# 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
- Nested SVM
- ...

# BookE

# Book3S

# Book3S

# Book3S



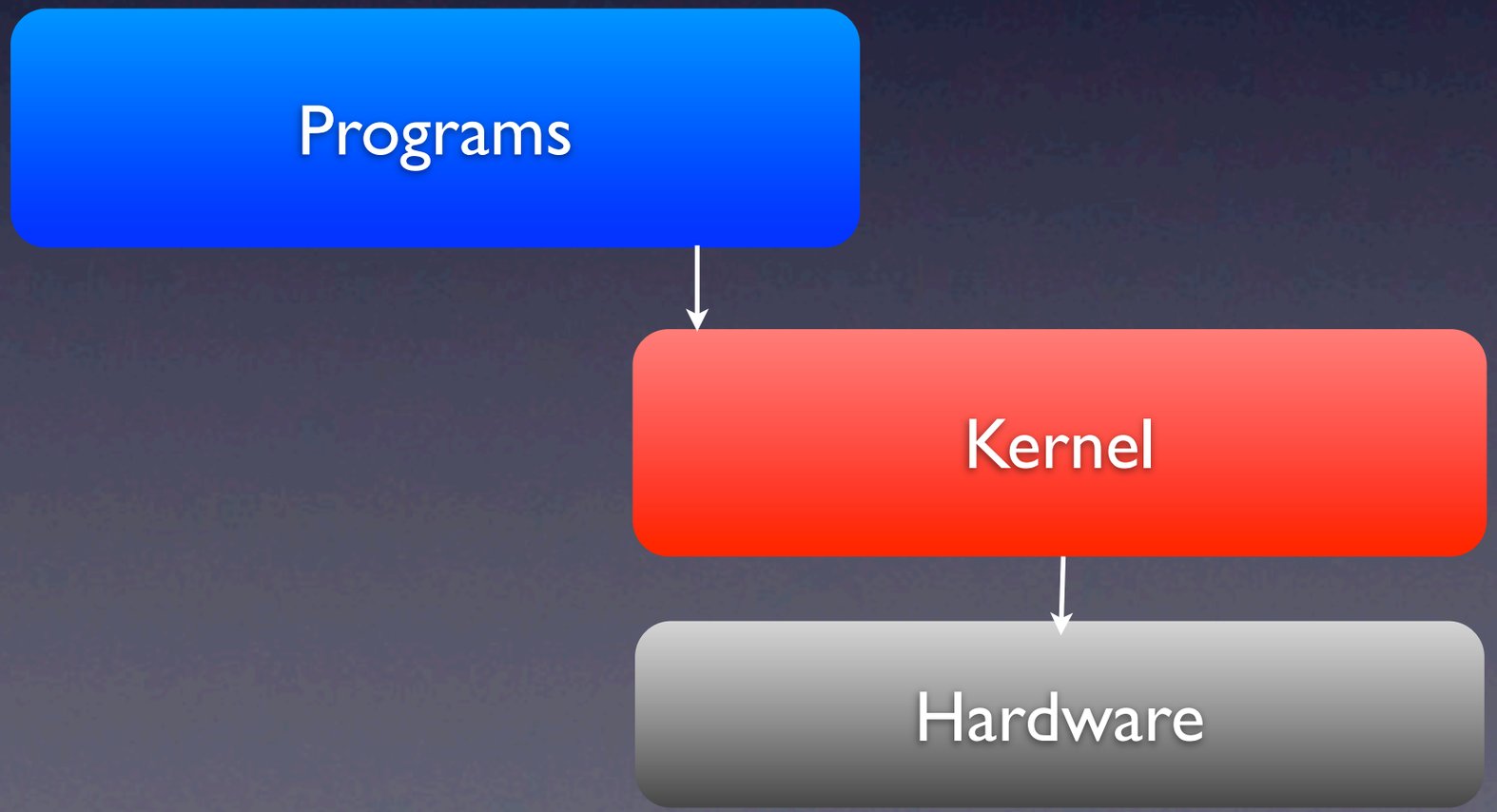

# PowerPC

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

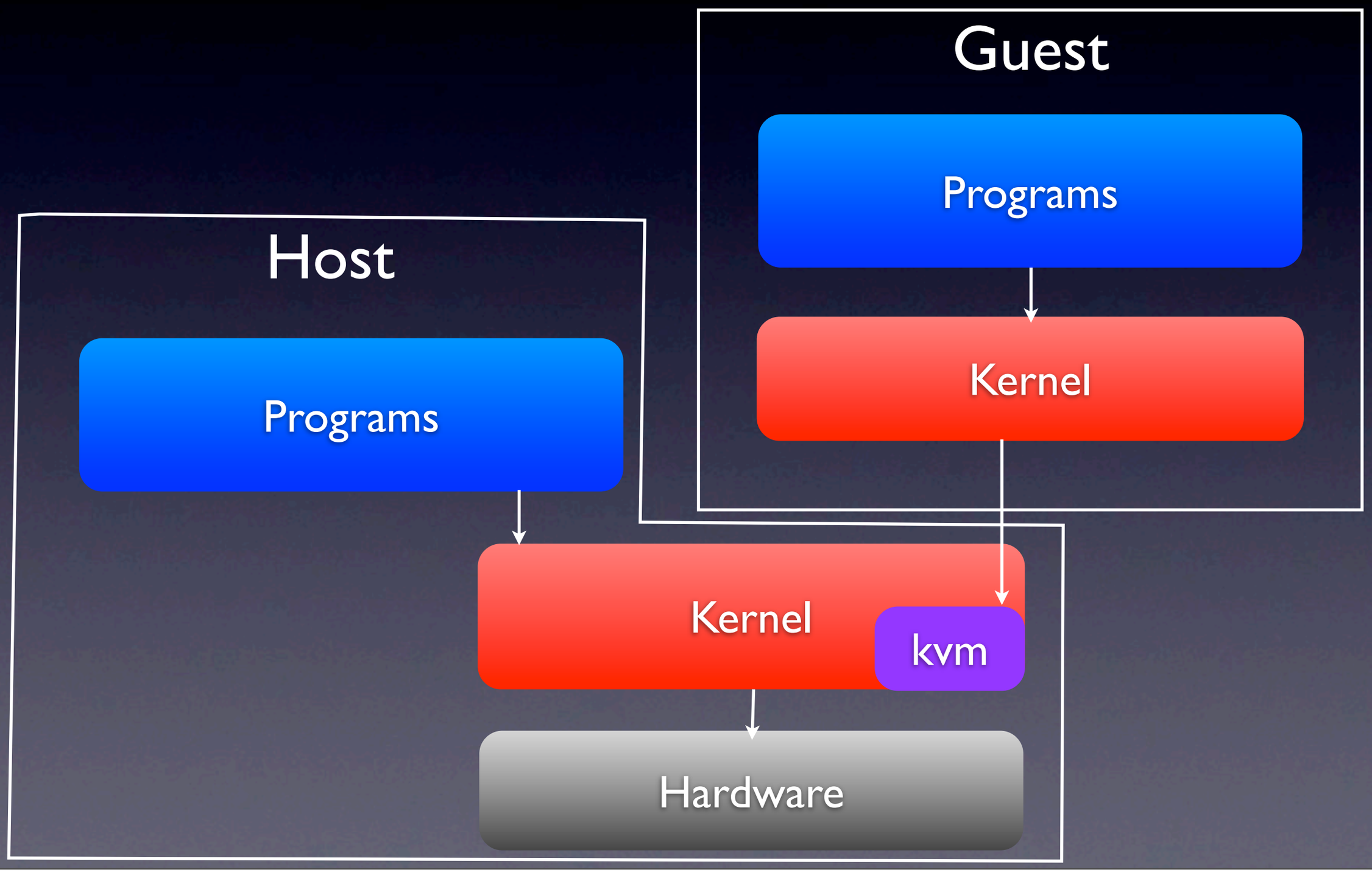
# Book3S vs BookE

- Server market / Embedded market
- HTAB MMU / Soft TLB
- Instruction set compatible in user mode
- Different kernel mode instructions

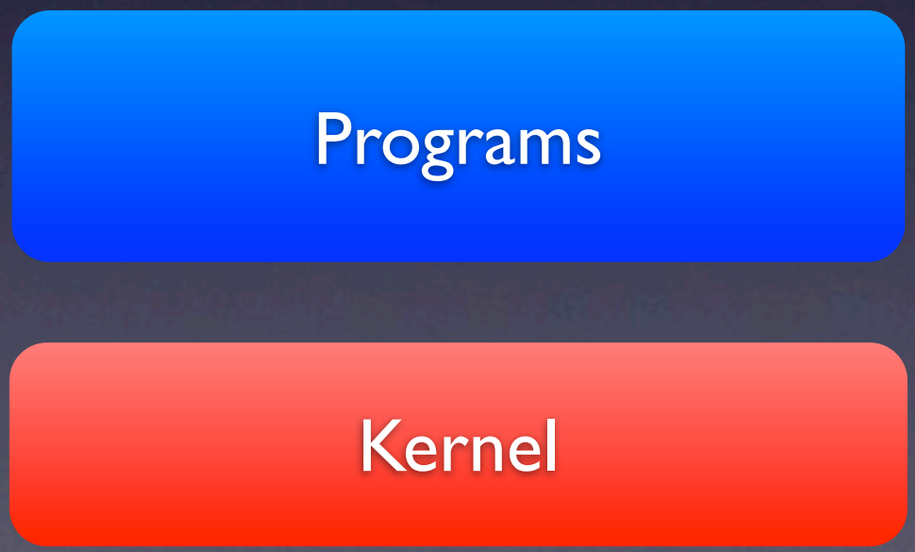
# What is KVM?



