Многозадачность: Процессы и потоки

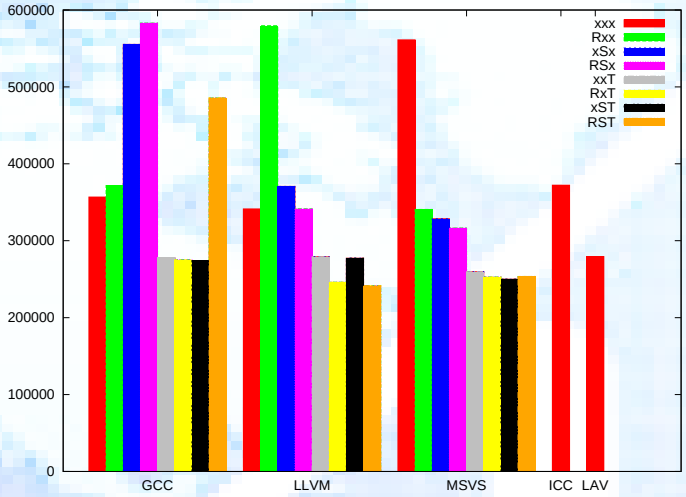
Евгений Иванович Клименков

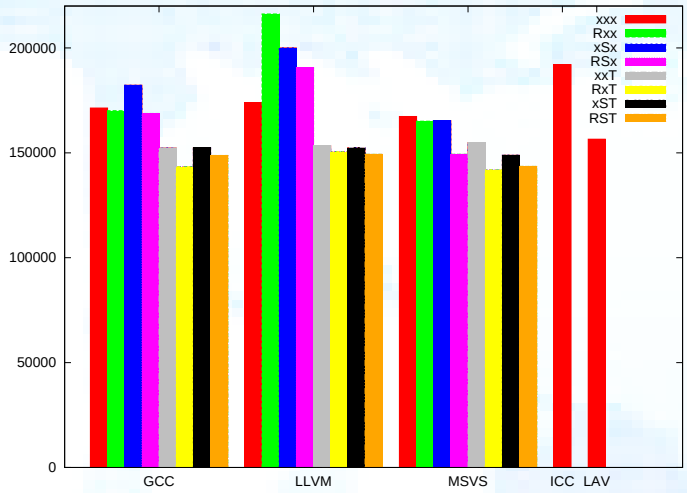
osisp2019@gmail.com

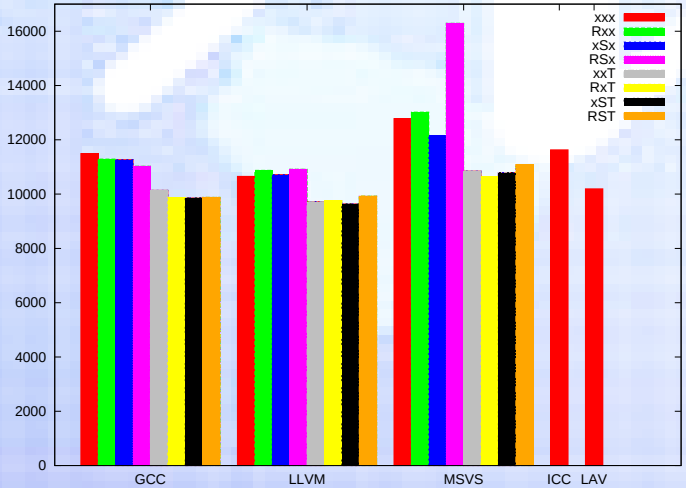
Белорусский Государственный Университет
Информатики и Радиоэлектроники

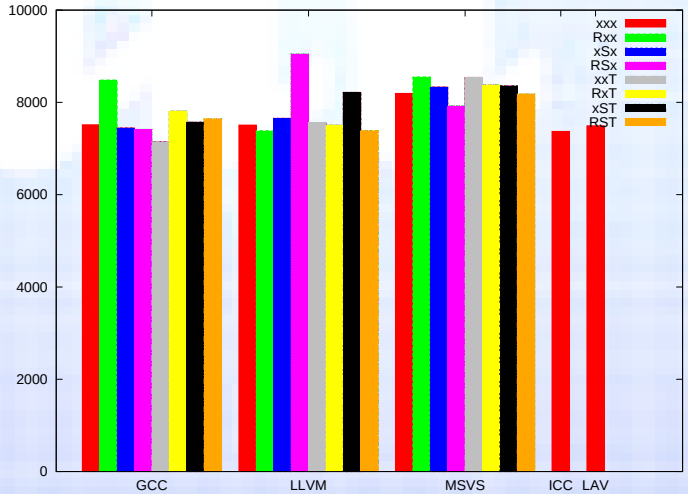
2019

Еще раз про сложность процессора :-)









Функции многозадачности

- + Упрощение архитектуры и реализации программного обеспечения
- + Утилизация многопроцессорности
- + Организация интерактивных вычислительных сред

= борьба со сложностью

Опять Процессор

Для полноценного функционирования, ОС нужна аппаратная поддержка

На примере IA-32:

- Аппаратная защита
- Два режима функционирования: привелигерованный и защищенный
- Привелигированные инструкции
- Управление и защита доступа к физическому адресному пространству
  - Сегментация памяти
  - Расстраничивание памяти

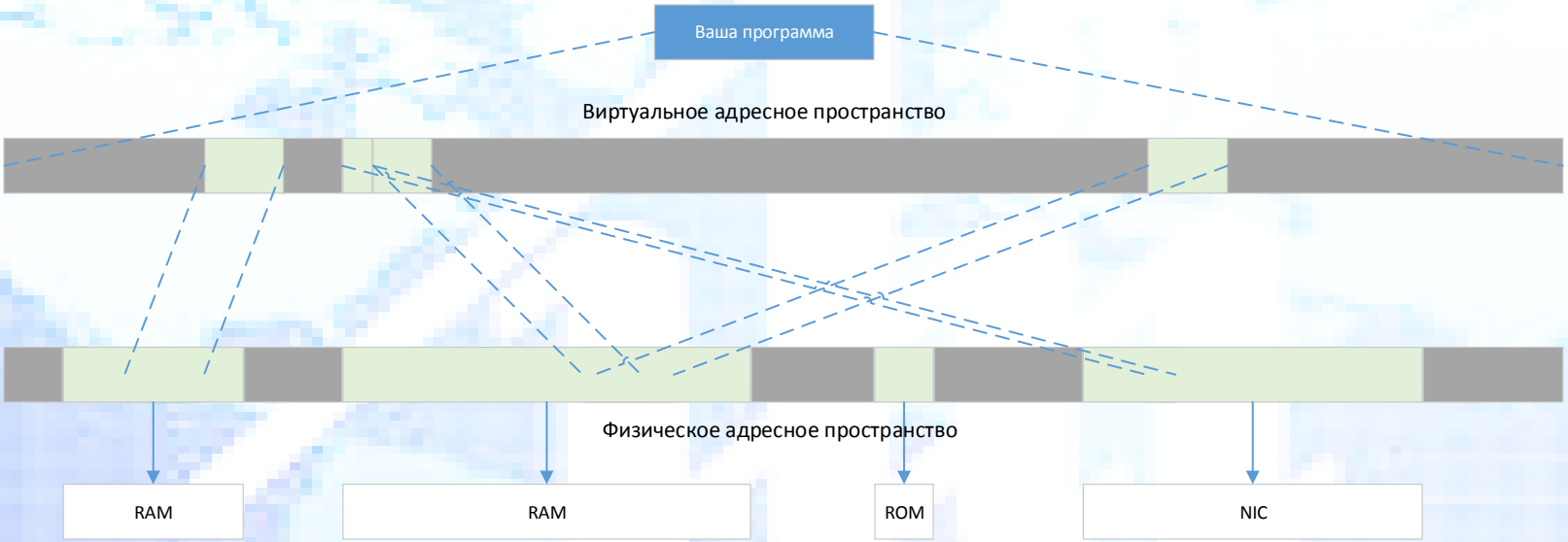
Опять Процессор

Для полноценного функционирования, ОС нужна аппаратная поддержка

На примере IA-32:

- Аппаратная защита
- Два режима функционирования: привелигерованный и защищенный
- Привелигированные инструкции
- Управление и защита доступа к физическому адресному пространству
  - Сегментация памяти
  - Расстраничивание памяти

Модель памяти



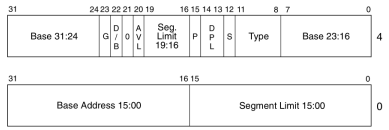
Опять Процессор

Физическое адресное пространство - это аппаратное представление набора адрессов с назначенными им обслуживающими аппаратными устройствами. ФАС единообразно для всех компонентов системы. ФАС не защищен. Виртуальное адресное пространство - это искусственно создаваемая и поддерживаемая ОС абстракция ФАС которая произвольным и управляемым образом отображается на ФАС. ВАС уникален для каждого процесса и за пределами доступа для аппаратных компонентов системы. ВАС защищен.

