# Введение

# Архитектура системы команд

# Ассемблер

# Общее описание системы

ПОЧЕМУ ЭТО ЗДЕСЬ, А НЕ В ОПИСАНИИ ПРОЦЕССОРА И ЧТО ЗА КОШМАР С ПУСТЫМИ МЕСТАМИ?

**Интерфейс wishbone для передачи данных**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название | | Назначение | Направление | Разрядность | |
| CLK\_I | | Тактовый сигнал | IN | 1 бит | |
| RST\_I | | Сигнал сброса (инициализации) | IN | 1 бит | |
| ADR\_O | | Выходная шина адреса | OUT | 16 бит | |
| ADR\_I | | Входная шина адреса | IN | 16 бит | |
| DAT\_I | | Выходная шина данных | IN | 16 бит | |
| DAT\_O | | Входная шина данных | OUT | 16 бит | |
| WE\_O | | Запись/чтение | OUT | 1 бит | |
| TGD\_I | Вспомогательный шины для передачи необязательных сигналов(знаки четностии др.) | | IN | | 16 бит |
| TGD\_O | См. TGD\_I | | OUT | | 16 бит |
| ACK\_I | Оповестительный сигнал – ведомое устройство подтверждает упешную обработку данных | | IN | | 1 бит |
| CYC\_O | Оповестительный сигнал – ведущее устройство начало цикл чтения/записи с ведомым | | OUT | | 1 бит |
| STALL\_I | Ведомое устройство не готово принимать данные | | IN | | 1 бит |
| ERR\_I | Сигал ошибки | | IN | | 1 бит |
| LOCK\_O | Блокирующий сигнал – запрещает прерывание | | OUT | | 1 бит |
| RTY\_I | Интерфейс не готов к передаче данных, необходимо пробовать еще раз | | IN | | 1 бит |
| SEL\_O | Говорит, какой байт считывается в шине данных | | OUT | | 2 бита |
| STB\_O | Ведомый интерфейс работает если сигнал = 1 | | OUT | | 1 бит |

**Интерфейс wishbone для передачи инструкций**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CLK\_I | | Тактовый сигнал | IN | 1 бит | |
| RST\_I | | Сигнал сброса (инициализации) | IN | 1 бит | |
| ADR\_O | | Выходная шина адреса | OUT | 16 бит | |
| ADR\_I | | Входная шина адреса | IN | 16 бит | |
| WE\_O | | Запись/чтение | OUT | 1 бит | |
| TGD\_I | Вспомогательные шины для передачи необязательных сигналов | | IN | | 16 бит |
| TGD\_O | См. TGD\_I | | OUT | | 16 бит |
| ACK\_I | Оповестительный сигнал – ведомое устройство подтверждает успешную обработку данных | | IN | | 1 бит |
| CYC\_O | Оповестительный сигнал – ведущее устройство начало цикл чтения/записи с ведомым | | OUT | | 1 бит |
| STALL\_I | Ведомое устройство не готово принимать данные | | IN | | 1 бит |
| ERR\_I | Сигал ошибки | | IN | | 1 бит |
| LOCK\_O | Блокирующий сигнал – запрещает прерывание | | OUT | | 1 бит |
| RTY\_I | Интерфейс не готов к передаче данных, необходимо пробовать еще раз | | IN | | 1 бит |
| SEL\_O | Говорит, какой байт считывается в шине данных | | OUT | | 2 бита |
| STB\_O | Ведомый интерфейс работает если сигнал = 1 | | OUT | | 1 бит |

# Описание блоков ядра процессора

# Описание контроллера доступа в память

**Карта памяти**

. . .

. . .

addr [9:0] (согласно ISA)

ROM

RAM

I/O

**0**0 000000000

abs\_addr [10:0]

abs\_addr [10] – признак принадлежности адреса (1 – кRAM, 0 – к ROMили к I/O).

0**1** 000000000

01 11111111

1 0000000000

1 1111111111

. . .

abs\_addr [9] – признак принадлежности адреса (1 – кI/O,

0 – к ROM). Рассмотрение этого признака не имеет смысла для RAM.

0x000

0x00C

0x200

0x3FF

0x400

Максимум 512 инструкций по 16 бит. Один адрес относится к одному байту в оперативной памяти.

00 000001100

7 инструкций по 16 бит.

0x7FF

**Начальная загрузка**

Процессор начинает работу (PC перекидывается в область RAM) при условии, что в первом регистре, принадлежащим области I/Oзагружены все единицы. Наличие там единиц означает, что все данные по UARTзагружены в ОЗУ.

