For Distribution

Устройство управления памятью в процессорах MIPS

www.imgtec.com

Общие задачи устройства управления памятью

Memory Management Unit (MMU)

- **Предоставить доступ программы к объему виртуальной памяти, который превосходит объем физической оперативной памяти**
  - Рассматривалась как неактуальная задача для микроконтроллеров
  - Становится теоретически реализуемо с PIC32MZ
- **Ограничить доступ пользовательских программ к памяти операционной системы**
  - Полезно для некоторых систем со встроенными процессорами
- **Защитить пользовательские программы друг от друга**
  - Полезно для некоторых систем со встроенными процессорами

Два способа реализации MMU в архитектуре MIPS

- **Гибкий, используя так называемый буффер ассоциативной трансляции**
  - Translation Lookaside Buffer (TLB)
  - Позволяет защитить не только операционную систему от пользовательских программ, но и пользовательские программы друг от друга
- **Фиксированный**
  - Fixed Mapping Translation (FMT)
  - Требует минимум аппаратных ресурсов
- **И с FMT, и с TLB защита достигается трансляцией виртуальных адресов в физические с исключением в случае доступа пользовательской программы к запрещенным для нее адресам**

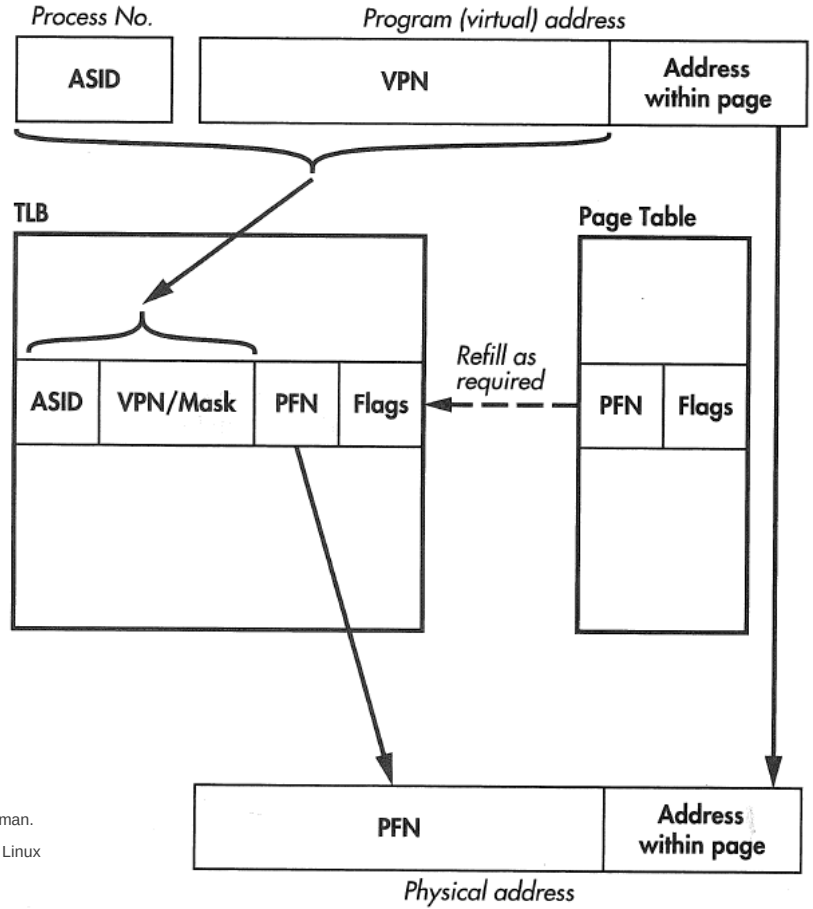
For Distribution

Буффер ассоциативной трансляции
Translation Lookaside Buffer (TLB)

Что такое TLB?

Translation Lookaside Buffer

- **Буффер ассоциативной трансляции**
- **На входе**
  - Виртуальный адрес, Virtual Address
    - Используется программой
  - Идентификатор адресного пространства
    - Address Space Identifier, ASID
- **На выходе**
  - Физический адрес, Physical Address
    - Используется контроллером памяти



Источник:

Dominic Sweetman.

See MIPS Run Linux

Что позволяет TLB?

Универсальное средство быстрого преобразования адресов

- **Позволяет скрывать память операционной системы от непривилегированного кода**
- **Позволяет нескольким регионам памяти выглядеть как последовательный кусок**
- **Позволяет разместить программу в любой части физической памяти**
- **Позволяет адресовать бОльшую по размеру физическую память, чем доступно виртуальных адресов**
- **Позволяет подгружать куски программы с внешнего устройства по надобности**