Сегментация

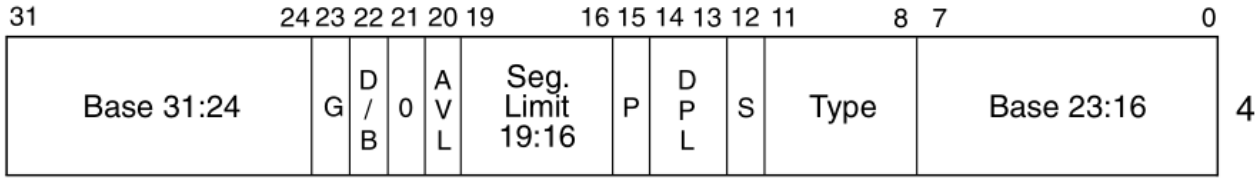
Сегмент – непрерывный участок виртуального адресного пространства произвольного размера отображающийся на заданный адресс ФАС.
На примере IA-32:

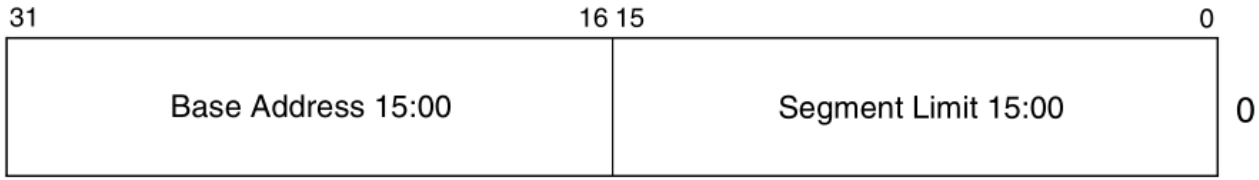
- Сегментные регистры (CS, DS, SS, ES, ...)
- Регистры-указатели на таблицы сегментов (GDTR, LDTR, IDTR)
- Таблицы сегментов



AVL — Available for use by system software
BASE — Segment base address
D/B — Default operation size (0 = 16-bit segment; 1 = 32-bit segment)
DPL — Descriptor privilege level
G — Granularity
LIMIT — Segment Limit
P — Segment present
S — Descriptor type (0 = system; 1 = code or data)
TYPE — Segment type

Сегментация





AVL — Available for use by system software
BASE — Segment base address
D/B — Default operation size (0 = 16-bit segment; 1 = 32-bit segment)
DPL — Descriptor privilege level
G — Granularity
LIMIT — Segment Limit
P — Segment present
S — Descriptor type (0 = system; 1 = code or data)
TYPE — Segment type

Сегментация

Жутко неудобна

- для создания программ на прикладном уровне
- для управления памятью на системном уровне

Современные ОС создают иллюзию отсутствия механизма сегментации – FLAT Memory Model.

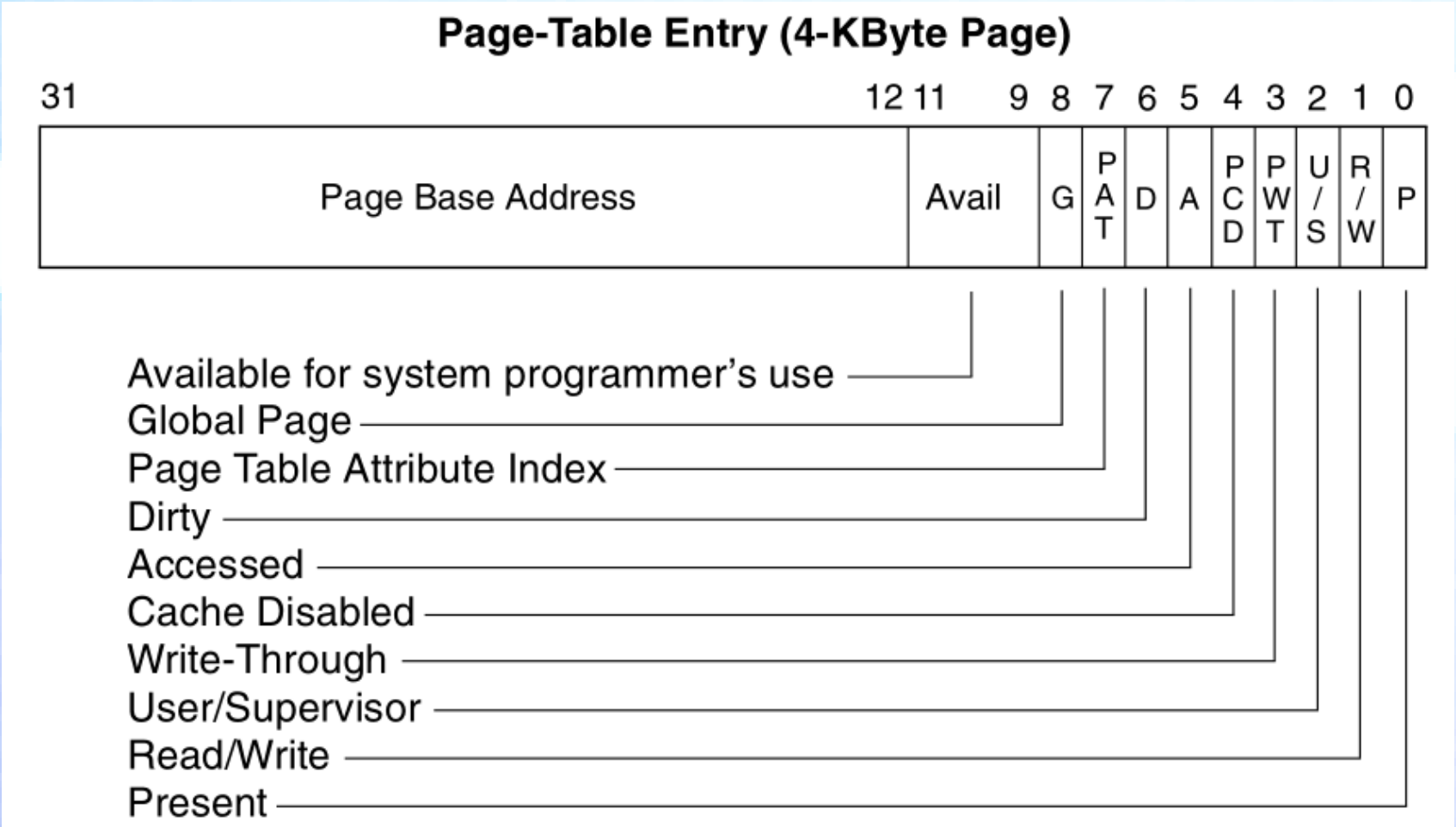
Paging

Виртуальное адресное пространство представляется в виде массива страниц, каждая из которых имеет один и тот же размер и выровненное на размер страницы смещение.
Отображение виртуального адресного пространства через трансляцию на основе таблицы страниц.

