Low Level Playground

Bruno Cardoso Lopes
João Batista Correa

About...

- O objetivo desta apresentação é introduzir conceitos a respeito de BIOS e PCI, ilustrando seu funcionamento e possíveis formas de exploração destes recursos.
- Fornecer ponteiros?

Agenda

- BIOS
- PCI
- Interagindo com hardware
- Low level Playground

BIOS

Introdução a BIOS

BIOS

- Basic Input and Output System
- Firmware executado durante inicialização do computador
- Provê ambiente de execução necessário para o sistema operacional
- BIOS != CMOS

BIOS Layout

- Processador executa diretamente Bloco de Boot
- BB verifica checksums e descomprime componentes compromidos
- Também possui algum código de testes e de inicialização de hardware

BIOS Layout

- *call* não pode ser executado durante execução do BB. Memória não inicializada, não existe pilha
- RAM ainda não foi testada neste momento
- BB salta p/ system BIOS, que cuida das demais tarefas de inicialização, como POST

POST

- Power On Self Test
- Utiliza estrutura chamada POST Jump Table
- Possui offsets de procedimentos de inicialização que estão no mesmo segmento que a PJT
- Inicializa componentes de hardware (p.e. código de expansion roms de PCIs)

POST Jump Table

E000:6000	dw 1234h	Controlador de Teclado
...	dw 4321h	CPU Flags
...	...	Rotina Dummy
...	...	Vetor de interrupção
E000:6...	...	Controlador de Vídeo

Gerenciador de Interrupções da BIOS

- x86 possui interrupções de hardware e software
- (Advanced) Programmable Interrupt Controller controla hardware interrupt requests
- (A)PIC precisa ser inicializado antes de habilitar qualquer interrupção no sistema
- o (A)PIC é inicializado pelo bloco de boot

PCI

Introdução a PCI

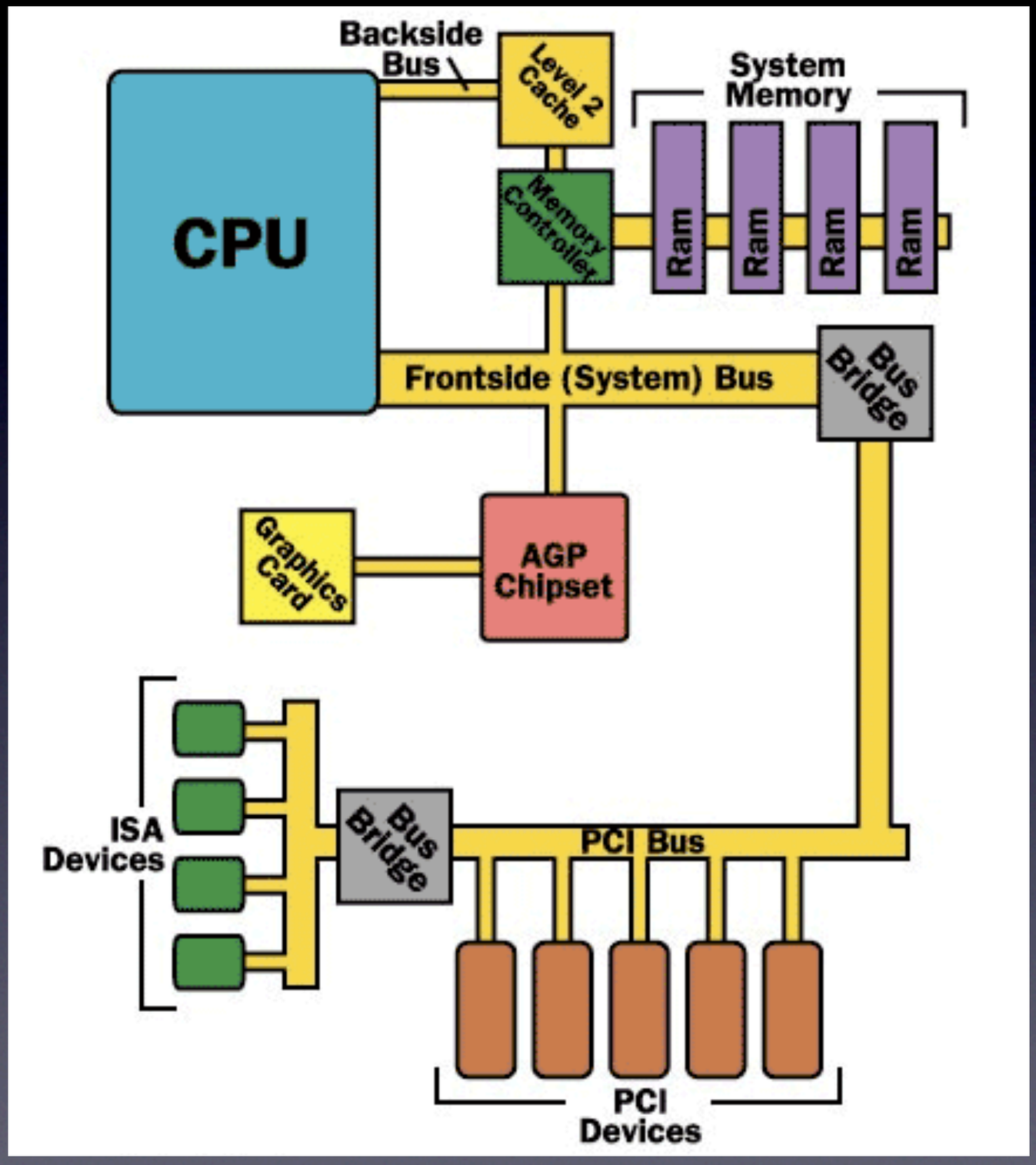
Barramentos

- Bus (Barramento)
- Interliga componentes em um computador
- Permite facil adição/remoção de componentes