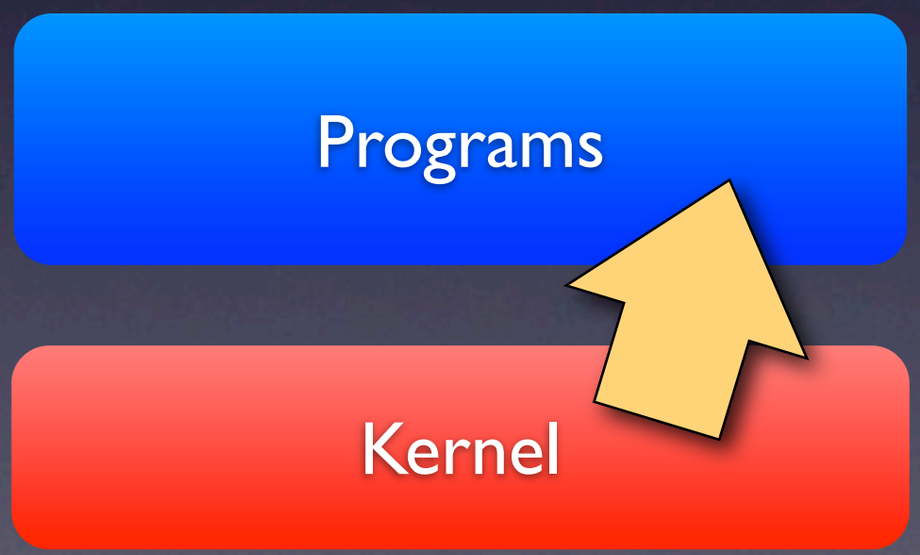
# What is KVM?