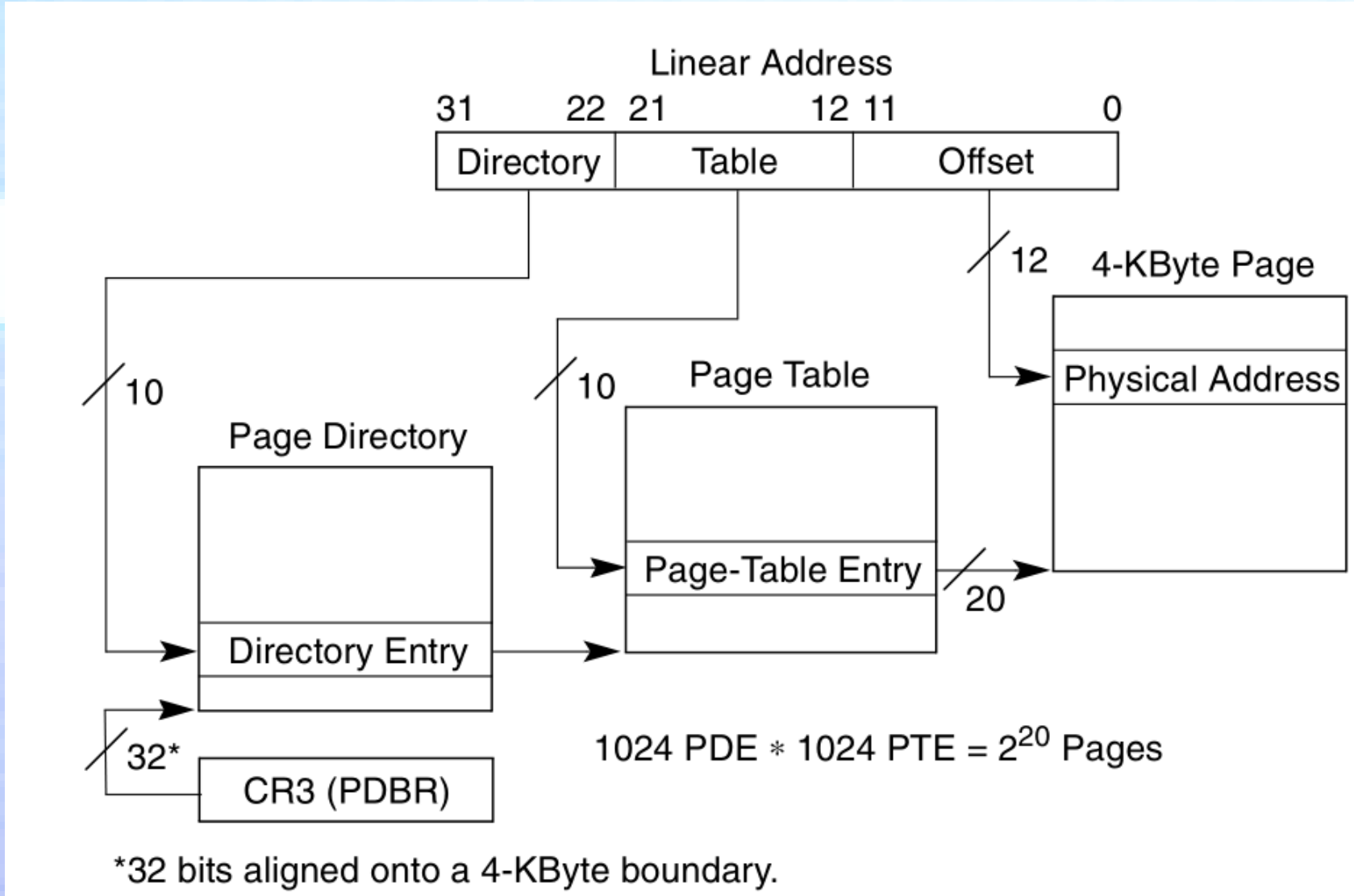
- Адрес страницы в ВАС определяет ее индекс в таблице страниц
- Содержимое элемента таблицы определяет адрес соответствующей страницы ФАС для страницы ВАС.

Современные ОС создают иллюзию отсутствия механизма сегментации – FLAT Memory Model.

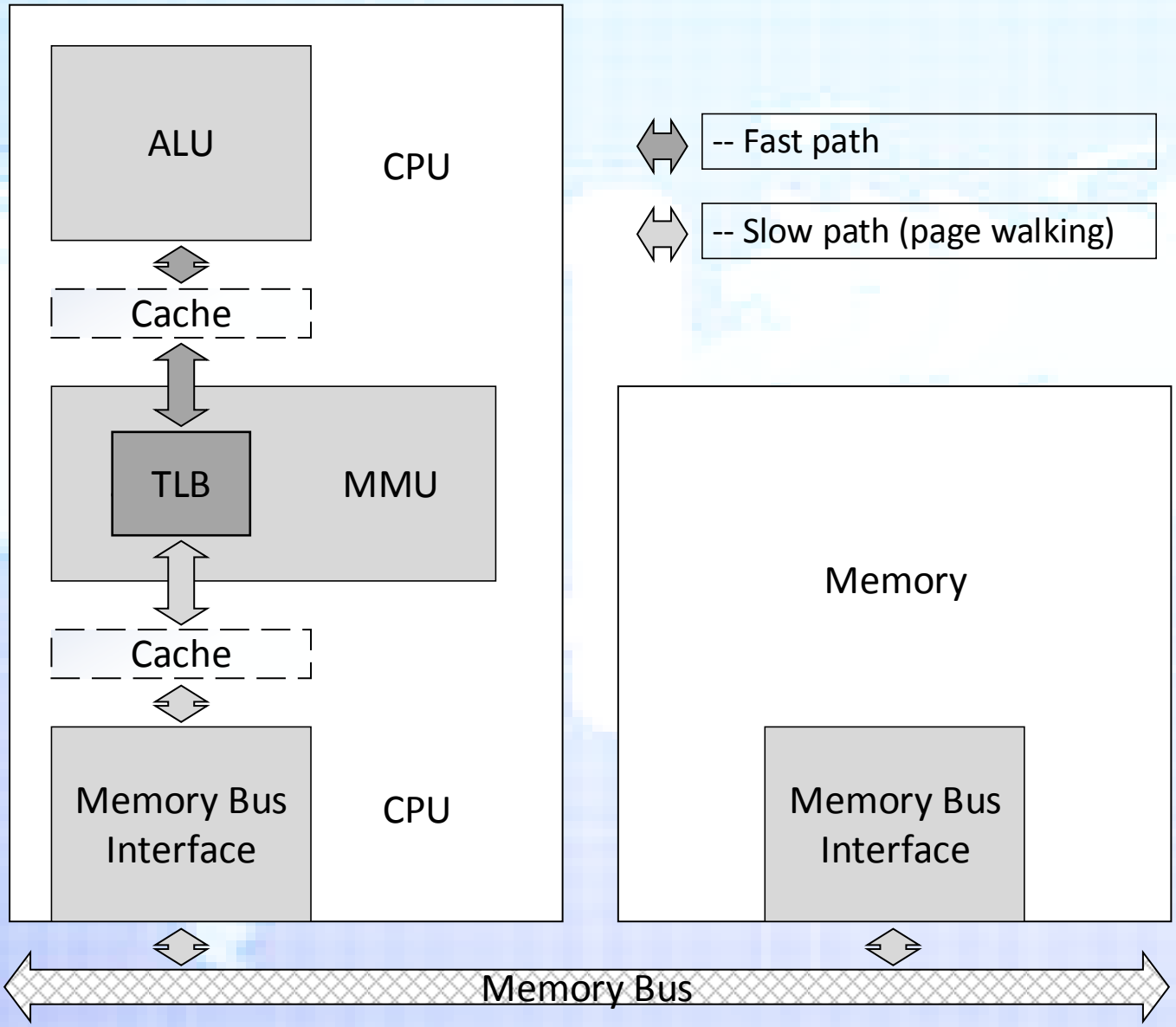
Pageing


Page-Table Entry (4-KByte Page)
31
12 11
9 8 7 6 5 4 3 2 1 0
Page Base Address
Avail
G
PAT
D
A
PCD
PWT
U/S
R/W
P
Available for system programmer's use
Global Page
Page Table Attribute Index
Dirty
Accessed
Cache Disabled
Write-Through
User/Supervisor
Read/Write
Present

