# My Why

- Build Service
- SUSE Studio

# Questions