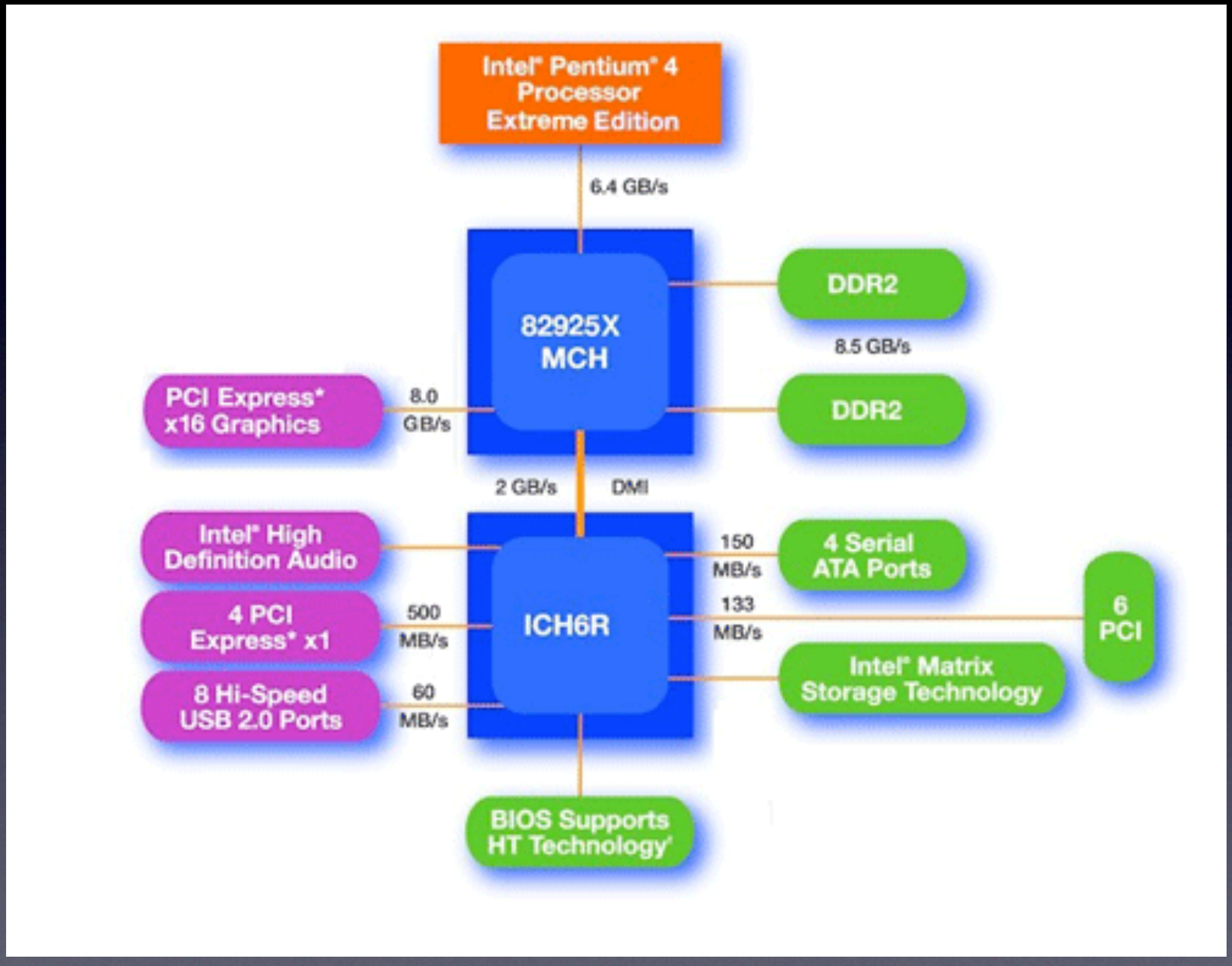
Barramentos

- Tipos de barramentos:
  - System (frontside) bus: interliga processador e memória (mais rápido do sistema)
  - PCI bus, USB, Firewire, ... todos mais lentos que o system bus.
  - Se conectam ao system bus através de bus bridges.