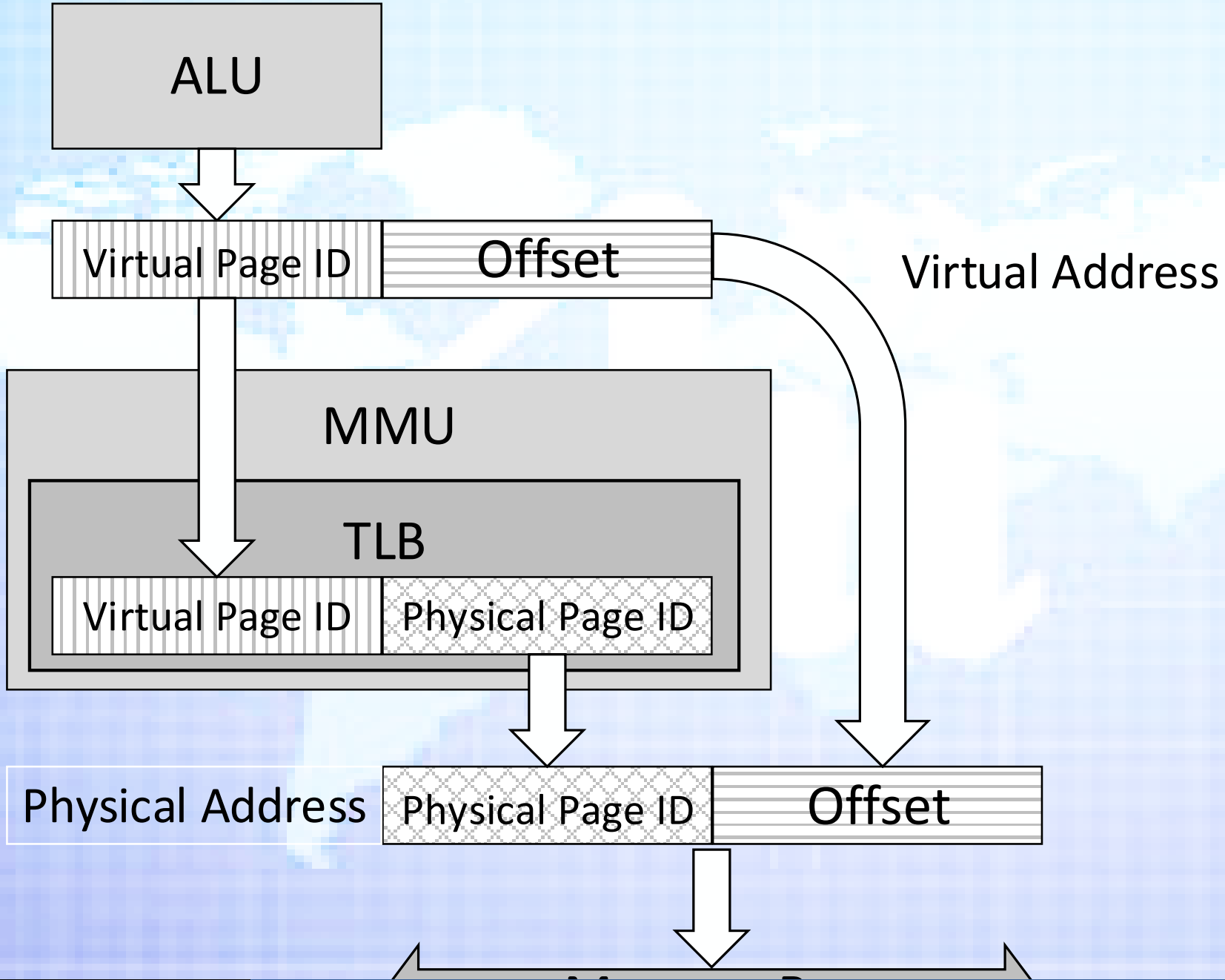
Pageing

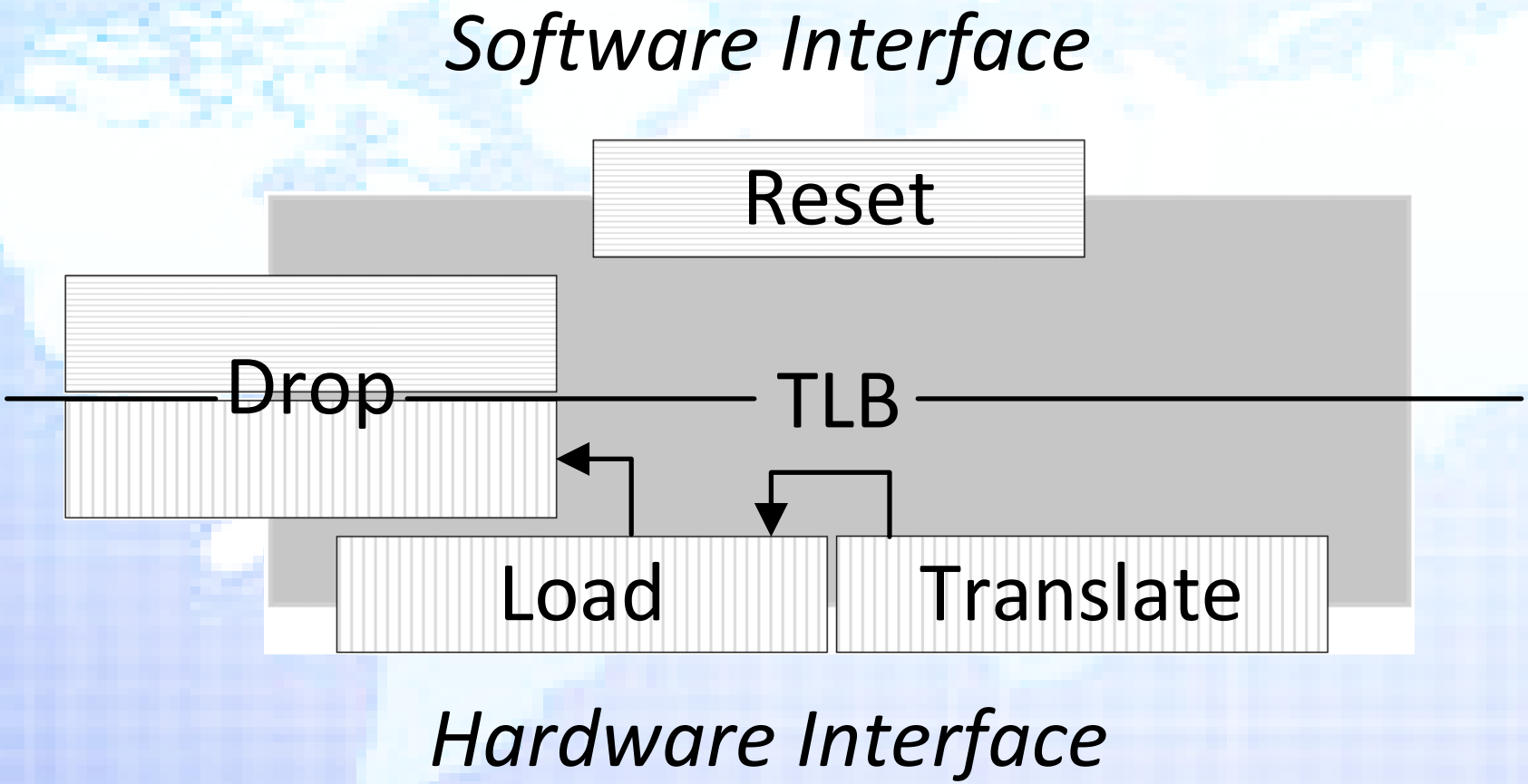

Linear Address
31
22
21
12
11
0
Directory
Table
Offset
12
4-KByte Page
Physical Address
10
Page Directory
10
Page Table
Page-Table Entry
20
Directory Entry
32*
CR3 (PDBR)
1024 PDE * 1024 PTE = 2^20 Pages
*32 bits aligned onto a 4-KByte boundary.

