in

About Me

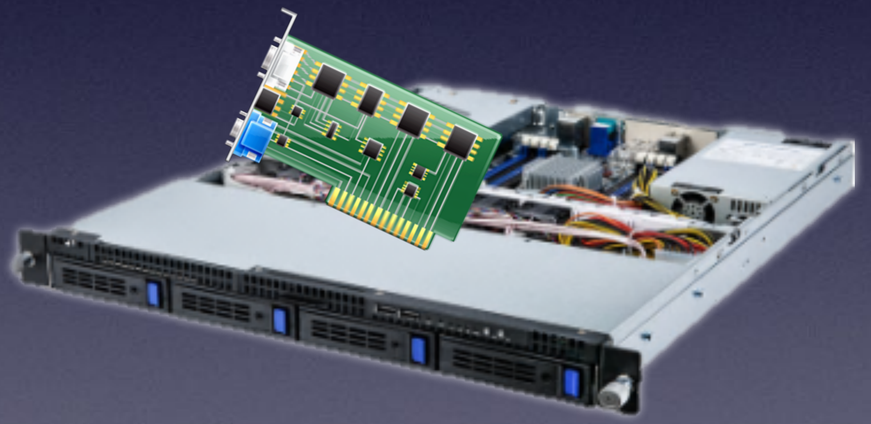
- Alexander Graf
- KVM and QEMU developer for SUSE
  - Server class PowerPC KVM port
  - Nested SVM
- Founding member of SUSE ARM team
- U-Boot UEFI support

ARM

x86

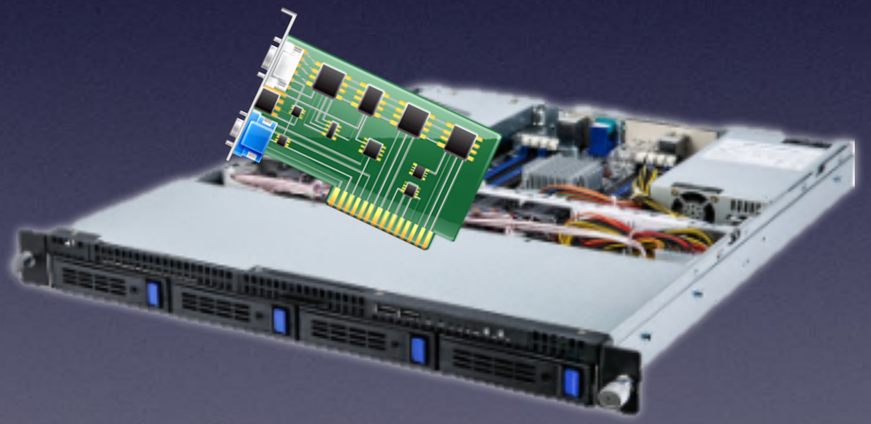
ARM

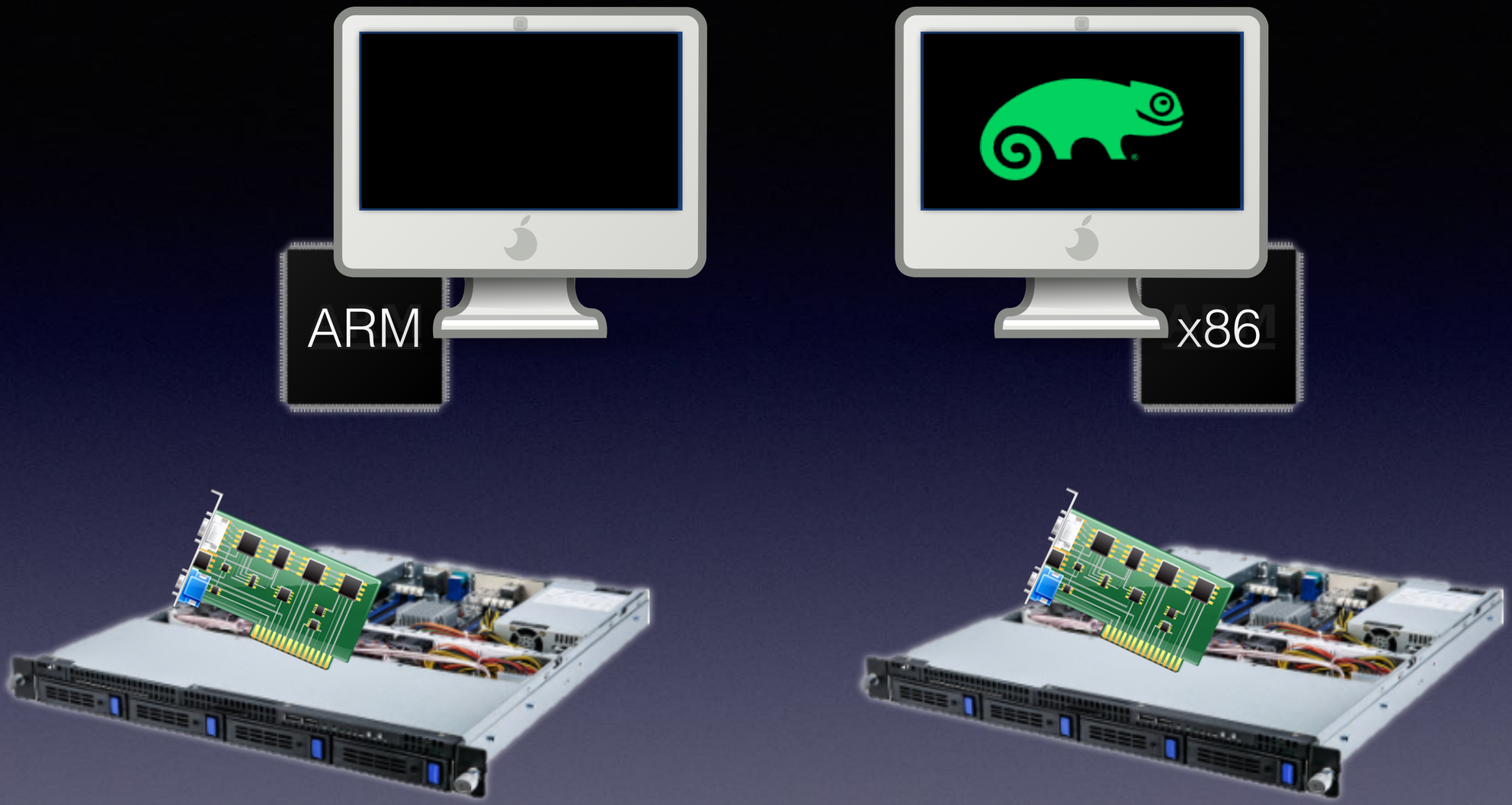
x86

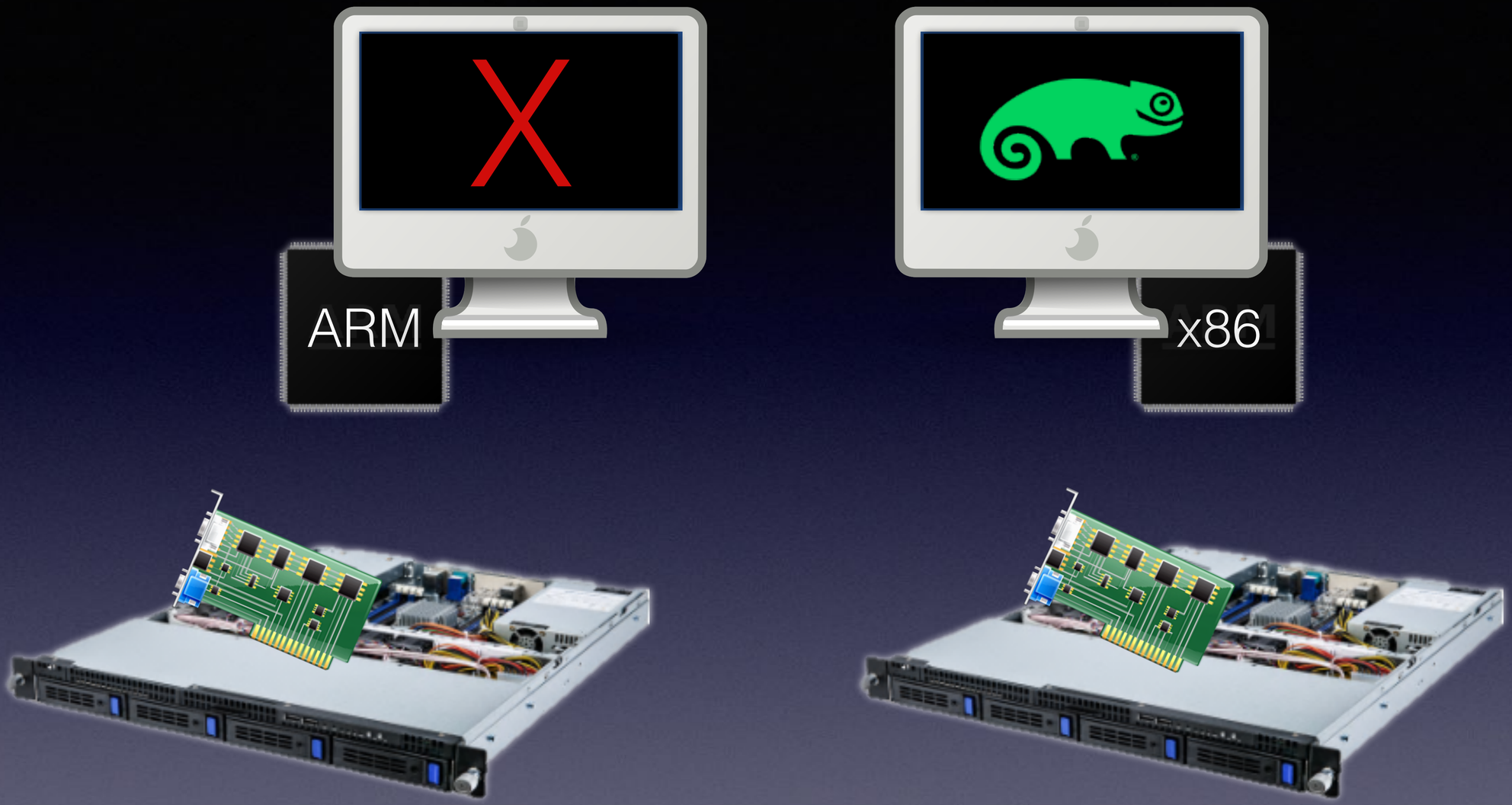


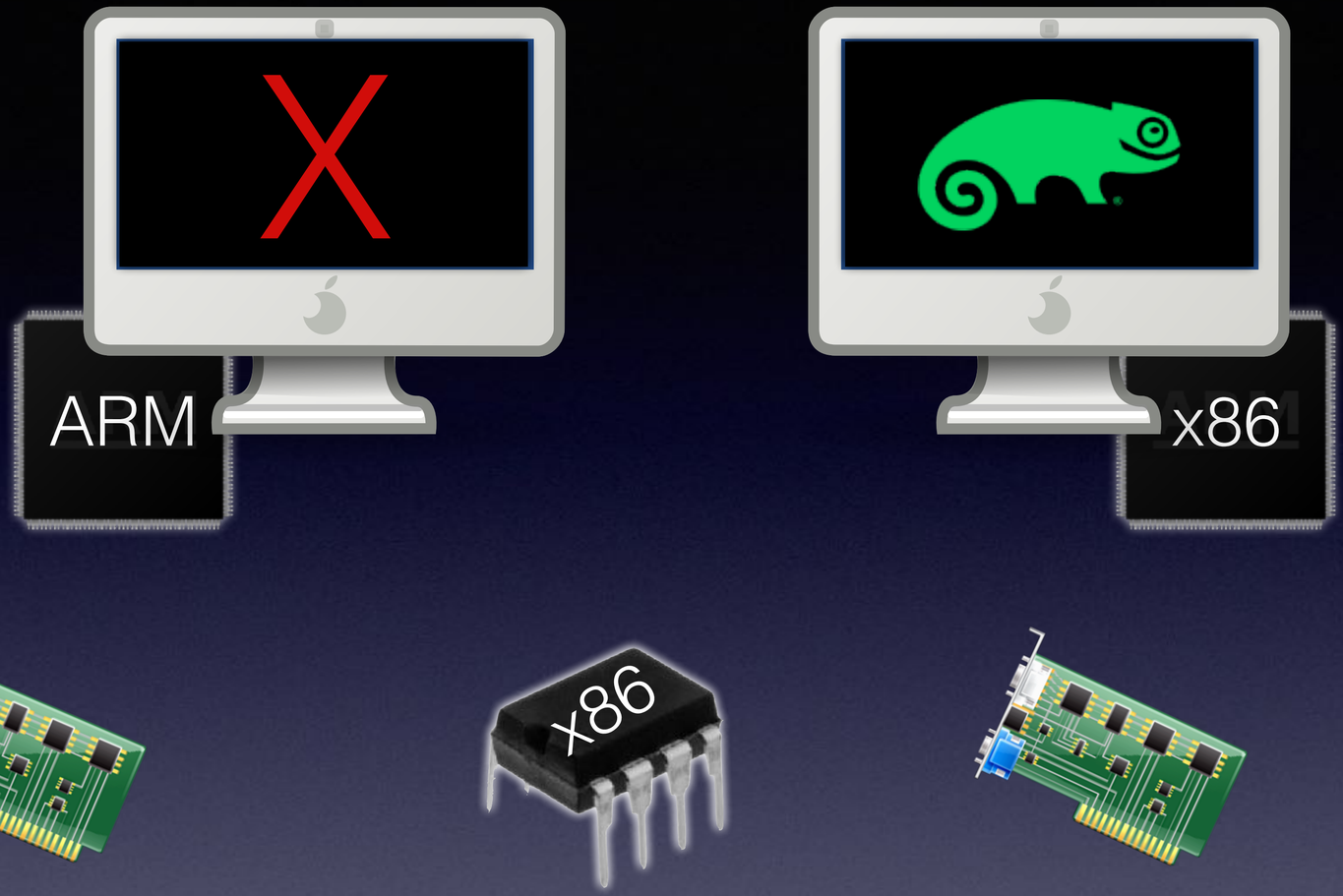

ARM


x86




ARM
x86


X
ARM
x86


X
ARM
x86
x86


Linaro
connect


Linaro

x0	x1	x2	x3	x4
x5	x6	x7	x8	x9
x10	x11	x12	x13	x14
x15	x16	x17	x18	x19
x20	x21	x22	x23	x24
x25	x26	x27	x28	x29
x30	xzr			