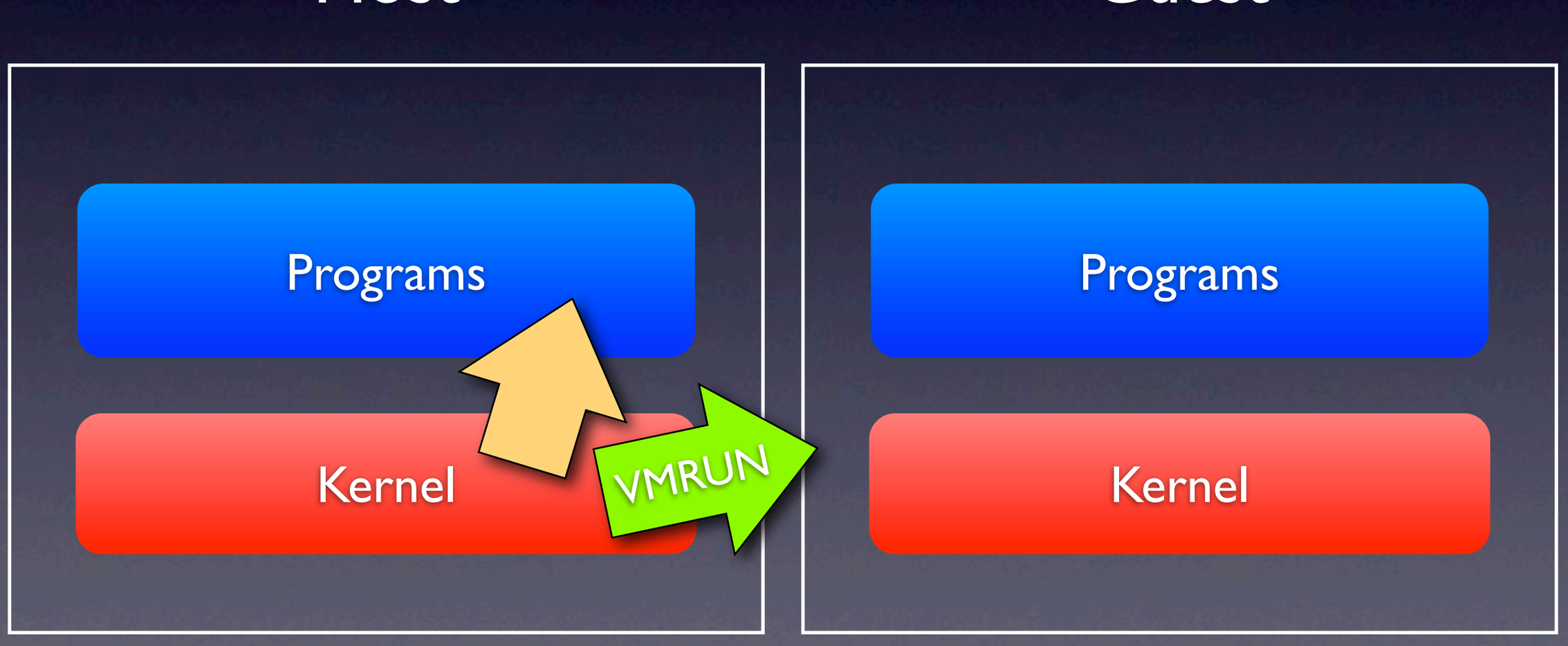
# Virtualization on x86



# Virtualization on x86



# Virtualization on x86

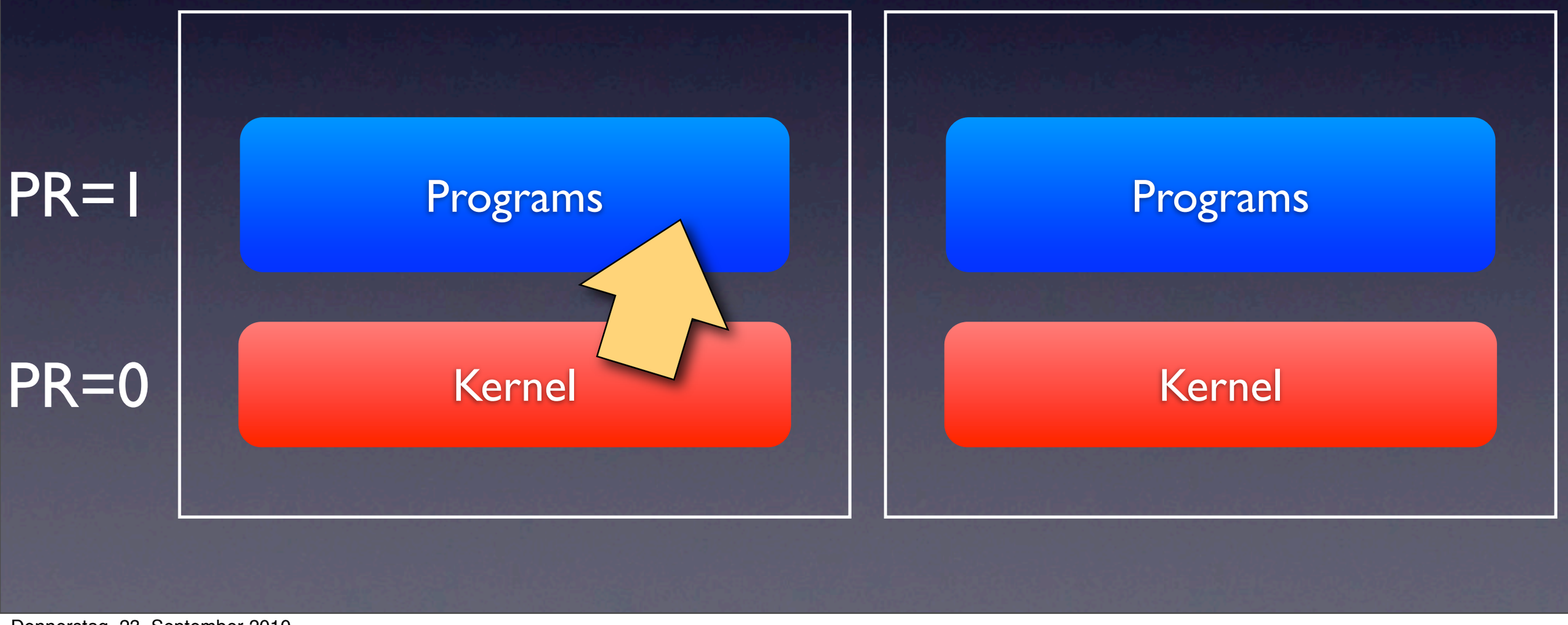


# Virtualization on x86

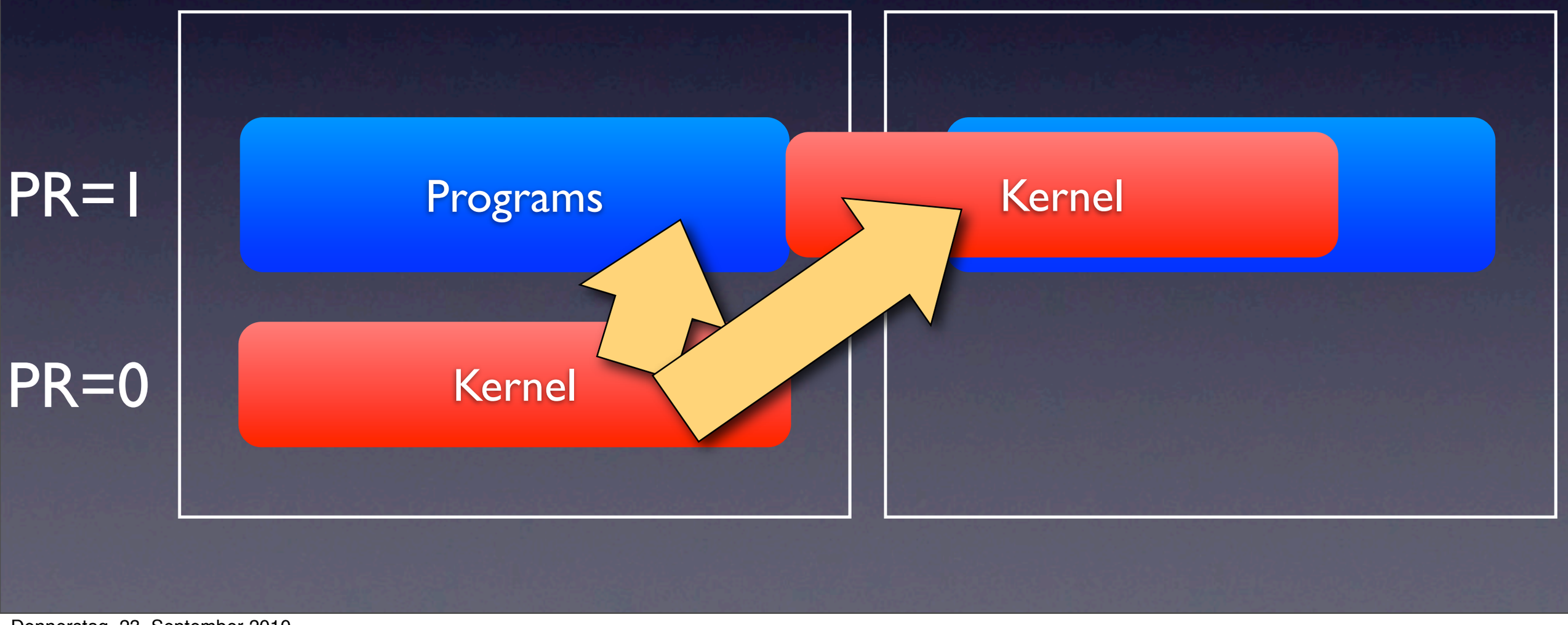
- New instructions for world switch
- Defined conditions to get back to host

# KVM on PowerPC

# The PR=1 Trick



# The PR=1 Trick



# The PR=1 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=1 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=1 Trick

## Example Linux Kernel Code

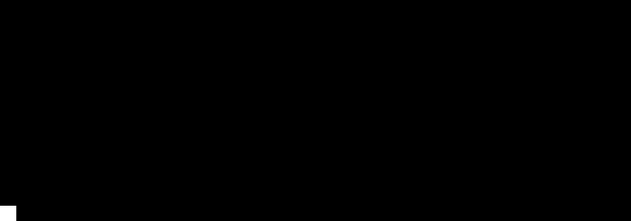
```
c000000000000380:       7c 42 13 78     mr      r2,r2
c000000000000384:       7d b1 43 a6     mtsprg  1,r13
c000000000000388:       7d b3 42 a6     mfsprg  r13,3
c00000000000038c:       f8 6d 01 60     std     r3,352(r13)
c000000000000390:       7c 73 02 a6     mfdar   r3
c000000000000394:       f9 2d 01 20     std     r9,288(r13)
c000000000000398:       7d 20 00 26     mfcr    r9
```

kvm

# The PR=1 Trick

## Example Linux Kernel Code

```
c000000000000380:       7c 42 13 78     mr      r2,r2
c000000000000384:       7d b1 43 a6     mtsprg  1,r13
c000000000000388:       7d b3 42 a6     mfsprg  r13,3
c00000000000038c:       f8 6d 01 60     std     r3,352(r13)
c000000000000390:       7c 73 02 a6     mfdar   r3
c000000000000394:       f9 2d 01 20     std     r9,288(r13)
c000000000000398:       7d 20 00 26     mfcr    r9
```



# The PR=1 Trick

## Example Linux Kernel Code

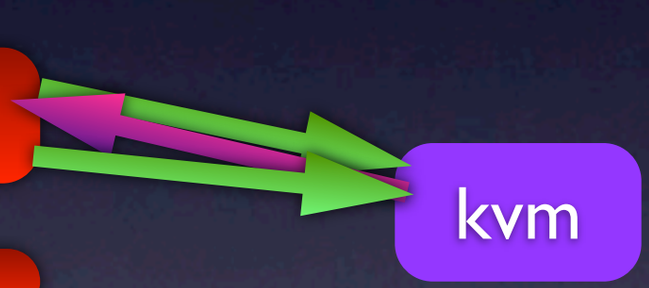
```
c000000000000380:       7c 42 13 78     mr      r2,r2
c000000000000384:       7d b1 43 a6     mtsprg  1,r13
c000000000000388:       7d b3 42 a6     mfsprg  r13,3
c00000000000038c:       f8 6d 01 60     std     r3,352(r13)
c000000000000390:       7c 73 02 a6     mfdar   r3
c000000000000394:       f9 2d 01 20     std     r9,288(r13)
c000000000000398:       7d 20 00 26     mfcr    r9
```

kvm

# The PR=1 Trick

## Example Linux Kernel Code

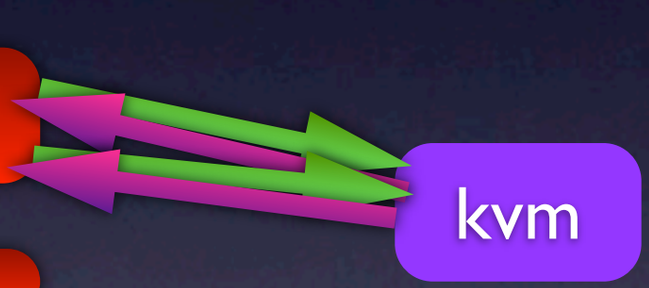
```
c000000000000380:       7c 42 13 78     mr      r2,r2
c000000000000384:       7d b1 43 a6     mtsprg  1,r13
c000000000000388:       7d b3 42 a6     mfsprg  r13,3
c00000000000038c:       f8 6d 01 60     std     r3,352(r13)
c000000000000390:       7c 73 02 a6     mfdar   r3
c000000000000394:       f9 2d 01 20     std     r9,288(r13)
c000000000000398:       7d 20 00 26     mfcr    r9
```



# The PR=1 Trick

## Example Linux Kernel Code

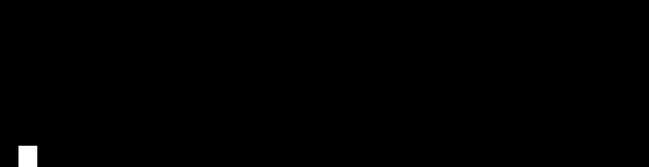
```
c000000000000380:       7c 42 13 78     mr      r2,r2
c000000000000384:       7d b1 43 a6     mtsprg  1,r13
c000000000000388:       7d b3 42 a6     mfsprg  r13,3
c00000000000038c:       f8 6d 01 60     std     r3,352(r13)
c000000000000390:       7c 73 02 a6     mfdar   r3
c000000000000394:       f9 2d 01 20     std     r9,288(r13)
c000000000000398:       7d 20 00 26     mfcr    r9
```



