Aleste LX

Система упраления памятью и расширения

h2w

27 декабря 2025 г.

Содержание

1. Базовые принципы

Система построена на процессоре Z80 с расширенным 24-битным адресным пространством (16 МБ) через механизм банкового переключения, аналогичный MSX, но с расширенной функциональностью. Поддерживает два основных режима работы для обратной совместимости и расширенной функциональности.

2. Режимы работы

Состояние системы определяется двумя независимыми параметрами: режимом привилегий и набором устройств ввода/вывода. `S = {supervisor_mode, native_mode, mmio_userlock}`

2.1. Режим привилегий (Privilege Mode)

Определяет уровень доступа к ресурсам системы.

- **User Mode**: Выполнение прикладного кода. Ограниченный доступ к слотам памяти и портам ввода/вывода.
- **Supervisor Mode**: Выполнение кода ядра (привилегированный режим). Полный доступ ко всем ресурсам системы. Аппаратный вход по событиям (trap). После сброса система всегда запускается в этом режиме.

2.2. Режим совместимости (Compatibility Mode)

Определяет, какой набор устройств ввода/вывода и механизм управления памятью активен.

- **Legacy Mode**: Полная аппаратная и программная совместимость с Amstrad CPC. Используется портовая адресация CPC. В режиме `supervisor` этот режим игнорируется, всегда действует `native`.
- **Native Mode**: Расширенный режим с новой функциональностью, использованием современного набора портов ввода/вывода и полноценным слотовым механизмом.

2.3. Таблица режимов работы

Режим	Supervisor	Native	Эффективный режим	Описание
Сброс	1	1	Supervisor + Native	Начальное состояние после сброса
Trap	1	(не меняется)	Supervisor + (Native или Legacy)	Аппаратный переход в супервизор
Программный	из GLOBAL_CTRL	из GLOBAL_CTRL	Как установлено	Управление через порт F0h
User	0	0	User + Legacy	Совместимость с CPC
User	0	1	User + Native	Расширенный режим

Важно: В режиме Supervisor бит `native_mode` не играет роли, всегда используется Native-механизм доступа к памяти и устройствам.

3. Организация памяти

3.1. Логическое и физическое пространство

- **Физическое адресное пространство**: 24 бита (16 МБ).
- **Логическое адресное пространство Z80**: 16 бит (64 КБ).
- **Логические слоты**: 64 КБ логического пространства делится на 4 слота по 16 КБ:
  - **Slot 0**: 0000-3FFF (CPC RAM, базовая память)
  - **Slot 1**: 4000-7FFF (CPC ROM, ПЗУ системы)
  - **Slot 2**: 8000-BFFF (User extended memory)
  - **Slot 3**: C000-FFFF (Supervisor memory, привилегированная)
- **Банковое переключение**: Для отображения 16КБ логического слота на 16МБ физической памяти используется 8-битный регистр страницы (банка). `physical_address = {bank_reg[7:0], cpu_a[13:0]}`

3.2. MMIO Пространство: Концепция HI и LO

Последние 64 КБ физического адресного пространства (адреса `FF0000h - FFFFFFh`) зарезервированы для памяти устройств ввода/вывода (MMIO). Это пространство логически разделено на две части:

- **`MMIO_HI` (Адреса `FF4000h - FFFFFFh`)**: Эта область предназначена для **эмуляции Legacy-устройств** Amstrad CPC. Устройства в этой области **жёстко привязаны** к своим физическим адресам (например, `Gate Array` на `FF7F00h`, `CRTC` на `FFBC00h`). В Legacy-режиме обращение к портам CPC (7FXXh, BCXXh и т.д.) транслируется в прямые обращения к этим фиксированным адресам в MMIO_HI через механизм полного отображения адреса шины Z80 `A[15:0]`.
- **`MMIO_LO` (Адреса `FF0000h - FF3FFFh`, 16КБ)**: Эта область предназначена для **новых устройств** системы Aleste LX и доступна через механизм банкового переключения (окно). Пространство организовано в 128 страниц по 128 байт.

Важно: Адрес устройства в MMIO-пространстве является частью его архитектурного контракта. Это позволяет не только CPU, но и другим мастерам шины (например, **DMA-контроллеру**) напрямую обращаться к периферийным устройствам по их фиксированным адресам на быстрой шине Wishbone, что критично для производительности. Однако MMU не отображается в этом пространстве и является устройством с доступом только от процессора.