- Frontside bus



- Barramentos

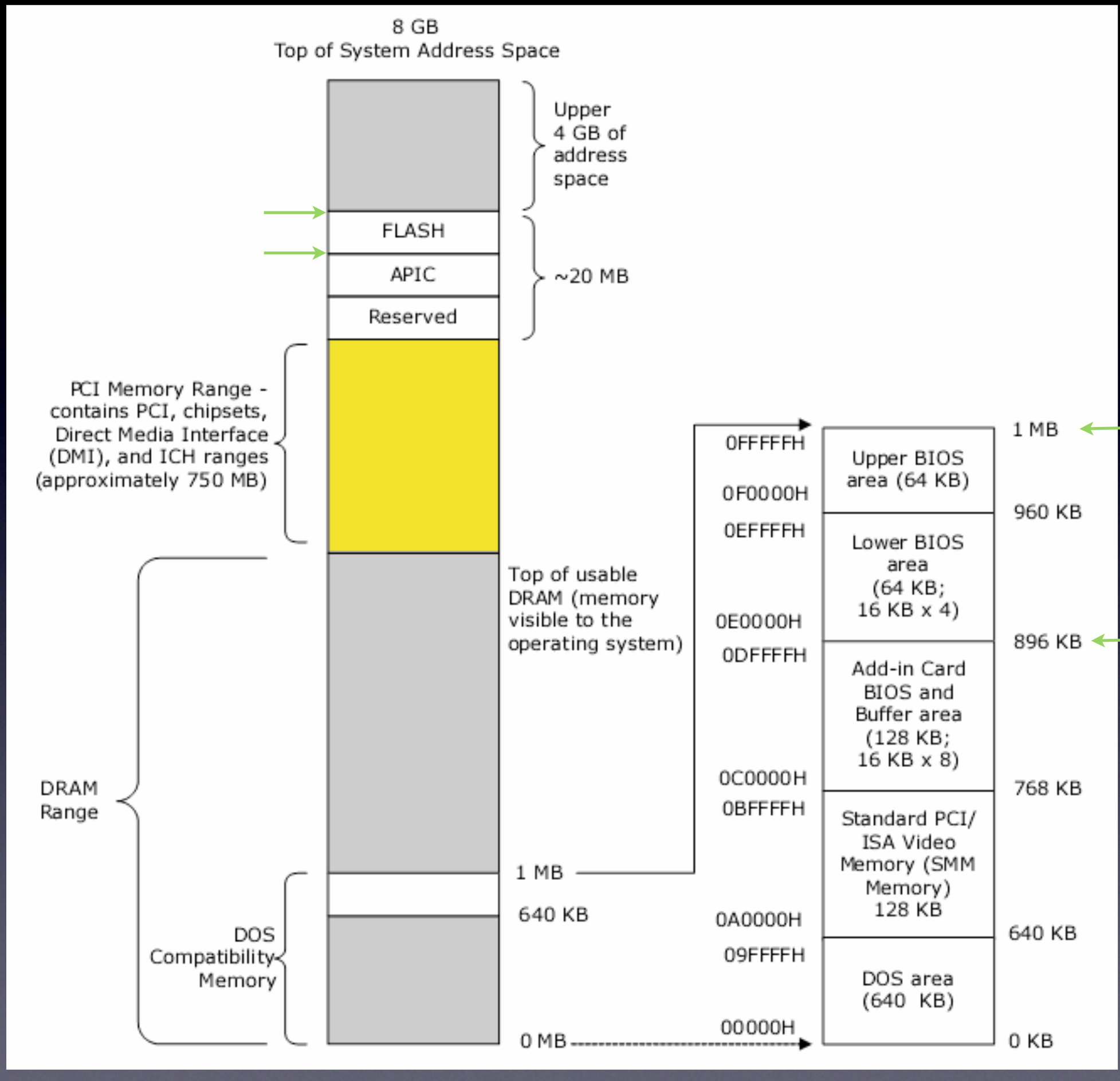


Barramentos

- RAM não é o único recurso que utiliza espaço de endereçamento do processador
- Hardware também é mapeado neste espaço
- Northbridge é responsável pela organização do espaço de endereçamento do sistema, principalmente controlador de memória