# The PR=1 Trick

## Example Linux Kernel Code

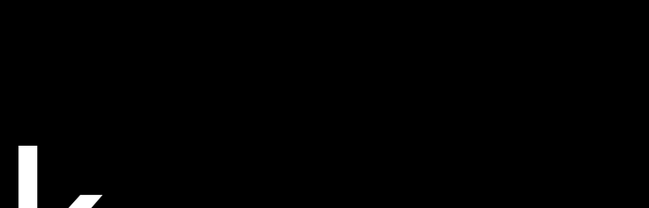
```
c000000000000380:       7c 42 13 78     mr      r2,r2
c000000000000384:       7d b1 43 a6     mtsprg  1,r13
c000000000000388:       7d b3 42 a6     mfsprg  r13,3
c00000000000038c:       f8 6d 01 60     std     r3,352(r13)
c000000000000390:       7c 73 02 a6     mfdar   r3
c000000000000394:       f9 2d 01 20     std     r9,288(r13)
c000000000000398:       7d 20 00 26     mfcr    r9
```



# The PR=1 Trick

## Example Linux Kernel Code

```
c000000000000380:       7c 42 13 78     mr      r2,r2
c000000000000384:       7d b1 43 a6     mtsprg  1,r13
c000000000000388:       7d b3 42 a6     mfsprg  r13,3
c00000000000038c:       f8 6d 01 60     std     r3,352(r13)
c000000000000390:       7c 73 02 a6     mfdar   r3
c000000000000394:       f9 2d 01 20     std     r9,288(r13)
c000000000000398:       7d 20 00 26     mfcr    r9
```



# The PR=1 Trick

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

# PV Framework

```
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
```

# PV Framework

```
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
```

# PV Framework

```
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
```

# PV Framework

```
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
```

# PV Framework

```
c0000000000000380:        7c 42 13 78     mr      r2,r2
c0000000000000384:        7d b1 43 a6     std     r13,-SPRG1(0)
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
```

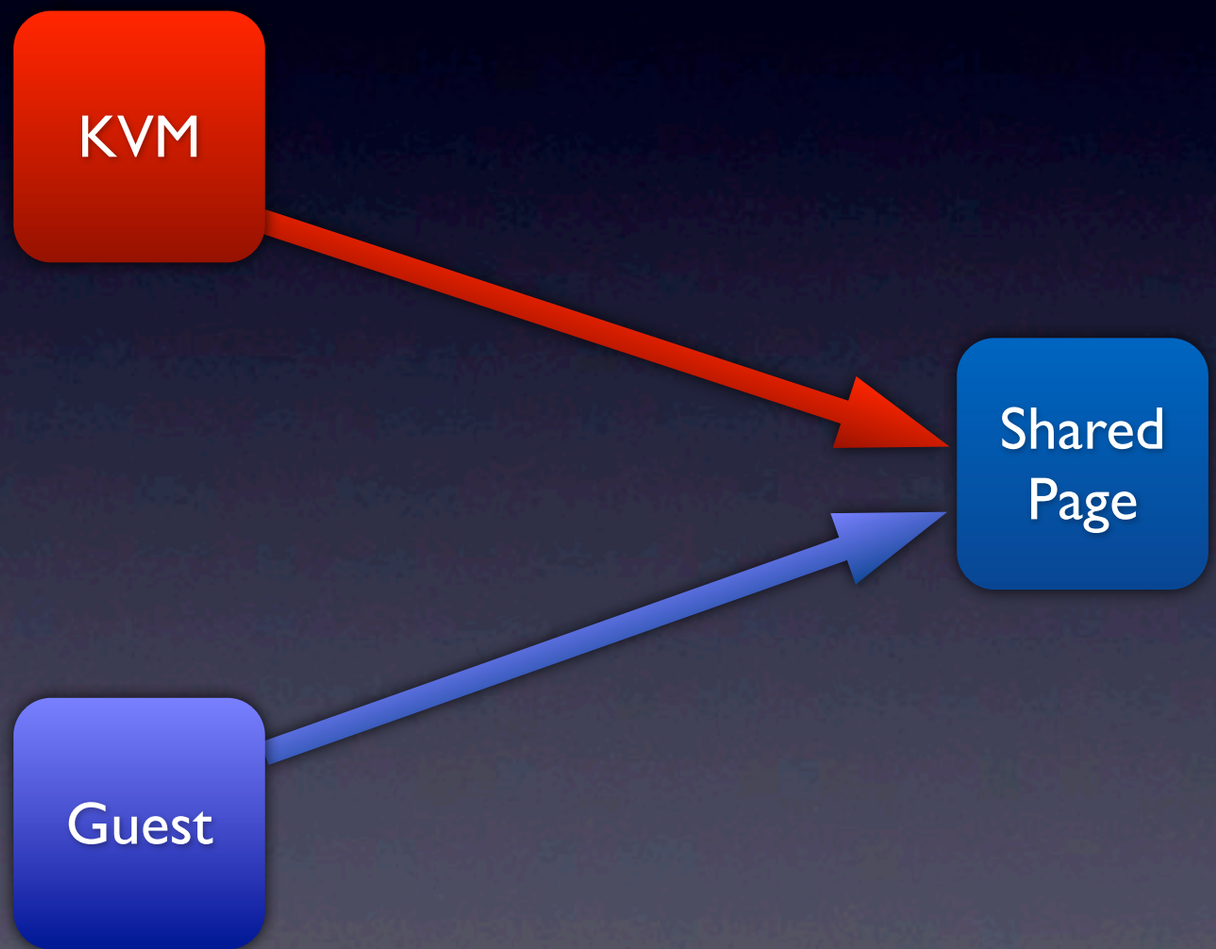
# PV Framework

```
c0000000000000380:      7c 42 13 78     mr      r2,r2
c0000000000000384:      7d b1 43 a6     std     r13,-SPRG1(0)
c0000000000000388:      7d b3 42 a6     ld      r13,-SPRG3(0)
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
```

# PV Framework

```
c0000000000000380:        7c 42 13 78     mr      r2,r2
c0000000000000384:        7d b1 43 a6     std     r13,-SPRG1(0)
c0000000000000388:        7d b3 42 a6     ld      r13,-SPRG3(0)
c000000000000038c:        f8 6d 01 60     std     r3,352(r13)
c0000000000000390:        7c 73 02 a6     ld      r3,-DAR(0)
c0000000000000394:        f9 2d 01 20     std     r9,288(r13)
c0000000000000398:        7d 20 00 26     mfcr    r9
```

