**ОСОБЕННОСТИ АРХИТЕКТУРЫ МП ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ (ЦОС)**

**ПРЕДПОСЫЛКИ ПЕРЕХОДА К ЦОС**

* МАССОВЫЙ ПЕРЕХОД ОТ АНАЛОГОВЫХ СИСТЕМ ОБРАБОТКИ СИГНАЛОВ К ЦИФРОВЫМ (С НАЧАЛА 80-х ГОДОВ) – ОБУСЛОВЛЕН УЛУЧШЕНИЕМ ТЕХНИЧЕСКИХ ХАРАКТЕРИСТИК МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ:

1. ВЫСОКОЕ БЫСТРОДЕЙСТВИЕ (МЛН. ОПЕРАЦИЙ/СЕК)
2. РАЗРЯДНОСТЬ (16 БИТ И БОЛЕЕ)
3. БОЛЬШОЙ ОБЪЕМ ПАМЯТИ (64 Кбайт)

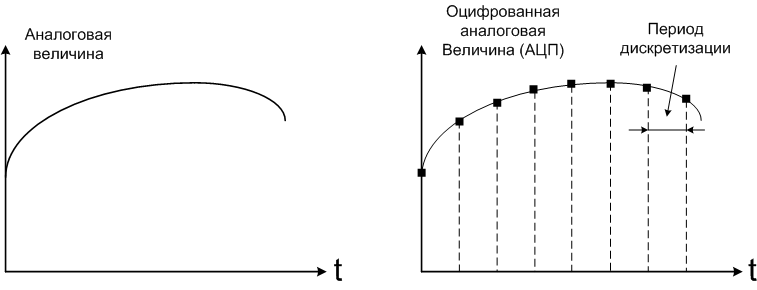
* ДАННЫЕ ХАРАКТЕРИСТИКИ СТАЛИ ОТВЕЧАТЬ СПЕЦИФИЧЕСКИМ ТРЕБОВАНИЯМ ДЛЯ РЕАЛИЗАЦИИ АЛГОРИТМОВ ПОТОКОВОЙ ОБРАБОТКИ ДАННЫХ, ПРЕДСТАВЛЕННЫХ В ЦИФРОВОЙ ФОРМЕ В РЕАЛЬНОМ ВРЕМЕНИ

* ЦИФРОВЫЕ СИСТЕМЫ УСТРАНЯЮТ НЕДОСТАТОК АНАЛОГОВЫХ СИСТЕМ, СВЯЗАННЫЙ С ЗАВИСИМОСТЬЮ ИХ ХАРАКТЕРИСТИК ОТ СТАБИЛЬНОСТИ ПАРАМЕТРОВ СХЕМОТЕХНИЧЕСКИХ ЭЛЕМЕНТОВ (ОПЕРАЦИОННЫЕ УСИЛИТЕЛИ, R, C)

**ИДЕЯ ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ**

* КВАНТОВАНИЕ ПО УРОВНЮ И ДИСКРЕТИЗАЦИЯ ВО ВРЕМЕНИ АНАЛОГОВЫХ СИГНАЛОВ И ИХ ПРЕОБРАЗОВАНИЕ В ЦИФРОВУЮ ФОРМУ
* АЛГОРИТМИЧЕСКАЯ ОБРАБОТКА ЦИФРОВЫХ СИГНАЛОВ С ЗАДАННОЙ ТОЧНОСТЬЮ И БЫСТРОДЕЙСТВИЕМ
* АЛГОРИТМИЧЕСКАЯ БАЗА И ВЫЧИСЛИТЕЛЬНЫЕ ОПЕРАЦИИ НАД ОЦИФРОВАННЫМИ СИГНАЛАМИ УНИФИЦИРОВАНЫ И НЕ ЗАВИСЯТ ОТ ОБЛАСТИ ПРИМЕНЕНИЯ
* ОБРАТНОЕ ПРЕОБРАЗОВАНИЕ ИЗ ЦИФРОВОЙ В АНАЛОГОВУЮ ФОРМУ

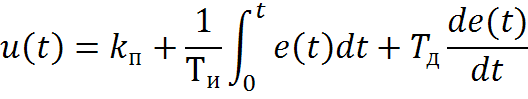
**КВАНТОВАНИЕ И ДИСКРЕТИЗАЦИЯ СИГНАЛОВ**