rax	rbx	rcx	rdx	rbp
rsp	rsi	rdi	r8	r9
r10	r11	r12	r13	r14
r15				


ARM

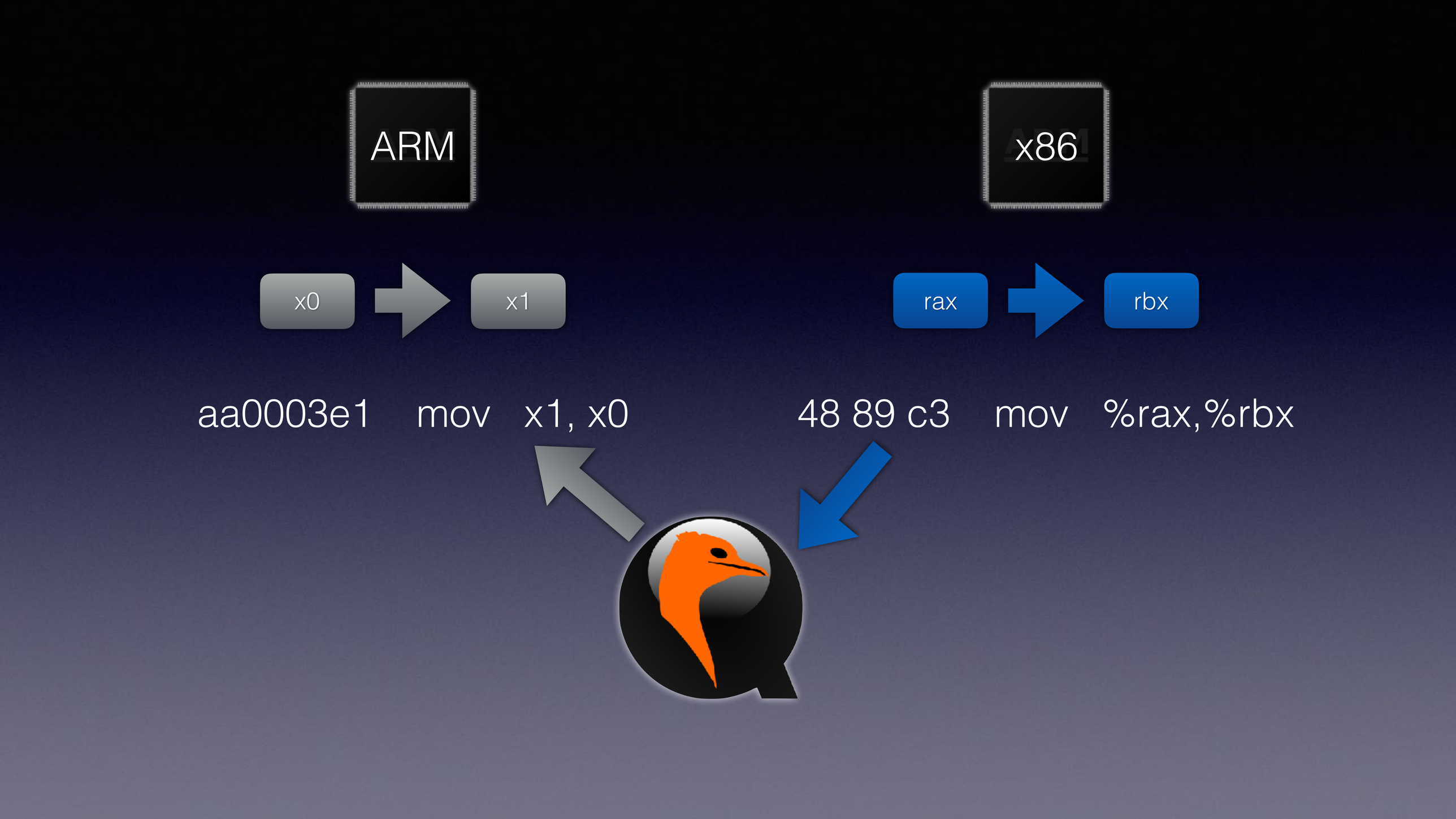

x0
x1


x86

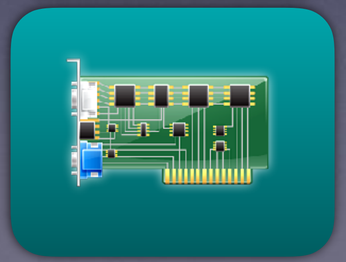

rax
rbx

aa0003e1 mov x1, x0

48 89 c3 mov %rax,%rbx

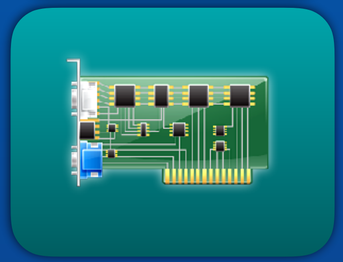

ARM
x86
x0
x1
rax
rbx
aa0003e1 mov x1, x0
48 89 c3 mov %rax,%rbx

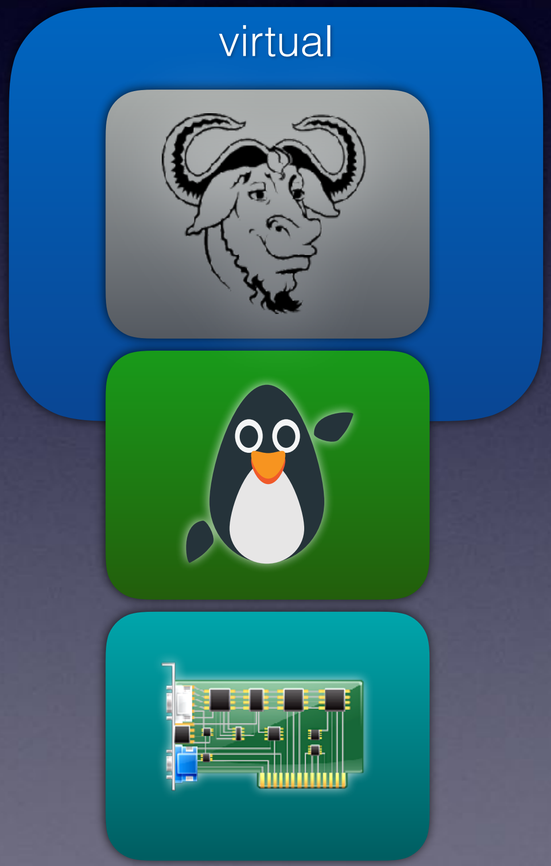

PONY

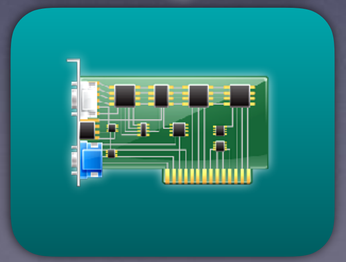



PONY

virtual

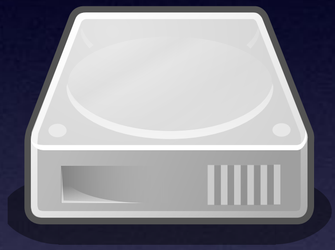



virtual

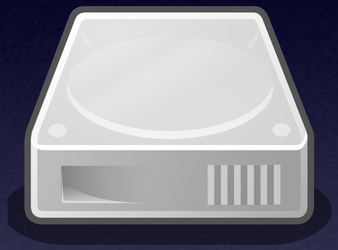


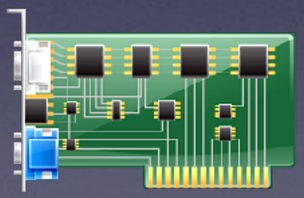

PONY


PONY

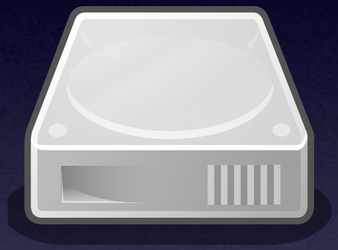