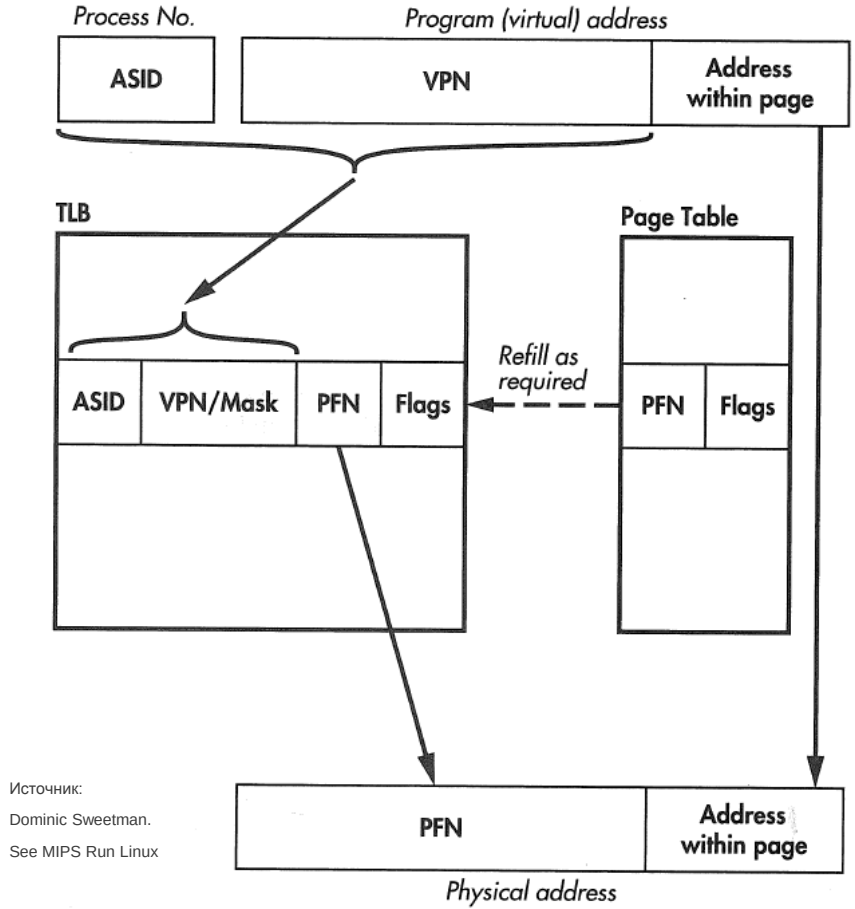
Что еще позволяет TLB?

TLB также ассоциирует с адресом различные атрибуты

- **Атрибут запрета чтения**
- **Атрибут запрета записи**
  - Альтернативно - индикатор, что на страницу с данным адресом происходила запись
- **Атрибут запрета исполнения**
- **Атрибуты кэшируемости и когерентности**

Страницы TLB

- **Память разбивается на страницы**
- **Страница может иметь размеры 4KB, 16KB, 64KB, 256KB, 1MB, 4MB, 16MB, 64MB**
  - Особый случай – 1KB (не поддерживается в PIC32MZ)
  - Разные страницы могут иметь разный размер
- **Номер виртуальной страницы**
  - Virtual Page Number, VPN
  - Виртуальный адрес = VPN + адрес внутри страницы
- **Номер физического фрейма**
  - Physical Frame Number, PFN
  - Физический адрес = PFN + адрес внутри страницы

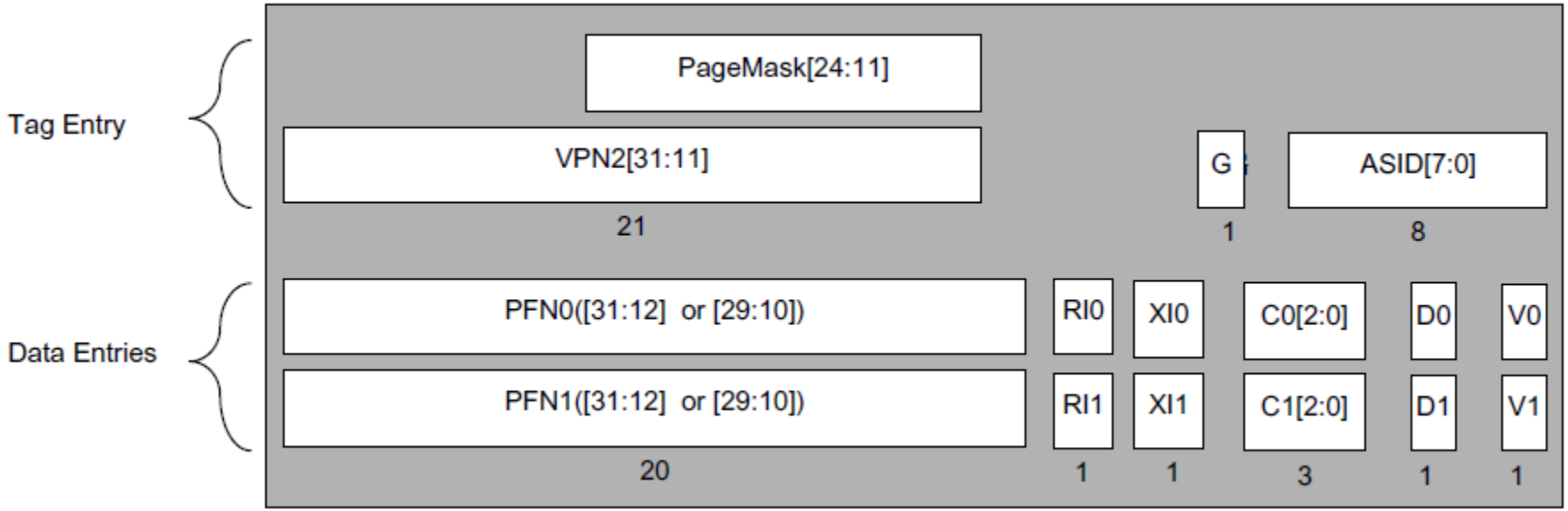


Источник:
Dominic Sweetman.
See MIPS Run Linux

Запись трансляции в TLB MIPS microAptiv UP

TLB Entry

- Делится на запись тега и две записи данных
- Описывает две трансляции – четной и нечетной страницы



Размер TLB

- **Ассоциативная память очень дорогая**
  - Content-addressable memory (CAM)
  - К каждой записи требуется компаратор
- **Типичный TLB в MIPS – от 16 до 64 трансляций**
- **Трюк с двойными записями родился для экономии на компараторах**
  - Согласно Майку Гупте, одного из дизайнеров MIPS R4000 (1991) :
  - Процессор с 64 одинарными записями TLB не влазил в floorplan
  - Влез дизайн с 24 двойными записями (48 трансляций)
  - Решение стало стандартным в начале 1990-х по требованию Микрософта
    - Микрософт и MIPS были членами комитета Advanced Computing Environment (ACE)
    - Микрософт портировал на MIPS Windows NT
  - В 1999 году трюк с двойными записями просочился в стандарт MIPS32