****

**ОБЛАСТИ ПРИМЕНЕНИЯ**

* РАДИОТЕХНИЧЕСКИЕ СИСТЕМЫ (ФИЛЬТРАЦИЯ, СПЕКТРАЛЬНЫЙ АНАЛИЗ);
* СИСТЕМЫ РАСПОЗНАВАНИЯ ОБРАЗОВ;
* МУЛЬТИМЕДИЙНЫЕ СИСТЕМЫ;
* СИСТЕМЫ УПРАВЛЕНИЯ (РЕГУЛЯТОРЫ)

**ТИПОВЫЕ ЗАДАЧИ ЦОС (регуляторы)**

* Уравнение ПИД-регулятора: 

u(t) - управляющий сигнал,

е(t) - рассогласование между заданным и фактическим значением регулируемой величины

Эквивалентное конечно-разностное уравнение:

u(k) = u(k-1)+K0e(k)-K1e(k-1)+K2e(k-2)

Ki – постоянные коэффициенты

e(k) – дискретные отсчеты (выборки)

k – номер отсчета

**ТИПОВЫЕ ЗАДАЧИ ЦОС (цифровая фильтрация)**

* ЦЕЛЬ ФИЛЬТРАЦИИ – ИЗМЕНЕНИЕ СТРУКТУРЫ СИГНАЛА (УДАЛЕНИЕ ПОМЕХ, ВЫДЕЛЕНИЕ ПОЛЕЗНОГО СИГНАЛА)
* ДИСКРЕТНАЯ МОДЕЛЬ ФИЛЬТРА n-го ПОРЯДКА:

**u(k)+a1u(k-1)+a2u(k-2)+….+ anu(k-n)=**

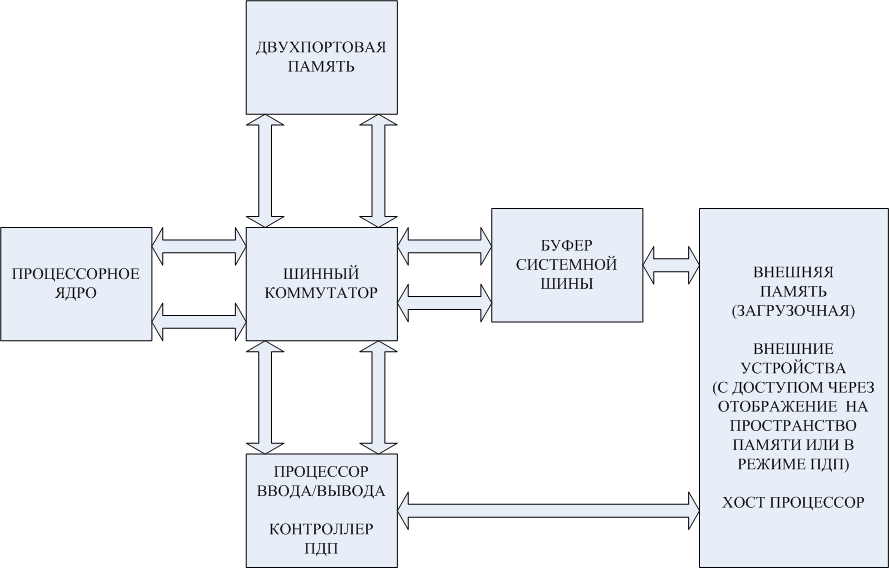
**b0e(k)+b1e(k-1)+b2e(k-2)+…+bne(k-n)**

* e(k) – отсчеты входного сигнала, u(k) – отсчеты выходного сигнала фильтра
* ВЫЧИСЛЕНИЯ u(k) ДОЛЖНЫ ВЫПОЛНЯТЬСЯ КАЖДЫЙ РАЗ С ПРИХОДОМ ОЧЕРЕДНОГО ОТСЧЕТА ВХОДНОГО СИГНАЛА e(k)
* ЕСЛИ ВСЕ КОНСТАНТЫ ai РАВНЫ НУЛЮ, ТО ФИЛЬТР ЯВЛЯЕТСЯ НЕРЕКУРСИВНЫМ

