**ШИНЫ**

**Адресная шина (АВ)**

|  |  |
| --- | --- |
| 2 | 12 |
| Номер устройства | Адрес |

Первые два бита отвечают за выбор устройства

0 – ничего

1 – RAM

2 – RCM

3 - СТЕК

**Контрольная шина (СВ)**

CB[7..0]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  | 0 – ничего  1 – остановить WRITE | 0 – ничего  1 – остановить OF and EXEC | 0 – ничего  1 – перезаписать адрес IF на правильный | 0 – ничего  1 – записать в AR с шины данных | 0 – ничего  1 – выставить AR на шину адреса | 0 – чтение  1 - запись |

CB[15..8]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | 11 | 10 | 9 | 8 |
|  |  |  |  | Сменить регистры конвейера | Перезаписать регистры конвейера | Запустить конвейер EXEC & WRITE | Запустить конвейер IF & OF |

**Внутренняя управляющая шина (ICB) и внутренние шины блоков.**

Контролирует направление связи всех внутренних шин с основными. Внутренняя память блока, содержащего микрокоманды имеет ширину 48 бит: ICB[15..0] CB[15..0] AB[15..0].

Младший байт ICB управляет связями всех внутренних шин. ICB[7..0]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Память микрокоманд в шины блока | | Шины блока в общие шины | | | Общие шины в шины блока | | |
| CB | AB | CB | AB | DB | CB | AB | DB |
| 0 – линия разорвана  1 – линия связана в указанном направлении | | | | | | | |

Старший байт ICB отвечает за различные взаимодействия внутри конкретного блока. В частности старший бит ICB[15] обычно используется для защелкивания входных данных (статическое управление).

**RAM**

Чтение происходит за два такта. Первый такт сменяется адрес, второй так – на шину выставляются правильно прочитанные данные.

**СТЕК**

Стек активируется при выставлении старших бит адресной шины (AB[13..12]) в 1.

Микрокомандами для стека служат младшие биты адресной шины:

|  |  |
| --- | --- |
| Младшие биты | Микрокоманда |
| 0 | - |
| 1 | Увеличить указатель стека |
| 2 | Уменьшить указатель стека |
| 3 | Записать данные в стек |
| 4 | Прочитать данные из стека |
| 5 | Прочитать регистр флагов из стека |

**Извлечение инструкций, Instruction Fetch (IF)**

В reg[39..0] ложится полная инструкция. В reg[40] сохраняется текущее предсказание.

Если команда являлась прыжком, то второй операнд перезаписывается блоком на (текущий адрес + 1).

Блок имеет следующую последовательность операций:

|  |  |  |
| --- | --- | --- |
| Номер ICB | Фронт Срез | Значение |
| 8 | / | Сохранить прочитанную инструкцию (первая половина такта) |
| 9 | \ | В зависимости от предсказания сменить текущий адрес |

Необходимо учесть, что память с инструкциями (IROM) имеет динамическое управление. Переключение инструкции происходит по фронту каждого такта.

Также блок имеет возможность заменить текущий адрес по внешнему сигналу.

**Контрольный модуль (СМ)**

Включает в себя бесконечный итератор, который проходит следующие шаги:

|  |  |
| --- | --- |
| Номер шага | Значение |
| 0 | Запустить все стадии конвейера |
| 1 | Сохранить результаты каждой стадии |
| 2 | Сменить промежуточные регистры |

**Промежуточные регистры**

reg[63]: 1 – работаем 0 – пауза

IF -> OF AND EXEC

|  |  |
| --- | --- |
| 63..40 | 39..0 |
| - | Прочитанная инструкция |

OF AND EXEC -> WRITE

|  |  |  |
| --- | --- | --- |
| 55..48 | 47..32 | 31..0 |
| Флаги операции | Результат операции | Операнды команды |

**Архитектура системы команд (АСК)**

Команда вместе с операндами занимает 40 бит

|  |  |  |
| --- | --- | --- |
| 39..32 | 31..16 | 15..0 |
| Команда | Операнд 1 | Операнд 2 |

Биты команды имеют следующий формат

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3..0 |
| Команда является прыжком | 2 операнда | 1 операнд | 0 операндов | Номер команды |

Полный список реализуемых команд

|  |  |  |  |
| --- | --- | --- | --- |
| Обозначение | Битовое представление | HEX представление | DEC представление |
| NOTZ | 00100001 | 21 | 33 |
| AND | 01000010 | 42 | 66 |
| INCS | 00100011 | 23 | 35 |
| ROL | 00100100 | 24 | 36 |
| MOV | 00100101 | 25 | 37 |
| JMP | 10010110 | 96 | 150 |
| PUSH | 00100111 | 27 | 39 |
| POP | 00011000 | 18 | 24 |
| HLT | 00011001 | 19 | 25 |
| JBZ | 10011010 | 9A | 154 |

**Извлечение операндов (OF)**

В блоке извлечения операндов находится два регистра, отвечающие за временное сохранение операндов.