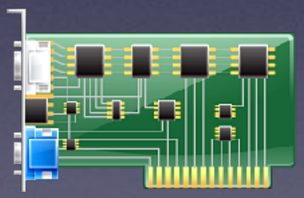

PONY



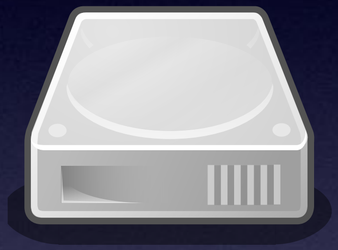


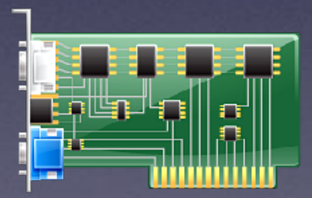

PONY

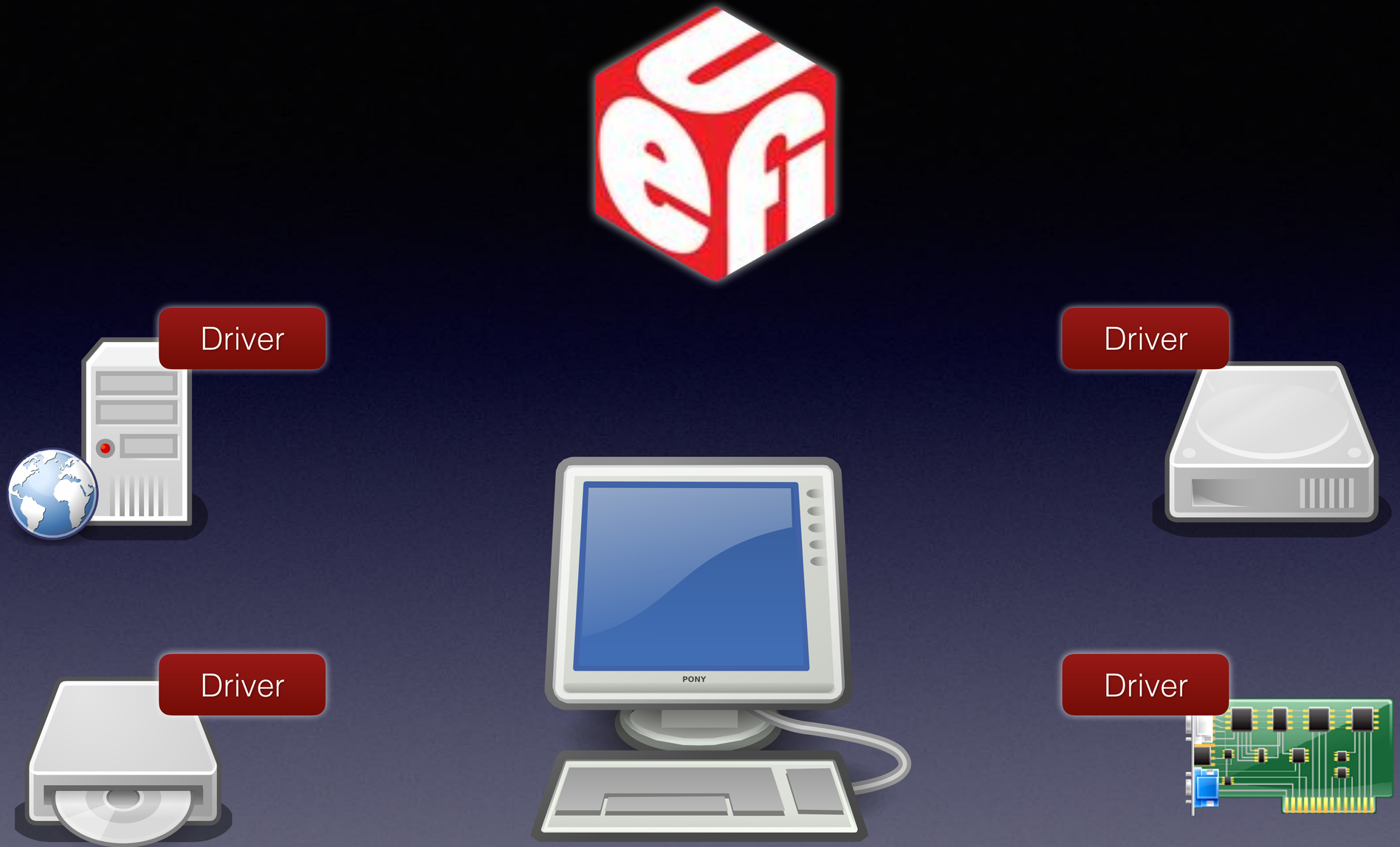


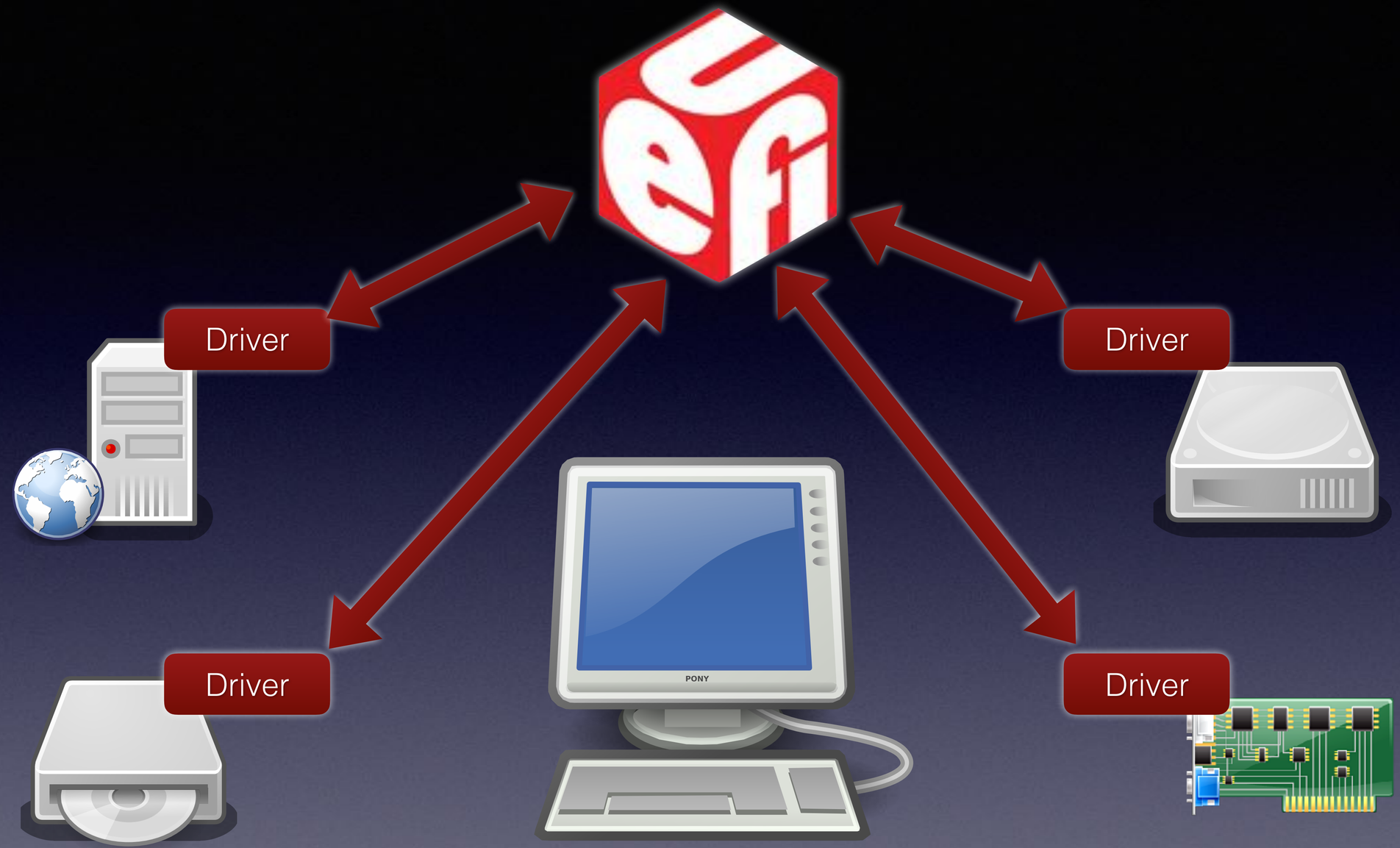


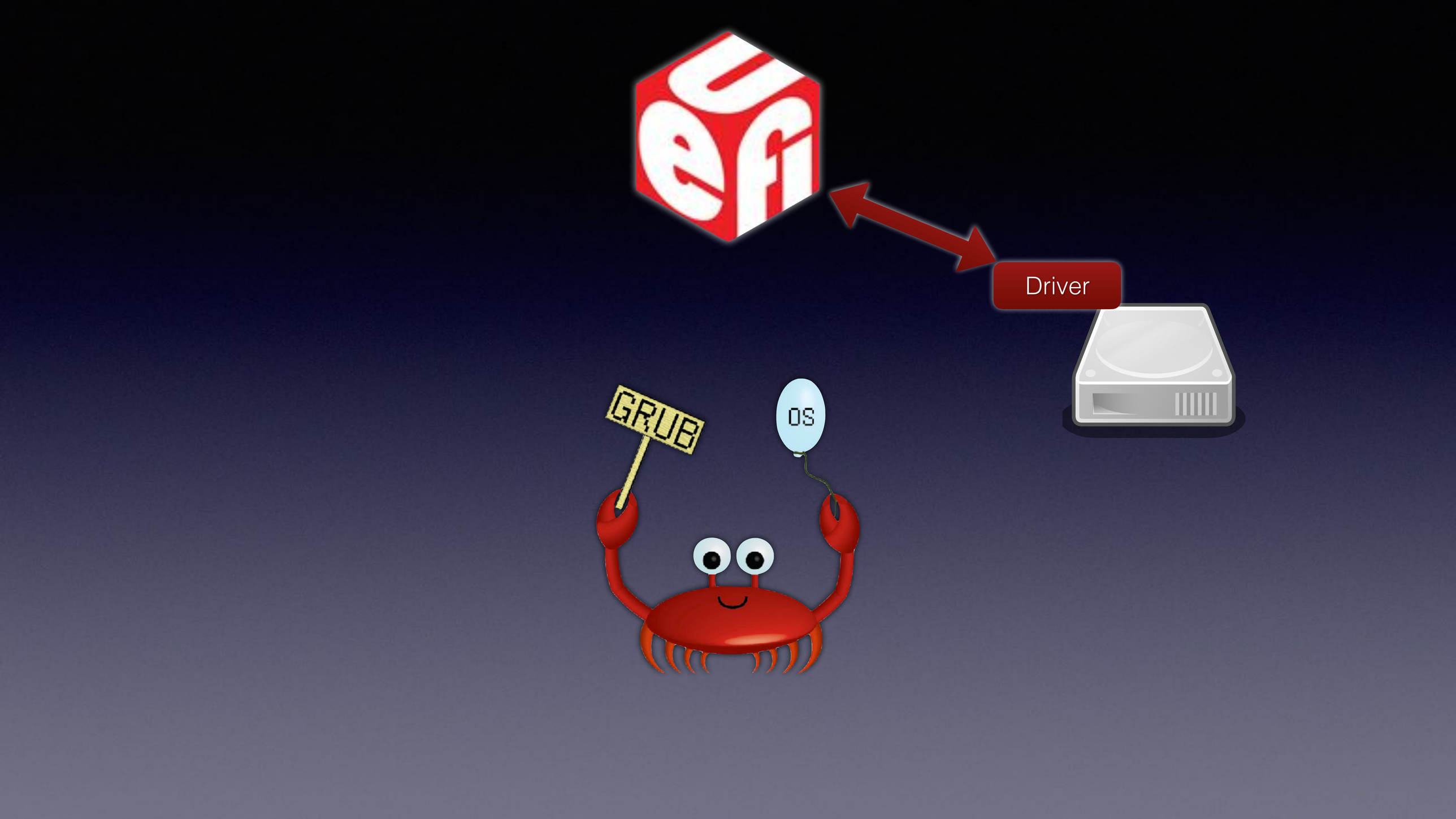

PONY

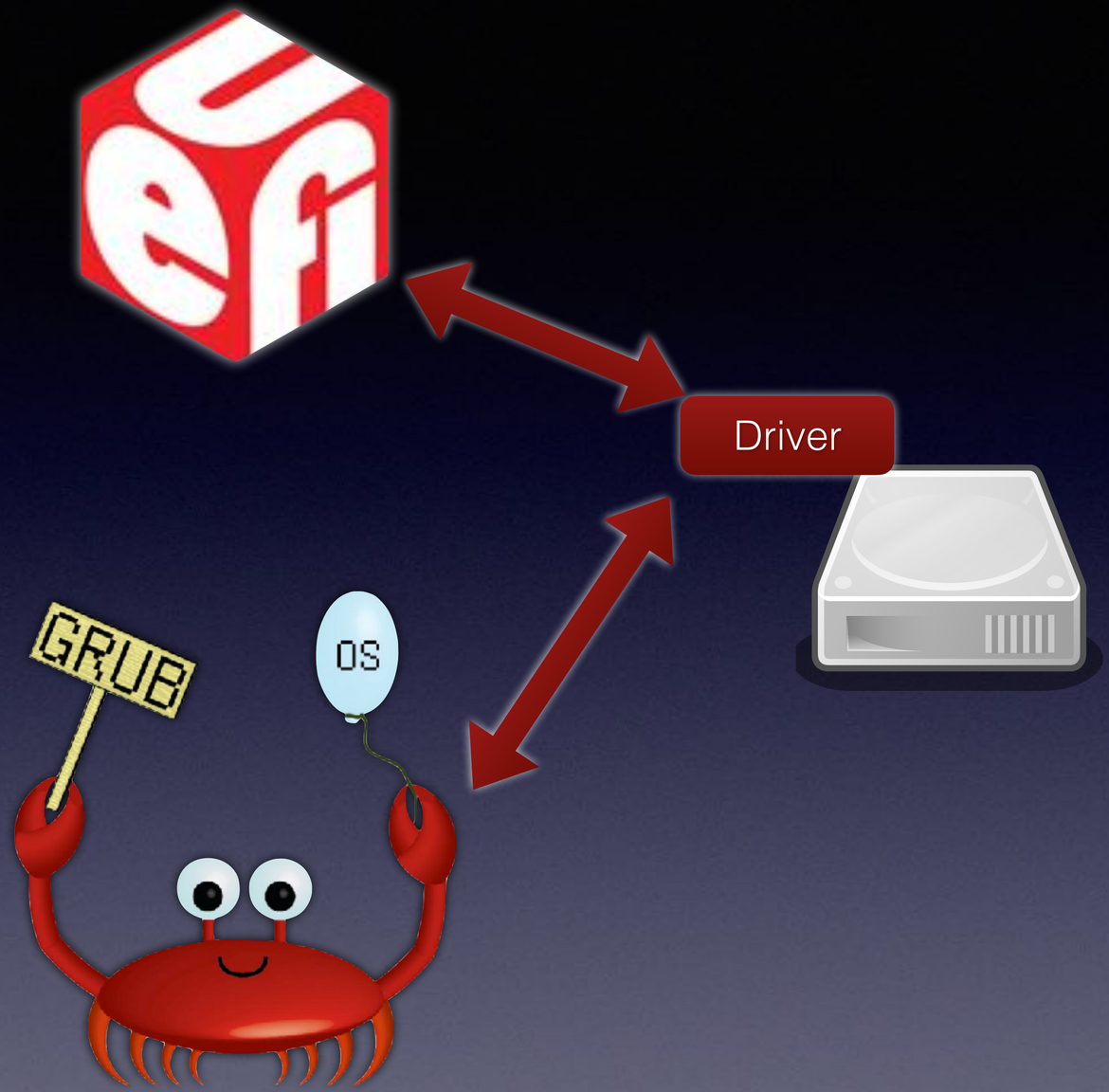


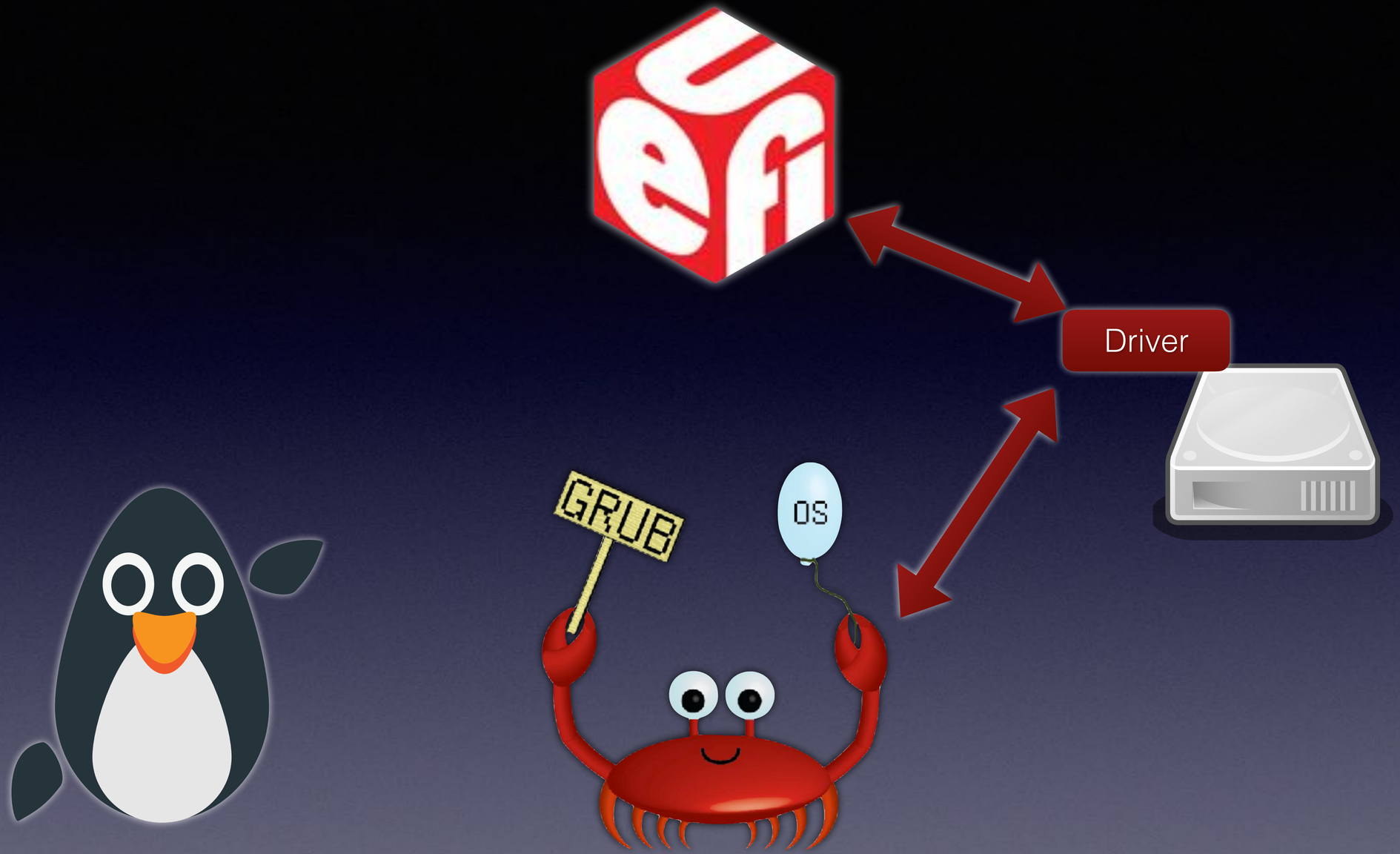


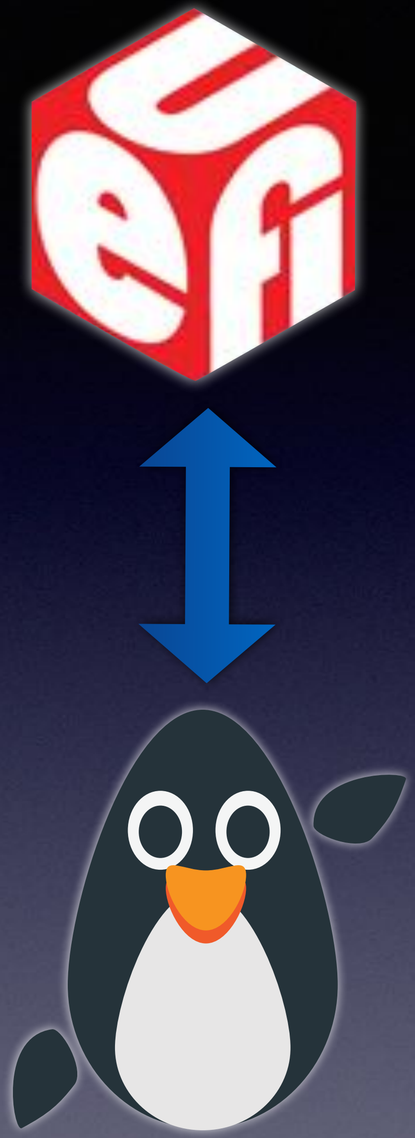

Driver
Driver
Driver
Driver
PONY

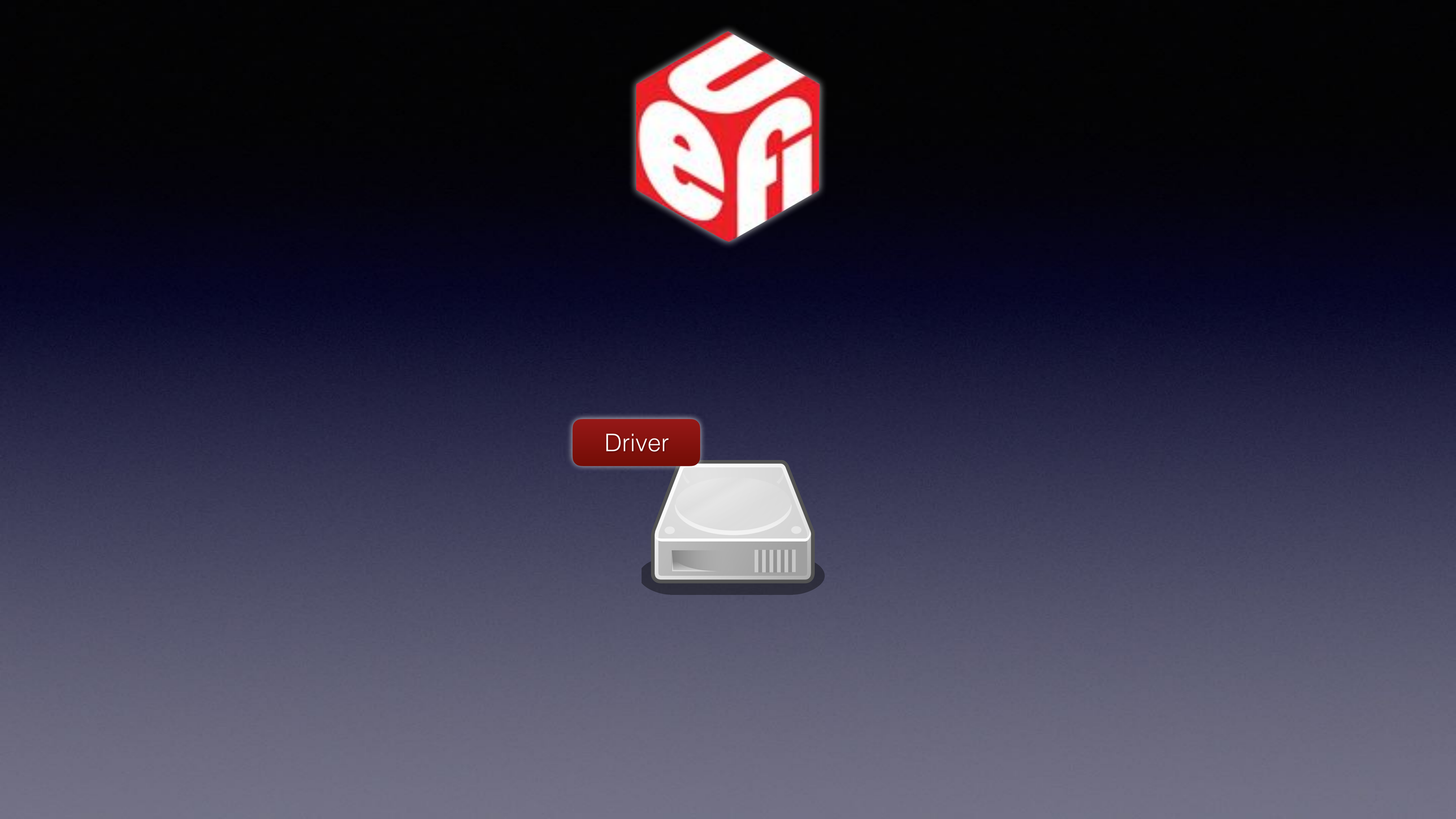

Driver
Driver
Driver
Driver
PONY


Driver
GRUB
OS


Driver
GRUB
OS


Driver
GRUB
OS




Driver


gBS->InstallMultipleProtocolInterfaces(🟦)
Driver

gBS->InstallMultipleP)