3.3. Организация MMIO_LO

Базовый принцип: 128 страниц × 128 байт = 16,384 байта (16КБ)
Формула преобразования адреса:

```
physical_address = 0xFF0000 + (MMIO_PAGE << 7) + A[6:0]
```

где:

- `MMIO_PAGE` - 7-битный регистр (значения 0-127)
- `A[6:0]` - младшие 7 бит адреса от CPU (порт 00-7Fh)

Карта размещения устройств в MMIO_LO:

Страница	Адресный диапазон	Размер	Устройство	Описание
0	FF0000-FF001F	32	PIC Controller	Контроллер прерываний
0	FF0020-FF003F	32	IPC Mailbox	Системный почтовый ящик
0	FF0040-FF005F	32	System Timer	Системный таймер
0	FF0060-FF007F	32	RTC Controller	Часы реального времени
0	FF0080-FF00BF	64	MMU Extension	Расширенный доступ к MMU
0	FF00C0-FF00CF	16	Reserved	Зарезервировано
0	FF00D0-FF00DF	16	Reserved	Зарезервировано
0	FF00E0-FF00EF	16	Reserved	Зарезервировано
0	FF00F0-FF00FF	16	Reserved	Зарезервировано
1	FF0080-FF00FF	128	Reserved	Зарезервировано
2	FF0100-FF017F	128	Legacy Emulation	Эмуляция Legacy устройств
4	FF0200-FF027F	128	DMA Controller	Контроллер DMA
6	FF0300-FF037F	128	Graphics Chip	Графический чип
8	FF0400-FF047F	128	Sound Chip	Звуковой чип
...	...	...	...	...

Примечание: Устройства размещаются с выравниванием по 32 байта для простоты декодирования.

4. Система регистров управления

4.1. Управляющие регистры (Native Mode)

Доступны через порты ввода/вывода в диапазоне F0-FF.

- **`MMIO_WINDOW` (Ports `00h-7Fh`)**: 128-байтное окно для доступа к MMIO_LO.
  - Доступ: RW (в зависимости от привилегий)
  - Формула: `mmio_physical_address = FF0000h + {MMIO_PAGE[6:0], cpu_a[6:0]}`
  - В User Mode при `mmio_userlock=1` доступ запрещен
- **`MMIO_PAGE` (Port `F1h`)**: 7-битный регистр выбора страницы MMIO.
  - Доступ: RW (в зависимости от привилегий)
  - Диапазон: 0-127 (7 бит)
  - В User Mode при `mmio_userlock=1` доступ запрещен
- **`SYS_CALL_CMD_PORT` (Port `F2h`)**: Порт для вызова команд операционной системы.
  - Доступ: W (всегда)
  - В Legacy Mode обращение к `F2XXh` автоматически перенаправляется сюда
- **`GLOBAL_CTRL` (Port `F0h`)**: Главный регистр управления режимами.

Бит	Группа	Назначение
0	Functional	**supervisor_mode**: 1=Режим супервизора (устанавливается аппаратно или программно).
1	Functional	**native_mode**: 1=Нативный режим, 0=Legacy режим.
2	Functional	**supervisor_hook**: 1=Включить аппаратный захват (trap) по адресам 0038h/0066h.
3	Reserved	Для будущего расширения функциональности.
4	Security	**mmio_user_unlock**: 1=Разрешить прямой доступ к MMIO 0=Заблокировать доступ -> только SysCall **← По умолчанию 0!**
5-7	Reserved	Для будущих битов безопасности.

- **`CLOCK_CTRL` (Port `F3h`)**: Регистр управления тактовой частотой CPU.
  - Бит 0-3: Делитель частоты шины для получения тактовой процессора
  - Доступ: RW (только в Supervisor режиме)
- **`SUPER_SLOT` (Port `F9h`)**: Выбор активного слота для режима **supervisor**.
- **`USER_SLOT` (Port `FBh`)**: Выбор активного слота для режима **user**.
- **`BANK_0` (Port `FCh`)**: Выбор страницы памяти для слота 0 (0000-3FFF) в текущем режиме.
- **`BANK_1` (Port `FDh`)**: Выбор страницы памяти для слота 1 (4000-7FFF) в текущем режиме.

- **`BANK_2` (Port `FEh`)**: Выбор страницы памяти для слота 2 (8000-BFFF) в текущем режиме.
- **`BANK_3` (Port `FFh`)**: Выбор страницы памяти для слота 3 (C000-FFFF) в текущем режиме.
  - Доступ к BANK_0-3: RW (только в Native/Supervisor режимах)

4.2. Расширенный доступ к MMU

- **`MMU_EXT` (Addresses `FF0080h-FF00BFh`)**: Расширенный доступ к состоянию маппера.
  - **Доступ**: RW (только CPU в Native/Supervisor режимах)
  - **Важно**: DMA и другие мастера шины **не имеют доступа** к этим регистрам
  - Предназначен для быстрого сохранения/восстановления состояния всего маппера

4.3. Регистры совместимости (Legacy Mode)

Предназначены для эмуляции окружения Amstrad CPC. Доступны через порты вида `XXYYh`.