Значение полей тега в записи TLB

- **VPN2 [31:13] – Virtual Page Number**
  - Номер виртуальной страницы, деленный на 2
- **PageMask [28:11]**
  - Фактически определяет размер страницы, маскируя определенные биты виртуального адреса, чтобы они не участвовали в сравнении
- **ASID [7:0] – Address Space Identifier**
  - Идентифицирует процесс, тред или другой примитив операционной системы, для которого делается эта трансляция
- **G – Global Bit**
  - Трансляция является «глобальной», поле ACID игнорируется

Поле PageMask - подробнее

- **PageMask – фактически определяет размер страницы, маскируя определенные биты виртуального адреса, чтобы они не участвовали в сравнении:**

PageMask	Page Size	Even/Odd Bank Select Bit
00_0000_0000_0000_0000	1KB	VAddr[10]
00_0000_0000_0000_0011	4KB	VAddr[12]
00_0000_0000_0000_1111	16KB	VAddr[14]
00_0000_0000_0011_1111	64KB	VAddr[16]
00_0000_0000_1111_1111	256KB	VAddr[18]
00_0000_0011_1111_1111	1MB	VAddr[20]
00_0000_1111_1111_1111	4MB	VAddr[22]
00_0011_1111_1111_1111	16MB	VAddr[24]

Значение полей данных в записи TLB - 1

- **PFN0, PFN1 ([31:12] или [29:10]) – Physical Frame Number**
  - Верхние биты физического адреса для четной и нечетной страницы
  - [29:10] – для конфигурации ядра MIPS, которая позволяет страницы в 1KB
    - Конфигурация MIPS microAptiv UP для PIC32MZ не позволяет страницы в 1KB; 4KB минимум
  - Для страниц больше 4KB используется только часть битов
- **V0, V1 - Valid**
  - Если не установлен, попытка доступа к странице вызывает исключение TLB Invalid

Значение полей данных в записи TLB - 2

- **RI0, RI1 – Read Inhibit**
  - Вызывает прерывание при попытке чтения страницы
  - Для работы требует, чтобы Cop0 PageGrain.RIE = 1 (Read Inhibit Enable)
- **XI0, XI1 – Execute Inhibit**
  - Вызывает прерывание при попытке чтения страницы для исполнения (fetch)
  - Для работы требует, чтобы Cop0 PageGrain.XIE = 1 (Execute Inhibit Enable)
- **D0, D1 – “Dirty” («Грязный») или Write Enable**
  - Если бит установлен, то на страницу можно писать
  - Если бит не установлен, то происходит исключение TLB Modified
  - Обработчик исключения может установить бит и при вытеснении страницы из TLB записать ее содержимое куда-нибудь (применяется для «больших» операционных систем)

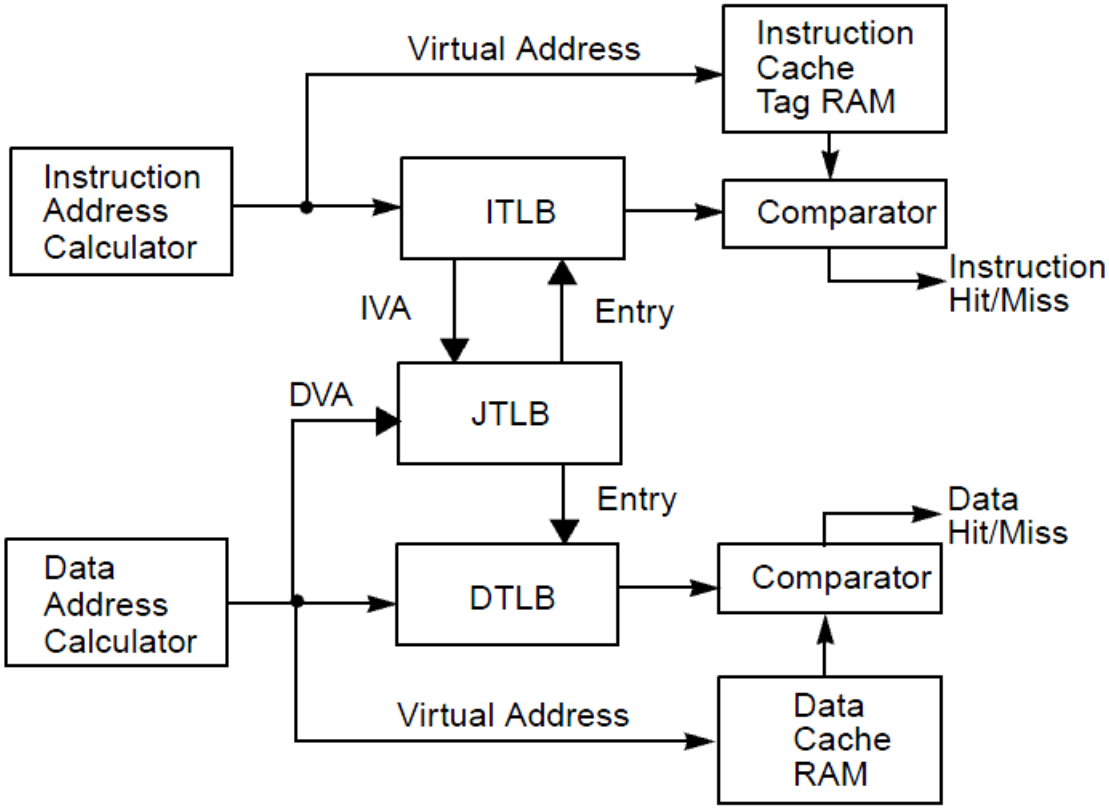
Значение полей данных в записи TLB - 3

- **C0 [2:0], C1 [2:0]**
  - Cacheability and Coherence Attributes (CCA)
  - Атрибуты кэшируемости и когерентности