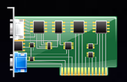
efi_status_t (*supported)(…)
efi_status_t (*start)(…)
efi_status_t (*stop)(…)

u32 version
void *image_handle
void *driver_handle

Driver

Driver

PCI driver



Driver

PCI driver

Driver

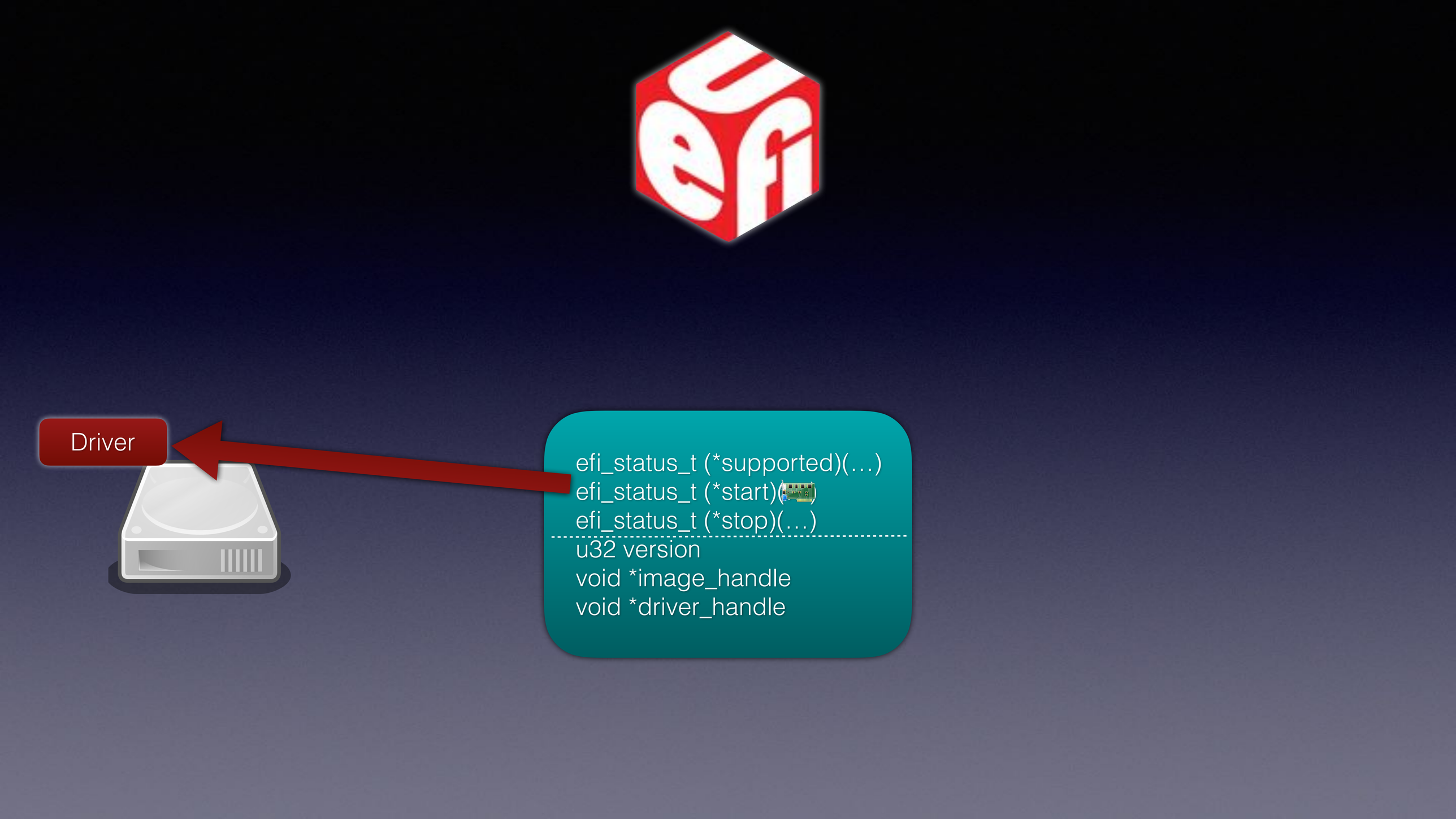
efi_status_t (*supported)(
efi_status_t (*start)(…)
efi_status_t (*stop)(…)

u32 version
void *image_handle
void *driver_handle

Driver

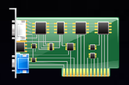
efi_status_t (*supported)(…)
efi_status_t (*start)(…)
efi_status_t (*stop)(…)

u32 version
void *image_handle
void *driver_handle


Driver
efi_status_t (*supported)(…)
efi_status_t (*start)(
efi_status_t (*stop)(…)
u32 version
void *image_handle
void *driver_handle

gBS->InstallMultipleProtocolInterfaces(🟦)

Driver



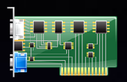
gBS->InstallMultipleProtocolInterfaces( )

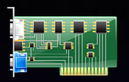
Driver

u64 revision
struct block_io_media *m

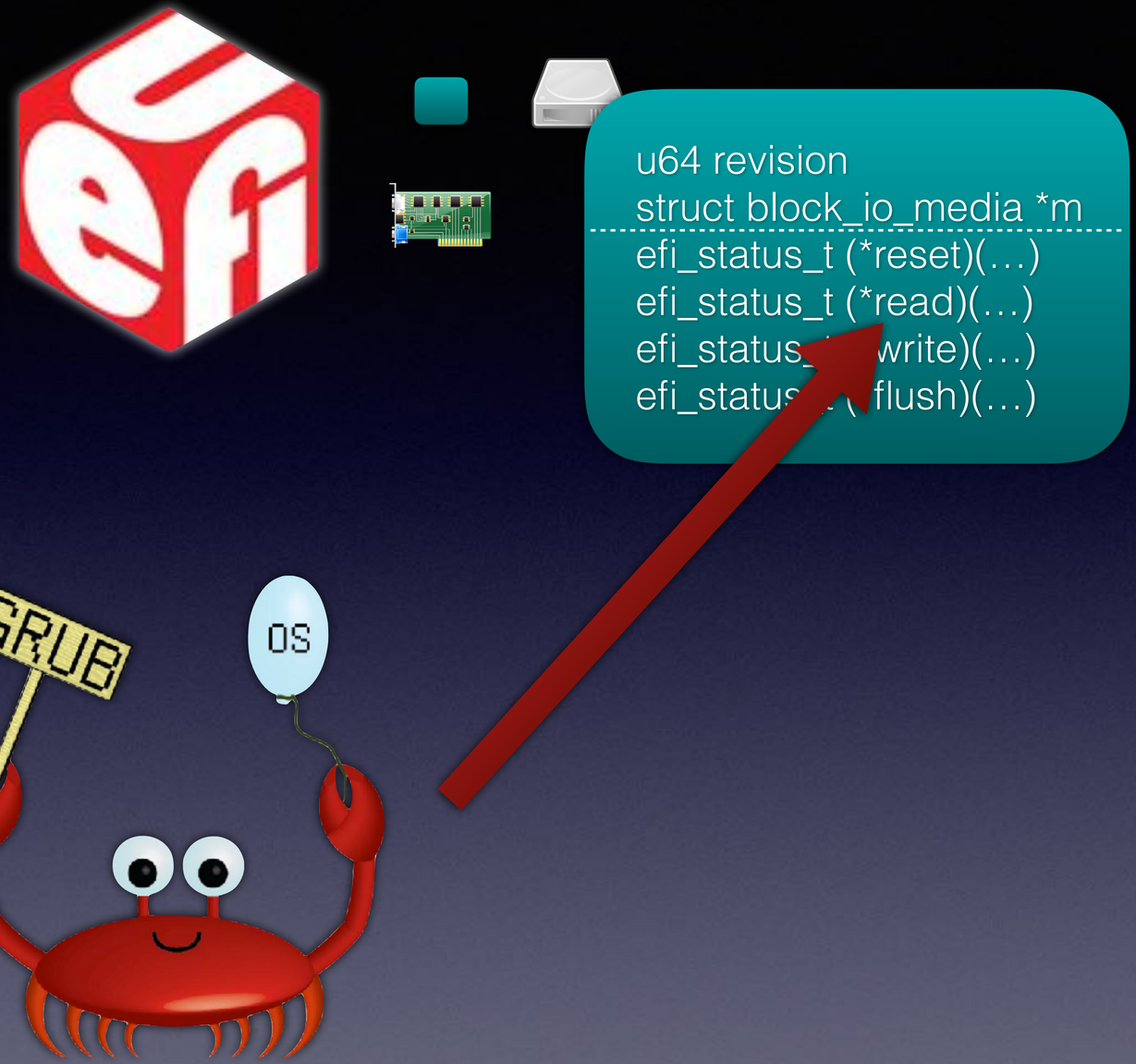
efi_status_t (*reset)(…)
efi_status_t (*read)(…)
efi_status_t (*write)(…)
efi_status_t (*flush)(…)

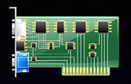







GRUB
OS


u64 revision
struct block_io_media *m
efi_status_t (*reset)(…)
efi_status_t (*read)(…)
write)(…)
flush)(…)
GRUB
OS



virtual

u64 revision
struct block_io_media *m

efi_status_t (*reset)(…)
efi_status_t (*read)(…)
efi_status_t (*write)(…)
efi_status_t (*flush)(…)


```
0x00:    01 00 00 00 00 00 00 00
0x08:    f0 ee dd cc bb aa 99 88
0x10:    00 ee dd cc bb aa 99 88
0x18:    00 ed dd cc bb aa 99 88
0x20:    00 ec dd cc bb aa 99 88
0x28:    00 eb dd cc bb aa 99 88
```


```
0x00:    01 00 00 00 00 00 00 00
0x08:    f0 ee dd cc bb aa 99 88
0x10:    00 ee dd cc bb aa 99 88
0x18:    00 ed dd cc bb aa 99 88
0x20:    00 ec dd cc bb aa 99 88
0x28:    00 eb dd cc bb aa 99 88
```


```
u32 id
efi_status_t (*reset)(…)
```

```
0x00:    01 00 00 00 XX XX XX XX
0x08:    f0 ee dd cc bb aa 99 88
```

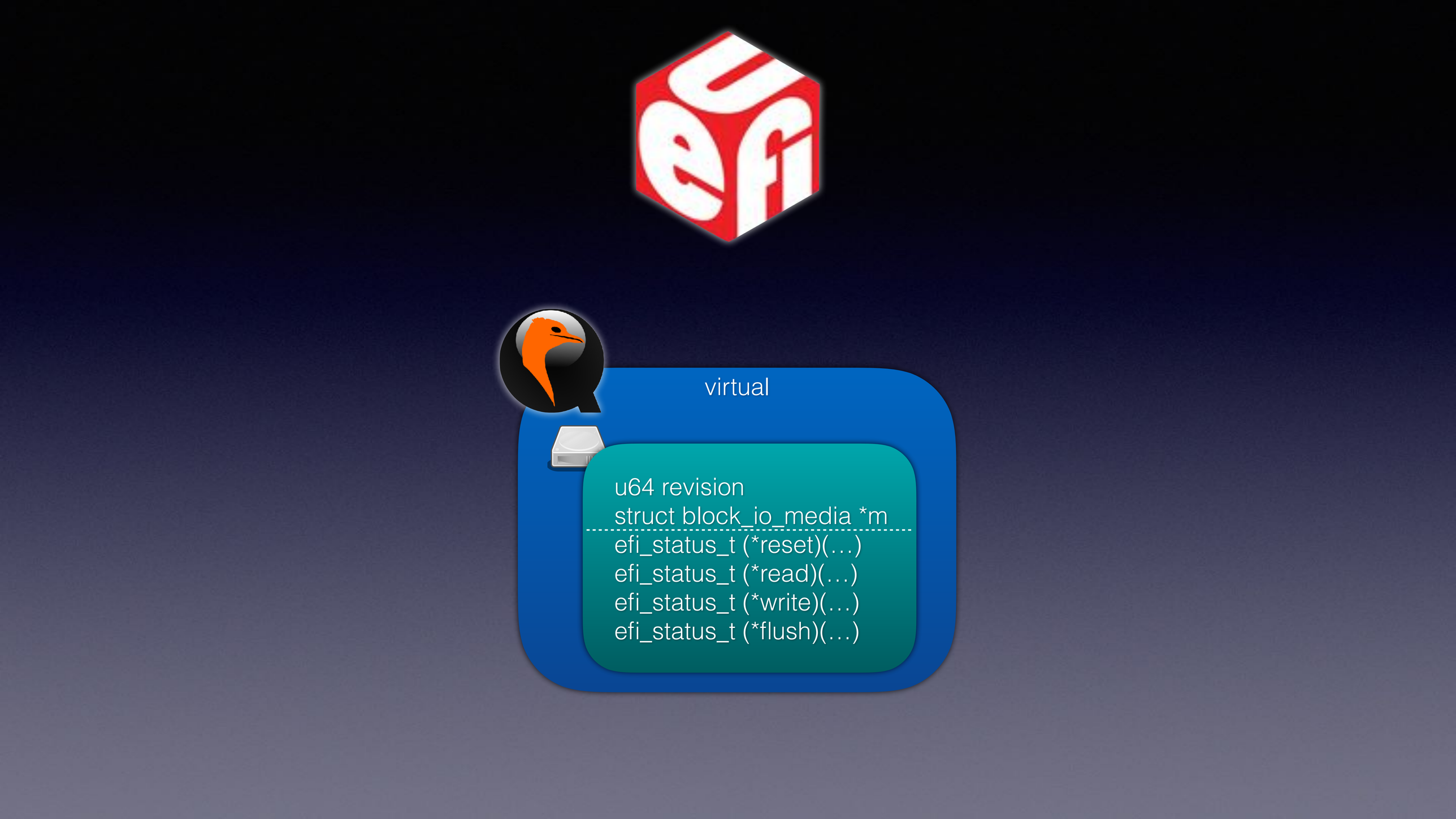

```
u32 id
efi_status_t (*reset)(…)
```