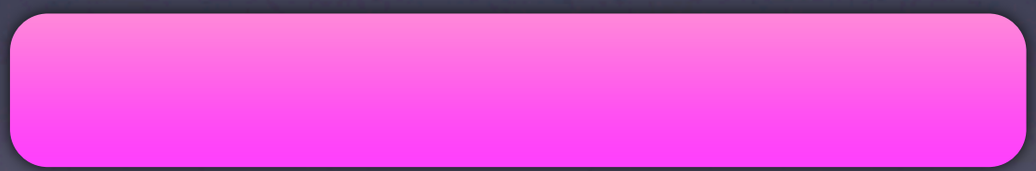
# PV Framework



# Virtual Memory

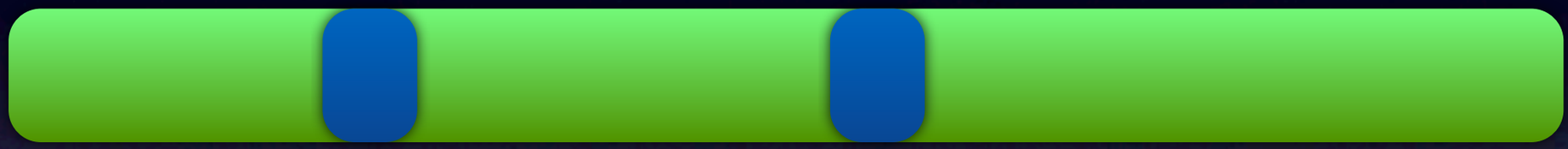
Virtual Memory

Physical Memory

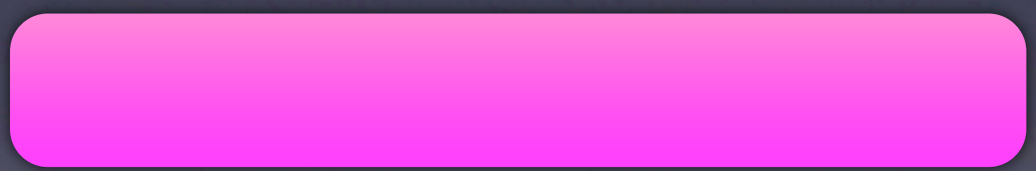


# Virtual Memory

Virtual Memory



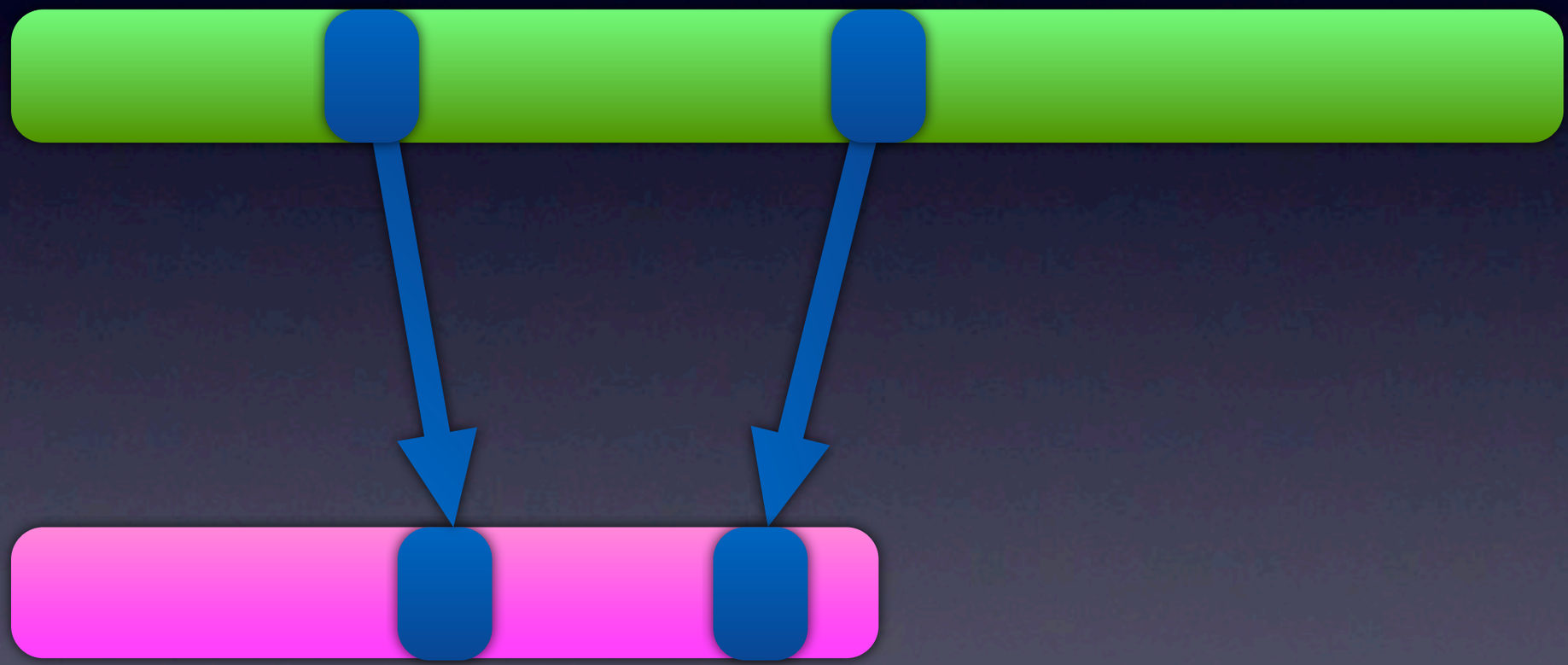
Physical Memory



# Virtual Memory

Virtual Memory

Physical Memory



# Virtual Memory

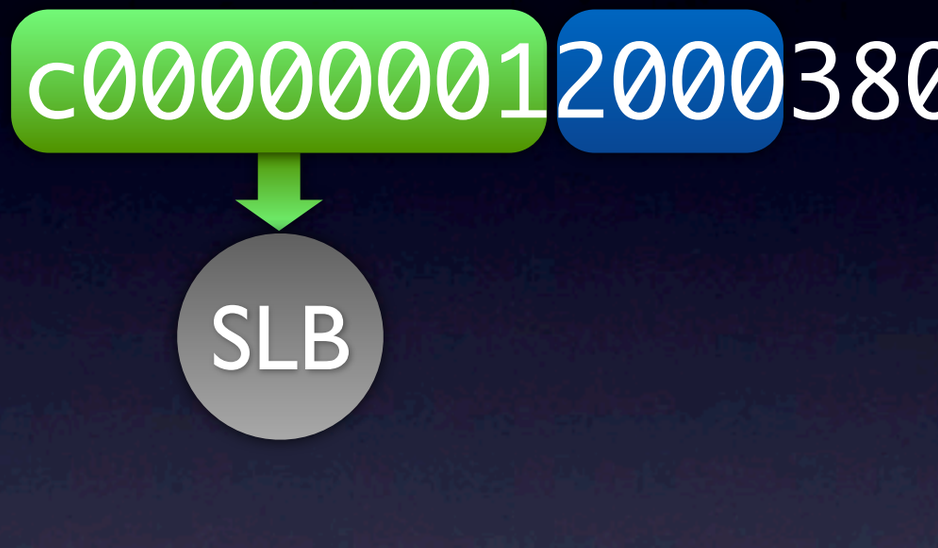
Effective Address c0000000120000380

Virtual Address

Real Address

# Virtual Memory

Effective Address




Virtual Address

Real Address

# Segment Lookaside Buffer

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

# Segment Lookaside Buffer

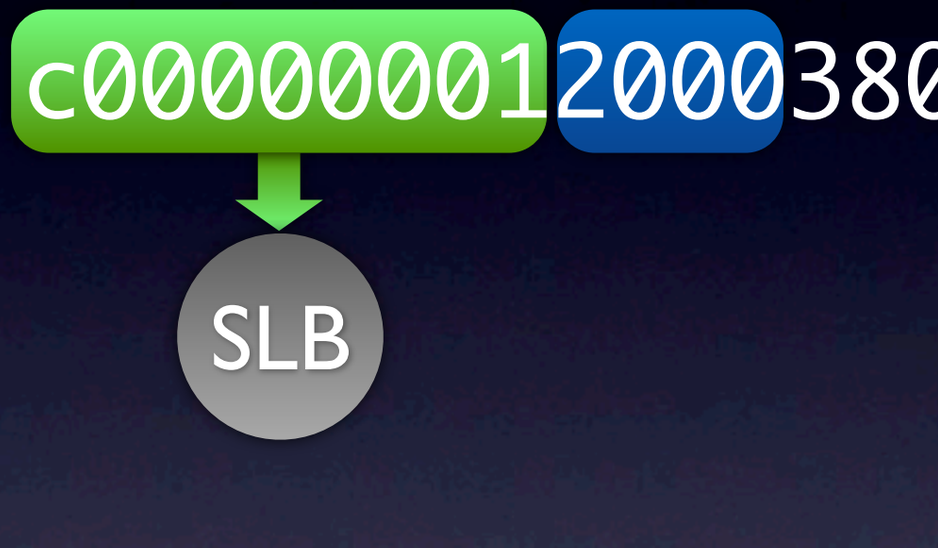
| ESID | VSID |
|---|---|
| c00000000 | 408f92c94 |
| d00000000 | f09b89af5 |
| c00000007 | 45cb97751 |
| 0 | ef72a166e |
| f | faaa19203 |
| c00000001 | 414f012af |
| ... | ... |