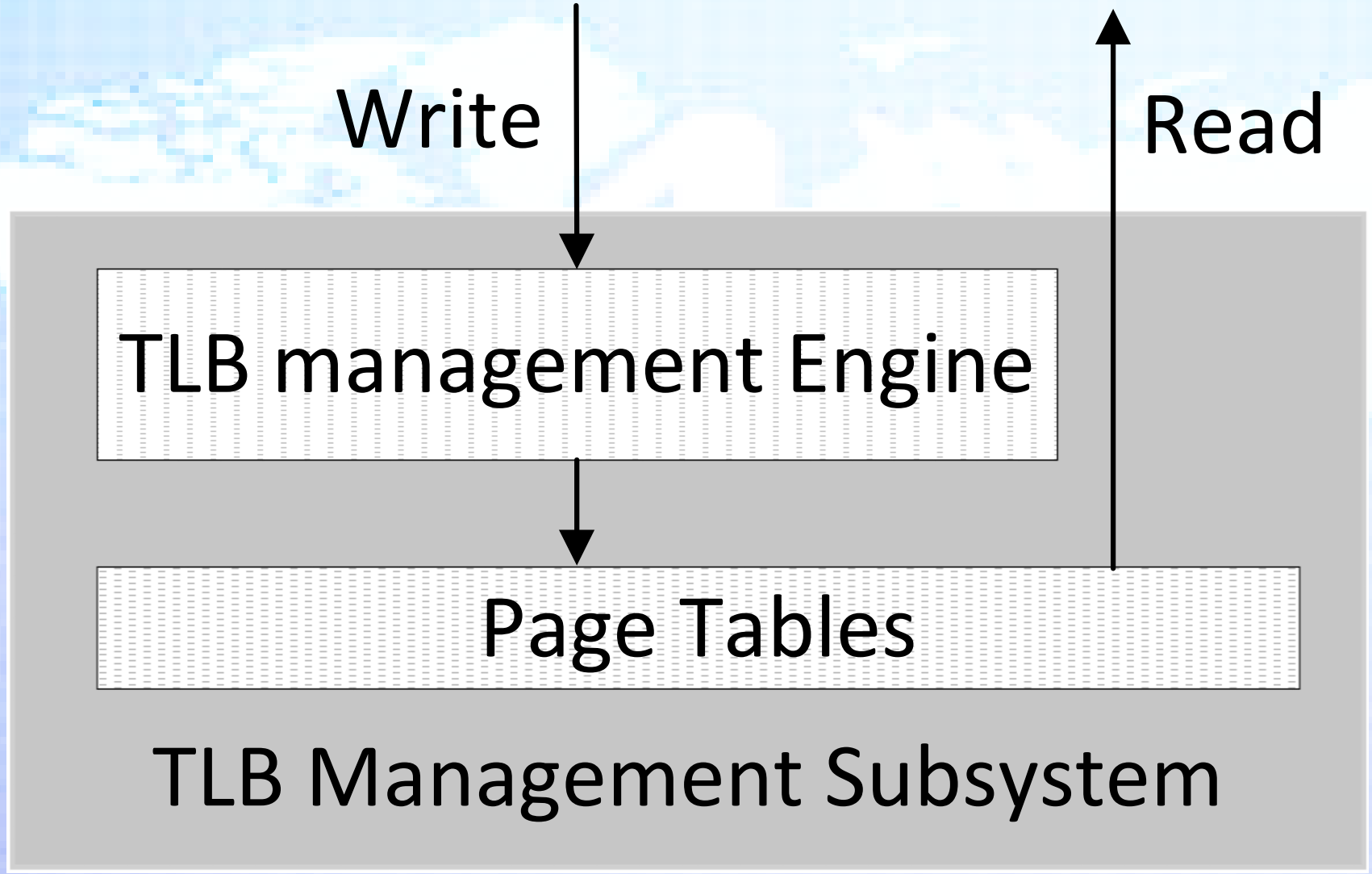
TLB


ALU
CPU
Cache
TLB
MMU
Cache
Memory Bus Interface
CPU
-- Fast path
-- Slow path (page walking)
Memory
Memory Bus Interface
Memory Bus