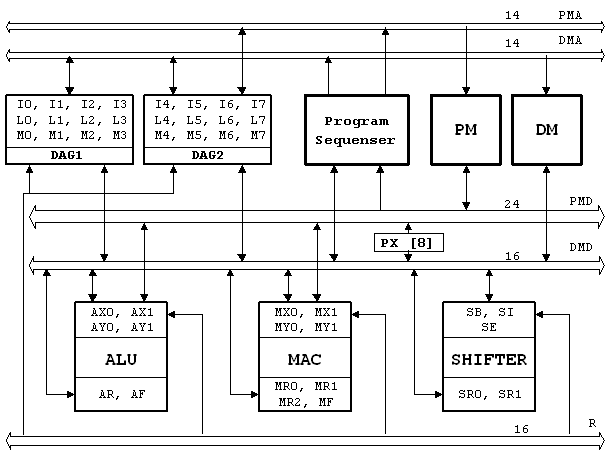
**ОТЛИЧИТЕЛЬНЫЕ ЧЕРТЫ АРХИТЕКТУРЫ МП ЦОС**

1. СУПЕРГАРВАРДСКАЯ АРХИТЕКТУРА (SHARC)
2. КОНВЕЙЕР КОМАНД
3. АППАРАТНОЕ УМНОЖЕНИЕ
4. УМНОЖЕНИЕ С НАКОПЛЕНИЕМ
5. РЕЖИМ НАСЫЩЕНИЯ ПРИ ПЕРЕПОЛНЕНИИ
6. АППАРАТНАЯ ПОДДЕРЖКА ЦИКЛИЧЕСКИХ ВЫЧИСЛЕНИЙ (ПРОГРАММНЫЕ ЦИКЛЫ, ЦИКЛИЧЕСКИЙ БУФЕР ДАННЫХ)
7. ИНВЕРСИЯ БИТОВ АДРЕСА (ДЛЯ РЕАЛИЗАЦИИ АЛГОРИТМОВ БПФ)

**СИСТЕМА НА КРИСТАЛЛЕ С СУПЕРГАРВАРДСКОЙ АРХИТЕКТУРОЙ (SHARC)**

****

**СТРУКТУРНАЯ СХЕМА ЯДРА (ПРОТОТИП SHARC)**

****

DAG - ГЕНЕРАТОРЫ АДРЕСА ОПЕРАНДА

1. ГЛАВНЫЙ АТРИБУТ SHARC – НАЛИЧИЕ 4-х НЕМУЛЬТИПЛЕКСИРУЕМЫХ ШИН.
2. ЗА ОДИН ПРОЦЕССОРНЫЙ ЦИКЛ ДОПУСКАЕТСЯ

ДВА ОБРАЩЕНИЯ К ПАМЯТИ ПРОГРАММ И ОДНОКРАТНОЕ ОБРАЩЕНИЕ К ПАМЯТИ ДАННЫХ

3. ЭФФЕКТИВНАЯ РАБОТА С МАССИВАМИ КОНСТАНТ (ПАМЯТЬ ПРОГРАММ) И МАССИВАМИ ОТСЧЕТОВ СИГНАЛОВ (ПАМЯТЬ ДАННЫХ)

**ВЫЧИСЛИТЕЛЬНЫЕ УСТРОЙСТВА МП ЦОС**

* ALU – арифметико-логическое устройство выполняет логические и арифметические операции
* MAC – умножитель-аккумулятор выполняет умножение, а также умножение со сложением и вычитанием (накоплением) за один процессорный цикл
* SHIFTER – устройство сдвига производит логические и арифметические сдвиги, а также операции над числами с плавающей запятой
* Обмен данными между вычислительными устройствами ALU, MAC и SHIFTER осуществляется по шинам DMD, PMD и R

**ОРГАНИЗАЦИЯ ПАМЯТИ МП ЦОС**

* PM - память программ предназначена для хранения 16Кб (214) 24-битовых слов (ДЛИНА ЛЮБОЙ КОМАНДЫ)
* 14-ти разрядный адрес на шину PMA выставляют входящий в состав Program sequencer счетчик команд, (ИЗВЛЕЧЕНИЕ КОМАНДЫ), ИЛИ DAG2 (СЧИТЫВАНИЕ ДАННЫХ)
* на шину данных памяти программ PMD считываются соответственно команды или данные (особенность супергарвардской архитектуры)
* DM - память данных предназначена для хранения 16КБ (214) 16-битовых слов