C[2:0]	Coherency Attribute
000	Cacheable, noncoherent, write-through, no write-allocate
001	Cacheable, noncoherent, write-through, write-allocate
010	Uncached
011	Cacheable, noncoherent, write-back, write-allocate
100	Maps to entry 011b*
101	Maps to entry 011b*
110	Maps to entry 011b*
111	Maps to entry 010b*
* These mappings are not used on the microAptiv UP processor cores but do have meaning in other MIPS Technologies implementations. Refer to the MIPS32 specification for more information.	

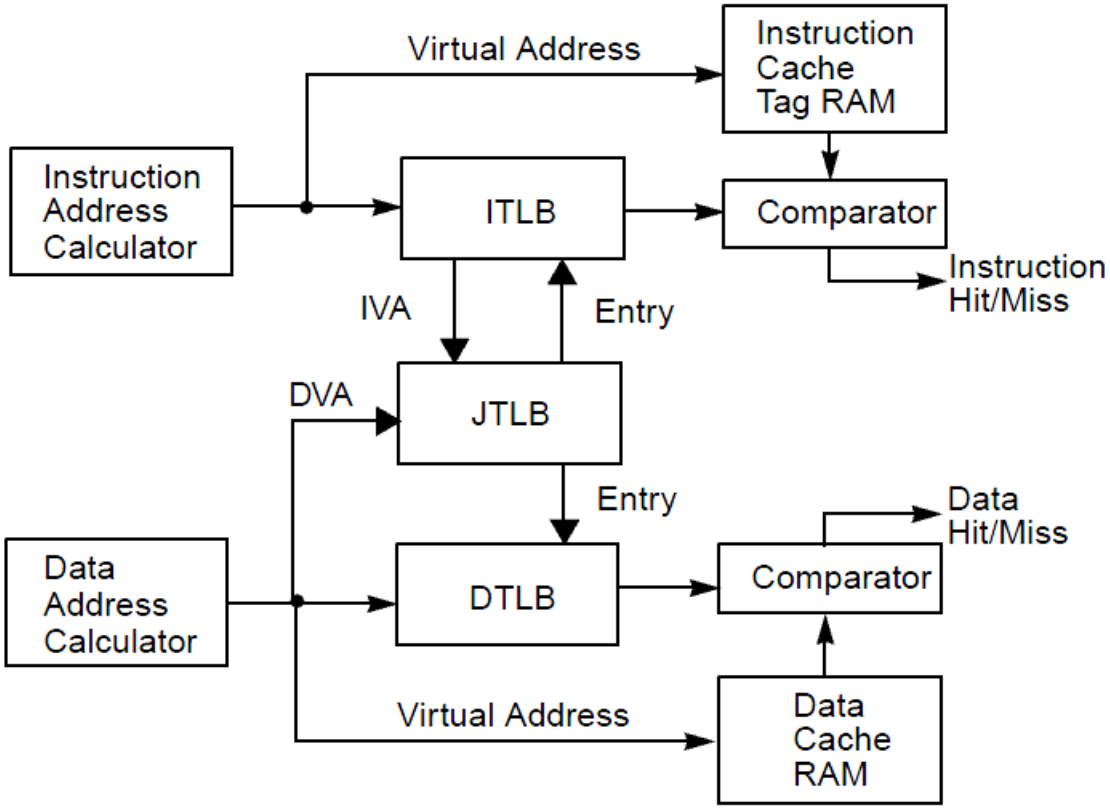
Хардверная реализация MMU в MIPS microAptiv UP

- Вместо одной ассоциативной памяти MMU использует три
  - ITLB, DTLB, JTLB
- Это трюк для повышения производительности
  - Дополнительный уровень кэширования
- Данная структура невидима для программиста
  - Для программы все выглядит как один TLB



Параметры реализации MMU в MIPS microAptiv UP

- TLB инструкций
  - Instruction TLB (ITLB)
  - 4 записи
  - Издержка промаха 2 цикла
- TLB данных
  - Data TLB (DTLB)
  - 4 записи
  - Издержка промаха 1 цикл
- Совмещенный TLB
  - Joint TLB (JTLB)
  - 16 записей



Регистры Cop0 для работы с TLB - 1

- **EntryHi – содержит VPN2 и ASID для чтения, записи и других операций**
- **PageMask – см. один из предыдущих слайдов**
- **EntryLo0, EntryLo1 – содержат PFN, V, RI, XI, D, C и G**
  - Бит G (Global) для записи ставится как G = EntryLo0.G and EntryLo1.G
- **PageGrain – содержит биты RIE и XIE разрешающие RI и XI в EntryLo**
  - При этом главное назначение Page Grain – поддержка страниц в 1KB (не в PIC32MZ)

Регистры Cop0 для работы с TLB - 2

- **Config1 – поле MMUSize содержит размер TLB минус единица**
  - Если MMUSize = 0, то TLB отсутствует и ядро использует FMT MMU (случай PIC32MX)
  - В PIC32MZ поле содержит число 15 (16 двойных записей TLB)

- **Index – индекс для чтения записи инструкцией TLBR и записи TLBWI**
  - Также служит для зондирования TLB инструкцией TLBP (TLB Probe)
  - TLBP записывает в регистр индекс найденной записи
    - Если не найдена – выставляет старший бит P (Probe Failure)

Что делать, если нужно много трансляций?