# Segment Lookaside Buffer

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

# Virtual Memory

Effective Address

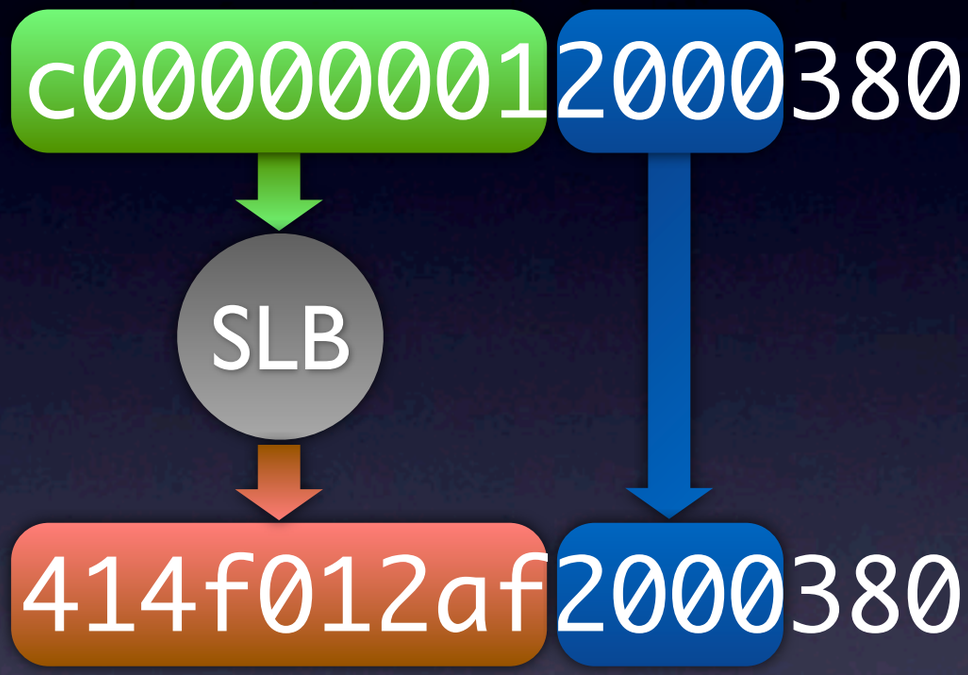



Virtual Address

Real Address

# Virtual Memory

Effective Address

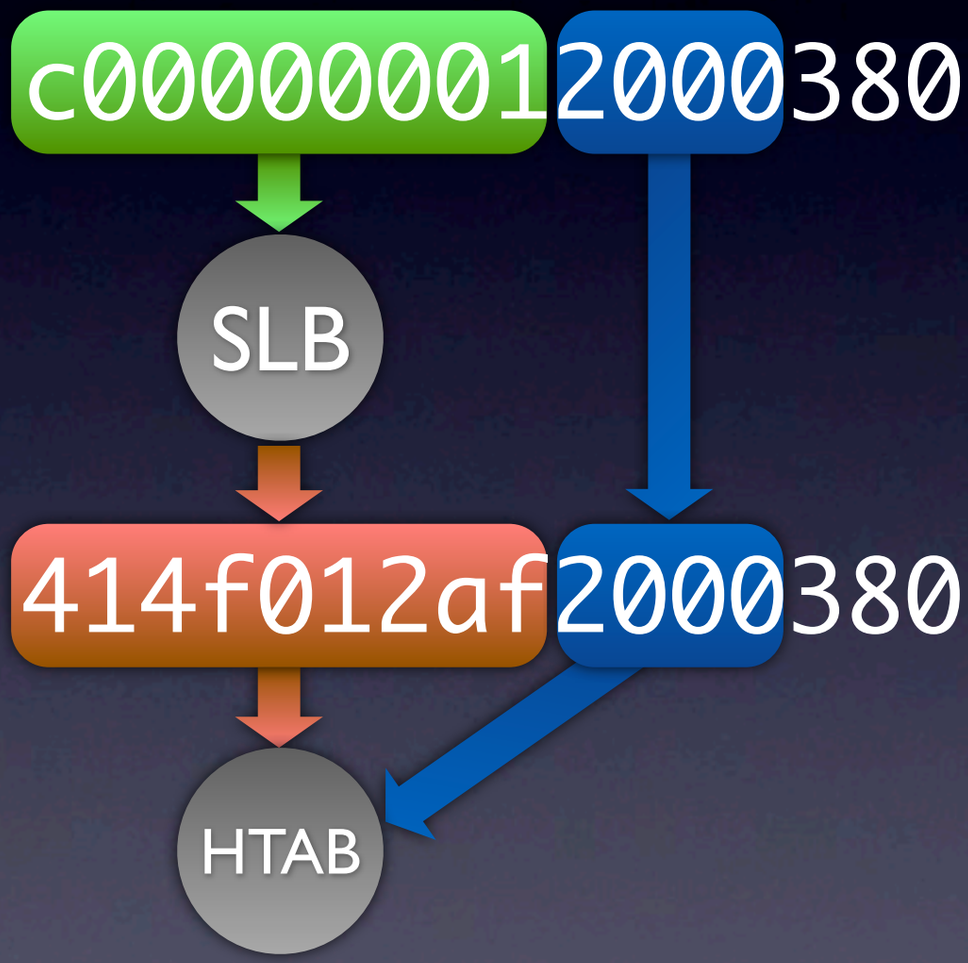



Virtual Address

Real Address

# Virtual Memory

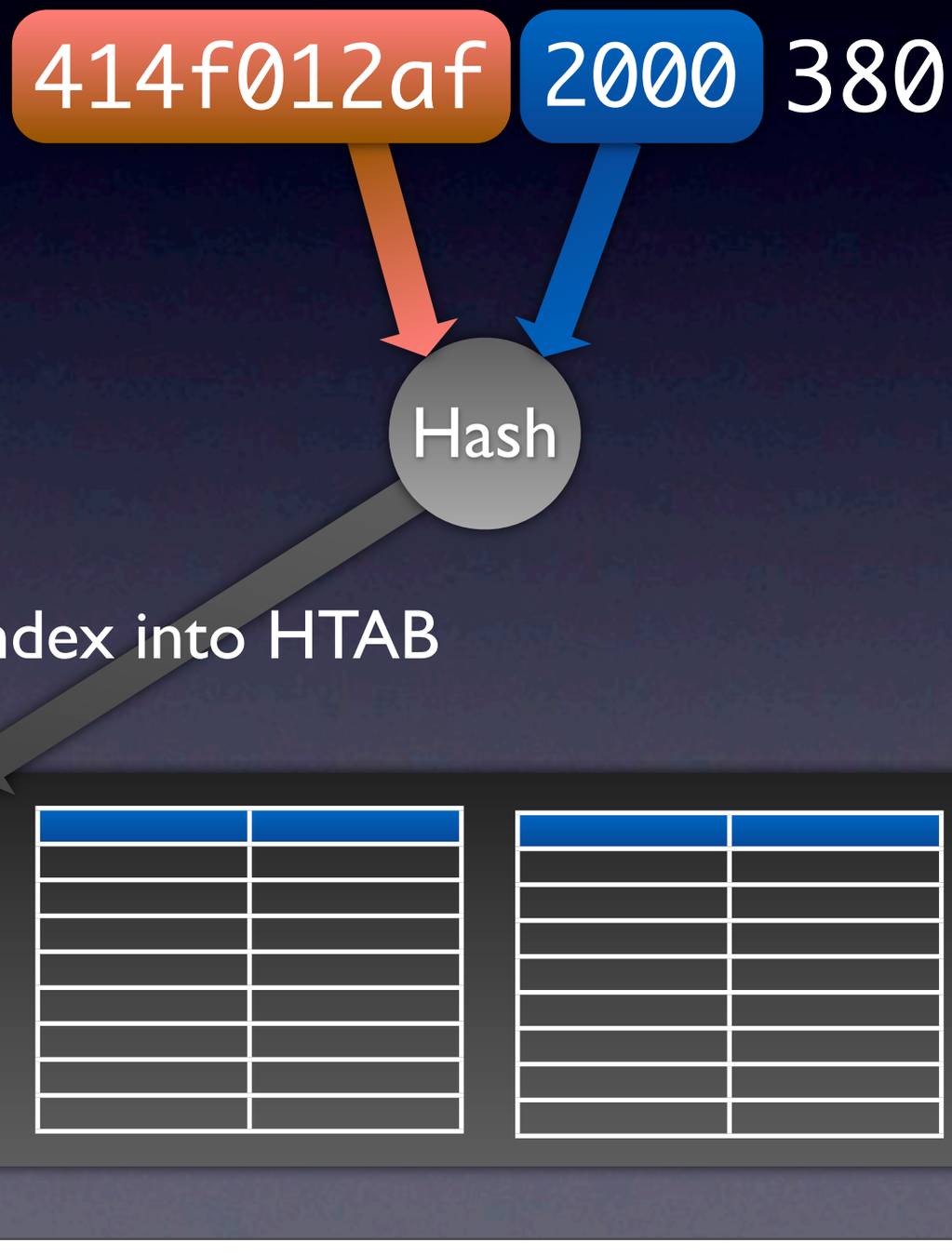
Effective Address




Virtual Address

Real Address

# 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

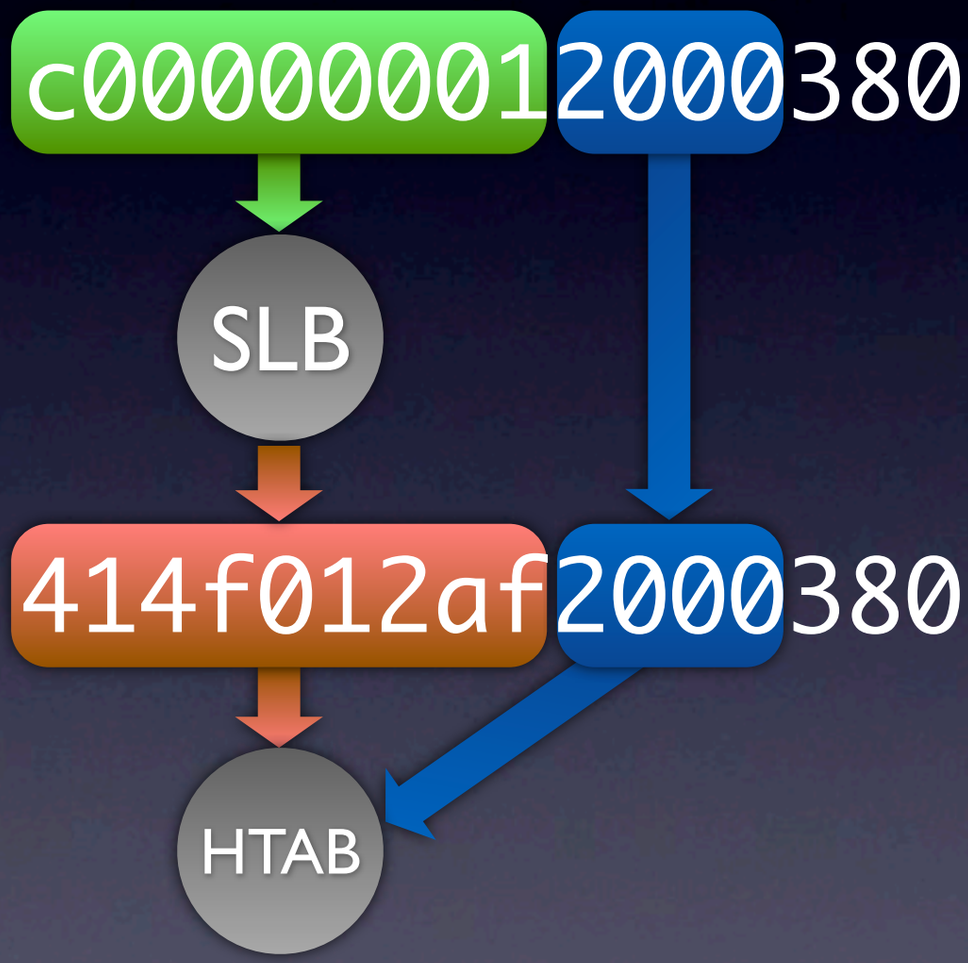