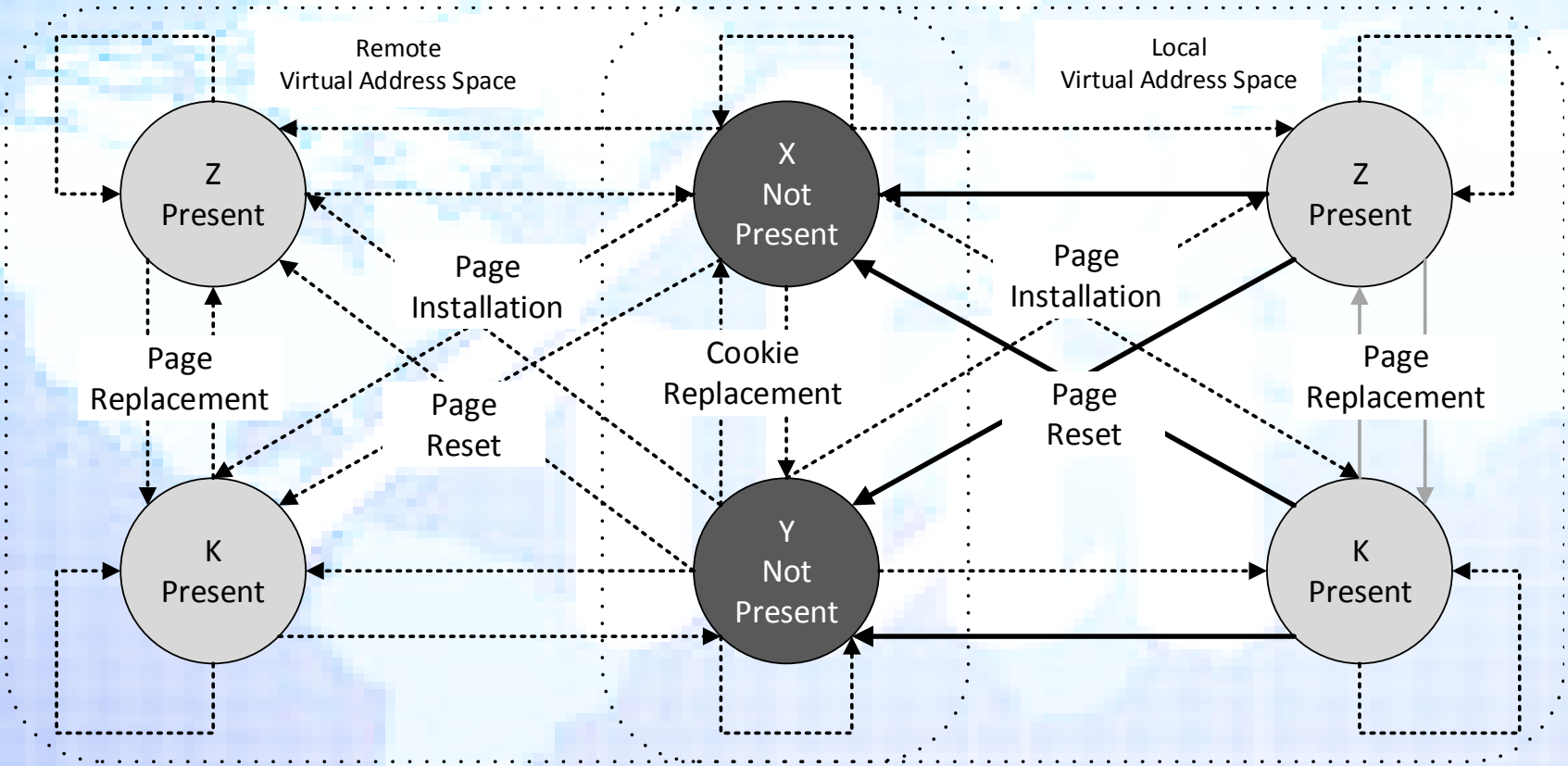
TLB


ALU
Virtual Page ID
Offset
Virtual Address
MMU
TLB
Virtual Page ID
Physical Page ID
Physical Address
Physical Page ID
Offset

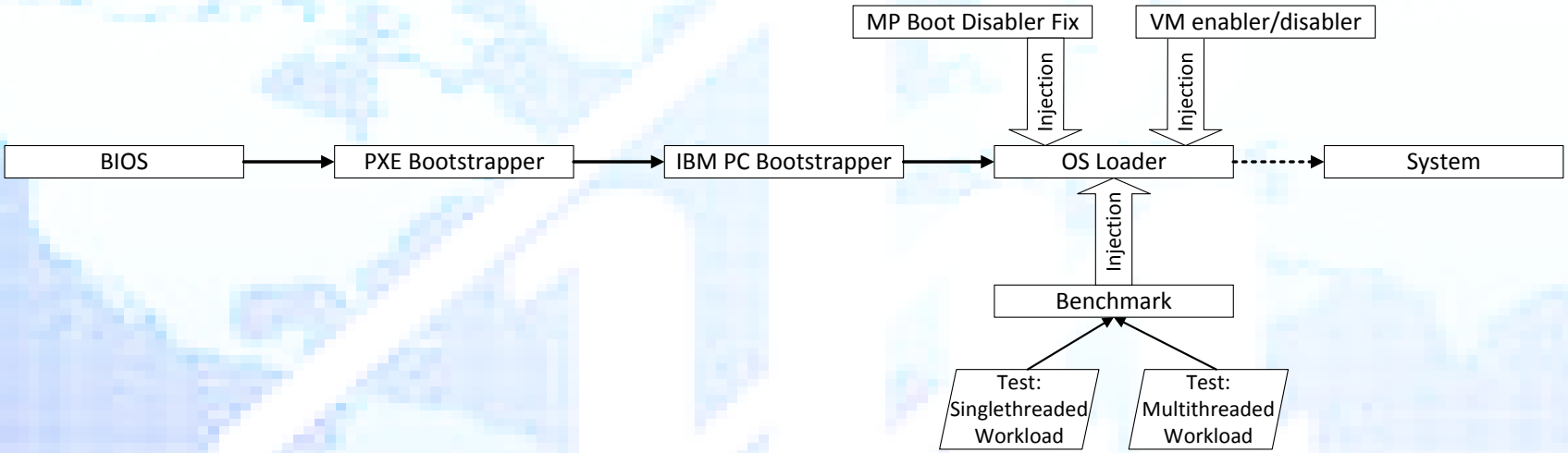

Software Interface
Reset
Drop
TLB
Load
Translate
Hardware Interface


Write
Read
TLB management Engine
Page Tables
TLB Management Subsystem

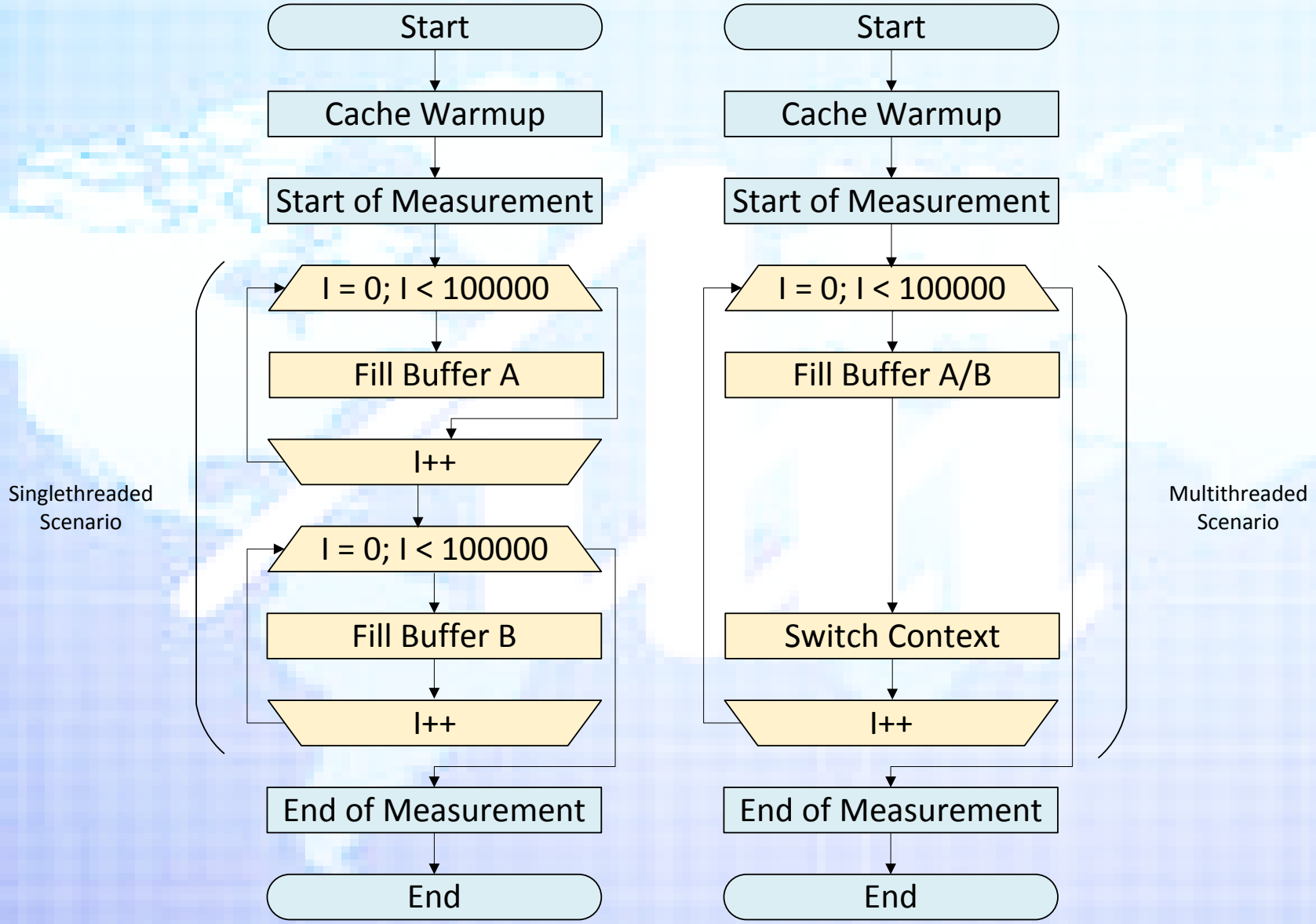
TLB


Remote
Virtual Address Space
Local
Virtual Address Space
Z
Present
X
Not
Present
Z
Present
K
Present
Y
Not
Present
K
Present
Page
Installation
Page
Installation
Page
Replacement
Page
Reset
Cookie
Replacement
Page
Reset
Page
Replacement

