Operating Systems

Physical and Virtual Memory

Me

February 18, 2016

План

1. Расположение физической памяти.
2. Понятие процесса и виртуальная память.
3. Сегментация памяти как способ защиты.
4. Paging и Page Fault.
5. Свободная память - бесполезная память.

Как выглядит физическая память?

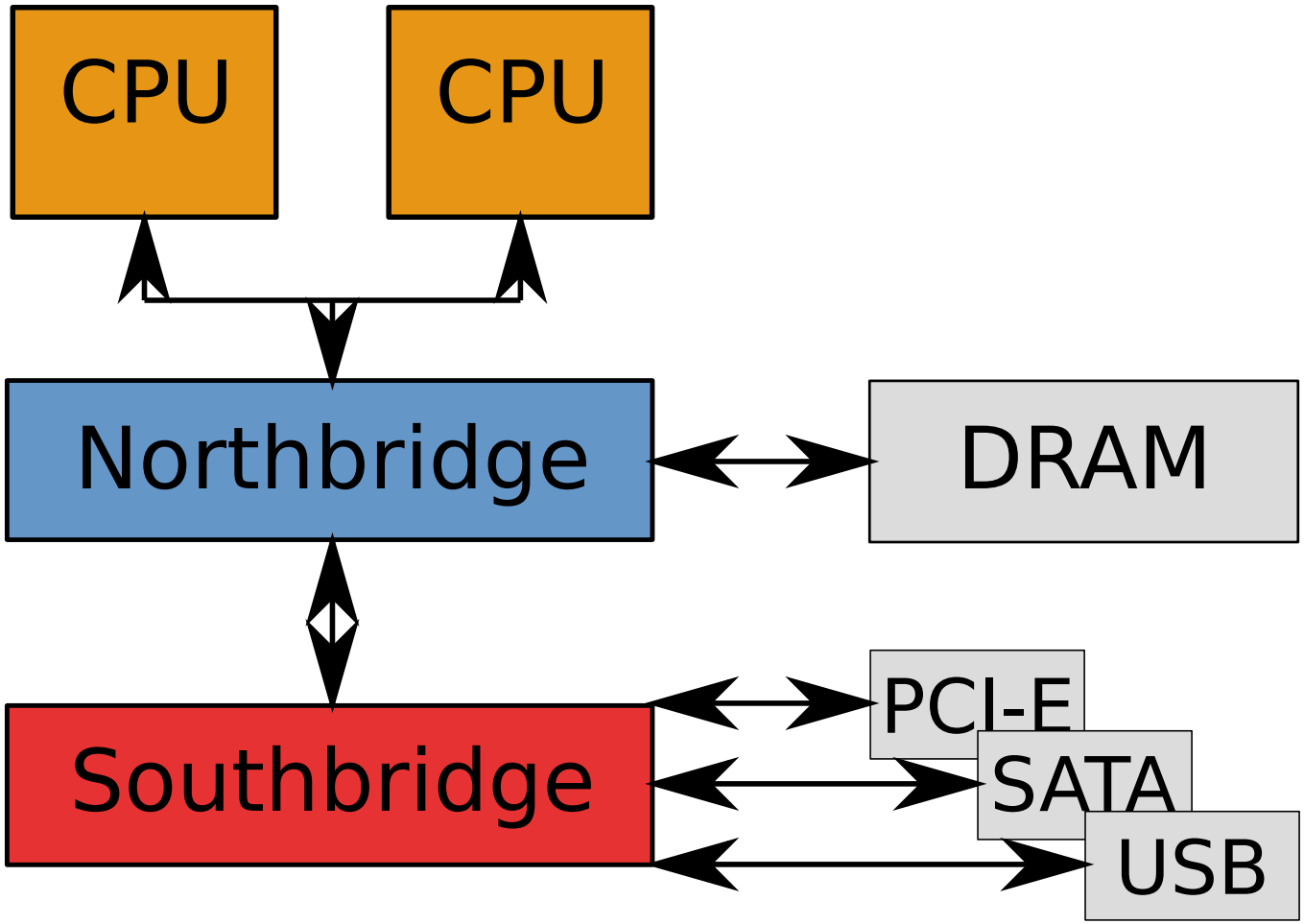

Figure : Classical UMA Architecture

Как выглядит физическая память?