PTEG

| VSID | Real Page |
| --- | --- |
| 408f92c94 | 5000 |
| 414f012af | 12000 |
| … | … |

414f012af 2000 → 12000

# Virtual Memory

Effective Address

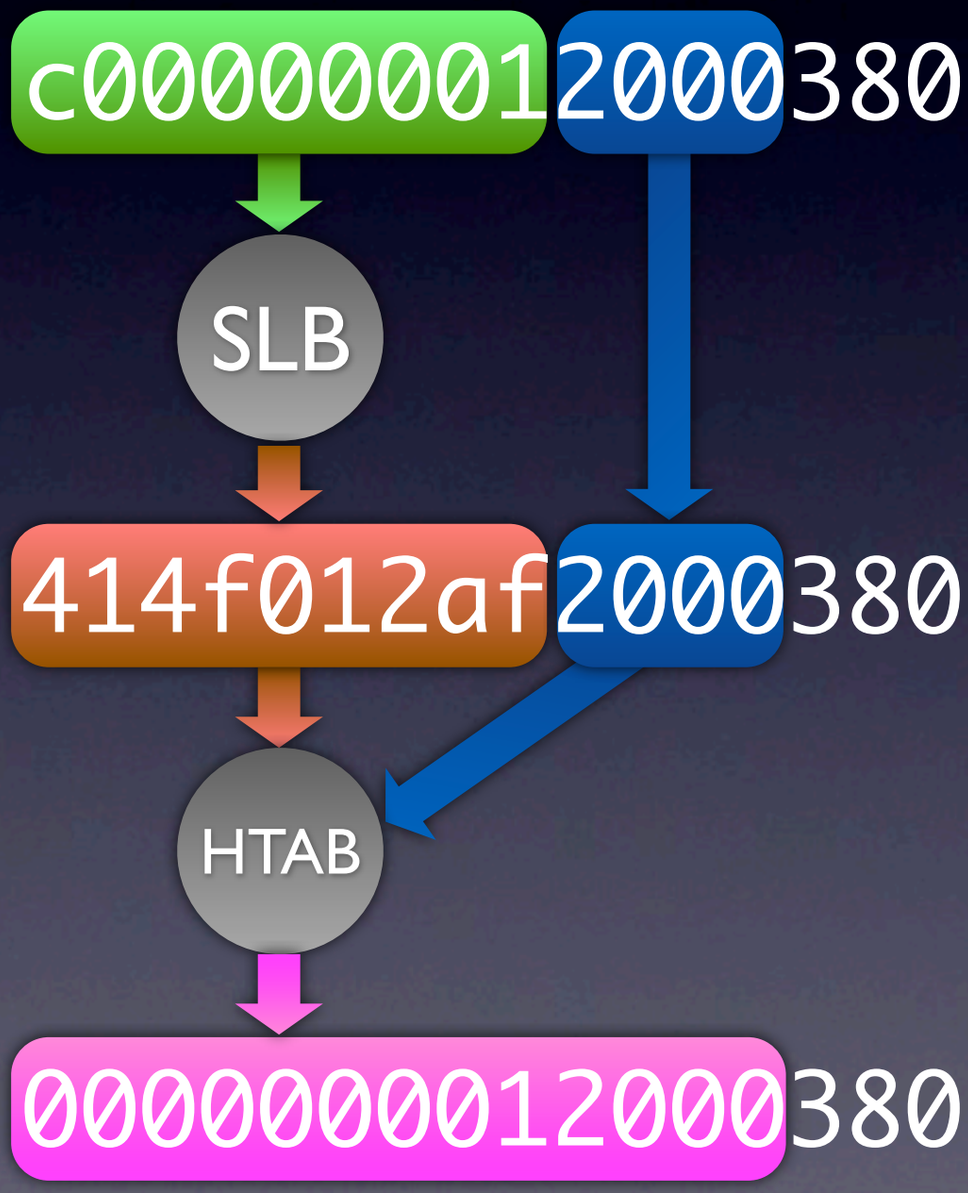



Virtual Address

Real Address

# Virtual Memory

Effective Address




Virtual Address

Real Address

# 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 |

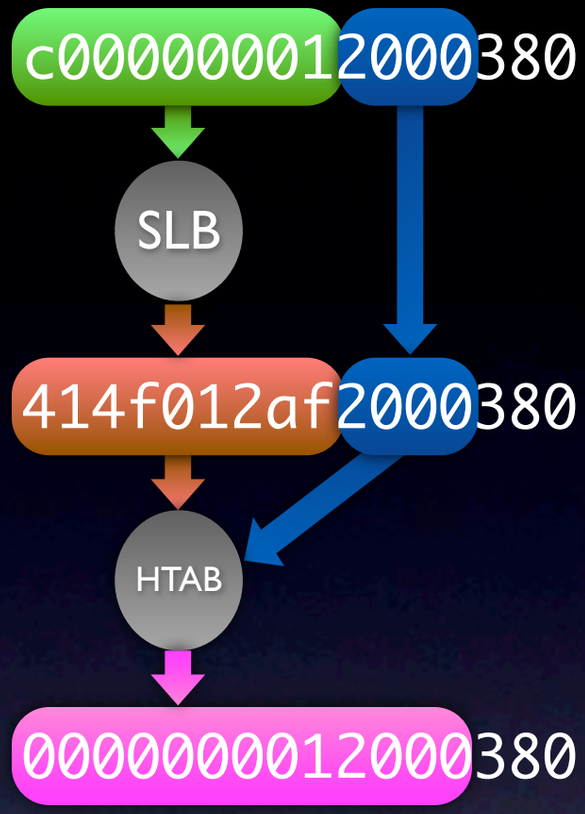
# Shadow Paging (Pages)