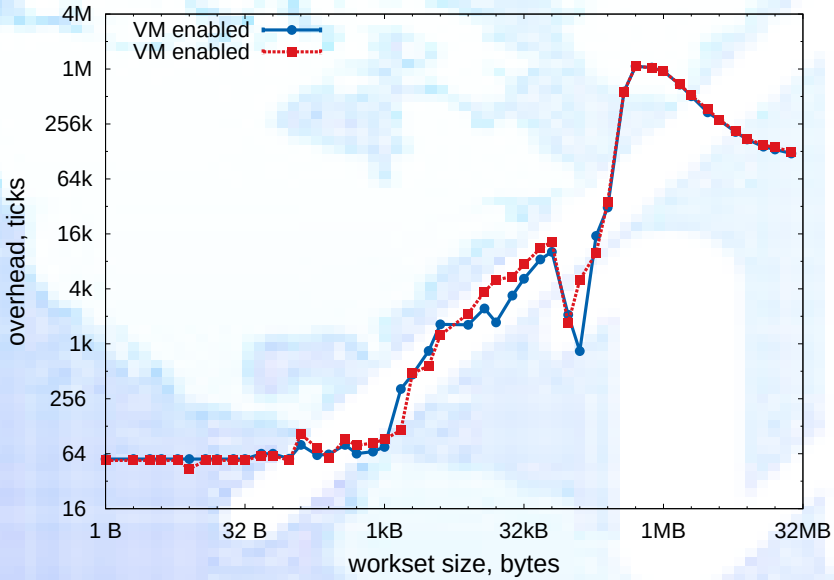
Цена вопроса

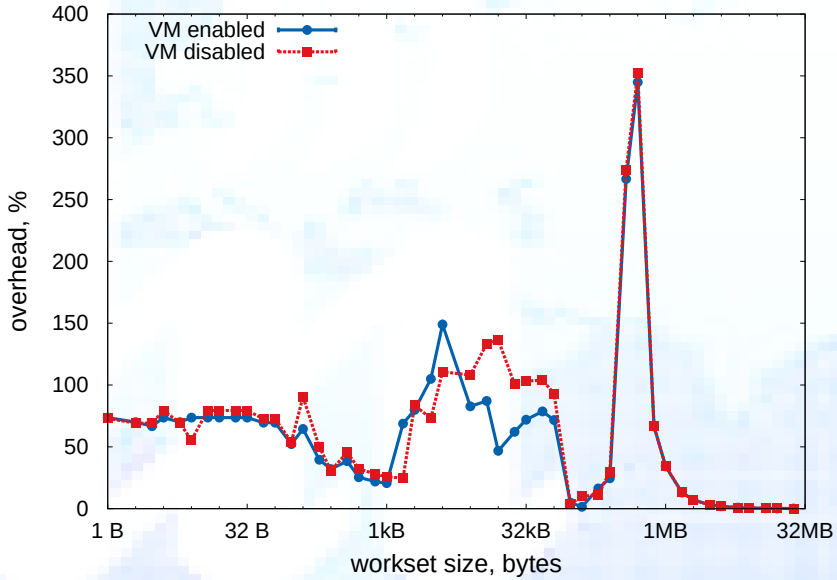


Цена вопроса

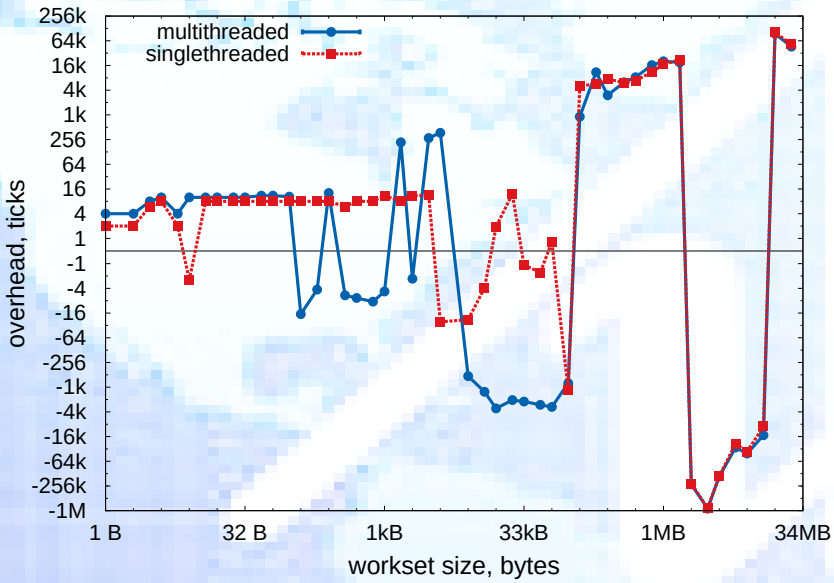


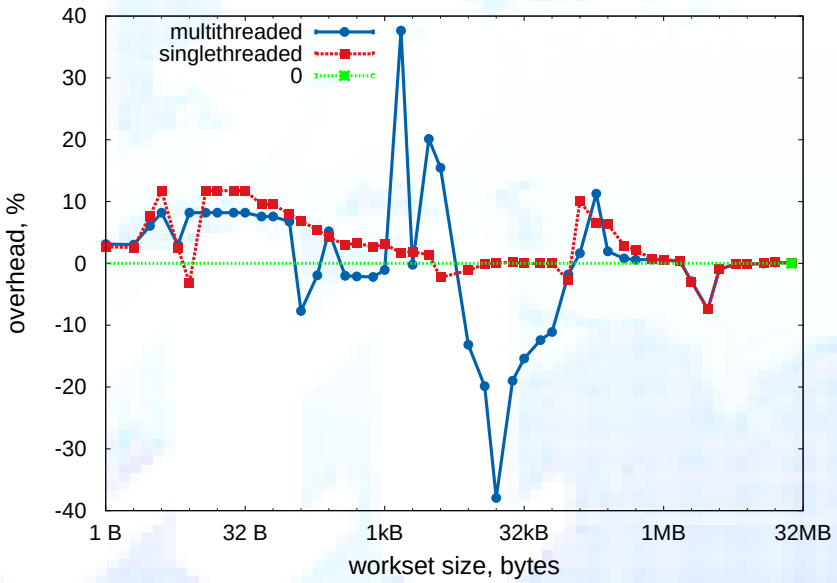
Цена вопроса



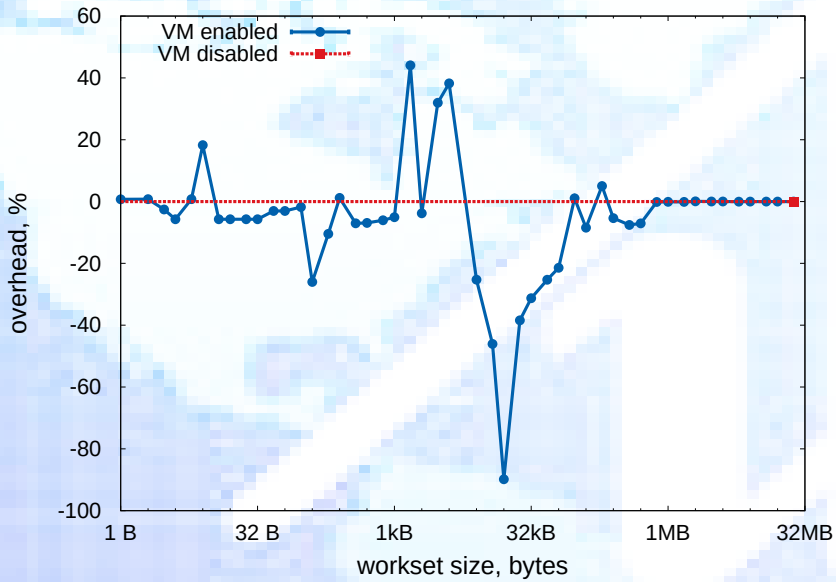


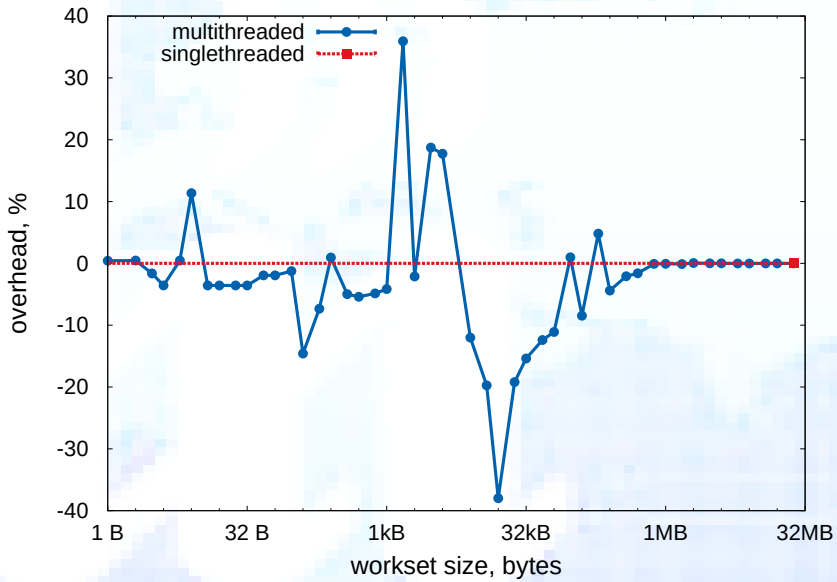
Цена вопроса





Цена вопроса





Процесс

Процесс - это программа во время выполнения (UNIX).

Процесс

НЕВЕРНО! На экзамене накажу за такое определение!

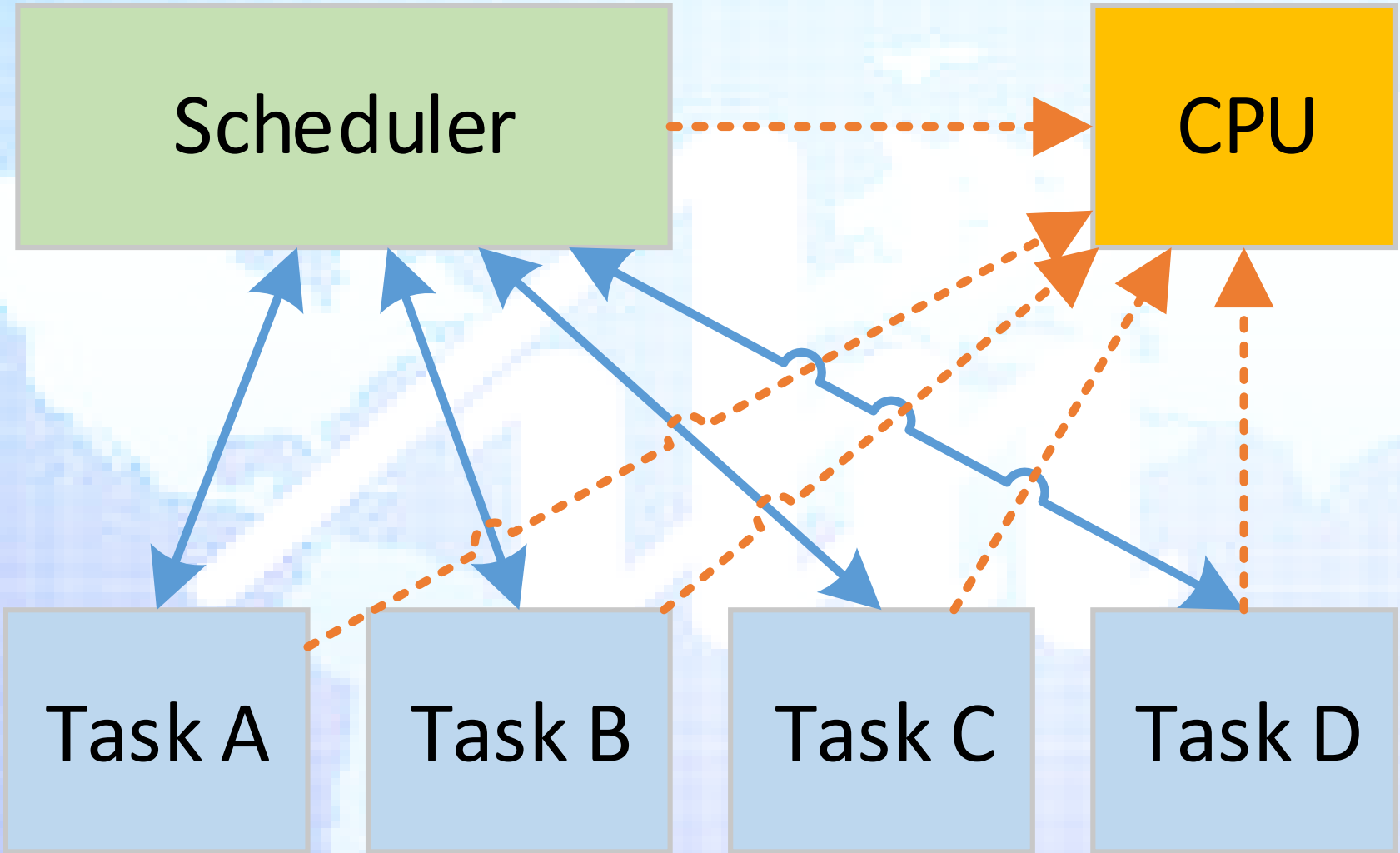
Процесс

Процесс - это идентифицируемая абстракция совокупности взаимосвязанных системных ресурсов на основе отдельного и независимого виртуального адресного пространства в контексте которой организуется выполнение потоков.

Поток выполнения

Поток выполнения - это элементарная единица планирования. Поток всегда привязан и выполняется в контексте процесса*. Процесс всегда имеет как минимум один поток выполняющийся в нем.

Модель многозадачности



Многозадачность

- Кооперативная
- Вытесняющая

Кто принимает решение о том когда следует переключиться между задачами?

От простого к сложному

Поток инструкций

Процессор по умолчанию выполняет инструкции последовательно. Регистр IP указывает на текущую/следующую инструкцию. В процессе выполнения каждой инструкции значение регистра увеличивается.

Потоки инструкций являются аппаратной абстракций (инструкции условного и безусловного переходов) ==> if/else, for, while.