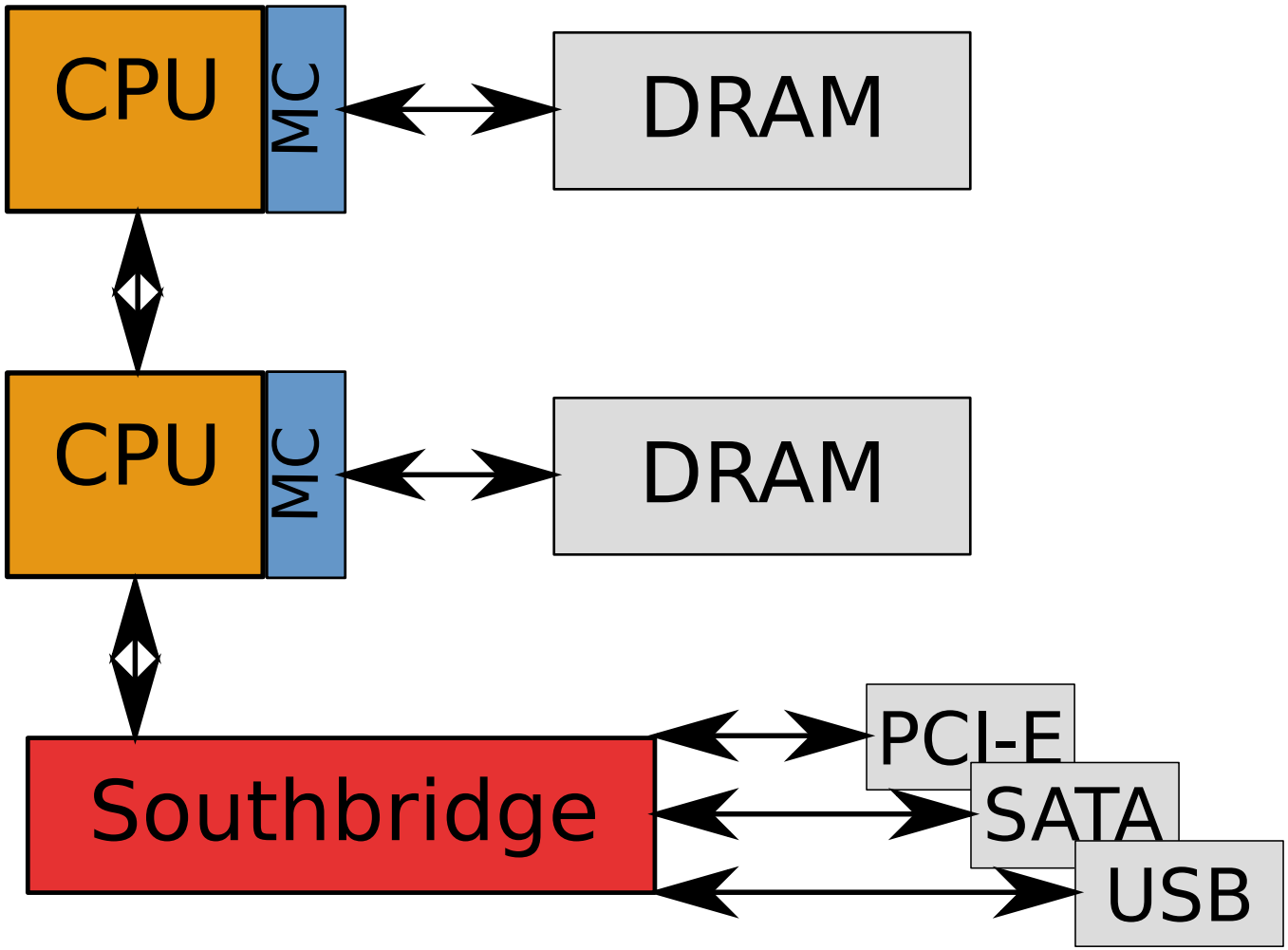

Figure : NUMA Architecture

Как выглядит физическая память?

Память не однородна:

- адреса могут вообще быть не доступны - память не отображена никуда;
- адреса могут быть отображены на устройства - особые правила доступа/кеширования;
- разные адреса могут иметь разное время доступа с разных CPU (NUMA);

Как выглядит физическая память?

Память не однородна:

- адреса могут вообще быть не доступны - память не отображена никуда;
- адреса могут быть отображены на устройства - особые правила доступа/кеширования;
- разные адреса могут иметь разное время доступа с разных CPU (NUMA);

Нужна карта памяти!

Где взять карту памяти?

- из документации чипсета

Где взять карту памяти?

- из документации чипсета
- из device tree
  - кто-то все равно должен взять документацию чипсета и описать память в нужном формате и передать загрузчику

Где взять карту памяти?