- Для многих систем количество трансляций в TLB недостаточно
- В таком случае таблица трансляций хранится в памяти, а TLB используется как кэш трансляций
- Если процессор не находит трансляцию в TLB, происходит исключение TLB Refill
- Обработчик исключения загружает трансляцию из памяти и записывает ее в случайную запись TLB

Структура таблицы страниц в памяти

- Таблица трансляций в памяти представляет собой линейный массив структур, состоящих из значений регистров Cop0 EntryLo0 и EntryLo1
- Структуры выровнены на границу 16 байт для совместимости с MIPS64

Адрес	Содержимое
PTEBase + 0	EntryLo0 для страницы 0
PTEBase + 4	
PTEBase + 8	EntryLo1 для страницы 1
PTEBase + 12	
PTEBase + 16	EntryLo0 для страницы 2
PTEBase + 20	
PTEBase + 24	EntryLo1 для страницы 3

Регистр Cop0 Context

PTEBase	BadVPN2	0

- **PTEBase [31:23] – Page Table Entry Base**
  - Записывается пользователем (операционной системой)
  - Определяет базовый адрес таблицы страниц в памяти
    - Весь регистр Context становится виртуальным адресом таблицы страниц, когда BadVPN2 = 0
- **BadVPN2 [22:4]**
  - Записывается процессором во время исключений TLB Refill, TLB Invalid и других
  - Соответствует битам [31:13] виртуального адреса, вызвавшего исключение
  - Во время исключения весь регистр Context образует адрес записи в таблице страниц
    - Context = { PTEBase, BadVPN2, 4 бита нулей для выравнивания }

Регистры Cop0 Random, Wired и команда TLBWR

- **Random**
  - Содержит псевдослучайное число в пределах от Wired до Config1.MMUSize
  - Используется командой TLBWR, TLB Write Random

- **Wired**
  - Содержит нижнюю границу для значения Random
  - Необходим, чтобы при использовании TLB как кэша трансляций из памяти, некоторые записи не выталкивались бы из TLB

- **Команда TLBWR**
  - Записывает содержимое регистров EntryHi, PageMask, EntryLo0 и EntryLo1 в запись TLB, индексом которой в TLB является регистр Random

Минимальный обработчик исключения TLB Refill

```
    .set    noreorder           // Выключить слишком умную оптимизацию
                                // ассемблером
tlb_miss_32:
    mfc0    k1,  C0_Context     // Взять адрес записи в таблице страниц
    lw      k0,  0(k1)          // Загрузить EntryLo0 из таблицы в памяти
    lw      k1,  8(k1)          // Загрузить EntryLo1 из таблицы в памяти
    mtc0    k0,  C0_EntryLo0
    mtc0    k1,  C0_EntryLo1
    ehb                         // Гарантия, что в регистры записалось
                                // перед использованием

    tlbwr                       // Записать EntryHi, PageMask,
                                // EntryLo0, EntryLo1 в случайное место TLB
                                // При TLB Refill, EntryHi уже установлен
    eret                        // Вернуться из исключения
```

Как экономить на размере таблицы страниц

- **Таблица страниц для 32-битного адресного пространства – 8MB**
  - Для 64-битного вообще необозримая
- **Для сокращения размеров саму таблицу можно хранить в виртуальной памяти, отображаемой с помощью TLB**
  - Context.PTEBase не обязан указывать на неотображаемой адрес
- **Когда страницы нет в TLB во время TLB Refill и вообще любого исключения (Cop0 Status.EXL=1), происходит другое исключение TLB Invalid на общий вектор исключений (не вектор для TLB Refill)**
  - Второе прерывание позволяет подновить таблицу страниц в памяти
  - При этом второе исключение возвращается прямо в код, вызвавший первое исключение
  - Исключение TLB Refill происходит снова, но на этот раз с обновленной таблицей и без последующего TLB Invalid

Другой пример реализации: MIPS32 Linux

- **Двухуровневая таблица страниц в памяти**

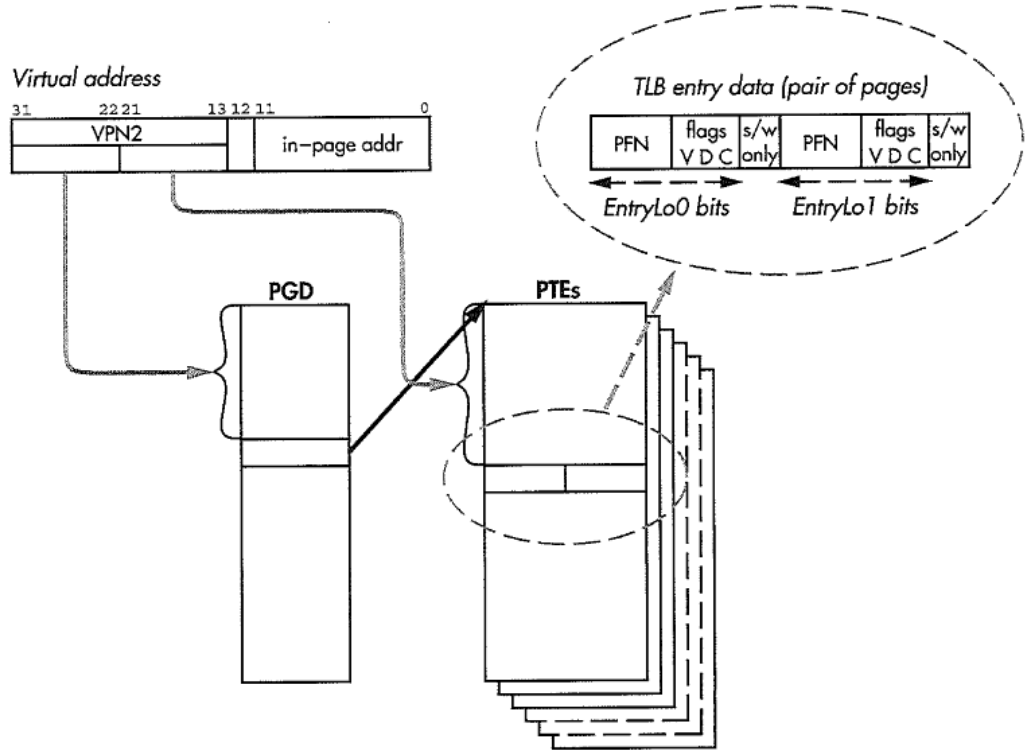


FIGURE 14.4 Linux two-level page table (32-bit MIPS design).