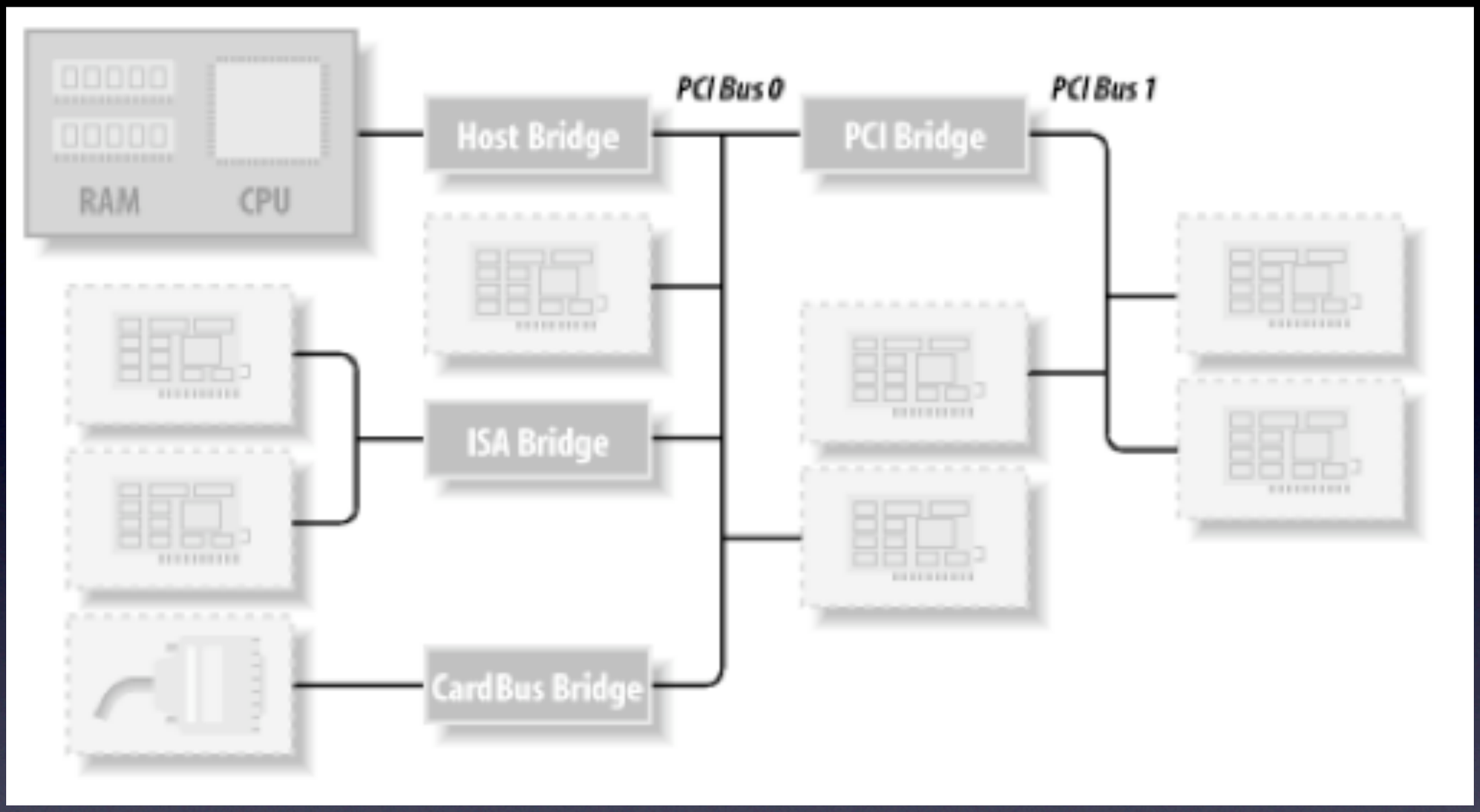
Barramentos

- Hardwares mapeados em memória: devices PCI, PCI-Express, HyperTransport, APIC, VGA e a BIOS ROM.
- Controlador de memória decide onde ler ou escrever uma requisição da CPU
- A operação pode ser encaminhada p/ a RAM, VGA-RAM ou southbridge


8 GB
Top of System Address Space
Upper 4 GB of address space
FLASH
APIC
Reserved
~20 MB
PCI Memory Range - contains PCI, chipsets, Direct Media Interface (DMI), and ICH ranges (approximately 750 MB)
Top of usable DRAM (memory visible to the operating system)
1 MB
640 KB
DRAM Range
DOS Compatibility Memory
0 MB
0FFFFFH
0F0000H
0EFFFFH
0E0000H
0DFFFFH
0C0000H
0BFFFFH
0A0000H
09FFFFH
00000H
1 MB
960 KB
896 KB
768 KB
640 KB
0 KB
Upper BIOS area (64 KB)
Lower BIOS area (64 KB; 16 KB x 4)
Add-in Card BIOS and Buffer area (128 KB; 16 KB x 8)
Standard PCI/ ISA Video Memory (SMM Memory) 128 KB
DOS area (640 KB)

PCI

- Barramentos de 32 ou 64 bits.
- PCI e PCI-Express são mapeados para a mesma região: não pode haver overlap.
- PCI: 256 buses, 32 devices por bus, e 8 funções por device.
- 2 barramentos PCI são conectados através de uma PCI-to-PCI bridge.