Простейшая модель задачи состоит из независимо хранящегося значения регистра IP.

От простого к сложному

Поток управления
Поток инструкций расширенный стеком. Становится доступна дополнительная и фундаментальная абстракция управления - функция (инструкции вызова и возврата из функции). Программа становится реентерабельной.
Пример программной реализации: легковесные coroutines (требуют поддержки компилятора).
coroutine = IP + Стек.

От простого к сложному

Простейший программный поток (они же green thread, он же fiber)
Поток управления + состояние регистров процессора.
Упрощение жизни для программиста. Отличие от потока управления только в том, кто отвечает за сохранение состояния регистров процессора.

От простого к сложному

Программный поток с временем.
Программный поток + временная характеристика.
Планировик способен принимать более взвешенные решения и проводить более справедливое планирование.

От простого к сложному

Поток (он же поток ядра) = программный поток + атрибуты ОС.
thread_create(), CreateThread().

От простого к сложному

Поток ядра + состояние памяти (виртуальное адресное пространство) = процесс.
fork(), CreateProcess().

От простого к сложному

Совокупность процессов + небольшое состояние ядра = контейнер.
Совокупность процессов + состояние ОС = виртуальная машина.

Жизненный цикл задачи

Задача всегда создается синхронно другой задачей.
fork() - причина всех недопониманий различий между потоком и задачей.