- **`RMR` / `MMR` (Port `7FXXh`)**: Регистры управления памятью и графикой CPC.
- **`ROM_SEL` (Port `DFXXh`)**: Выбор банка ROM для верхней области памяти (C000-FFFF).
- **`SYSCALL_LEGACY` (Port `F200h`)**: Вызов функции операционной системы. Автоматически перенаправляется в Native порт `F2h`.

5. Механизмы переключения режимов

5.1. Программное переключение

Запись в регистр `GLOBAL_CTRL` (порт `F0h`): `S ← {data[0], data[1]}` (устанавливаются режимы supervisor и native).

5.2. Аппаратное переключение (Trap) в Supervisor Mode

Вход в привилегированный режим осуществляется аппаратно при выполнении условия: `trap = (supervisor_hook =` 1) ∧ (сигнал M1 активен) ∧ (address_bus `= 0038h v address_bus =` 0066h)=

5.3. Переключение через Syscall

1. Протокол системного вызова
   - Код функции: Передается в регистре A.
   - Аргументы: Передаются через регистры. Стандартное соглашение:
     - BC --- 1-й аргумент (хэндл файла, номер устройства)
     - DE --- 2-й аргумент (указатель на данные или размер)
     - HL --- 3-й аргумент (указатель на буфер или дополнительный параметр)
   - Возвращаемое значение: В регистре A (статус) или в HL (результат).
   - Сохранение регистров: Обработчик системного вызова в супервизоре сохраняет и восстанавливает все регистры, кроме тех, что используются для возврата значения.

2. Пример корректного вызова

```
; Подготовка аргументов для syscall_write код( функции 0x02)
LD A, 2          ; A = Код функции 'write'
LD BC, file_handle ; BC = й1- арг.: хэндл файла
LD DE, data_size ; DE = й2- арг.: размер данных
LD HL, data_buffer ; HL = й3- арг.: указатель на буфер
CALL syscall     ; Вызов обертки
; Проверка возвращаемого значения в A успех(0=, иначе код ошибки)
```

3. Реализация обертки **Native Mode**:

```
syscall:
    OUT (F2h), A    ; !!! ВОЛШЕБНАЯ КОМАНДА !!!
    RET             ; Сюда программа вернется после выполнения syscall
```

Legacy Mode:

```
syscall_legacy:
    OUT (F200h), A  ; !!! ВОЛШЕБНАЯ КОМАНДА !!!
    RET
```

Аппаратная трансляция:

```
if (native_mode == 1 && io_addr == 8'hF2)
    syscall_trigger = 1;
else if (native_mode == 0 && io_addr == 16'hF200)
    syscall_trigger = 1;
```

4. Supervisor-mode обработчик

```
Syscall_Dispatcher:
    ; A уже содержит код функции
    CP 0
    JP Z, Syscall_OpenFile
    CP 1
    JP Z, Syscall_ReadFile
    CP 2
    JP Z, Syscall_WriteFile ; <--- Переход сюда
    ; ... и тд..

Syscall_WriteFile:
    ; Параметры уже в регистрах: Акод=, BC=file_handle, HL=buffer, DE=size
    ; Осталось только выполнить работу.
    ; ... обработка ...
    RET ; Возврат из диспетчера
```

5.4. Выход из Supervisor Mode

Выход является **многошаговым процессом**, синхронизированным с циклом команд процессора:

1. **Инициация выхода**: Программа ядра выполняет команду `OUT` с данными `0` в регистр `GLOBAL_CTRL`.

```
LD A, 00000000b  ; supervisor=0, native=0 или 1
OUT (F0h), A     ; запись в GLOBAL_CTRL
```

2. **Детекция**: Аппаратура детектирует эту запись и устанавливает внутренний флаг `EXIT_SUPERVISOR_F = 1`.

3. **Исполнение RETN**: Следующая команда (обычно `RETN`) исполняется полностью в супервизорном режиме.

4. **Синхронизация по M1**: Аппаратура ожидает **следующий** цикл шины `M1`.

5. **Переключение**: При активации `M1` сбрасывается режим `supervisor_mode`.