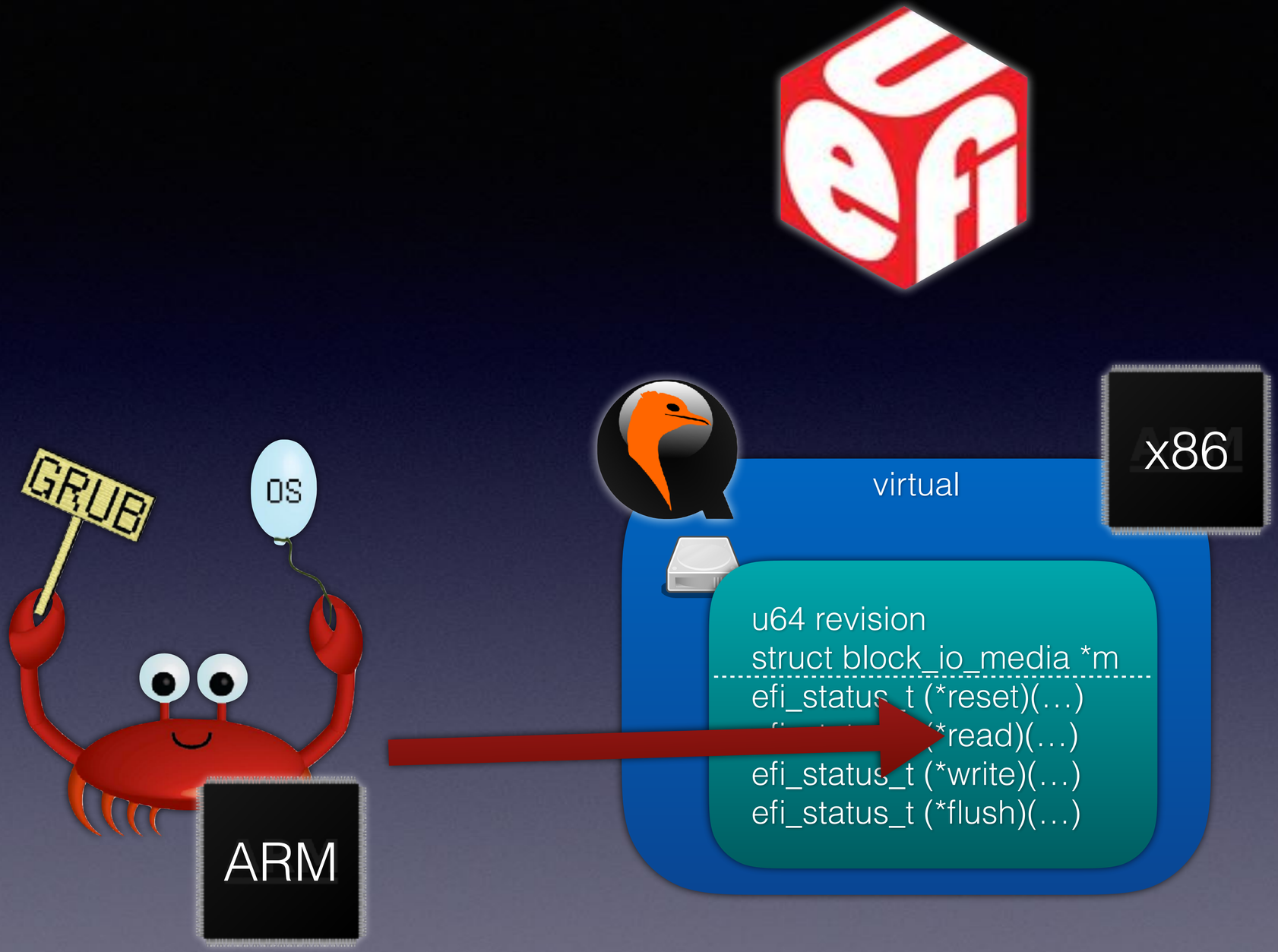
```
0x00:    01 00 00 00 XX XX XX XX
0x08:    f0 ee dd cc bb aa 99 88
```

ARM x86

u32 id
efi_status_t (*reset)(…)

```
0x00:    01 00 00 00 XX XX XX XX
0x08:    f0 ee dd cc bb aa 99 88
```


virtual
u64 revision
struct block_io_media *m
efi_status_t (*reset)(…)
efi_status_t (*read)(…)
efi_status_t (*write)(…)
efi_status_t (*flush)(…)


GRUB
OS
ARM
virtual
x86
u64 revision
struct block_io_media *m
efi_status_t (*reset)(…)
*read)(…)
efi_status_t (*write)(…)
efi_status_t (*flush)(…)

efi_status_t (*read)(struct block_io *this, u32 media_id, u64 lba, ulong buffer_size, void*buffer)

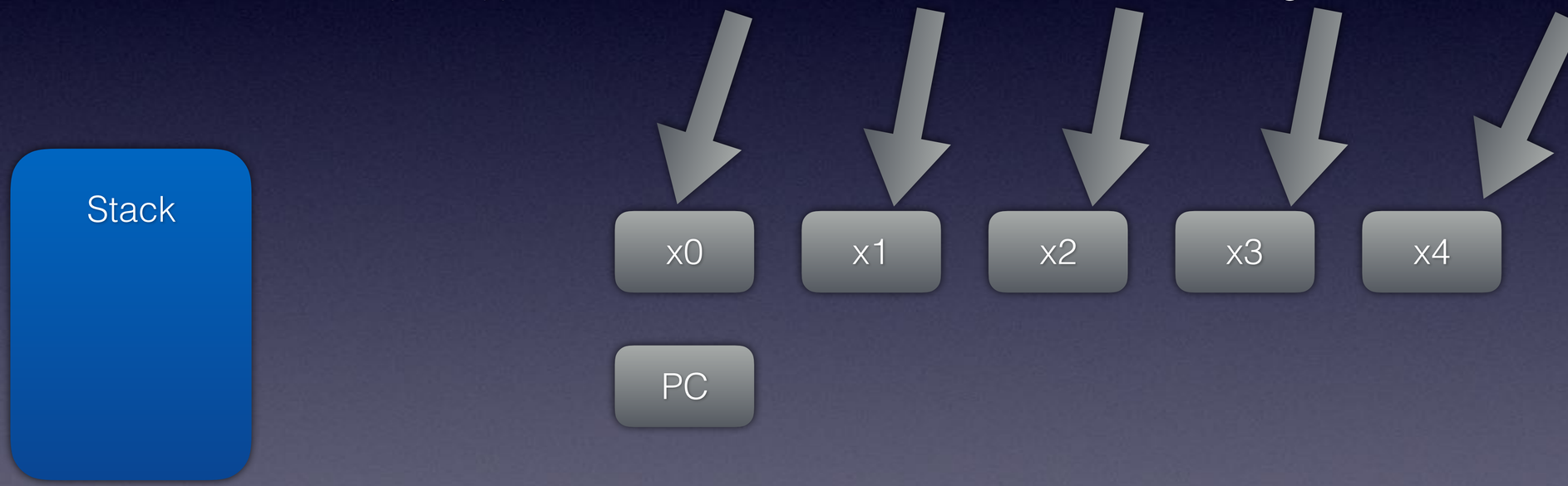
efi_status_t (*read)(struct block_io *this, u32 media_id, u64 lba, ulong buffer_size, void*buffer)

Stack

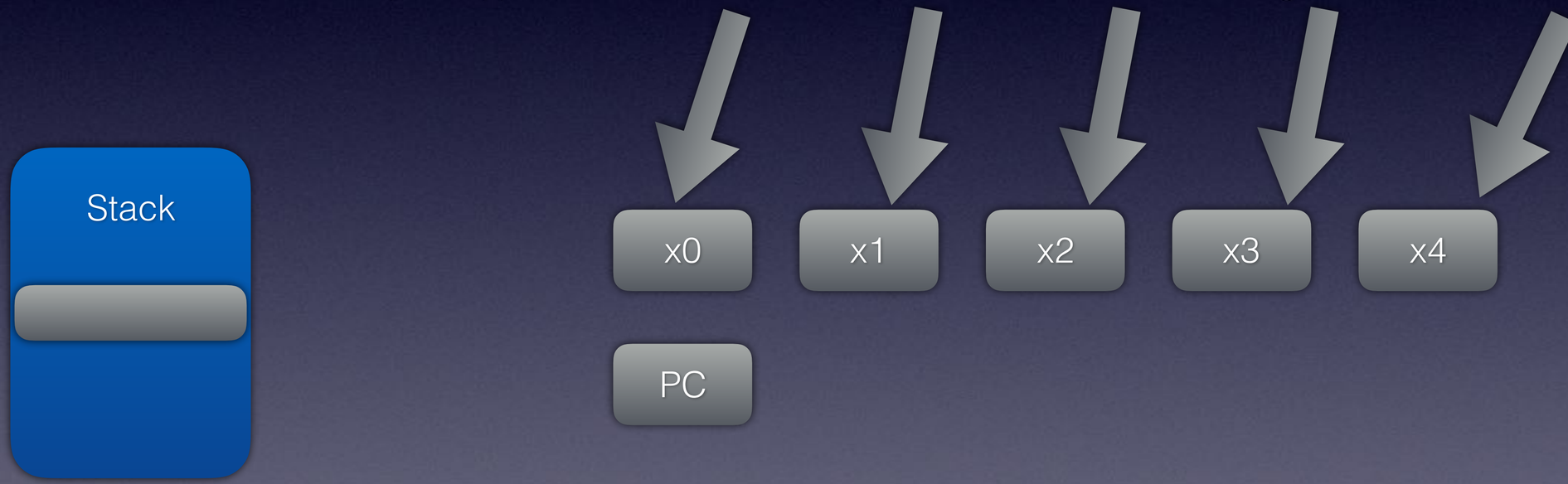
x0 x1 x2 x3 x4

PC

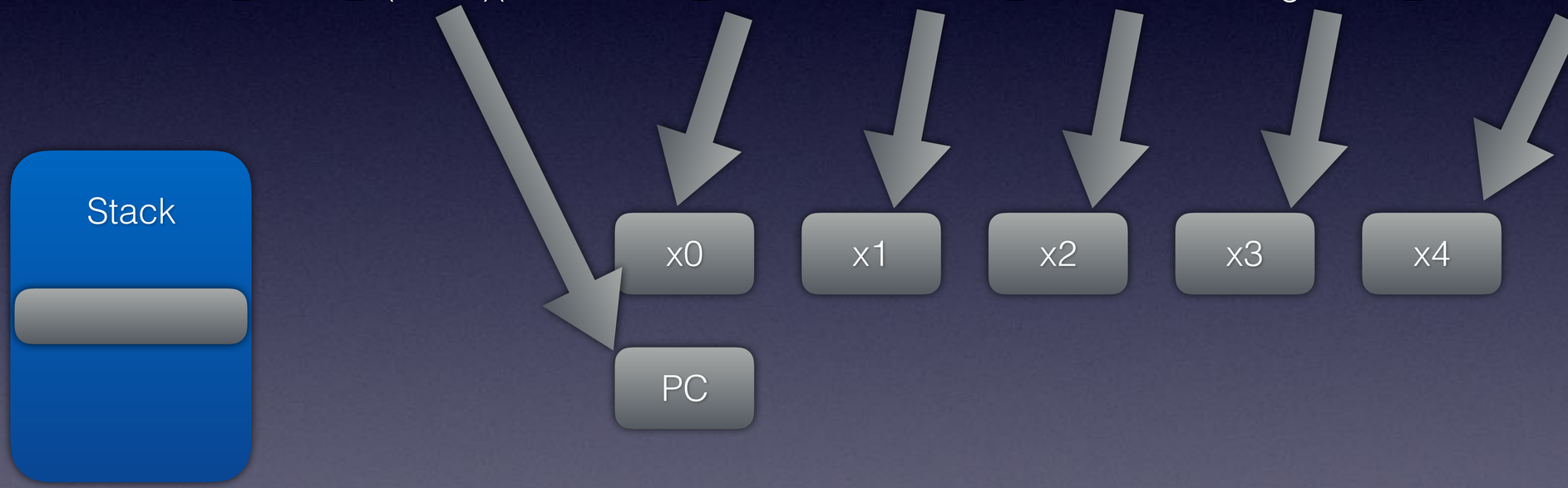
efi_status_t (*read)(struct block_io *this, u32 media_id, u64 lba, ulong buffer_size, void*buffer)

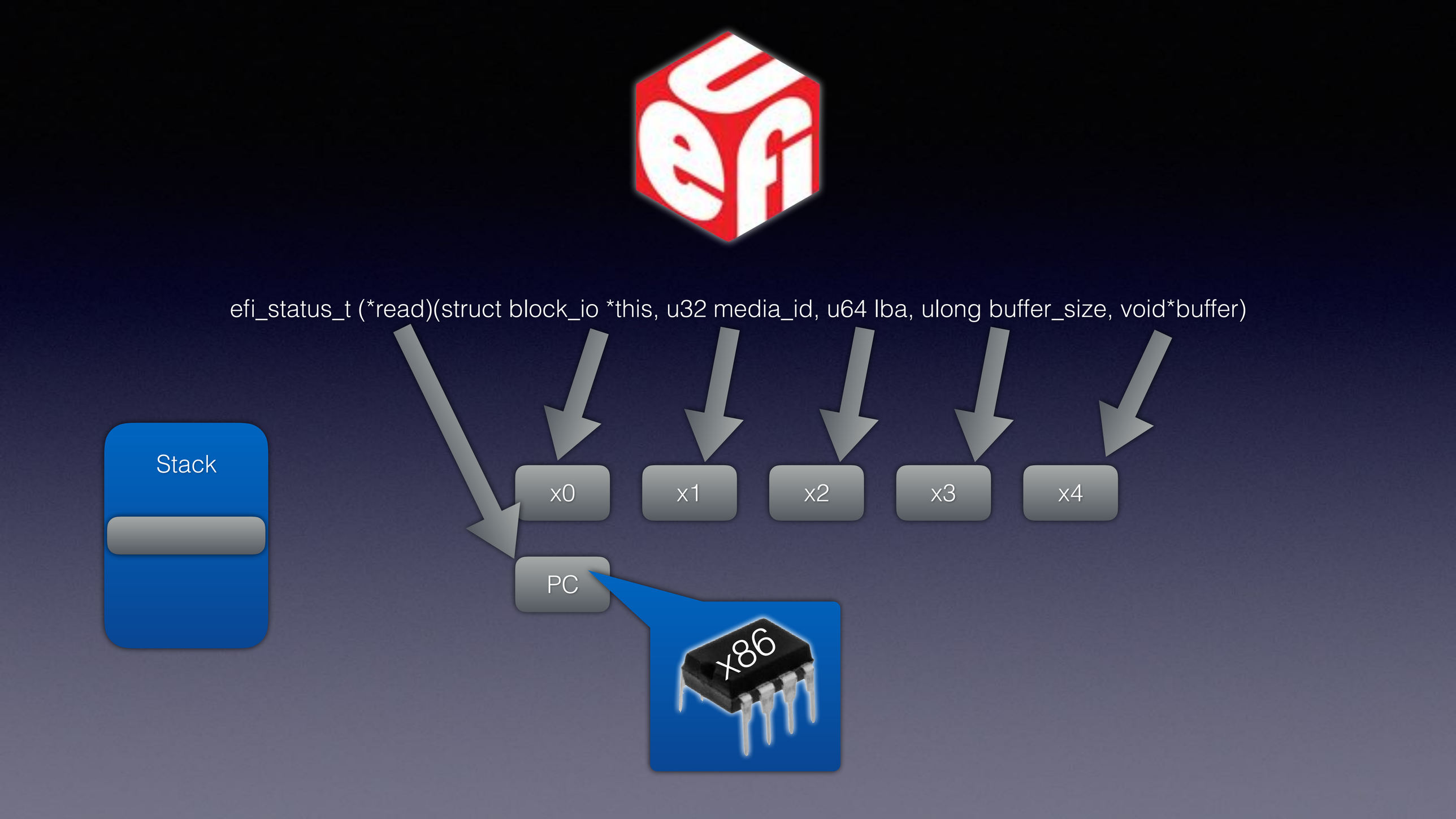


efi_status_t (*read)(struct block_io *this, u32 media_id, u64 lba, ulong buffer_size, void*buffer)