**Ассемблерный код, записанный в ROM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес PC | | | Ассемблерный код | Описание |
| DEC | HEX | BIN |
| 0 | 00 | 0000 | NOR t0, zero, zero | Заполнение регистра t0 всеми единицами |
| 2 | 02 | 0110 | ADDI t1, zero, 0100 | Присвоение регистру t1 значения «4» длядальнейшего его использования приизменении PC (PC→PC+4 –перескок через строку)а также для перехода по адресу «4» (PC→4) во время ветвления |
| 4 | 04 | 0010 | LDL 0x200\* | Загрузка в регистр MR (L) из памяти по адресу 0x200\* |
| 6 | 06 | 0100 | LDH 0x200\* | Загрузка в регистр MR (H) из памяти по адресу 0x200\* |
| 8 | 08 | 1000 | BNE t1, t0, MR | При невыполнении условия равенства t0 и MR, PC→PC+t1,т.е. +4 |
| 10 | 0A | 1010 | JMP 0x200\* (здесь addr[9:0] = 000000000) | Переход PC поадресу 0x200\* (переход PC к первому адресу RAM)JMP позволяет перемещаться только вRAM или по RAM! (см. комментарий) |
| 12 | 0C | 1100 | JR t1 | Переход PC по адресу, записанному в t1=4 (т.е. в этом цикле переходак RAM не будет. PCостанется в ROM). |

\*Звездочкой отмечены абсолютные адреса (abs\_addr)

Комментарий

*По поводу JMP и JAL. Это единственные инструкции, которые имеют фиксированный девятый бит регистраabs\_addr [9], равный для JMP – 0, а для JAL – 1. Так как вся программа будет находиться в RAM, то с JAL все хорошо. Ветвимся по RAM. С JMP ситуация хуже. Тут Даниэлю придется передавать в логику, связывающую память и процессор abs\_addr с девятым битом, равным единице, а не нулю. В итоге, указанный в ISAaddr, будет являться локальным адресом в RAM при использовании JMP и JAL. Для остальных инструкций типа М адрес будет абсолютным.*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| загрузка  из памяти  в регистр | MR[15:0]MEM[abs\_addr] | LDL abs\_addr | 1 | 0 | 0 | 1 | 0 | abs\_addr | |
| MR[31:16]MEM[abs\_addr] | LDH abs\_addr | 1 | 0 | 0 | 1 | 1 | abs\_addr | |
| сохранение  из регистра  в память | MEM[abs\_addr]MR[15:0] | STL abs\_addr | 1 | 0 | 1 | 0 | 0 | abs\_addr | |
| MEM[abs\_addr]MR[31:16] | STH abs\_addr | 1 | 0 | 1 | 0 | 1 | abs\_addr | |
| безусловный переход  по адресу | PCJumpAddr | JMP loc\_addr | 1 | 0 | 1 | 1 | 0 | 0 | loc\_addr |
| вызов подпрограммы по адресу | LR PC + 2; PCJumpAddr | JAL loc\_addr | 1 | 0 | 1 | 1 | 0 | 1 | loc\_addr |

**Схема организации памяти**



# Описание модулей периферии

Для того, чтобы отличать инструкции от сигналов блоку периферии, введем header. Он будет отправляться первым, и в зависимости от его значения будет либо передана инструкция (значит следующие 16 бит - код инструкции), либо блок периферии будет записывать в регистр какое-то значение (после него сразу идет следующий header). Регистр [3:0]CTRL служит для управления блоком периферии.

**Формат header'a:**

|  |  |
| --- | --- |
| **Кодировка** | **Действие** |
| 00000000 | Передача инструкции (следующие 16 бит) |
| 11111111 | Запись в регистр 1111 - все инструкции загружены |
| 00001111 | Запись в регистр 0011 - разрешение на передачу от платы к ПК |

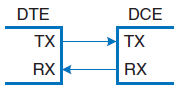
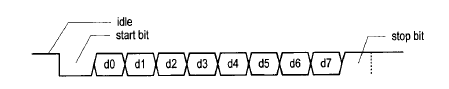
Имя регистра

|  |  |  |  |
| --- | --- | --- | --- |
| Биты | Поле | Режим доступа | Описание |
| 31 | status | RO |  |
| 30:28 | something | RW |  |
|  |  |  |  |

**UART**

UART (Universal Asynchronous Receiver-Transmitter) - универсальный асинхронный приемник-передатчик, интерфейс для последовательной передачи данных.

UART состоит из приемника (rx) и трансмитера (tx). Передача данных в UART осуществляется по одному биту в равные промежутки времени. Этот временной промежуток определяется заданной скоростью UART и для конкретного соединения указывается в бодах (количество бит в секунду в нашем случае, но это верно только для двоичного кодирования). Помимо собственно информационного потока UART автоматически вставляет в поток синхронизирующие метки, так называемые стартовый и стоповый биты. При приёме эти лишние биты удаляются из потока. В нашем дизайне используется 2 стоповых бита (чтоб затормозил наверняка). Также может посылаться бит четности, но у нас он не используется. Idle line имеет высокий уровень, стартовый бит -низкий, а стоповые биты - тоже высокий.

****

До начала передачи приемник и передатчик должны согласоваться о параметрах передачи - бод рейту, количеству битов данных, количеству стоповых битов, использованию бита четности.

Для того, чтобы приемник мог распознать биты в потоке информации, используется схема выборки с запасом. Обычно частота выборки в 16 раз больше бод рейта - каждый бит сэмплируется 16 раз. В середине каждого бита происходит сохранение его значения. Очевидно, и приемник и передатчик работают на сдвиговых регистрах.

# Функциональное тестирование и верификация

# Функциональная модель

# Описание тестового окружения

# Прототипирование