6. **Выборка в User Mode**: Выборка команды по адресу возврата происходит уже в User Mode.

Итог: Команда `RETN` выполняется в супервизоре, а следующая за ней команда --- уже в пользовательском режиме.

5.5. Определение активного слота

Активный слот выбирается в зависимости от текущего режима привилегий: `current_slot = (supervisor_mode =` 1) ? SUPER_SLOT : USER_SLOT=

5.6. Алгоритм работы маппера

Обращение к регистрам банков `BANK_0`...=BANK_3= перенаправляется в один из 16-ти внутренних регистров маппера в зависимости от активного слота.

```
void write_to_bank_register(uint8_t address_low_bits, uint8_t data) {
    page_index = (current_slot * 4) + (address_low_bits & 0b00000011);
    internal_mapper_registers[page_index] = data;
}
```

5.7. Политика безопасности доступа

- **User Native Mode**: При `mmio_userlock=1` запрещен доступ к портам 00-7F и F1, F3, F9, FB-FF. Разрешен только F2 (SysCall).
- **User Legacy Mode**: При `mmio_userlock=1` Разрешен F200 (SysCall) и все стандартные CPC-порты.
- **Supervisor Mode**: Доступ разрешен всегда.

5.8. Системный вызов для снятия блокировки доступа к MMIO

`SYS_MMIO_ACCESS_REQUEST` (Код функции: `0xFE`)

- **Назначение**: Запрос на снятие блокировки прямого доступа к MMIO.
- **Вход**: `A = 0xFE`, `BC = 0x0001` (флаг запроса на MMIO).

- **Выход**: `A = Код результата` (0: разрешено, 1: запрещено).
- **Поведение**: Ядро проверяет цифровую подпись программы. В случае успеха выставляет `mmio_userlock = 0`.

6. Детальная информация о регистрах MMU нативного режима

6.1. Регистр GLOBAL_CTRL

Полная таблица регистра приведена в разделе 4.1.

6.2. Расширенный доступ к мапперу

Предназначен для использования системой для быстрого сохранения и восстановления состояния всего маппера.

WB Address	Slot	CPU Page	Назначение
FF00E0	0	0000	Слот 0, банк для 0000-3FFF
FF00E1	0	4000	Слот 0, банк для 4000-7FFF
FF00E2	0	8000	Слот 0, банк для 8000-BFFF
FF00E3	0	C000	Слот 0, банк для C000-FFFF
FF00E4	1	0000	Слот 1, банк для 0000-3FFF
FF00E5	1	4000	Слот 1, банк для 4000-7FFF
FF00E6	1	8000	Слот 1, банк для 8000-BFFF
FF00E7	1	C000	Слот 1, банк для C000-FFFF
FF00E8	2	0000	Слот 2, банк для 0000-3FFF
FF00E9	2	4000	Слот 2, банк для 4000-7FFF
FF00EA	2	8000	Слот 2, банк для 8000-BFFF
FF00EB	2	C000	Слот 2, банк для C000-FFFF
FF00EC	3	0000	Слот 3, банк для 0000-3FFF
FF00ED	3	4000	Слот 3, банк для 4000-7FFF
FF00EE	3	8000	Слот 3, банк для 8000-BFFF
FF00EF	3	C000	Слот 3, банк для C000-FFFF

Важно: Эти регистры доступны только CPU в Native/Supervisor режимах. DMA и другие мастера шины не имеют к ним доступа.

7. IPC Mailbox (Системный почтовый ящик)

Назначение: Обеспечить сверхбыстрый, детерминированный и безопасный обмен служебными команд и уведомлениями между **Супервизором (Ядром)** и **Драйверами**, работающими в пространстве ядра.

Адрес: `FF0020h-FF003Fh` (32 байта)

Аппаратная Реализация: Состоит из набора регистров:

- **`MAILBOX_CMD`**: Драйвер записывает сюда код действия (напр., `SND_BUF_EMPTY`, `DSK_IO_DONE`).
- **`MAILBOX_DATA_*`**: Параметры команды (номер канала, адрес и т.д.).
- **`MAILBOX_STATUS`**: Биты `FULL` (сообщение не прочитано) и `ACK` (подтверждение обработки).

- **MAILBOX_INT_CTRL**: Управление прерыванием по приему сообщения.

Протокол: Драйвер пишет команду и данные, аппаратура выставляет FULL и генерарует прерывание ядру (если разрешено). Ядро, обработав сообщение, сбрасывает флаг FULL.