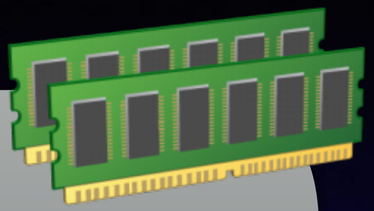


efi_status_t (*read)(struct block_io *this, u32 media_id, u64 lba, ulong buffer_size, void*buffer)

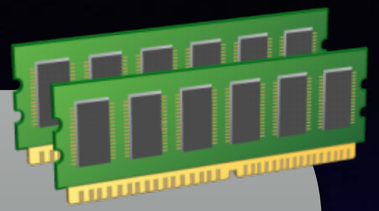



efi_status_t (*read)(struct block_io *this, u32 media_id, u64 lba, ulong buffer_size, void*buffer)
Stack
x0
x1
x2
x3
x4
PC
x86

NX



NX



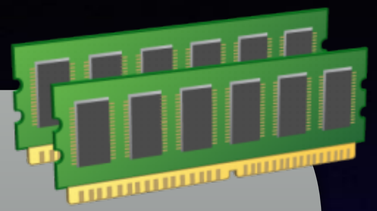
read, write, execute

read, write, execute

read, write, execute

read, write, execute

NX

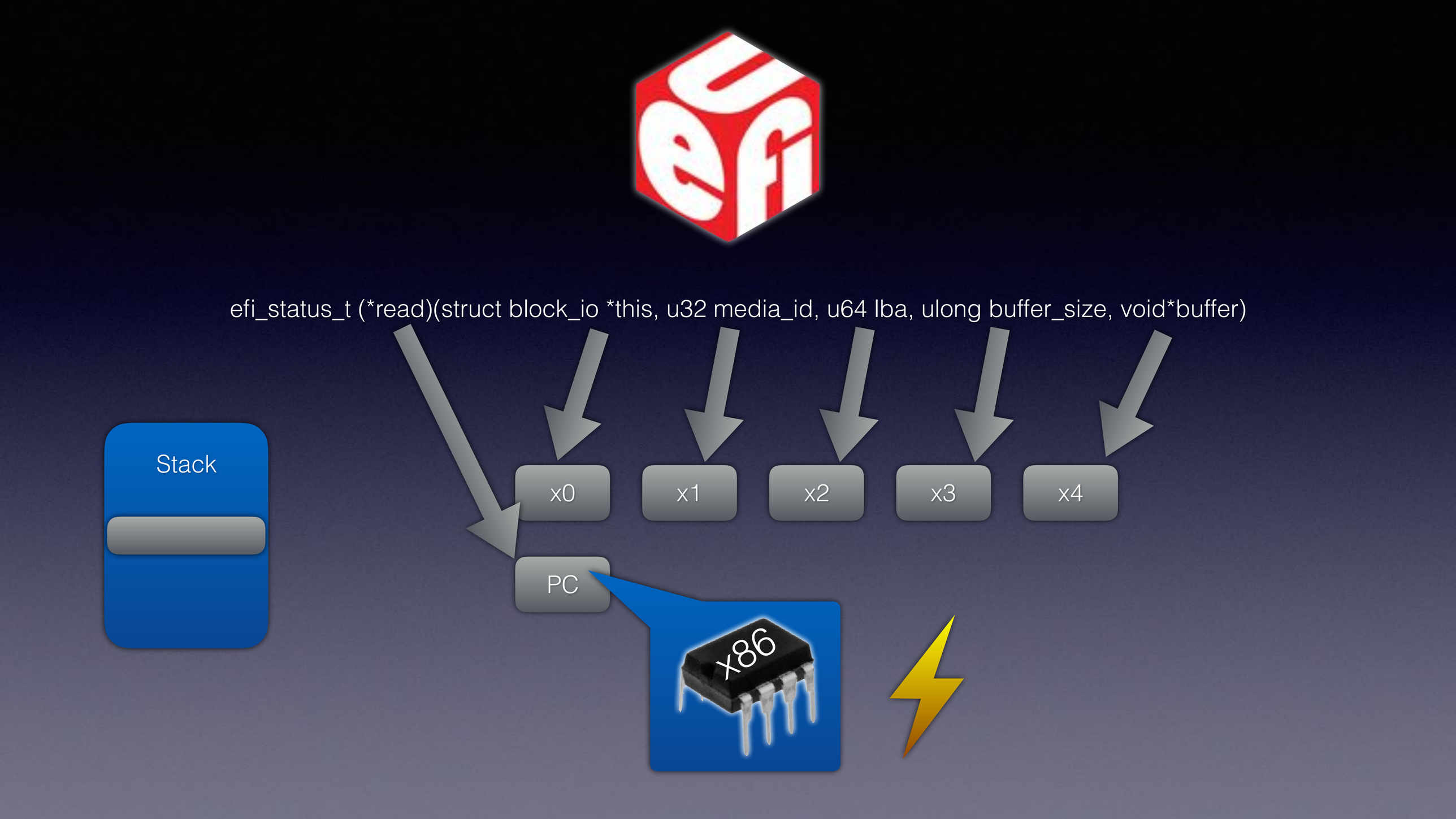