* 14-ти разрядные адреса для обращения к DM формируют DAG1 или DAG2

**УСТРОЙСТВА УПРАВЛЕНИЯ (Program Sequencer)**

* определяет порядок выполнения команд
* генерирует управляющие сигналы, необходимые для выполнения команд

* обеспечивает аппаратную поддержку циклических вычислений, условных переходов и управления прерываниями

**УСТРОЙСТВА УПРАВЛЕНИЯ (DAG1 и DAG2 )**

* Выполняют функцию вычисления исполнительных адресов операндов
* DAG1 и DAG2 являются независимыми генераторами адресов и обеспечивают одновременное обращение к памяти программ и данных
* Генераторы различаются по функциональным возможностям:

DAG1 генерирует адреса только для памяти данных «DM», а DAG2 для памяти данных и программ «PM»

* Функцию инверсии бит адреса поддерживают оба генератора (С НЕЗАВИСИМОЙ НАСТРОЙКОЙ НА ЭТОТ РЕЖИМ)

**ОСОБЕННОСТИ ОРГАНИЗАЦИИ DAG1/2**

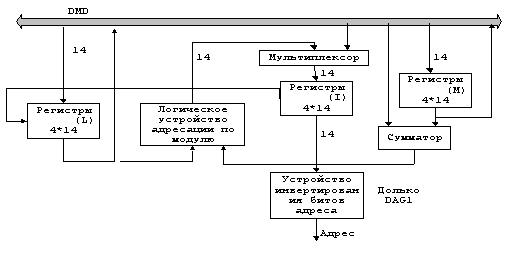
* позволяют аппаратно реализовать процедуру модификации исполнительного адреса по модулю, что обеспечивает возможность организации циклических буферов

* ОБЕСПЕЧИВАЮТ ГИБКОСТЬ НАСТРОЙКИ И АППАРАТНУЮ ПОДДЕРЖКУ РЕАЛИЗАЦИИ РЕЖИМОВ АДРЕСАЦИИ
* Пример НАСТРОЙКИ РЕЖИМОВ АДРЕСАЦИИ:

- DAG1 генерирует адреса памяти данных «DM» С реверсОМ бит адреса (перестановка бит в обратном порядке)

- DAG2 генерирует адреса для памяти программы PM и для памяти данных DM без реверса бит адреса

**СТРУКТУРА DAG – ГЕНЕРАТОРА АДРЕСА**

****

РЕГИСТРОВАЯ МОДЕЛЬ DAG:

1. I - индексные регистры указатели (I0-3 для DAG 1 и I4-7 для DAG2)

2. M - регистры пост-модификаторы

(M0-3 для DAG1 и M4-7 для DAG2)

3. L - регистры длины буфера

(L0-3 для DAG1 и L4-7 для DAG2)

DAG поддерживают адресацию данных структурированного типа,

организованных как линейные (L=0) или циклические массивы.

Регистры I и L содержат беззнаковые величины, а регистр модификатор M

– знаковую, что позволяет как инкрементировать, так и декрементировать

содержимое регистра-указателя при косвенной адресации

**ОСОБЕННОСТИ РЕГИСТРОВОЙ МОДЕЛИ DAG**

* DAG РЕАЛИЗУЮТ ОБРАЩЕНИЕ К ПАМЯТИ ПРОГРАММ И ДАННЫХ С АВТОМАТИЧЕСКОЙ ПОСТ-МОДИФИКАЦИЕЙ ИНДЕКСНЫХ РЕГИСТРОВ УКАЗАТЕЛЕЙ
* ПРИ ЗАДАНИИ РЕЖИМА ПОСТ-МОДИФИКАЦИИ РАЗРЕШАЕТСЯ ИСПОЛЬЗОВАТЬ ЛЮБОЕ ПАРОСОЧЕТАНИЕ РЕГИСТРОВ «I» И «М» ИЗ ОДНОГО И ТОГО ЖЕ DAG
* НОМЕР РЕГИСТРА ДЛИНЫ БУФЕРА L ОДНОЗНАЧНО ОПРЕДЕЛЯЕТ НОМЕР СООТВЕТСТВУЮЩЕГО ЕМУ ИНДЕКСНОГО РЕГИСТРА УКАЗАТЕЛЯ