Источник:

Dominic Sweetman.

See MIPS Run Linux

Инициализация TLB

Необходима в boot code перед любой работой с TLB

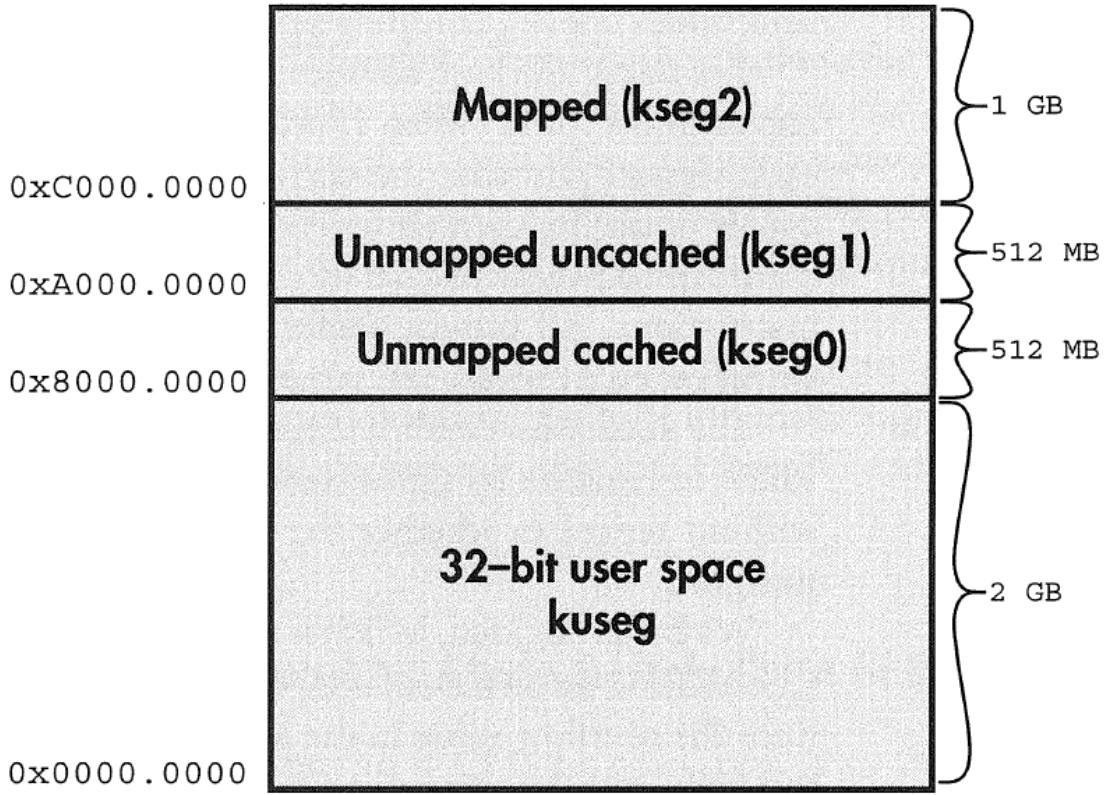
- **Инициализировать Context.PTEBase, Wired, PageGrain.RIE и PageGrain.XIE**
- **Записать нули в EntryHi, PageMask, EntryLo0 и EntryLo1**
- **Пройти в цикле все индексы TLB от нуля до Config1.MMUSize**
- **В теле цикла записывать Index, после чего выполнять инструкции EHB и TLBWI**
  - Инструкция EHB (Clear Hazard Barrier) нужна для гарантии, что запись в Index произошла перед выполнением TLBWI, которая записывает в TLB данные из EntryHi, PageMask, EntryLo0 и EntryLo1
- **Код, инициализирующий TLB, должен исполняться из области памяти, не использующий отображение с помощью TLB (kseg0 или kseg1)**