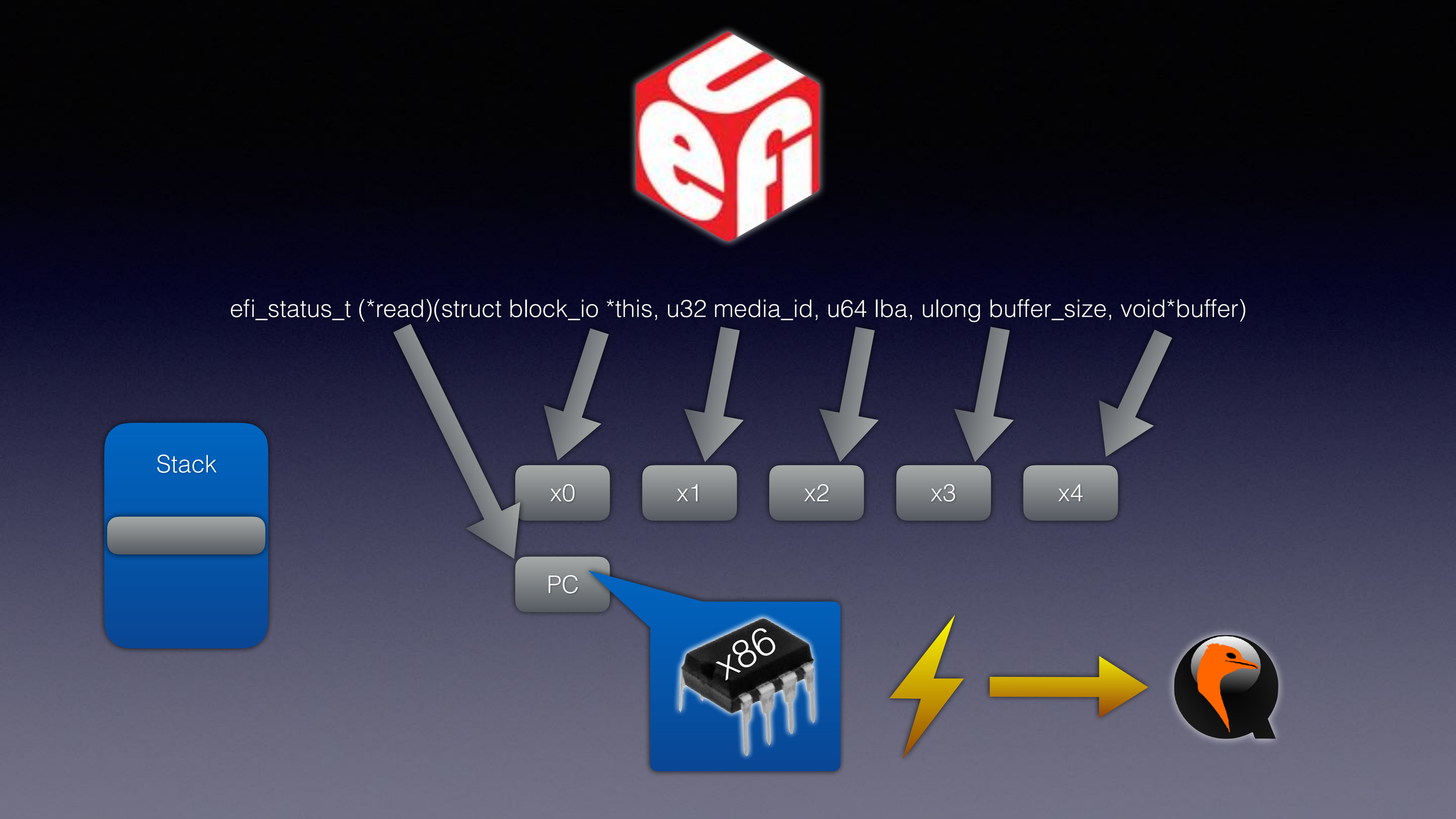
read, write, execute

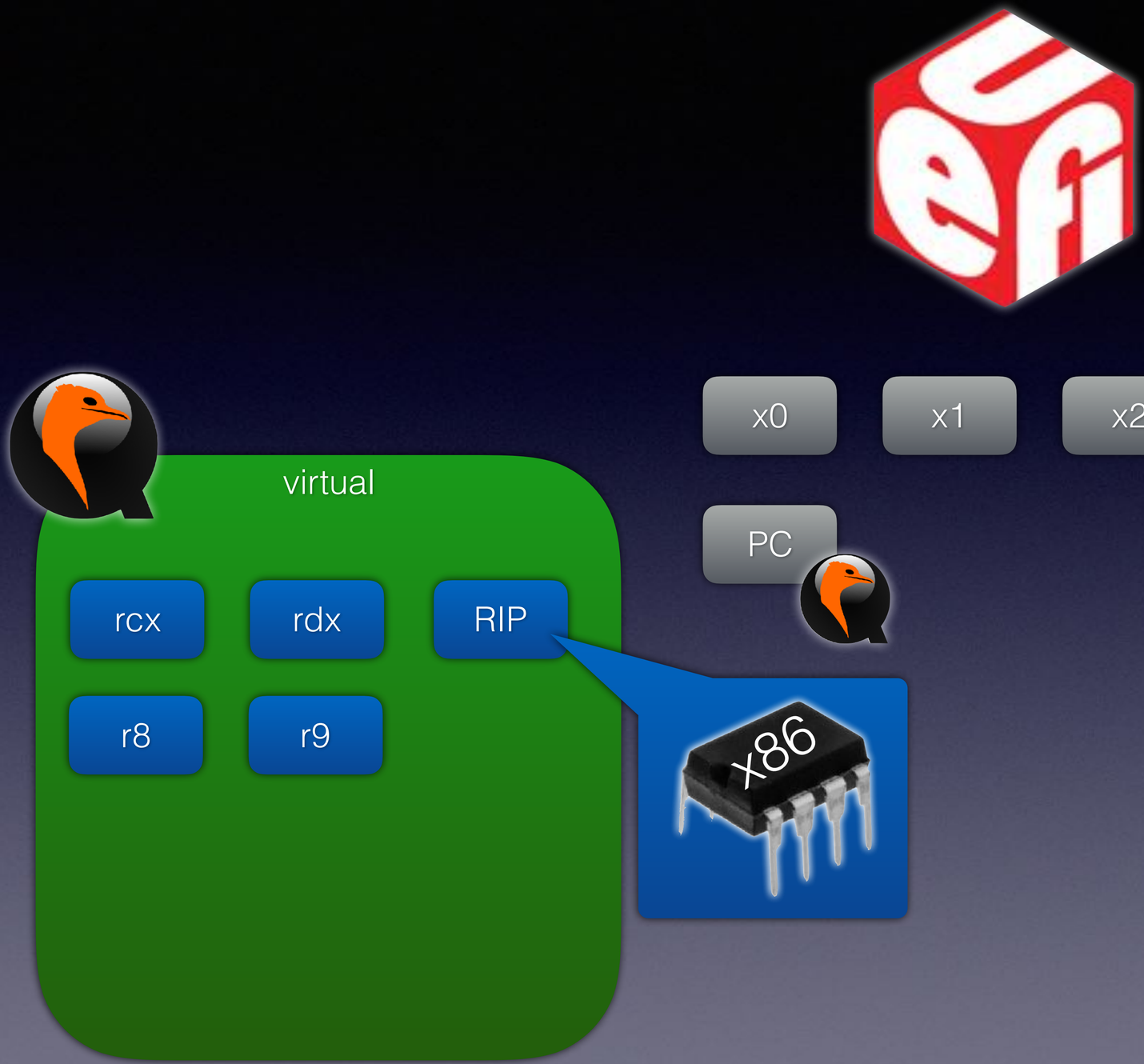
read, write

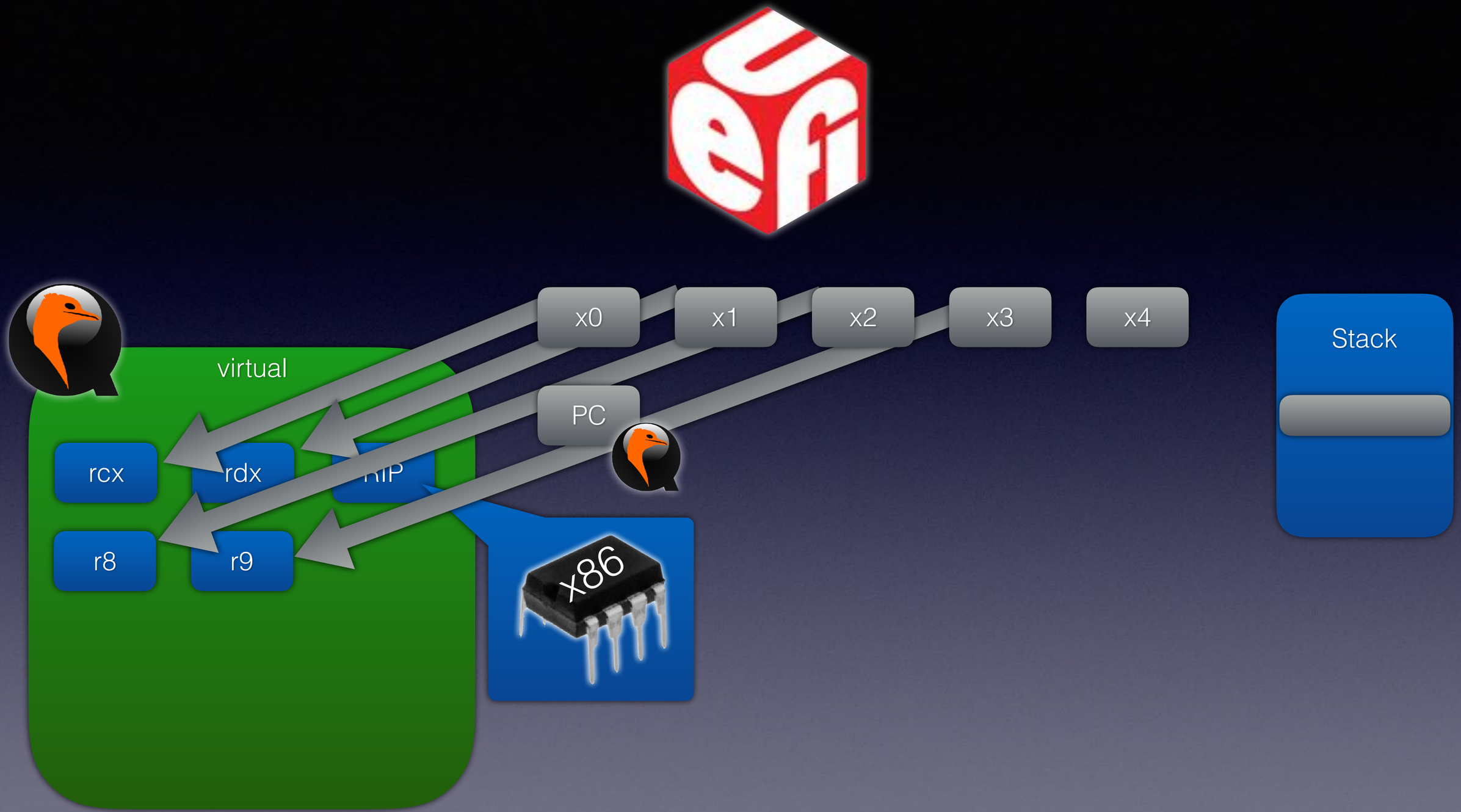
read, write, execute

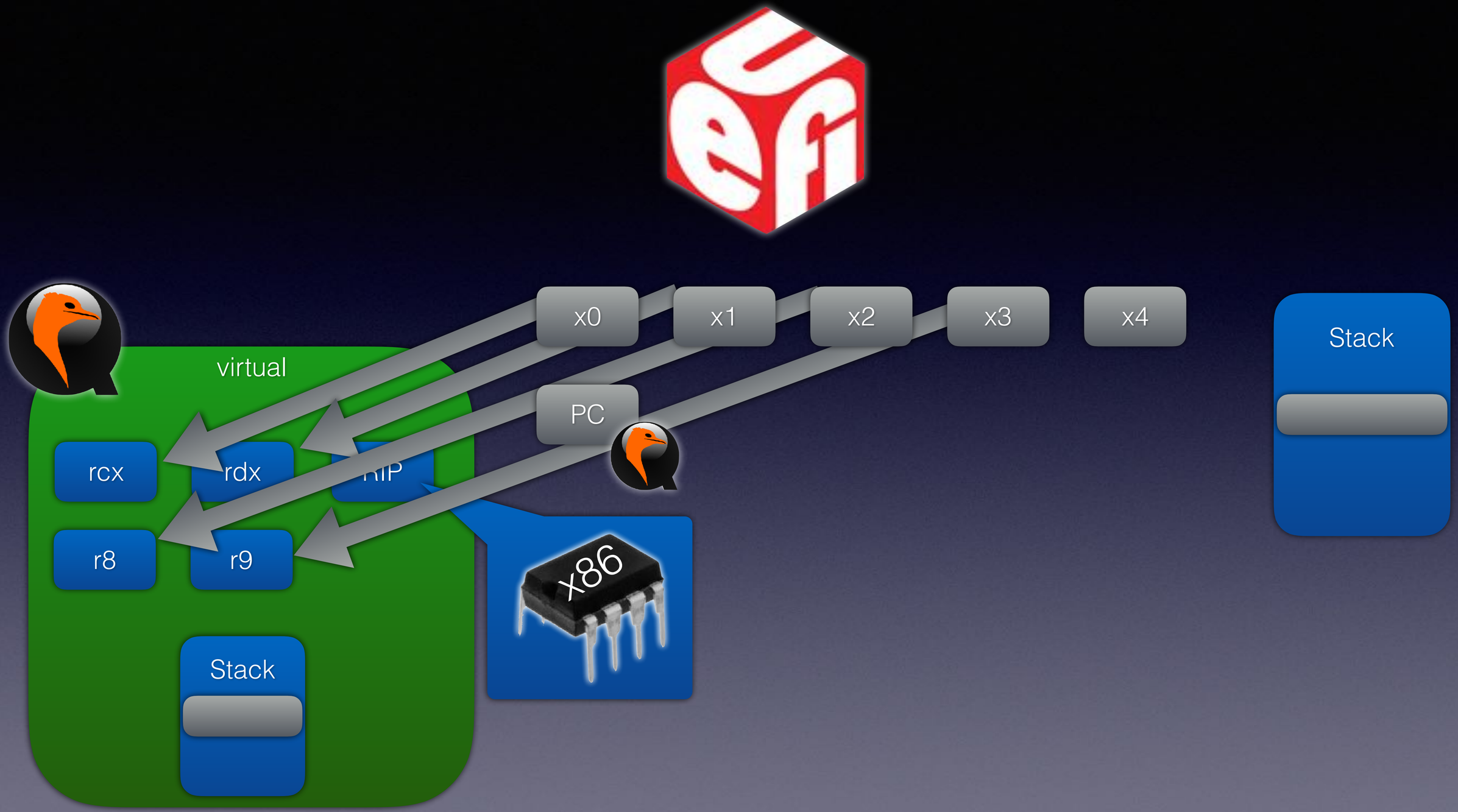
read, write, execute

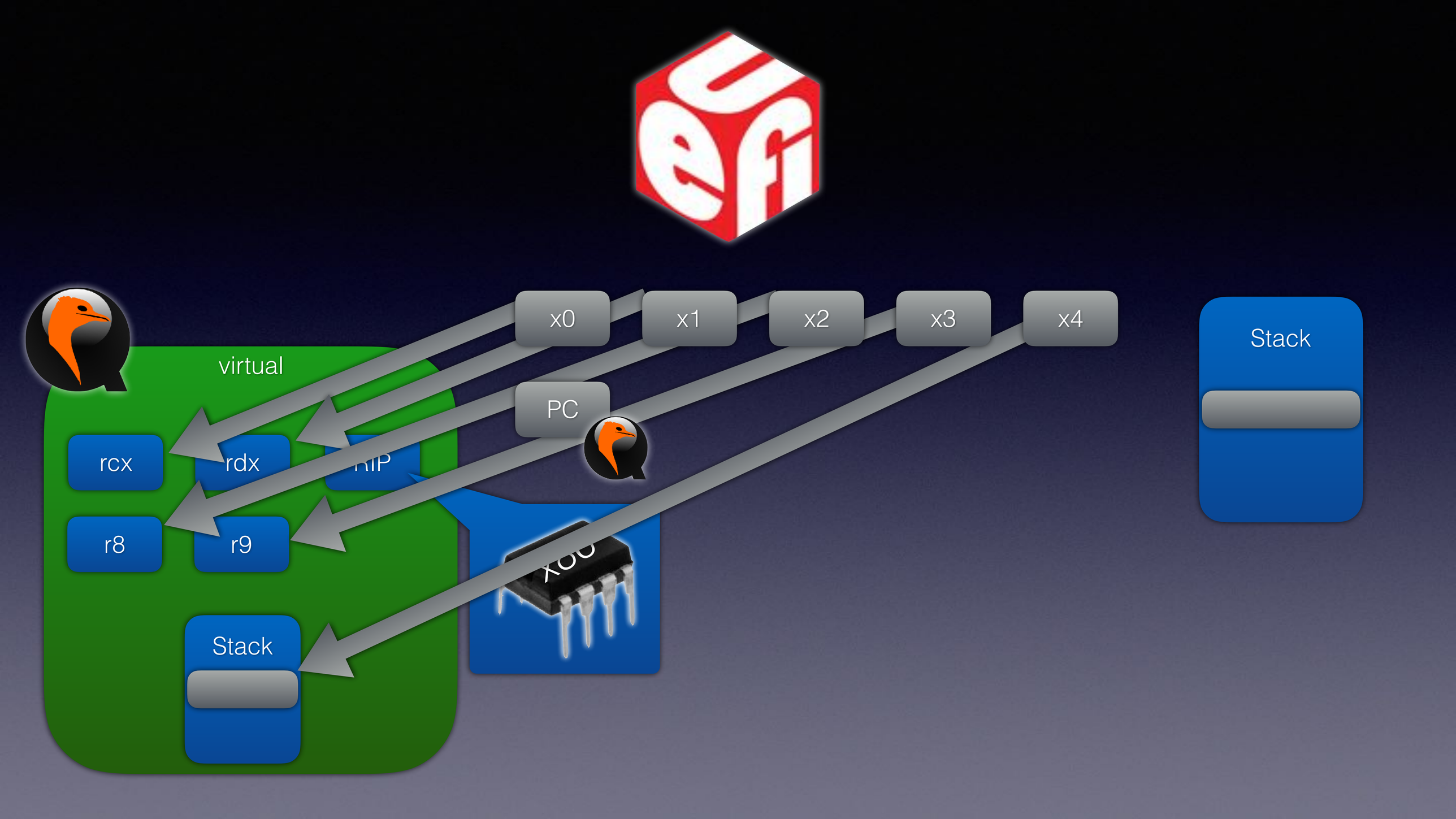

efi_status_t (*read)(struct block_io *this, u32 media_id, u64 lba, ulong buffer_size, void*buffer)
Stack
x0
x1
x2
x3
x4
PC
x86


efi_status_t (*read)(struct block_io *this, u32 media_id, u64 lba, ulong buffer_size, void*buffer)
Stack
x0
x1
x2
x3
x4
PC
x86