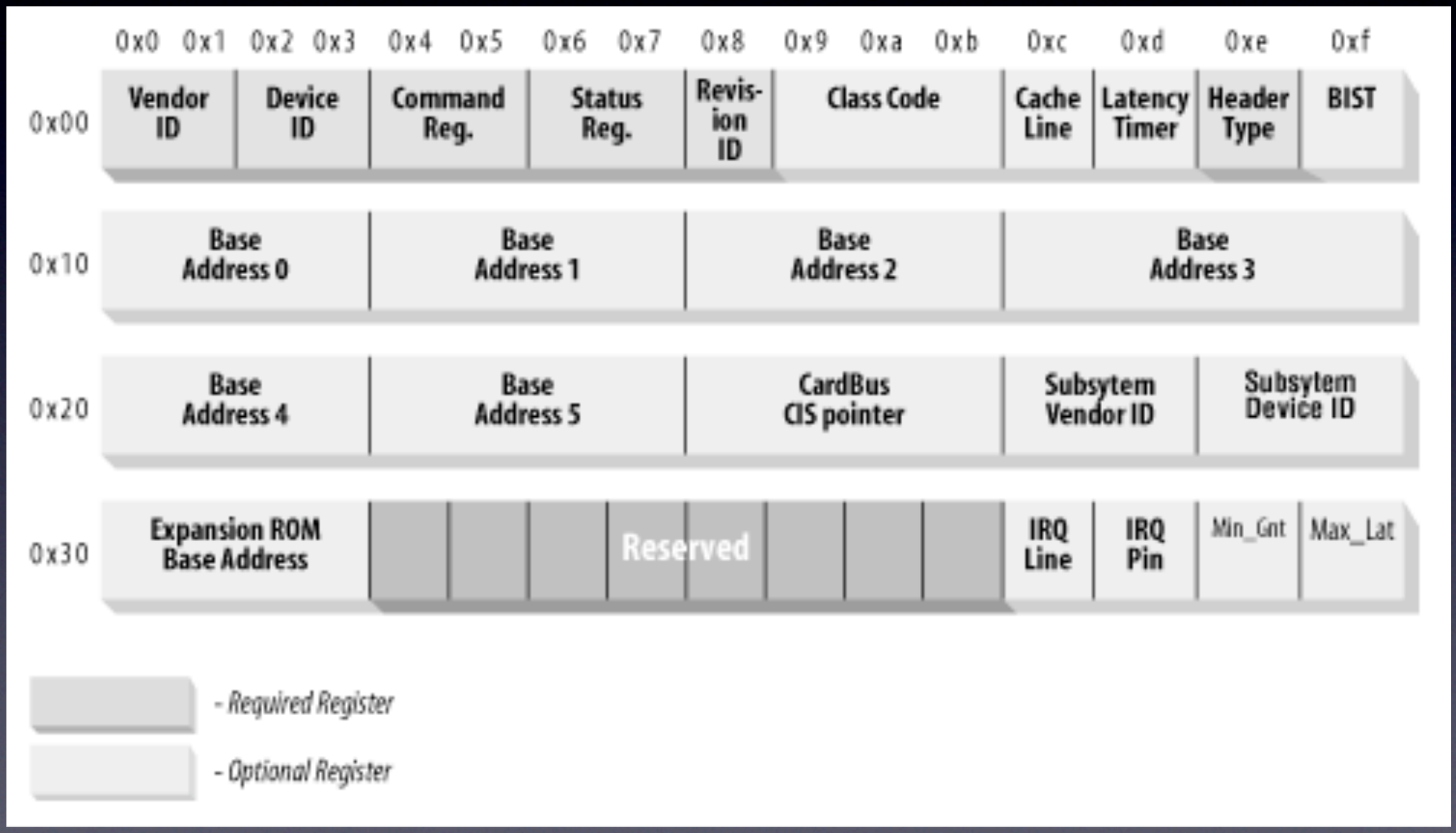


- Host Brigde: Interliga frontside bus ao barramento PCI.

PCI

- Registradores de configuração (256 bytes) para cada função em um device
- Cada processador define a maneira de acesso a estes registradores: Mapeado em memória ou através de I/O
- x86:
  - 0xCF8-0xCFB: porta de end
  - 0xCFC-0xCFF: porta de dados

- Registradores de configuração PCI



- Registradores de configuração para uma bridge PCI-to-PCI

31 — 16	15 — 0	
Device ID	Vendor ID	00h
Status	Command	04h
Class code	Revision ID	08h
BIST / Header type	Latency timer / Cache line size	0Ch
Base address registers		10h 14h
Secondary latency timer / Subordinate bus number / Secondary bus number / Primary bus number		18h
Secondary status	I/O limit / I/O base	1Ch
Memory limit	Memory base	20h
Prefetchable memory limit	Prefetchable memory base	24h
Prefetchable base upper 32 bits		28h
Prefetchable limit upper 32 bits		2Ch
I/O limit upper 16 bits	I/O base upper 16 bits	30h
Reserved		34h
Expansion ROM base address		38h
Bridge control / Interrupt pin / Interrupt line		3Ch

PCI

bus device function

0000:03:00.0 Ethernet controller [0200]: Marvell Technology Group Ltd. Device [11ab:436a] (rev 13)
Subsystem: Marvell Technology Group Ltd. Device [11ab:00ba]
...
Memory at 0000000090400000 (64-bit, non-prefetchable)
I/O ports at 2000 [disabled]
Expansion ROM at fffe0000 [disabled]
....

```
00: ab 11 6a 43 06 04 10 00 13 00 00 02 40 00 00 00
10: 04 00 40 90 00 00 00 00 01 20 00 00 00 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 ab 11 ba 00
30: 00 00 fe  ff   48 00 00 00 00 00 00 00 11 01 00 00
```