8. Математическая модель

8.1. Преобразование адреса

Алгоритм трансляции логического адреса в физический зависит от режимов.

```
physical_address =
    if (supervisor_mode == 1) or (native_mode == 1) then
        // Нативный или супервизорный режим: используем маппер
     { internal_mapper_registers[current_slot * 4 + page_number], cpu_a[13:0] }
    else
        // Legacy-режим: используем механику CPC
        legacy_cpc_mapping(cpu_a, RMR, MMR)
```

8.2. Автомат состояний

Изменение состояния системы можно описать как:

```
S(t+1) =
    if trap_condition then
        {1, S(t).native_mode} // Вход в Supervisor, Native режим не меняется
    else if exit_supervisor_pending & next_M1_cycle then
        {0, S(t).native_mode} // Выход в User, Native режим не меняется
    else if io_write_to_F0h then
        {data[0], data[1]}    // Программная установка обоих режимов
    else
        S(t)                  // Состояние не изменяется
```

9. Таблица регистров и адресов

Device / Description	RW	Legacy IO Address	Legacy WB Address	Native IO Address	Native WB Address	Size	CPU Access	DMA Access
MMIO_WINDOW	RW	-	-	00-7Fh	FF0000+{ PAGE, ADDR[6:0]} }	128	RW*	R*
GLOBAL_CTRL	RW	-	-	F0h	-	1	RW*	-
MMIO_PAGE	RW	-	-	F1h	-	1	RW*	-
SYSCALL	W	F200h	FFF200h	F2h	-	1	W	-
CLOCK_CTRL	RW	-	-	F3h	-	1	RW (Supervisor)	-
SUPER_SLOT	RW	-	-	F9h	-	1	RW*	-
USER_SLOT	RW	-	-	FBh	-	1	RW*	-
BANK_0	RW	-	-	FCh	-	1	RW*	-
BANK_1	RW	-	-	FDh	-	1	RW*	-
BANK_2	RW	-	-	FEh	-	1	RW*	-
BANK_3	RW	-	-	FFh	-	1	RW*	-
MMU_EXT	RW	-	-	через окно	FF0080h-FF00BFh	64	RW (CPU only)	-
PIC Controller	RW	-	-	через окно	FF0000h-FF001Fh	32	RW	RW
IPC Mailbox	RW	-	-	через окно	FF0020h-FF003Fh	32	RW	-
System Timer	RW	-	-	через окно	FF0040h-FF005Fh	32	RW	-
RTC Controller	RW	-	-	через окно	FF0060h-FF007Fh	32	RW	-
Legacy Emulation	RW	-	-	через окно	FF0100h-FF017Fh	128	RW	RW
DMA Controller	RW	-	-	через окно	FF0200h-FF027Fh	128	RW	-
Graphics Chip	RW	-	-	через окно	FF0300h-FF037Fh	128	RW	RW
Sound Chip	RW	-	-	через окно	FF0400h-FF047Fh	128	RW	RW
Gate Array	W	7FXXh	FF7FXXh	-	-	-	W	W
CRTC 6845	RW	BCXXh/BDXXh	FFBCXXh/FFBDXXh	-	-	-	RW	RW
Upper ROM Select	W	DFXXh	FFDFXXh	-	-	-	W	-
PPI 8255	RW	F4XXh-F7XXh	FFF4XXh-FFF7XXh	-	-	-	RW	RW

Условные обозначения:

- – * доступно только в Native/Supervisor режимах
- RW* - чтение/запись с ограничениями по привилегиям
- R* - чтение с ограничениями по привилегиям
- <<CPU only>> - доступен только процессору, не доступен через шину Wishbone

10. Legacy MMU регистры (только для совместимости с CPC)

10.1. Register RMR (Control Interrupt counter, ROM mapping and Graphics mode)

Это общий регистр, отвечающий за графический режим и конфигурацию ROM в CPC.

bit	Action
7	Must be 1
6	Must be 0
5	must be 0 on Plus machines with ASIC unlocked
4	irq_control - Interrupt generation control
3	upper_rom - 1=Upper ROM area disable, 0=Upper ROM area enable
2	lower_rom - 1=Lower ROM area disable, 0=Lower ROM area enable
1-0	graphic_mode - Graphics Mode selection

- `upper_rom` начинается с C000
- `lower_rom` начинается с 0000
- `graphic_mode` определяет режим графики (наследие CPC)

10.2. Upper ROM Select (DFXX)

Восьмибитный регистр, который выбирает 16КБ банк памяти в адресах C000-FFFF когда `upper_rom` равна 0.