**ОСОБЕННОСТИ СИСТЕМЫ КОМАНД**

* СИСТЕМА КОМАНД ОБЕСПЕЧИВАЕТ УПРАВЛЕНИЕ ВСЕМИ ВЫЧИСЛИТЕЛЬНЫМИ УСТРОЙСТВАМИ (АЛУ, УМНОЖИТЕЛЬ, сдвигатель)
* ДЛИНА ВСЕХ КОМАНД ОДИНАКОВА (24 бита)
* КАЖДАЯ КОМАНДА ВЫПОЛНЯЕТСЯ ЗА ОДИН ПРОЦЕССОРНЫЙ ЦИКЛ
* ВЫЧИСЛИТЕЛЬНЫЕ КОМАНДЫ ОБРАБАТЫВАЮТ 16-ти РАЗРЯДНЫЕ ОПЕРАНДЫ (ВОЗМОЖНА обработка С ПОВЫШЕННОЙ ТОЧНОСТЬЮ)
* ДВА РЕЖИМА АДРЕСАЦИИ – ПРЯМАЯ И КОСВЕННАЯ
* РАСШИРЕННОЕ КОМАНДНОЕ СЛОВО ПОЗВОЛЯЕТ ЗА ОДИН ПРОЦЕССОРНЫЙ ЦИКЛ ВЫПОЛНЯТЬ МНОГОФУНКЦИОНАЛЬНЫЕ ИНСТРУКЦИИ (ПАРНЫЕ СОЧЕТАНИЯ ИЗ ГРУПП ВЫЧИСЛИТЕЛЬНЫХ ИНСТРУКЦИЙ, ПЕРЕСЫЛОК И ОБРАЩЕНИЙ К ПАМЯТИ ПРОГРАММ И ДАННЫХ)

**КОМАНДЫ ПЕРЕСЫЛКИ**

|  |  |
| --- | --- |
| **№ п/п** | **Команда** |
| **1** | **Пересылка данных между регистрами** |
| **2** | **Загрузка непосредственного значения в регистр** |
| **3** | **Считывание из памяти данных (прямая адресация)** |
| **4** | **Считывание из памяти данных (косвенная адресация)** |
| **5** | **Считывание из памяти программ (косвенная адресация)** |
| **6** | **Запись в память данных (прямая адресация)** |
| **7** | **Запись в память данных (косвенная адресация)** |
| **8** | **Запись в память программ (косвенная адресация)** |

**МНОГОФУНКЦИОНАЛЬНЫЕ ИНСТРУКЦИИ**

* В МП ЦОС имеется возможность “уплотнения” процессорных циклов (т.е. за один процессорный цикл выполняется несколько команд - многофункциональных инструкций)
* ” Уплотнение” обеспечивается следующими особенностями архитектуры МП ЦОС:

1. Раздельные память программ и данных
2. Два независимых генератора адресов данных DAG1 и DAG2
3. Наличие общей шины «R» позволяет использовать результат выполнения текущей команды в качестве операнда следующей команды
4. Совмещение различных фаз обработки нескольких команд (конвейер команд)

**ПРИМЕРЫ КОМАНД (АЛГЕБРАИЧЕСКАЯ ЗАПИСЬ)**

* КОМАНДЫ ПЕРЕСЫЛКИ:

1. DM(I0,M1)=AY0; -- В ПАМЯТЬ ДАННЫХ ПО АДРЕСУ, КОТОРЫЙ НАХОДИТСЯ В РЕГИСТРЕ I0 ЗАПИСЫВАЕТСЯ ЗНАЧЕНИЕ ИЗ РЕГИСТРА АЛУ AY0. ПОСЛЕ ОБРАЩЕНИЯ К ПАМЯТИ I0= I0+M1 (КОСВЕННАЯ АДРЕСАЦИЯ С ПОСТ-МОДИФИКАЦИЕЙ ИНДЕКСНОГО РЕГИСТРА)
2. PM(I4,M5)=AX0;
3. AR=AX0+AY0;

* МНОГОФУНКЦИОНАЛЬНАЯ ИНСТРУКЦИЯ:

AX0=DM(I0,M0), AY0=PM(I4,M4); -- В ОДНОМ ПРОЦЕССОРНОМ ЦИКЛЕ В РЕГИСТРЫ АЛУ ОДНОВРЕМЕННО В КАЧЕСТВЕ ОПЕРАНДОВ СЧИТЫВАЮТСЯ ПО ОЧЕРЕДНОМУ ЭЛЕМЕНТУ МАССИВОВ, КОТОРЫЕ ХРАНЯТСЯ В ПАМЯТИ ПРОГРАММ И ПАМЯТИ ДАННЫХ