→ Registradores do espaço de configuração

PCI

- Relembrando...
- I/O Mapped:
  - in reg, reg : lê do I/O
  - out reg, reg : escreve no I/O
- Memory mapped:
  - leitura/escrita do hardware realizada de maneira transparente através da memória (endereços geralmente marcados como non-cacheable)

PCI

- Endereçamento de um registrador PCI : Expansion ROM Base Address

Bit	Significado	Expansion ROM
0-1	Não usado - alinhamento	“00”
2-7	Offset no espaço de configuração	“11 0000” = 30h
8-10	Function	“000”
11-15	Device	“0 0000”
16-23	Bus	“0000 0011” = 3h
24-30	RESERVED	“000 0000”
31	Enable Bit	“1”

PCI

- Endereçamento final: 0x800300c0
- Exemplo de leitura/escrita:

```
pushad
mov edx, 800300c0h

; Envia o endereçamento do registrador PCI
pela porta de endereço
mov bx, 0CF8h
out bx, edx
```

PCI

```
; lê da porta de dados PCI obtendo o conteúdo do
; registrador acessado
mov bx, 0CFCh
in edx, bx

; Como vimos na saída do lspci, edx conterá o valor
; 0xfffe0000. Escrevemos o mesmo valor de volta
; através da porta de dados.
out bx, edx

popad
ret
```

PCI

- BAR (Base Address Register) contêm os endereços de memória ou I/O onde os devices PCI são mapeados.
- BIOS inicializa o BAR de cada device PCI durante o durante o BOOT.
- O tamanho de memória necessário pelo device é descoberto:
  - Escrita/leitura do BAR
  - Decodificação do valor lido

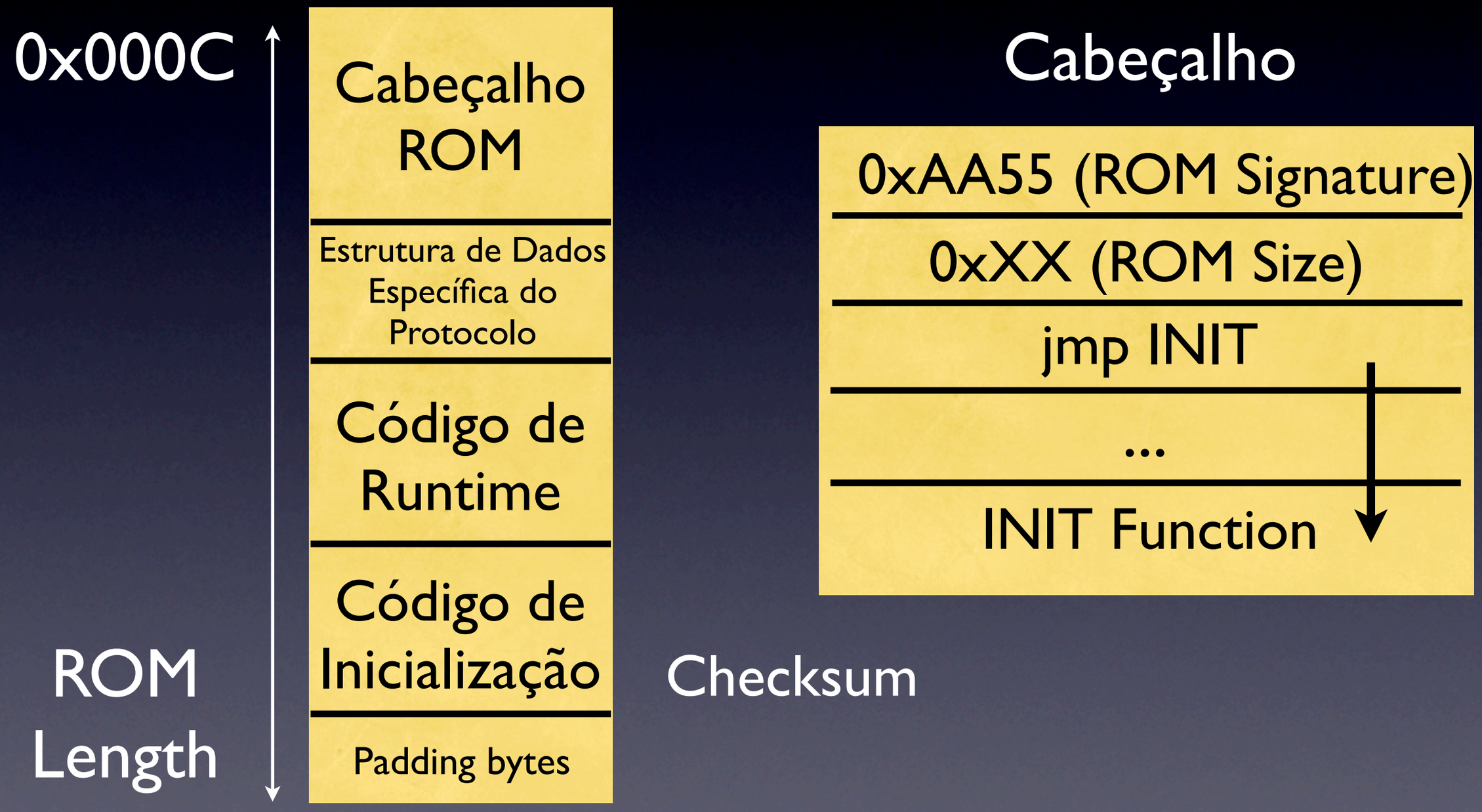
PCI-Express

- Mecanismo semelhante a PCI.
- RCBAR (root complex register block) utilizados para configuração.
  - Mapeado em memória
  - BIOS utiliza o mecanismo tradicional de conf PCI para inicializar o RCBAR.