10.3. Register MMR (RAM memory mapping)

Бит	Назначение
7	Must be 1
6	Must be 1
5-3	**bbb** - Старшие биты номера банка. Выбирают блок по 64КБ в расширенной памяти.
2-0	**ccc** - Конфигурация, определяющая младшие биты номера банка.

CPC 128 Memory map

-Address-	0	1	2	3	4	5	6	7
C000-FFFF	RAM_3	RAM_7	RAM_7	RAM_7	RAM_3	RAM_3	RAM_3	RAM_3
8000-BFFF	RAM_2	RAM_2	RAM_6	RAM_2	RAM_2	RAM_2	RAM_2	RAM_2
4000-7FFF	RAM_1	RAM_1	RAM_5	RAM_3	RAM_4	RAM_5	RAM_6	RAM_7
0000-3FFF	RAM_0	RAM_0	RAM_4	RAM_0	RAM_0	RAM_0	RAM_0	RAM_0

CPC 512 KB Memory map

Адрес	ccc=0	ccc=1	ccc=2	ccc=3	ccc=4	ccc=5	ccc=6	ccc=7
0000-3FFF	0	0	bbb*4 + 0	0	0	0	0	0
4000-7FFF	1	1	bbb*4 + 1	3	bbb*4 + 0	bbb*4 + 1	bbb*4 + 2	bbb*4 + 3
8000-BFFF	2	2	bbb*4 + 2	2	2	2	2	2
C000-FFFF	3	bbb*4 + 3	bbb*4 + 3	bbb*4 + 3	3	3	3	3

10.4. CPU Clock Control

Бит	Назначение
7-4	Reserved
3-0	**dddd** - Делитель частоты шины для получения тактовой процессора.

Регистр `CLOCK_CTRL` (F3h) доступен только в Supervisor режиме.

11. TAG-based Адресация Wishbone

11.1. Обзор

В FPGA реализации система использует TAG-based адресацию на шине Wishbone для упрощения декодирования:

- **Master устройства** выводят только сырые адреса
- **Slave устройства** используют 3-битные TAG сигналы для идентификации типа адресного пространства
- **Система interconnect** выполняет централизованное преобразование адреса в TAG

11.2. Address-to-TAG Mapping

```
// Address regions
logic mmio_space, mmio_4000_FFFF, mmio_native, mmio_legacy;
assign mmio_space     = (wb_adr_i[23:16] == 8'hFF);
assign mmio_4000_FFFF = (wb_adr_i[15] || wb_adr_i[14]);   // 0x4000-0xFFFF
assign mmio_native    = mmio_space && !mmio_4000_FFFF;    // 0xFF0000-0xFF3FFF
assign mmio_legacy    = mmio_space &&  mmio_4000_FFFF;    // 0xFF4000-0xFFFFFF

// 3-bit TAG encoding - mutually exclusive
always_comb begin
    wb_tag_o = 3'b000;
    if (mmio_native)       wb_tag_o = 3'b010;  // Native IO
    else if (mmio_legacy)  wb_tag_o = 3'b100;  // Legacy IO
    else                   wb_tag_o = 3'b001;  // Memory space
end
```

11.3. Кодировка TAG

```
TAG[2:0] Encoding:
000 - Reserved
001 - Memory space        (0x000000-0xFEFFFF) - Основная память
010 - Native IO Space     (0xFF0000-0xFF3FFF) - Новые устройства
100 - Legacy IO Space     (0xFF4000-0xFFFFFF) - Устройства совместимости
```

11.4. Преимущества подхода

- **Быстрое декодирование**: Slave устройства проверяют только TAG биты
- **Масштабируемость**: Легко добавлять новые типы адресных пространств
- **Совместимость**: Полное сохранение Legacy адресации CPC

12. Состояния после сброса

Регистр	Адрес	Значение после сброса	Описание
GLOBAL_CTRL	F0h	00000110b	supervisor=1, native=1, mmio_userlock=0
MMIO_PAGE	F1h	00000000b	Страница MMIO 0
SUPER_SLOT	F9h	00000000b	Слот 0 для Supervisor
USER_SLOT	FBh	00000000b	Слот 0 для User
BANK_0-BANK_3	FCh-FFh	00000000b	Банки памяти 0
CLOCK_CTRL	F3h	00000111b	Делитель частоты (зависит от реализации)