For Distribution

Карты виртуальных адресов в архитектуре MIPS

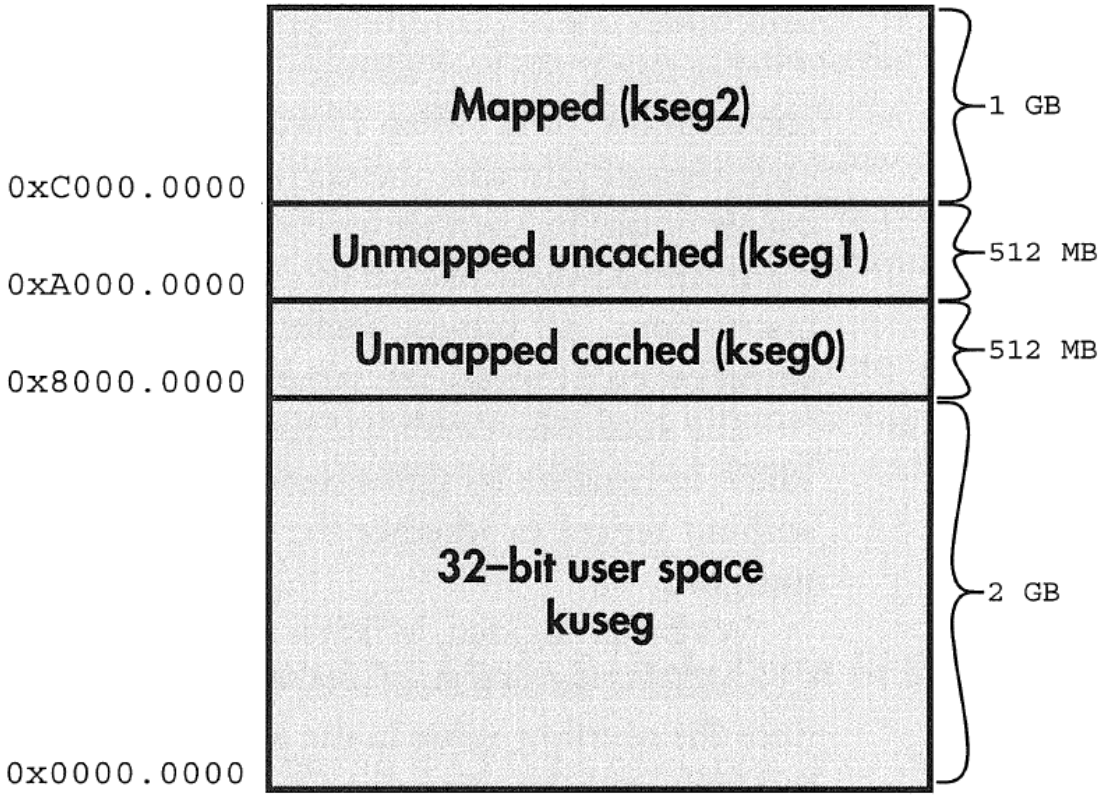
Классическая карта виртуальных адресов MIPS

- **Не все адреса транслируются TLB**
- **Сегменты kseg0 и kseg1**
  - Транслируются простым удалением верхних двух битов адреса
- **kseg0**
  - Кэшируемый
  - Используется для OS
- **kseg1**
  - Некэшируемый
  - Используется для загрузки – адрес 0xbfc00000
  - Используется для регистров ввода-вывода



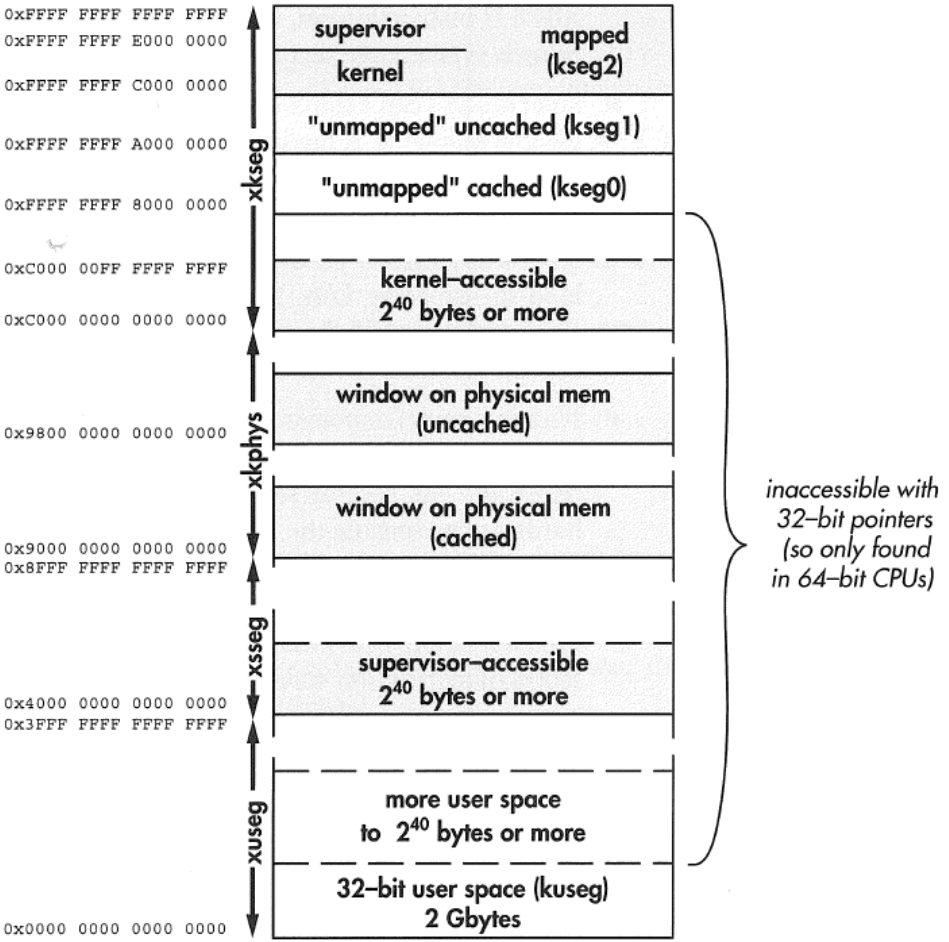
Классическая карта виртуальных адресов MIPS - 2

- **Пользователь использует только нижние два гигабайта - kuseg**
  - Отображаемые с TLB
  - В ядрах MIPS interAptiv и proAptiv пользователь может иметь доступ к 3.5GB с помощью нового механизма EVA (Enhanced Virtual Addressing)