Expansion ROMs

- Tipo de BIOS embarcada em placas
- Inicializa a placa antes da execução do SO
- Chamadas durante o POST pela BIOS
- Utilizando info presentes na Expansion ROM Base Address.

PCI XROM Layout



POST e PCI

- Carrega código p/ RAM
- Chama função INIT (set +w na memória)
- Utiliza byte no offset 02h p/ saber quanto de memória será necessário

PCI XROM Layout

- Encontrar "jmp INIT" no header
- XROM é chamada com *far call* pela BIOS
- Espera-se que *retf* finalize a XROM
- Codificação em 16 bits (real mode)
- ROM pode conter BEV (*Bootstrap Entry Vector)*, que é executado se o boot é realizado através do dispositivo PCI

INIT Function

- Prepara dispositivos p/ execução
- Escreve dados que serão utilizados pela BIOS ou drivers na memória
- Área de memória não writable em runtime
- Parâmetros - No do barramento, No do dispositivo e No da função

INIT Function

1. Verifica se PCI possui XROMBAR no seu espaço de registradores de config.
2. Se há registrador, POST mapeia e ativa ROM, procura assinatura nos 2 primeiros bytes (0xAA55)
3. POST procura imagem com código e o carrega na RAM

Interagindo com o Hardware

Aprendendo a ler e a escrever...

Manipulando a BIOS

- Entrar em Kernel Mode
- Possível através de LKMs
- Linux Kernel possibilita acesso ao espaço de endereçamento da BIOS através do arquivo virtual /dev/mem

Manipulando a BIOS

- Mapear endereços físicos da BIOS
- Criar ponteiro p/ início da BIOS no espaço virtual de endereçamento do processo
- Utilizar o ponteiro p/ manipular BIOS
- Operações de escrita exigem alguns requisitos

Manipulando a BIOS

- Configurar registradores p/ permitir leitura e escrita no espaço de endereços da BIOS
- Ler identificação do fabricante e do chip p/ determinar método a ser utilizado p/ acessar o conteúdo da BIOS
- Ler e escrever na BIOS conforme especificação do fabricante

Manipulando a BIOS

- LinuxBIOS (Freebios project)
  - Flashrom
    - Realiza Dump e escrita na BIOS
    - libpci fornece funções e estruturas
      - pci_access, pci_filter, pci_dev, flashchip
      - pci_filter_init(), pci_filter_match()...

Acessando PCI XROM

- Específico do fabricante
- Ctflasher (Linux)
- LKM
- Disponível no /proc/
- Suporta poucos devices (poucas placas de rede - Realtek, Via Rhine...)
- Código aberto, reuso de código

Low Level Playground

BIOS Modification

- Antes de se aventurar, é recomendável compreender bem *datasheets* e o funcionamento dos *chipsets*
- Injeção de código na BIOS
- Código será executado durante o boot

BIOS Code Injection

- Patch na POST Jump Table
  - Incluir rotina no binário da BIOS
  - Alterar um Jump p/ rotina injetada
- Redirecionar um jump do bloco de boot
  - Código precisa estar no bloco de boot
  - Espaço limitado, impossível usar pilha

POST Jump Table Patch

- Alterar jump da PJT c/ offset do código inserido

1. Entender e identificar estruturas da BIOS através de engenharia reversa
2. Encontrar Jump p/ rotina *dummy* na PJT
3. Montar código a ser injetado

POST Jump Table Patch

4. Encontrar padding bytes onde o procedimento será inserido (0xFFFF)
5. Modificar POST jump table e incluir jump p/ procedimento
6. Gravar BIOS modificada

- PJT possui limitações de endereçamento, uma vez que só utiliza ponteiros de 16 bits

BIOS Rootkits

- Combinação de Code Injection com técnicas de redirecionamento
- *Detour patching*
- Alvo não é PJT e sim o gerenciador de interrupções da BIOS

Detour Patching 13h

- Interrupção lê setor especificado do disco utilizando conjunto de comandos ATA
- Pode ser alterada para:
  - Identificar endereços dos setores relacionados ao Código do SO no HD
  - Alterar o conteúdo antes de retornar se o código lido é o Kernel do SO

Detour Patching 13h

- *Detour Patching*
- Redirecionar uma chamada a rotina p/ um código customizado
- Código customizado pode conter um kernel patcher, mas espaço é insuficiente

Detour Patching 13h

- Kernel patcher pode estar em BIOS module
- Inserir um novo POST entry após a inicialização do BIOS interrupt handler
- Procedimento chamado descomprime o código customizado e altera a rotina de gerenciamento da interrupção p/ pular p/ o código customizado