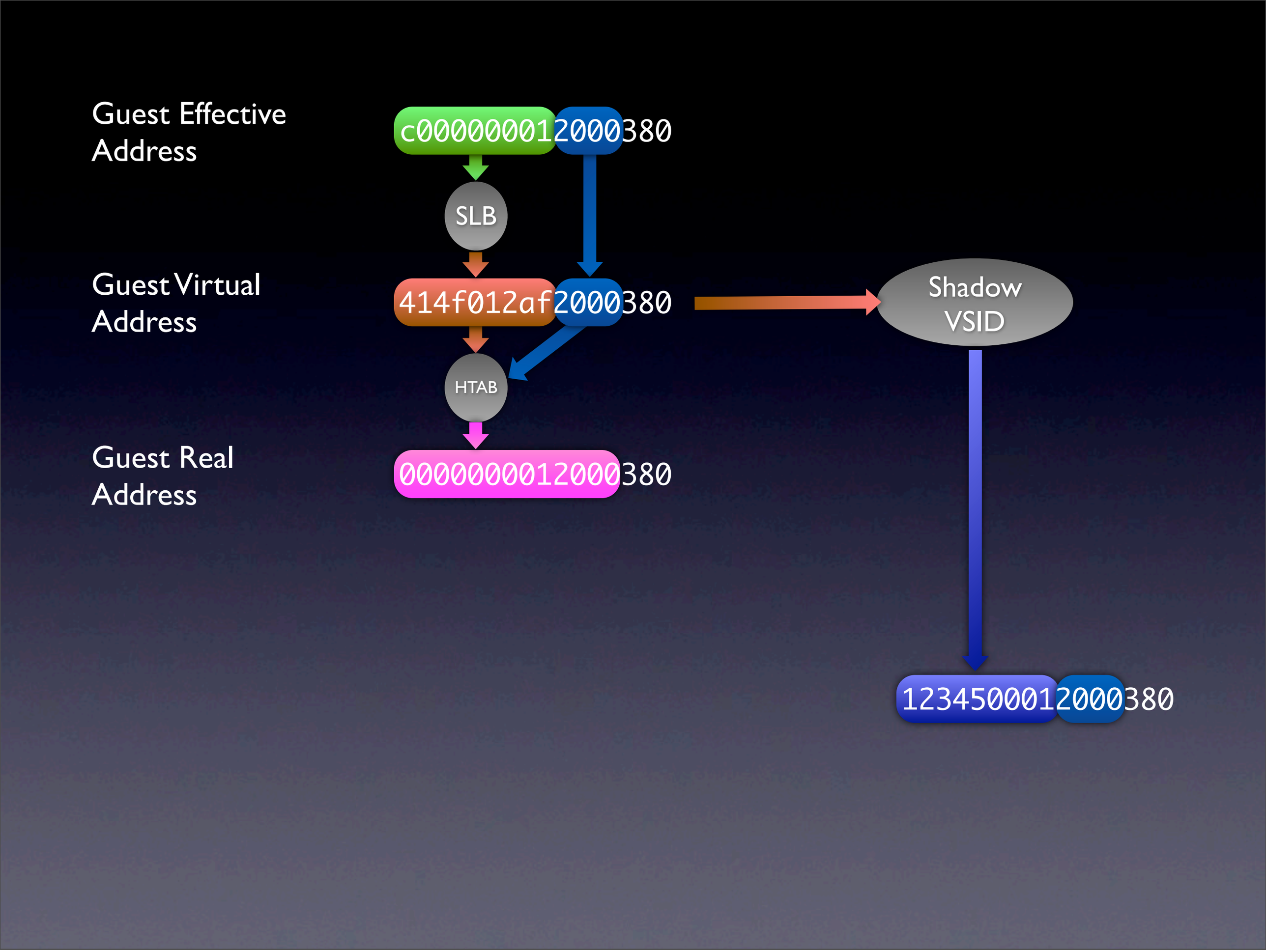
- Guest accesses an unmapped page
- Read guest SLB & HTAB
- Resolve Host Real Address
- Convert VSID to Shadow VSID
- Call Linux MMU helper to add host PTE

Guest Effective Address

Guest Virtual Address

Guest Real Address




Guest Effective Address
c0000000012000380
SLB
Guest Virtual Address
414f012af2000380
Shadow VSID
HTAB
Guest Real Address
0000000012000380
12345000012000380

Guest Effective Address

c000000012000380

SLB

Guest Virtual Address

414f012af2000380

Shadow VSID

HTAB

Guest Real Address

000000012000380

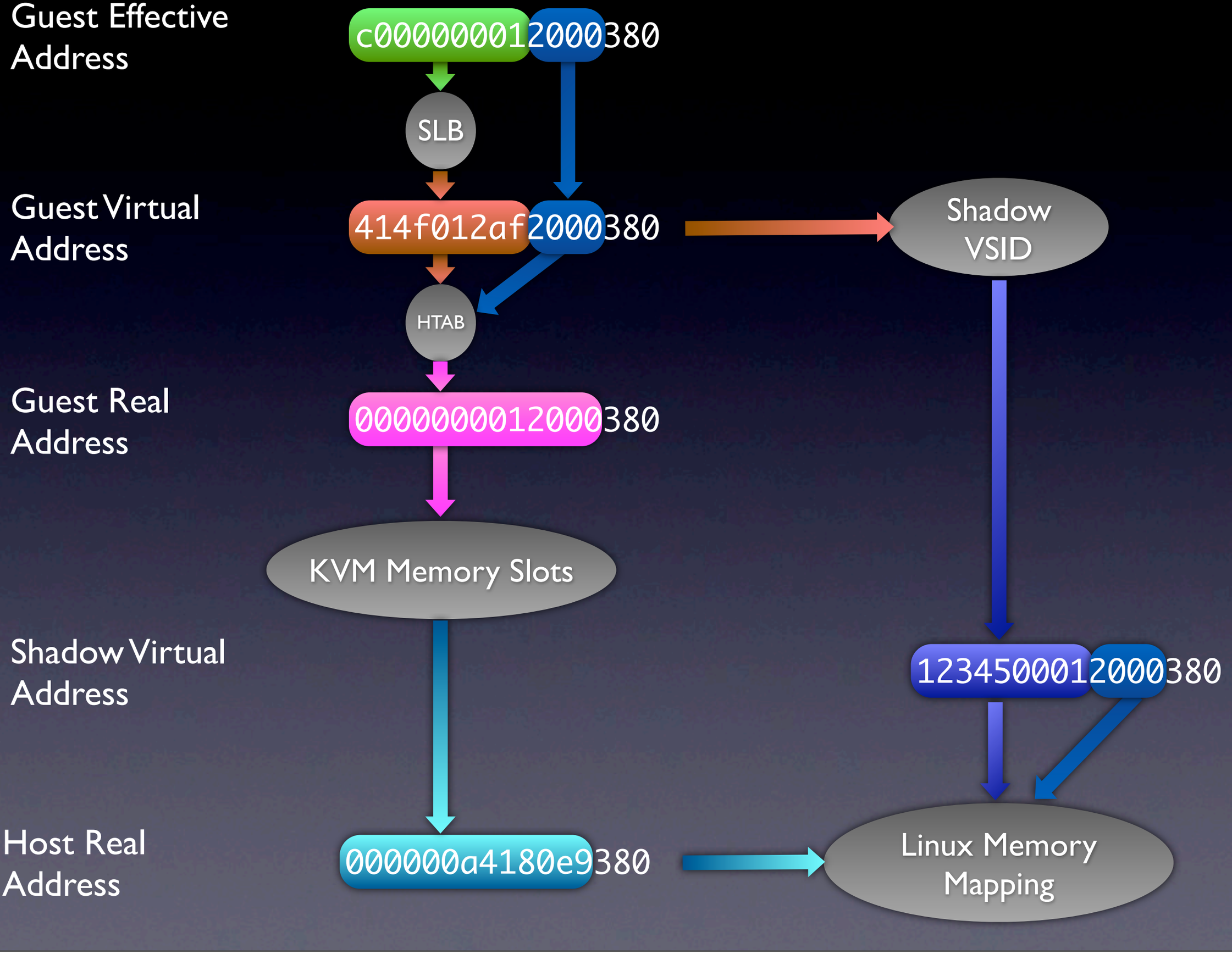
KVM Memory Slots

Shadow Virtual Address

12345000012000380

Host Real Address

000000a4180e9380


Guest Effective Address
c0000000012000380
SLB
Guest Virtual Address
414f012af2000380
Shadow VSID
HTAB
Guest Real Address
0000000012000380
KVM Memory Slots
Shadow Virtual Address
12345000012000380
Host Real Address
000000a4180e9380
Linux Memory Mapping

# Guest Entry

CPU

vCPU struct

PACA

# Guest Entry

CPU

vCPU struct

PACA

# Guest Exit

CPU

vCPU struct

PACA

# Guest Exit

CPU

vCPU struct

PACA

# Mac-on-Linux

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

# Why do I do this?

- Build Service
- SUSE Studio

# Status

| | |
|---|---|
| 440 | ✔ |
| e500v2 | ✔ |
| Book3S 32bit | ✔ |
| Book3S 64bit | ✔ |

# Questions