13. Примеры кода для разработчиков

13.1. Инициализация системы после сброса

```
start:
    ; Система запущена в Supervisor+Native режиме

    ; Настройка банков памяти для Supervisor
    LD A, 0
    OUT (F9h), A           ; SUPER_SLOT = 0

    ; Настройка банков памяти для User
    OUT (FBh), A           ; USER_SLOT = 0

    ; Настройка частоты CPU только( Supervisor)
    LD A, 00000111b        ; Делитель частоты
    OUT (F3h), A           ; CLOCK_CTRL

    ; Переход в User Native Mode
    LD A, 00000010b        ; supervisor=0, native=1
    OUT (F0h), A           ; GLOBAL_CTRL

    ; Теперь в User Native Mode
    ; ...
```

13.2. Запрос доступа к MMIO

```
request_mmio_access:
    LD A, 0FEh             ; SYS_MMIO_ACCESS_REQUEST
    LD BC, 0001h           ; флаг запроса на MMIO
    CALL syscall           ; через обертку
    OR A                   ; проверка результата
    RET NZ                 ; ошибка если A ≠ 0
```

```
    ; теперь mmio_userlock=0, доступ к MMIO разрешен
    RET
```

13.3. Обработчик аппаратного Trap

```
    ; Установка обработчика в Supervisor режиме
    LD A, 00000111b        ; supervisor=1, native=1, trap_enabled=1
    OUT (F0h), A           ; GLOBAL_CTRL

    ; Теперь при обращениях к 0038h/0066h будет аппаратный переход
    ; в Supervisor Mode
```

13.4. Сохранение/восстановление состояния маппера

```
save_mapper_state:
    ; Сохранение состояния маппера через MMU_EXT
    LD HL, save_buffer
    LD BC, 0040h           ; 64 байта
    LD A, 0                ; MMIO страница 0
    OUT (F1h), A           ; MMIO_PAGE
    LD A, 80h              ; Начало MMU_EXT области
    LD D, 16               ; 16 регистров
.save_loop:
    OUT (A), A             ; Чтение через MMIO_WINDOW
    IN A, (00h)            ; псевдокод(, зависит от реализации)
    LD (HL), A
    INC HL
    INC A
    DEC D
    JR NZ, .save_loop
    RET
```

14. Архитектурные возможности расширения: двойной доступ к устройствам

14.1. Текущая архитектура доступа

В текущей реализации существует **единый механизм доступа** к периферийным устройствам через MMIO_WINDOW:

`Пользовательская программа → MMIO_WINDOW (00-7Fh) → MMIO_PAGE → Устройство в MMIO_LO`

Этот подход **унифицирован** и работает для всех устройств.

14.2. Теоретическая возможность: двойной доступ

Архитектура **теоретически позволяет** реализовать двойной доступ к определенным устройствам:

1. Через MMIO_WINDOW (стандартный путь)

```
; Стандартный доступ через окно
LD A, page_number
OUT (F1h), A          ; MMIO_PAGE = страница устройства
LD A, value
OUT (reg_offset), A   ; запись через MMIO_WINDOW
```

2. Прямой доступ через фиксированные порты (теоретический)

```
; Теоретический прямой доступ если( реализовано)
LD A, value
OUT (C0h), A          ; прямое обращение к устройству
```

14.3. Как это могло бы работать

Последняя страница MMIO_LO (страница 127) могла бы быть проецирована в **старшие 128 байт портового пространства**:

```
Адресное отображение (теоретическое):
Порты 80h-FFh → Страница 127 MMIO_LO (FF8080h-FFFFFFh)
Исключение: Порты F0h-FFh уже заняты системными регистрами
```

Реализация (теоретическая):

```
// Теоретический двойной доступ
always_comb begin
    if (cpu_io_write && cpu_addr[7] == 1'b1) begin
        // Прямой доступ через порты 80h-FFh
        if (cpu_addr[7:0] >= 8'h80 && cpu_addr[7:0] <= 8'hEF) begin
            // Проецируем в страницу 127 MMIO_LO
            mmio_physical_addr = 24'hFF8000 + {7'd127, cpu_addr[6:0]};
            direct_access = 1'b1;
        end
    end
end
```

14.4. Почему это НЕ реализовано в текущей архитектуре

1. **Нет практической необходимости**:
   - Доступ через MMIO_WINDOW достаточно быстр (1-2 дополнительные инструкции)
   - Упрощает декодирование адресов
   - Унифицированный подход для всех устройств
2. **Ограниченное пространство портов**:
   - Порты 80h-FFh могут быть использованы Legacy-устройствами
   - Риск конфликтов с существующим Legacy-окружением CPC
3. **Архитектурная чистота**:

- Один механизм доступа → меньше ошибок
- Проще для понимания разработчиками
- Легче документировать и отлаживать