x0
x1
x2
x3
x4
Stack
PC
virtual
rcx
rdx
RIP
r8
r9
x86


x0
x1
x2
x3
x4
Stack
virtual
PC
rcx
rdx
RIP
r8
r9
x86

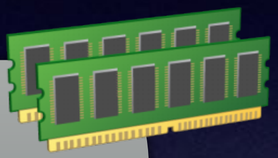

x0
x1
x2
x3
x4
Stack
virtual
PC
rcx
rdx
RIP
r8
r9
x86
Stack


x0
x1
x2
x3
x4
Stack
virtual
PC
rcx
rdx
r8
r9
Stack


virtual
x86

virtual

x86



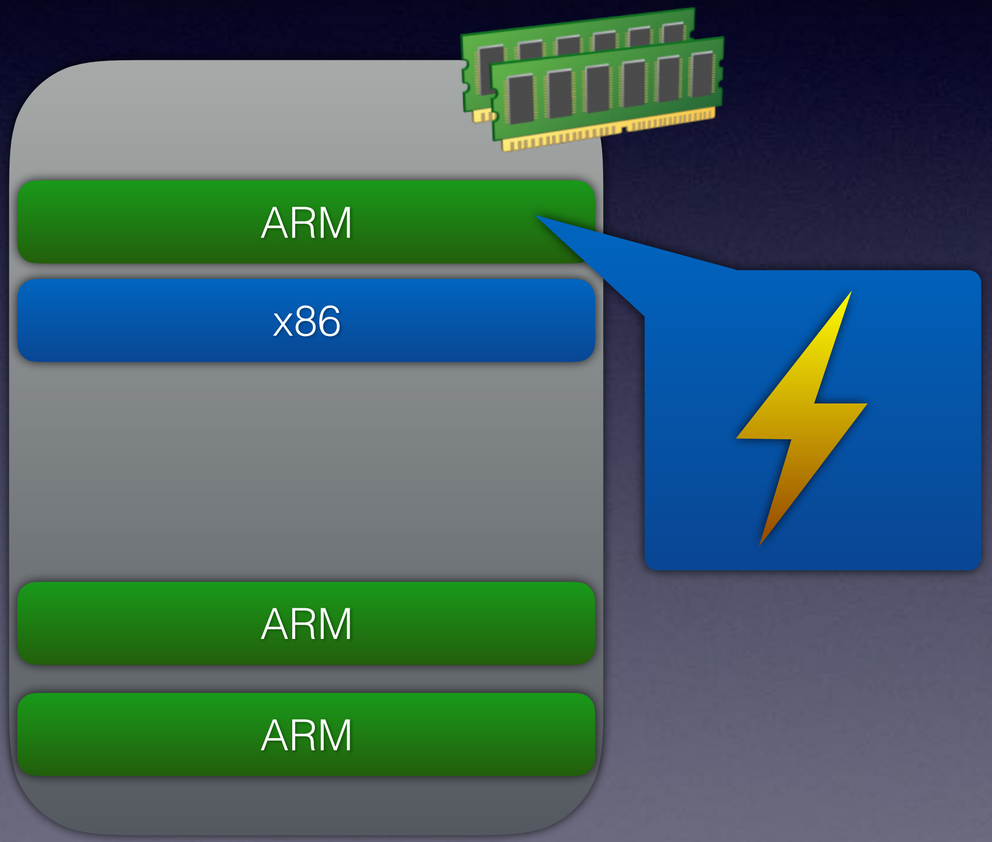
ARM

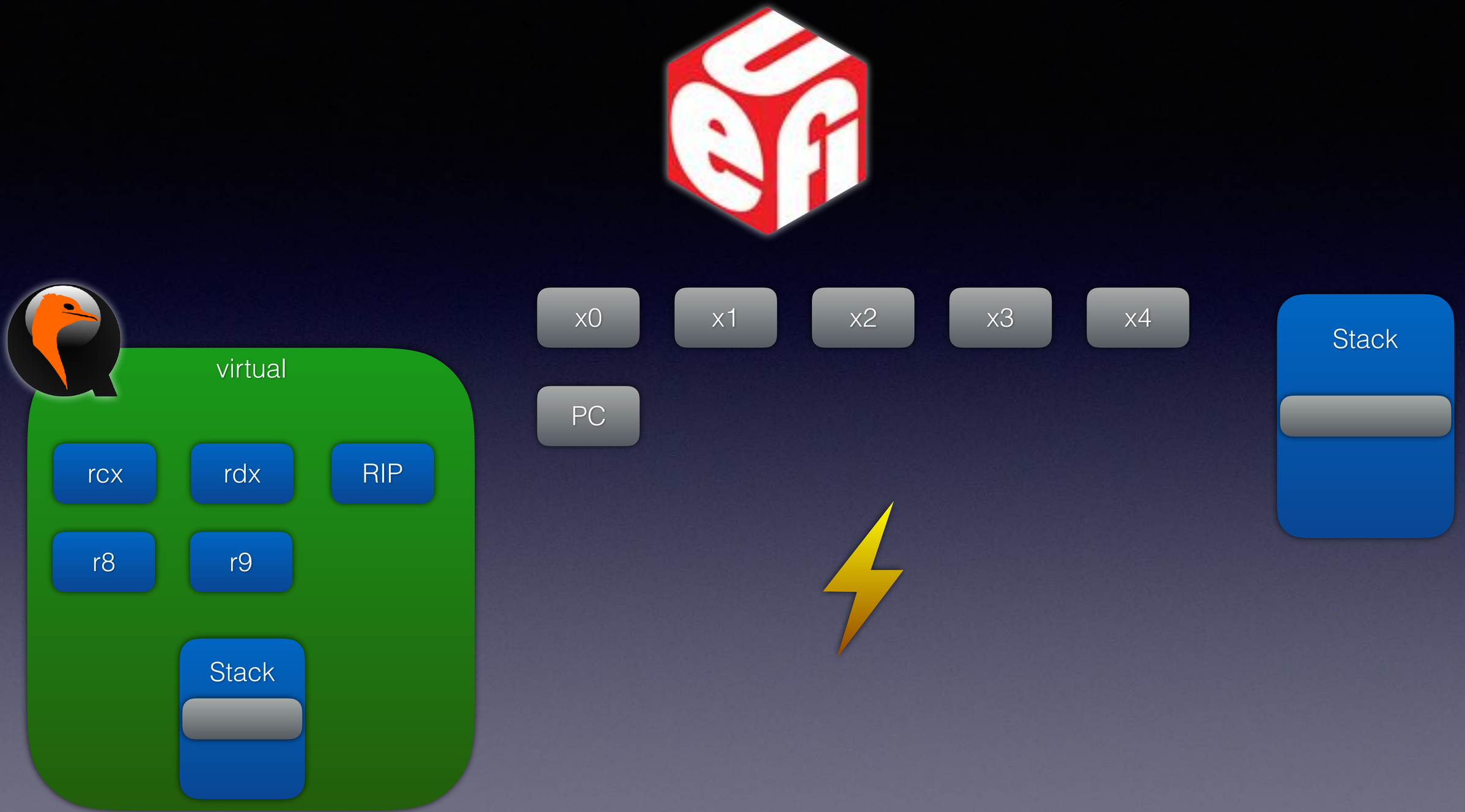
x86

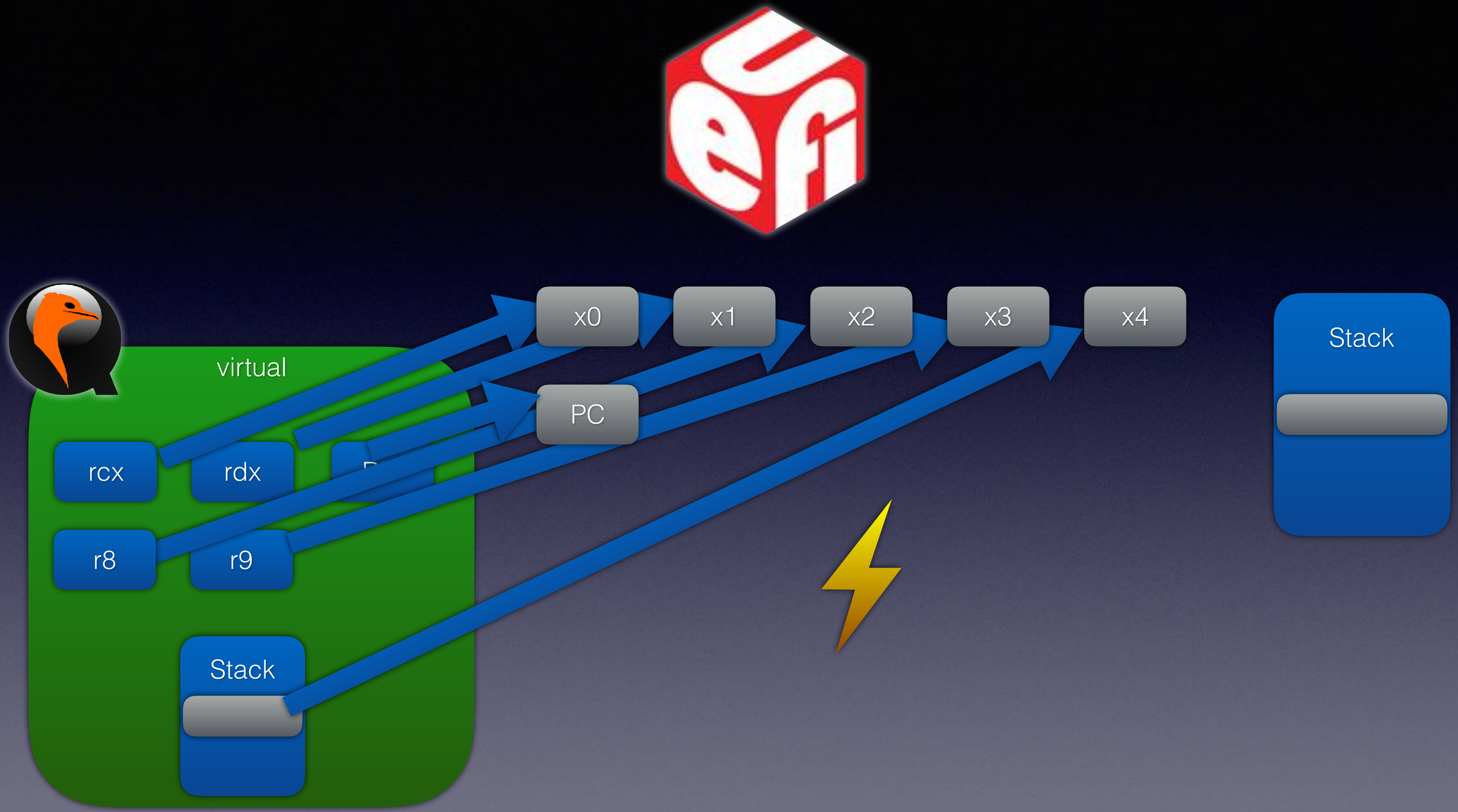
ARM

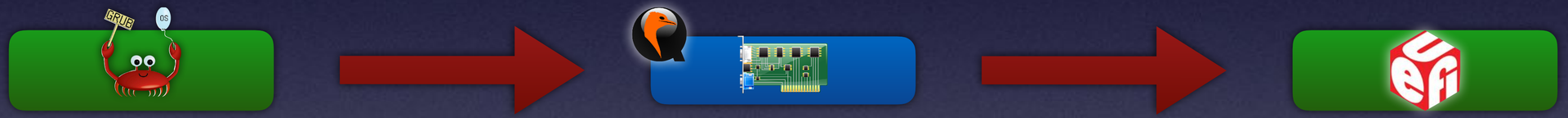
ARM

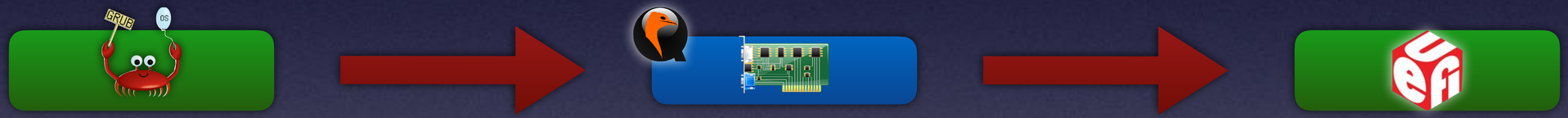

virtual
x86


ARM
x86
ARM
ARM


x0
x1
x2
x3
x4
PC
Stack
virtual
rcx
rdx
RIP
r8
r9
Stack


x0
x1
x2
x3
x4
PC
virtual
rcx
rdx
r8
r9
Stack
Stack


BRUG
OS



function level boundary


SUSE
SLES 12-SP3
Advanced options for SLES 12-SP3
Start bootloader from a read-only snapshot
The highlighted entry will be executed automatically in 3s.

Why TCG

Why TCG

- C
- Isolated
- LGPL
- Supports x86_64

How to Use

How to Use

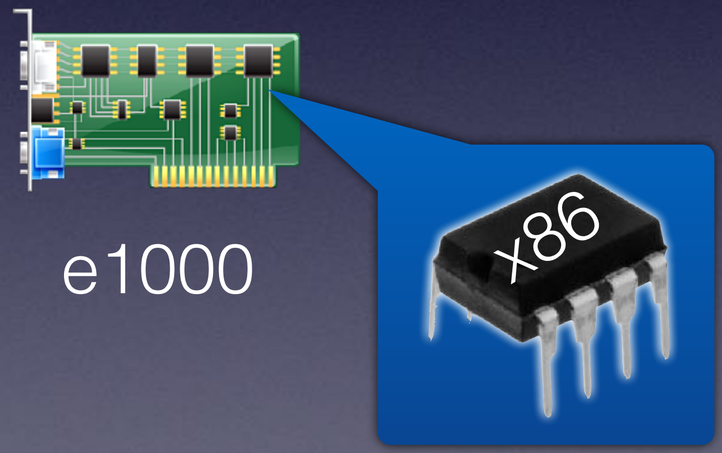
https://github.com/ardbiesheuvel/X86EmulatorPkg.git

How to Use

```
$ git clone https://github.com/ardbiesheuvel/edk2.git
$ cd edk2
$ git checkout origin/x86emu
$ git submodule add https://github.com/ardbiesheuvel/X86EmulatorPkg.git
$ echo "  X86EmulatorPkg/X86Emulator.inf" >> ArmVirtPkg/ArmVirtQemu.dsc
$ echo "  INF X86EmulatorPkg/X86Emulator.inf" >> ArmVirtPkg/ArmVirtQemuFvMain.fdf.inc
$ make -C BaseTools
$ . edksetup.sh
$ export GCC5_AARCH64_PREFIX=... (if you are on a non-aarch64 system)
$ build -a AARCH64 -t GCC5 -p ArmVirtPkg/ArmVirtQemu.dsc -b RELEASE
```

Demo

Demo

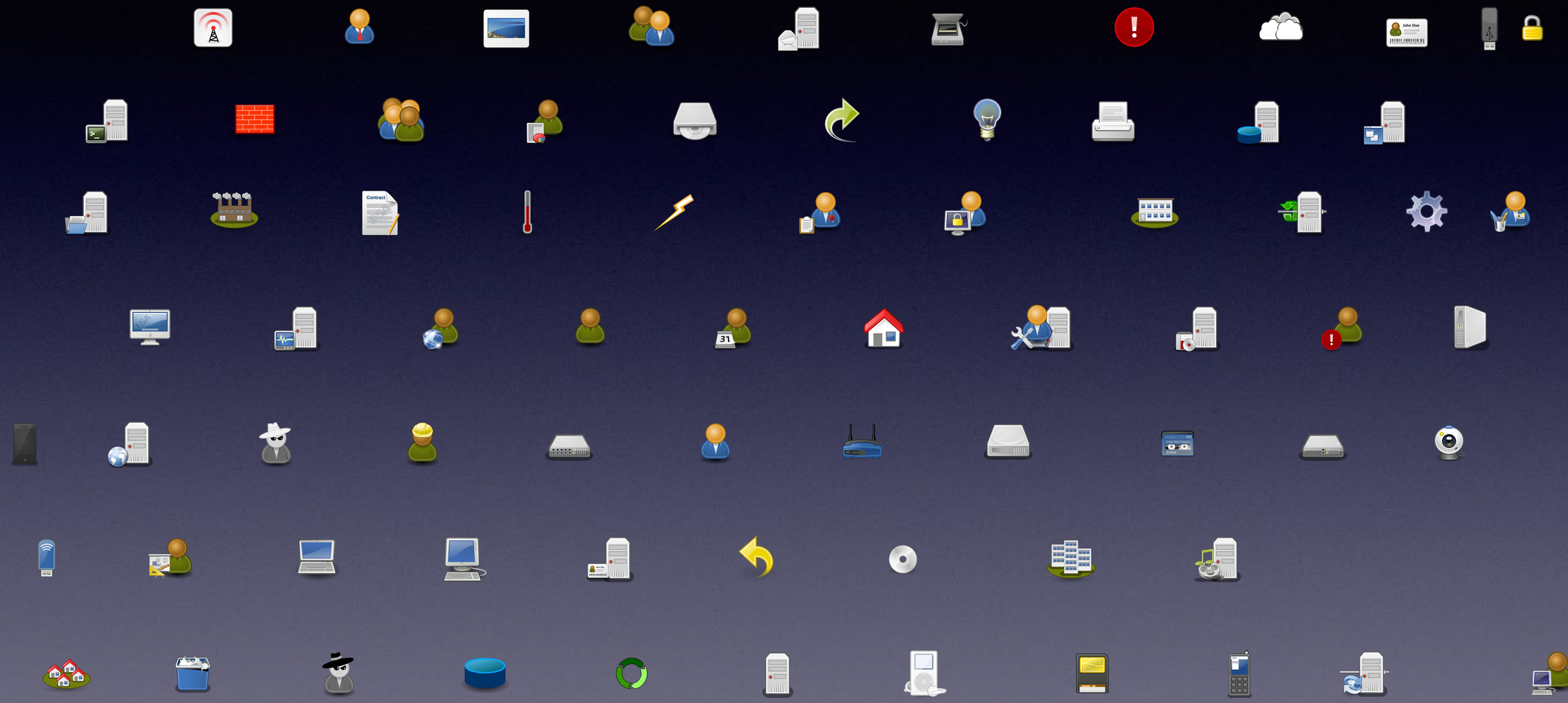


e1000

Backup

- EBC
- Cache Coherency

OSA Icons



Icons received from http://www.opensecurityarchitecture.org/cms/library/icon-library

Other Icons

http://findicons.com/icon/202613/folder_library

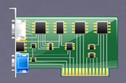
http://findicons.com/icon/download/234261/clock/128/png

http://findicons.com/icon/439269/button_power

https://fosdem.org/2017/schedule/event/grub_new_maintainers/attachments/slides/1768/export/events/attachments/grub_new_maintainers/slides/1768/slides.pdf

https://de.wikipedia.org/wiki/BeagleBoard#/media/File:Beagle_Board_big.jpg

https://thenounproject.com/term/folder-tree/27307/



https://commons.wikimedia.org/wiki/File:Crystal_Project_Hardware.png

emojione Icons



Icons received from http://www.opensecurityarchitecture.org/cms/library/icon-library

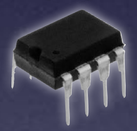
External Sources

https://commons.wikimedia.org/wiki/File:Raspberry_Pi_B%2B_illustration.svg

https://commons.wikimedia.org/wiki/File:Sd-card-1377140.svg

http://eu.mophie.com/shop/media/catalog/product/cache/3/small_image/270x330/9df78eab33525d08d6e5fb8d27136e95/u/s/usb-micro3-40-blk_usb-tip-detail_front-back_540px.jpg

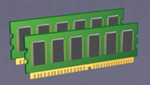
https://commons.wikimedia.org/wiki/File:Circle-icons-submarine.svg



https://commons.wikimedia.org/wiki/File:150-8-DIP.jpg

https://commons.wikimedia.org/wiki/File:Hdd_icon.svg

https://commons.wikimedia.org/wiki/File:ARM_CPU_icon.svg



http://findicons.com/icon/177982/memory#