- из документации чипсета
- из device tree
  - кто-то все равно должен взять документацию чипсета и описать память в нужном формате и передать загрузчику
- BIOS/UEFI или их аналог
  - BIOS int $0x15, функции 0xe820 или 0xe801
  - UEFI GetMemoryMap

Где взять карту памяти?

- из документации чипсета
- из device tree
  - кто-то все равно должен взять документацию чипсета и описать память в нужном формате и передать загрузчику
- BIOS/UEFI или их аналог
  - BIOS int $0x15, функции 0xe820 или 0xe801
  - UEFI GetMemoryMap
- спросить у загрузчика (где ее берет загрузчик - не наше дело)

Типичная карта памяти

Такие регионы памяти, например, может сообщать QEMU через multiboot загрузчик:

- `0x00000000-0x0009fbff, Available`
- `0x0009fc00-0x0009ffff, Reserved`
- `0x000f0000-0x000fffff, Reserved`
- `0x00100000-0x07ffdfff, Available`
- `0x07ffe000-0x07ffffff, Reserved`
- `0xfffc0000-0xffffffff, Reserved`

Понятие процесса

Процесс - контейнер ресурсов ОС:

Понятие процесса

Процесс - контейнер ресурсов ОС:

- ресурсы в ОС привязаны к процессам
  - память (свое адресное пространство у процессов)
  - файловые дескрипторы
  - другие ресурсы (сокеты, различные IPC и тд)

Понятие процесса

Процесс - контейнер ресурсов ОС:

- ресурсы в ОС привязаны к процессам
  - память (свое адресное пространство у процессов)
  - файловые дескрипторы
  - другие ресурсы (сокеты, различные IPC и тд)
- процессы изолированы друг от друга
  - на сколько это позволяет аппаратное обеспечение (далее просто HW)
  - некоторые процессы могут разделять общие ресурсы намеренно

Адресное пространство процесса

Адресное пространство процесса (виртуальное адресное пространство, VA) - это набор адресов доступных процессу для работы (мы называем эти адреса виртуальными):

Адресное пространство процесса

Адресное пространство процесса (виртуальное адресное пространство, VA) - это набор адресов доступных процессу для работы (мы называем эти адреса виртуальными):

- VA отображается на физическую память (далее PA)
  - paging - произвольное отображение
  - "один к одному", если HW не поддерживает трансляцию

Адресное пространство процесса

Адресное пространство процесса (виртуальное адресное пространство, VA) - это набор адресов доступных процессу для работы (мы называем эти адреса виртуальными):

- VA отображается на физическую память (далее PA)
  - paging - произвольное отображение
  - "один к одному", если HW не поддерживает трансляцию
- VA может быть аппаратно защищено
  - сегментирование - попытка обращения в чужой сегмент приводит к ошибке
  - paging устраняет саму возможность обратиться к чужой памяти

Адресное пространство процесса

Адресное пространство процесса (виртуальное адресное пространство, VA) - это набор адресов доступных процессу для работы (мы называем эти адреса виртуальными):

- VA отображается на физическую память (далее PA)
  - paging - произвольное отображение
  - "один к одному", если HW не поддерживает трансляцию
- VA может быть аппаратно защищено
  - сегментирование - попытка обращения в чужой сегмент приводит к ошибке
  - paging устраняет саму возможность обратиться к чужой памяти
- VA может быть неоднородным - в нем могут быть дыры

Адресное пространство процесса

VA процесса - это ресурс, который нужно аллоцировать и освобождать:

Адресное пространство процесса

VA процесса - это ресурс, который нужно аллоцировать и освобождать:

- ОС необходимо делить память между несколькими процессами - не нужно выдавать процессу сразу много памяти, которая скорее всего не будет использована;

Адресное пространство процесса

VA процесса - это ресурс, который нужно аллоцировать и освобождать:

- ОС необходимо делить память между несколькими процессами - не нужно выдавать процессу сразу много памяти, которая скорее всего не будет использована;
- разные части VA используются под разные нужды:
  - они находятся в разных регионах VA (стек растет вниз, его логично положить наверх)
  - могут иметь разные права/привелегии доступа (например, делать стек исполняемым - плохая затея с точки зрения безопасности)

Сегментация на примере x86

Вся память разбивается на сегменты:

- уровень привелегий доступа назначается каждому сегменту отдельно
- сегменты могут перекрываться, т. е. два сегмента с разными привелегиями могут описывать одну и ту же физическую память