Detour Patching 19h

- Interrupção lê setor de boot do HD ou dispositivo de boot e executa código lido
- Pode ser alterada para:
  - Patchear o boot loader do SO de forma que ele chame o código customizado
  - Código customizado esconde o rootkit e executa o loader do SO

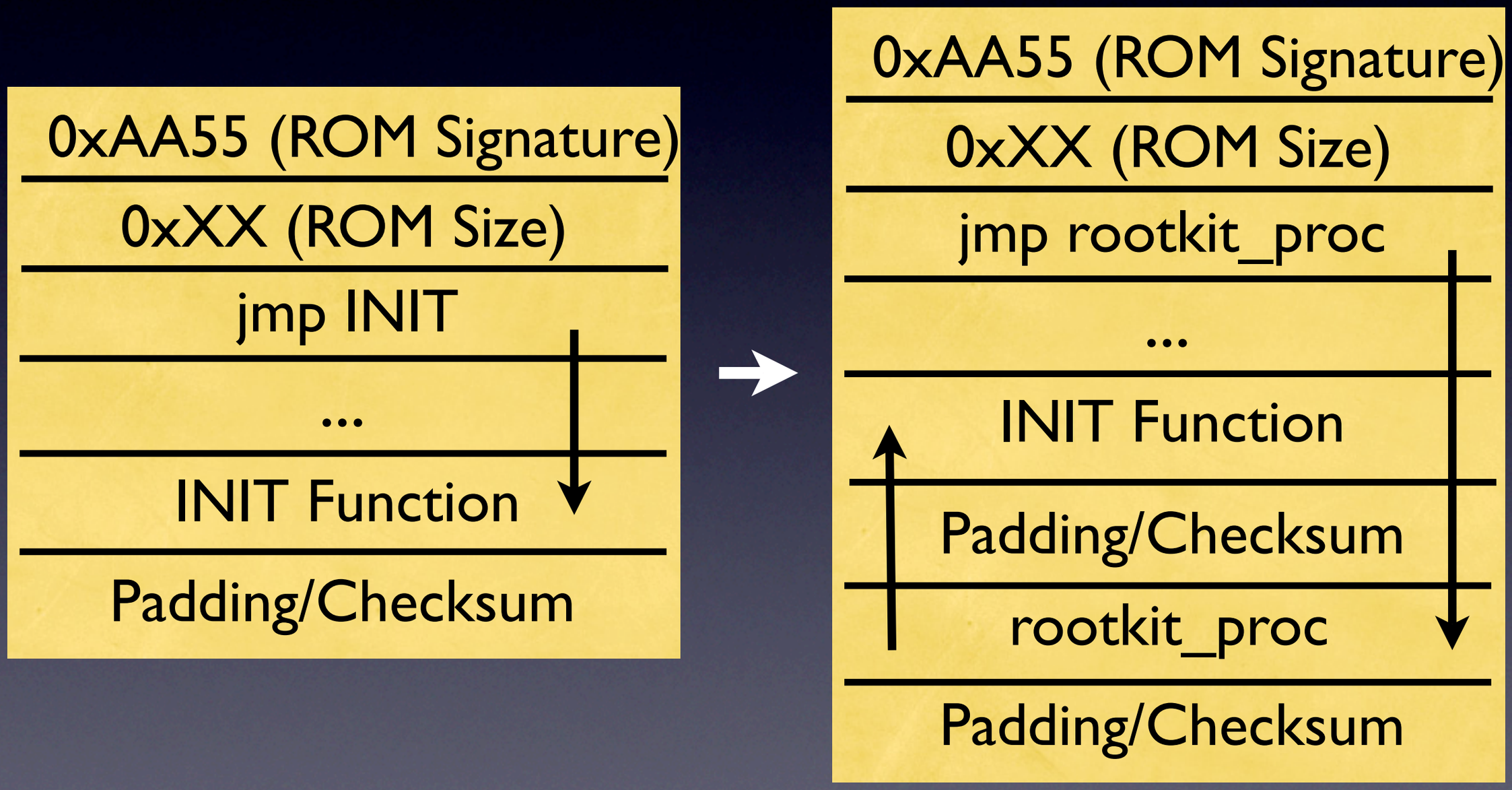
PCI Rootkits

- Detour patching
  - Limitado pelo tamanho do espaço livre no chip de Expansion ROM.
- Ideia direta
  - Vet. de interr. e expansion ROM da placa de vídeo já inicializados.
  - Instalação de hooks nos vetores de interrupção.

PCI Rootkits

- Alterar o ponto de entrada original do XROM
- Colocar o código no espaço livre
- Garantir que o tamanho final é múltiplo de 512 bytes e 8-bit checksum calculado corretamente.

PCI Rootkits



PCI Rootkits

- Limitações:
  - Placas de rede: Se a opção de “boot from LAN” estiver desabilitada, INIT não executado
  - Placa de vídeo: Não é possível realizar debug pelo display gráfico.

Low Level Playground

Bruno Cardoso Lopes
João Batista Correa