4. **Приоритеты разработки**:
   - MMU и системные регистры требуют эксклюзивного CPU-only доступа
   - Периферийные устройства работают хорошо через MMIO_WINDOW
   - Нет драйверов/ПО, требующих такого оптимизированного доступа

14.5. Когда это могло бы быть полезно (теоретически)

1. **Устройства с критической задержкой**:
   - Таймеры реального времени
   - Контроллеры прерываний
   - DMA-контроллер (хотя он управляет шиной)
2. **Часто используемые регистры**:
   - Статусные регистры устройств
   - Буферы быстрого обмена данными
3. **Обратная совместимость**:
   - Если нужно эмулировать устройство с фиксированными портами

14.6. Реализационные детали (для будущих версий)

Если в будущем потребуется реализовать двойной доступ:

```
module dual_access_device (
    input wire clk,
    input wire rst_n,
    // Стандартный MMIO доступ
    input wire [23:0] wb_adr_i,
    input wire [7:0] wb_dat_i,
    input wire wb_we_i,
    input wire wb_stb_i,
    // Прямой CPU доступ теоретический()
    input wire [7:0] cpu_io_addr,
    input wire [7:0] cpu_io_data,
    input wire cpu_io_write,
    // Выбор режима доступа
    input wire direct_access_enabled
);

// Регистры устройства
reg [7:0] device_regs[0:15];

always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        // Сброс
```

```
    end else begin
        // Приоритет: прямой доступ > MMIO доступ
        if (direct_access_enabled && cpu_io_write &&
            cpu_io_addr[7:0] == DEVICE_BASE_PORT) begin
            // Прямой доступ через порт
            device_regs[cpu_io_addr[3:0]] <= cpu_io_data;
        end else if (wb_stb_i && wb_we_i &&
                wb_adr_i[23:16] == 8'hFF &&
                wb_adr_i[15:14] == 2'b00) begin
            // Стандартный MMIO доступ
            device_regs[wb_adr_i[5:2]] <= wb_dat_i;
        end
    end
end
endmodule
```

14.7. Будущее развитие архитектуры

Текущая архитектура **оставляет возможность** для добавления прямого доступа в будущем:

Версия	Доступ к устройствам	Комментарий
Aleste LX v1.0	Только через MMIO_WINDOW	Текущая реализация
Aleste LX v2.0	+ Прямой доступ для CPU-critical	Если потребуется
Aleste LX v3.0	+ Прямой доступ для выбранных устройств	На усмотрение сообщества

Решение о реализации прямого доступа будет приниматься на основе:

- Запросов от разработчиков сообщества
- Анализа производительности реальных приложений
- Обратной совместимости с существующим ПО

14.8. Рекомендации для разработчиков периферии

1. **Всегда проектируйте** для доступа через MMIO_WINDOW
2. **Не рассчитывайте** на прямой доступ через порты
3. **Если критична производительность**:
    - Оптимизируйте алгоритмы доступа
    - Используйте DMA для передачи данных
    - Кэшируйте часто используемые значения
4. **Для совместимости** с Legacy-устройствами:
    - Используйте механизм эмуляции в MMIO_HI
    - Не занимайте потенциальные Legacy-порты

15. Резюме ключевых изменений

1. **Обновленные порты управления**: F0-FF вместо D0-DF
    - `GLOBAL_CTRL`: F0h (был D7h)
    - `MMIO_PAGE`: F1h (был D3h)
    - `SYSCALL`: F2h (был D4h)
    - `CLOCK_CTRL`: F3h (новый)
    - `SUPER_SLOT`: F9h (был D9h)
    - `USER_SLOT`: FBh (был DBh)
    - `BANK_0-3`: FCh-FFh (были DCh-DFh)
2. **Исправленная битовая карта GLOBAL_CTRL**:
    - Бит 0: supervisor_mode (был native_mode)
    - Бит 1: native_mode (был supervisor_mode)
3. **Добавлен TAG-based механизм адресации**: 3-битные TAG для Wishbone
4. **Четкое разделение доступа**:
    - MMU_EXT доступен только CPU
    - CLOCK_CTRL доступен только в Supervisor
5. **Добавлены примеры кода** и таблица состояний сброса
6. **Рекомендации для разработчиков периферии** добавлен раздел о развитии платформы.

Эта обновленная документация сохраняет все оригинальные концепции архитектуры Aleste LX, исправляет несоответствия и добавляет практические примеры для разработчиков.

Понял! Вы предлагаете **интересный компромисс**: теоретическая возможность vs практическая реализация.

Давайте добавим в документацию раздел о **двойном доступе к устройствам** (и почему это скорее теоретическая возможность):