Сегментация на примере x86

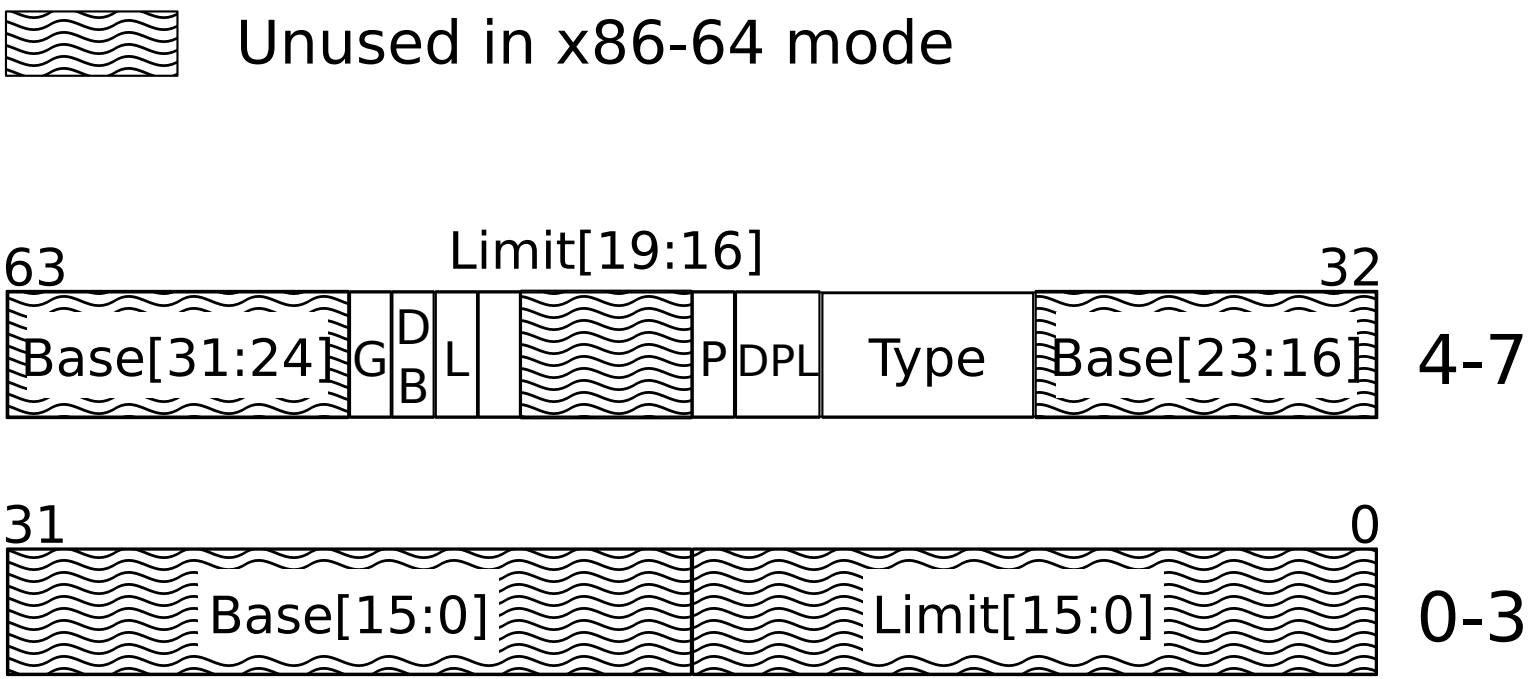

Figure : x86 segment data/code descriptor format

Сегментация на примере x86

Дескрипторы сегментов хранятся в таблицах GDT и LDT:

- GDT предполагается общей для всех процессов (не обязательно)
- LDT своя для каждого процесса (не обязательно)
- при обращении к памяти таблица и номер дескриптора в ней определяются используя селектор сегмента (16-битное значение в CS, SS, DS, ES, FS или GS)

Сегментация на примере x86

Data Selector (DS, ES)

DESCRIPTOR INDEX	T	RPL

Code Selector (CS)

DESCRIPTOR INDEX	T	CPL

Figure : Code and Data Segment Selectors

Сегментация на примере x86

Проверка привелегий:

- при записи селектора в сегментный регистр CPL (из CS) и RPL (то что мы записываем) должны быть меньше или равны DPL (в дескрипторе сегмента, на который мы ссылаемся);
- для селектора стека (SS) используются особые правила - RPL и DPL должны быть равны CPL.