Полная программа блока включает в себя загрузку обоих операндов. Управление количеством загружаемых операндов происходит за счет изменения длины выполняемой микропрограммы (константа длины для внутреннего итератора).

Даже если длина программы равна 0, то на выходе ready имеется мгновенный прыжок (101), который позволяет использовать этот выход для запуска следующего блока выполнения.

Блок полностью полагается на верную кодировку старших четырех бит команды. Если ни один бит, отвечающий за количество аргументов, не выставлен, то блок войдет в бесконечный цикл.

Внутренняя шина блока (ICB[15..8])

|  |  |
| --- | --- |
| Номер бита | Значение |
| 8 | Выставить на шину адреса адрес OP1 |
| 9 | Выставить на шину адреса адрес OP2 |
| 10 | Сохранить OP1 |
| 11 | Сохранить OP2 |

Микропрограмма извлечения операндов

|  |  |  |
| --- | --- | --- |
| ICB | CB | AB |
| Ничего не делаем | | |
| 00000000\_11000000 | 00000000\_00000000 | 00000000\_00000000 |
| Выставляем адрес первого операнда на шину и защелкиваем данные | | |
| 00000001\_10110001 | 00000000\_00000000 | 00000000\_00000000 |
| Читаем первый операнд и сохраняем его | | |
| 00000101\_10110001 | 00000000\_00000000 | 00000000\_00000000 |
| Читаем второй операнд | | |
| 00000010\_10110001 | 00000000\_00000000 | 00000000\_00000000 |
| Читаем второй операнд и сохраняем его | | |
| 00001010\_10110001 | 00000000\_00000000 | 00000000\_00000000 |

**Запись результата (WRITE)**

Внутренняя шина блока (ICB[15..8])

|  |  |
| --- | --- |
| Номер бита | Значение |
| 8 | 0 – выставить значение 1- выставить флаги |
| 9 | 0 – не выставлять 1 – выставить адрес результата |
| 10 | 0 – выставить значение на шину данных 1 – не выставлять |

Микропрограмма

|  |  |  |
| --- | --- | --- |
| ICB | CB | AB |
| Ничего не делаем | | |
| 00000000\_11000000 | 00000000\_00000000 | 00000000\_00000000 |
| Записываем регистр флагов и защелкиваем данные | | |
| 10000001\_11111000 | 00000000\_00000001 | 00100000\_00000000 |
| Записываем данные | | |
| 10000010\_10111000 | 00000000\_00000001 | 00100000\_00000000 |
| 10000010\_10111000 | 00000000\_00000001 | 00100000\_00000000 |

**Регистр флагов (FR)**

Имеет адрес 0 в пространстве регистров.

Значащим является только младший байт. Значения старшего байта не определены.

|  |  |
| --- | --- |
| Бит | Значение |
| 0 | 0 – (res != 0) 1 – (res == 0) |
| 1 | 0 – (res > 0) 1 – (res < 0) |
| 2 | Carry бит |
| 3 | Бит переполнения |
| 4 | Стек пуст ошибка чтения |
| 5 | Стек полон ошибка записи |
| 6 |  |
| 7 |  |

**Конвейер (convey)**

Конвейер имеет внутренний итератор, который отвечает за фазу (внутреннюю итерацию) работы конвейера. Основные фазы:

|  |  |
| --- | --- |
| Фаза | Значение |
| 0 | Запуск блоков IF, OF, EXEC, WRITE. |
| 1 |
| 2 | Сохранение результатов в промежуточных регистрах |
| 3 | Смена промежуточных регистров |

0 и 1 фазы выделены для работы всех блоков т.к. параллельно работают блоки (фаза 0 - IF || OF) а потом (фаза 1 - EXEC || WRITE).

Конвейер работает постоянно. Остановка происходит за счет своевременной установки битов reg[63] в 0.

**Операции**

**AND (DEC 66)**

|  |  |  |
| --- | --- | --- |
| ICB | CB | AB |
| Ничего не делаем | | |
| 00000000\_11000000 | 00000000\_00000000 | 00000000\_00000000 |
| Защелкиваем входные данные | | |
| 10000000\_11000000 | 00000000\_00000001 | 00100000\_00000000 |