**ФОРМАТЫ ПРЕДСТАВЛЕНИЯ ДАННЫХ**

|  |  |  |  |
| --- | --- | --- | --- |
| **Формат 1.15** | **Десятичный эквивалент** | **Формат 16.0**  **(беззнаковый/знаковый)** | **Десятичный эквивалент** |
| 0х0001 | 0.000031 | 0х0001 | 00001 |
| 0х7FFF | 0.999969 | 0х7FFF | 32767 |
| 0xFFFF | -0.000031 | 0xFFFF | -00001 |
| 0x8000 | -1.000000 | 0x8000 | -32768 |

Формат 16.0 - формат представления 16-ти разрядных целых беззнаковых чисел или целых чисел со знаком в дополнительном коде.

Формат 1.15 - это формат представления дробных

16-ти разрядных чисел в дополнительном коде.

Старший разряд – знак. 15 разрядов – дробная часть числа.

**НЕОБХОДИМОСТЬ ИСПОЛЬЗОВАНИЯ ЦИКЛИЧЕСКИХ БУФЕРОВ ДАННЫХ**

* РАССМОТРИМ ВЫРАЖЕНИЕ ДЛЯ РЕАЛИЗАЦИИ НЕРЕКУРСИВНОГО ФИЛЬТРА:

u(k) = b0e(k)+b1e(k-1)+b2e(k-2)+…+bne(k-n)

* ПРИ ВЫЧИСЛЕНИИ u(k) покомпонентно перемножаются и затем складываются элементы массивов e(k-i) и bi. Содержимое массива bi не изменяется, а содержимое массива e(k) необходимо обновлять ПРИ ПОЛУЧЕНИИ очередного отсчета e(k)
* При вычислении очередного u(k) из массива bi  всегда считывается последовательность значений b0, b1, b2,...bi,…bn. Поэтому перед каждым новым вычислением u(k) указатель должен переводиться на начало массива (коэффициент b0). Для автоматизации этого процесса буфер для хранения массива коэффициентов bi целесообразно сделать кольцевым

**ДИНАМИЧЕСКИ ОБНОВЛЯЕМЫЙ МАССИВ ОТСЧЕТОВ ВХОДНОГО СИГНАЛА e(k)**

* ПРИ ИСПОЛЬЗОВАНИИ линейной буферизации ОБНОВЛЯЕМОГО МАССИВА ВХОДНЫХ ОТСЧЕТОВ необходимо осуществлять переупорядочивание его элементов (n=5): e(4),e(3),e(2),e(1),e(0) → e(5),e(4),e(3),e(2),e(1) → …...
* ПРИ ИСПОЛЬЗОВАНИИ линейной буферизации НА ПЕРЕУПОРЯДОЧИВАНИЕ МАССИВА e(k) ЗАТРАЧИВАЕТСЯ СУЩЕСТВЕННОЕ ВРЕМЯ, ЧТО ЯВЛЯЕТСЯ КРИТИЧНЫМ МОМЕНТОМ ДЛЯ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ
* ВЫВОД: ДЛЯ МАССИВА e(k) ТАКЖЕ НЕОБХОДИМ ЦИКЛИЧЕСКИЙ БУФЕР

**РЕАЛИЗАЦИЯ ЦИКЛИЧЕСКОГО БУФЕРА ПРИ ПОМОЩИ DAG**

Ls- определяет длину буфера;

Is- хранит текущий (исполнительный)адрес для обращения к буферу;

Ms-модификатор (постоянная);

s - индекс группы регистров DAG, (s=0-3 для DAG1, s=4-7 для DAG2) (ПРИМЕМ ДОПУЩЕНИЕ ОБ ИДЕНТИЧНОСТИ s ДЛЯ «I» И «М»)

Исполнительный адрес элемента в кольцевом буфере (циклическом массиве) формируется по правилу:

А след.(Is)=(Is+Ms-B)mod Ls+B,

где В-базовый адрес массива

**ПРИМЕР ЦИКЛИЧЕСКОГО БУФЕРА**

|  |  |  |
| --- | --- | --- |
| **Адрес** | **Динамически изменяемое содержимое массива входных отсчетов** | **I0** |
| 0x3800 | X(0) | (3800+1-3800)mod3+3800=3801 |
| 0x3801 | X(1) | (3801+1-3800)mod3+3800=3802 |
| 0x3802 | X(2) | (3802+1-3800)mod3+3800=3800 |
| 0x3800 | X(3) | (3800+1-3800)mod3+3800=3801 |
| 0x3801 | X(4) | (3801+1-3800)mod3+3800=3802 |
| 0x3802 | X(5) | (3802+1-3800)mod3+3800=3800 |
| 0x3800 | X(6) | (3800+1-3800)mod3+3800=3801 |

**B=0x3800, I0=0x3800,M0=1, L0=3**

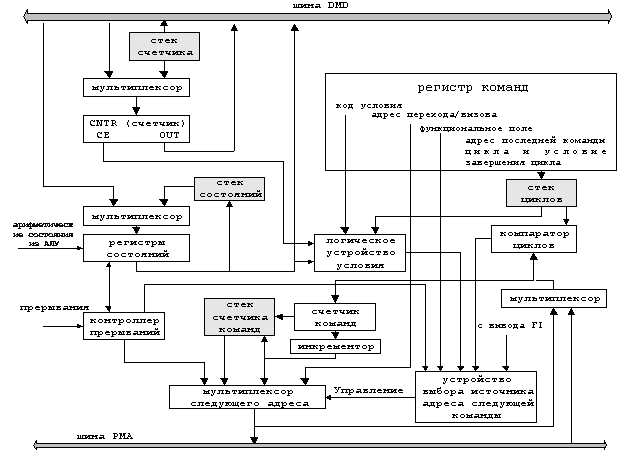
**СОСТОЯНИЯ БУФЕРА ПРИ ВВОДЕ ДАННЫХ**

|  |  |  |  |
| --- | --- | --- | --- |
| **Значение** | **Текущий адрес буфера (I0)** | **Формирование следующего адреса для обращения к буферу Аслед.=(I0+M0-B)mod L0+B,** | **Комментарий (команды чтения/записи в массив)** |
| е(0) | 0x3800 | A=(0x3800+1-0x3800)mod 3+0x3800=0x3801 | MX0=DM(I0,M0)/чтение е(0)/ |
| е(1) | 0x3801 | A=(0x3801+1-0x3800)mod 3+0x3800=0x3802 | MX0=DM(I0,M0) /чтение е(1)/ |
| е(2) | 0x3802 | A=(0x3802+1-0x3800)mod 3+0x3800=**0x3800** | MX0=DM(I0,M0) /чтение е(2)/ |
| Вычисление u(2) :u(2)=e(0)·b(2) + e(1)·b(1) + e(2)·b(0) | | | |
| x(3) | **0x3800** | A=(0x3800+1-0x3800)mod 3+0x3800=0x3801 | DM(I0,M0)=AR /запись e(3)/ |
| x(1) | 0x3801 | A=(0x3801+1-0x3800)mod 3+0x3800=0x3802 | MX0=DM(I0,M0) /чтение e(1)/ |
| x(2) | 0x3802 | A=(0x3802+1-0x3800)mod 3+0x3800=0x3800 | MX0=DM(I0,M0) /чтение e(2)/ |
| x(3) | 0x3800 | A=(0x3800+1-0x3800)mod 3+0x3800=**0x3801** | MX0=DM(I0,M0) /чтение e(3)/ |
| Вычисление u(3) : u(2)=e(1)·b(2) + e(2)·b(1) + e(3)·b(0) | | | |
| x(4) | **0x3801** | A=(0x3801+1-0x3800)mod 3+0x3800=0x3802 | DM(I0,M0)=AR /запись e(4)/ |
| x(2) | 0x3802 | A=(0x3802+1-0x3800)mod 3+0x3800=0x3800 | MX0=DM(I0,M0) /чтение e(2)/ |
| x(3) | 0x3800 | A=(0x3800+1-0x3800)mod 3+0x3800=0x3801 | MX0=DM(I0,M0) /чтение e(3)/ |
| x(4) | 0x3801 | A=(0x3801+1-0x3800)mod 3+0x3800=0x3802 | MX0=DM(I0,M0) /чтение e(4)/ |
| Вычисление u(4) : u(3)=e(2)·b(2) + e(3)·b(1) + e(4)·b(0) | | | |