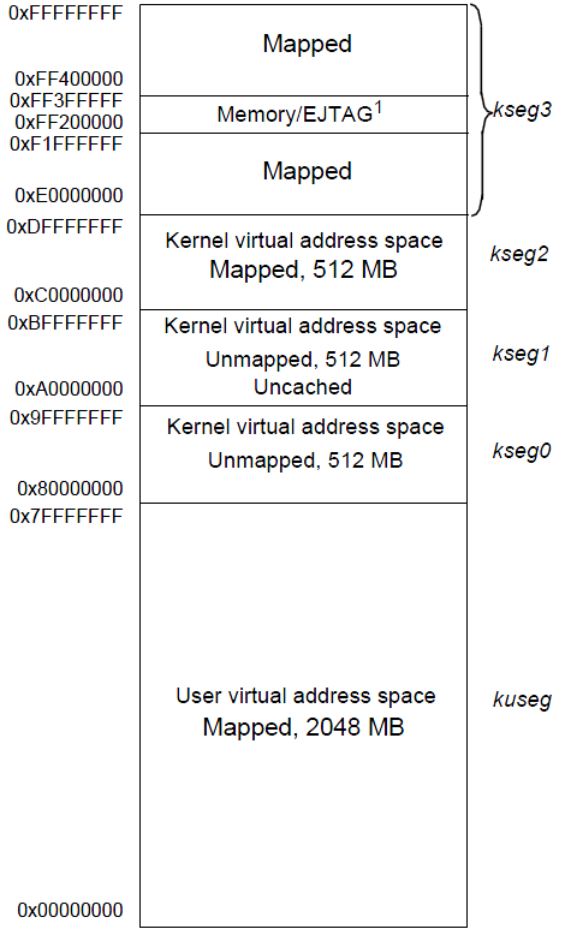
Для сравнения – карта виртуальных адресов MIPS64

- Если отрезать 32 верхних бита адреса – совместима с MIPS32
- Регион от 0x8000_0000 до 0xFFFF_FFFF_8000_0000 доступен только с 64-битными указателями
  - 64-битная карта «упакована» внутри 32-битной карты



Карта виртуальных адресов MIPS M4K и MIPS microAptiv UP

- Дополнительные детали – адреса для отладочного интерфейса EJTAG
- Для MIPS M4K TLB не используется, вместо этого используется FMT
  - Fixed Mapping Translation



For Distribution

Трансляция с фиксированным отображением
Fixed Mapping Translation (FMT)

Реализация MMU с помощью FMT

Fixed Mapping Translation – фиксированное отображение адресов

- **Доступные пользователю адреса в user-mode сдвигаются на 0x40000000**
  - Это способ имитации части поведения TLB без TLB как такового
- **Не защищает различные пользовательские программы друг от друга**
- **Требует минимального количества аппаратных ресурсов**
- **Опция FMT реализована на всех ядрах MIPS Technologies**
- **Используется в ядре MIPS M4K / микроконтроллере PIC32MX**

Трансляция виртуальных адресов в физические с помощью Fixed MappingTranslation

Figure 5 FMT Memory Map (ERL=0) in the M4K Core

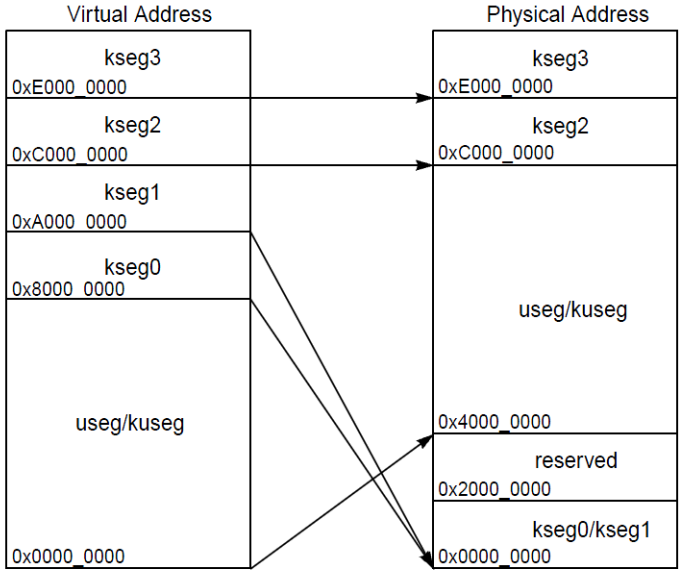
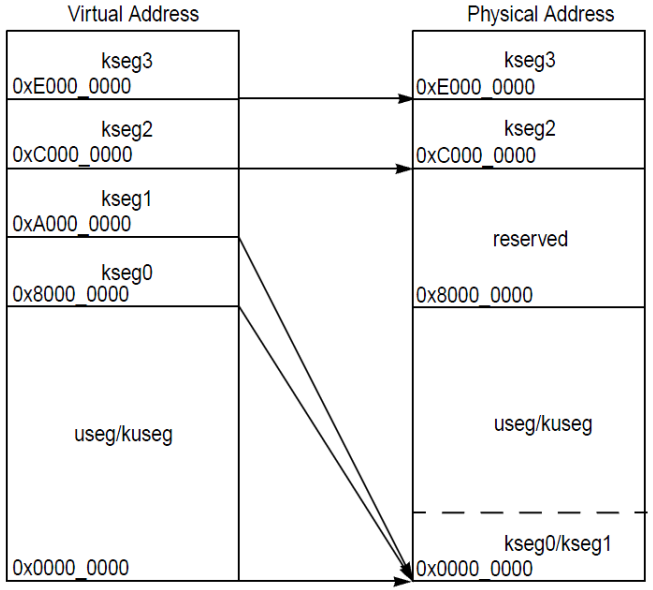


Figure 6 FMT Memory Map (ERL=1) in the M4K Core



Источник: MIPS32® M4K® Processor Core Datasheet March 4

For Distribution

Спасибо!