ВЫЧИСЛЕНИЯ В ДАННОЙ ТАБЛИЦЕ ВЫПОЛНЯЮТСЯ В ОБРАТНОМ ПОРЯДКЕ:

u(2)=e(2)·b(0) + e(1)·b(1) + e(0)·b(2); →u(2)=e(0)·b(2) + e(1)·b(1) + e(2)·b(0);

**СТРУКТУРА БЛОКА ФОРМИРОВАНИЯ АДРЕСА СЛЕДУЮЩЕЙ КОМАНДЫ**

****ОСОБЕННОСТЬ СТРУКТУРЫ БЛОКА – АППАРАТНАЯ ПОДДЕРЖКА ЦИКЛИЧЕСКИХ ВЫЧИСЛЕНИЙ:

1. АППАРАТНЫЕ СТЕКИ:

* СЧЕТЧИКА
* ЦИКЛОВ
* СЧЕТЧИКА КОМАНД
* СОСТОЯНИЙ

2. КОМПАРАТОР ЦИКЛОВ

**СЧЕТЧИК КОМАНД И СТЕК СЧЕТЧИКА КОМАНД**

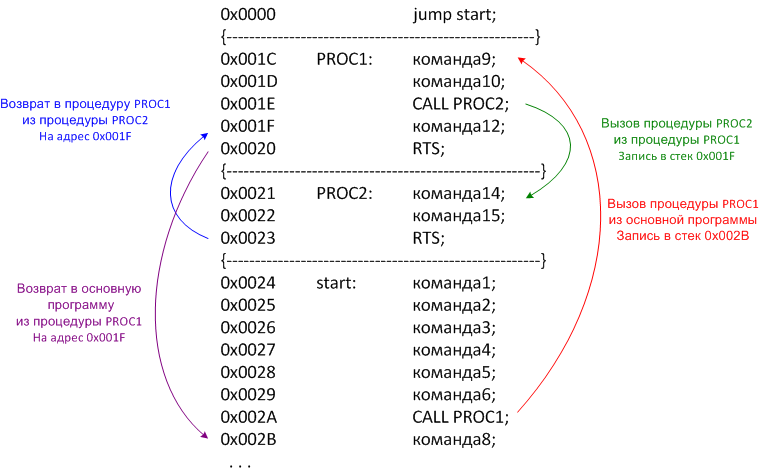
* Счетчик команд (PC) - регистр, в котором содержится адрес текущей команды. «РС» увеличивается на «1» инкрементором для формирования адреса следующей команды в случае последовательной выборки команд - длина команд одинакова
* АппаратнЫЙ стек счетчика команд (PC Stack) используется при:

1. Обращении к подпрограммам (процедуре) по команде CALL;

1. Обработке прерываний;
2. Организации циклов DO UNTIL для хранения адреса первой команды цикла;

* Глубина стека – не более 16 адресов

**ВЛОЖЕННЫЕ ПРОЦЕДУРЫ**

****

В данном примере из фрагмента программы, начинающегося с адреса 0х0024, осуществляется вызов процедуры PROC1,

которая в свою очередь вызывает процедуру PROC2.

**ТРАССИРОВКА ВЛОЖЕННЫХ ПРОЦЕДУР**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Адрес**  **команды** | **Команда** | **Адрес следующий команды**  **Счетчик команд (PC)** | **Содержимое стека счетчика**  **команд (PC Stack)** | |
| **0х0000** | **jump start** | **0x0024** | **1**  **2** | **0xXXXX**  **0xXXXX** |
| **0х0024** | **start: команда1** | **0х0025** | **1**  **2** | **0xXXXX**  **0xXXXX** |
| **0х0025** | **Команда2** | **0х0026** | **1**  **2** | **0xXXXX**  **0xXXXX** |
| **0х0029** | **Команда6** | **0х002А** | **1**  **2** | **0xXXXX**  **0xXXXX** |
| **0х002А** | **CALL PROC1** | **0x001C** | **1**  **2** | **0х002В**  **0xXXXX** |
| **0х001С** | **Команда9** | **0х001D** | **1**  **2** | **0х002В**  **0xXXXX** |
| **0х001D** | **Команда10** | **0х001Е** | **1**  **2** | **0х002В**  **0xXXXX** |
| **0х001Е** | **CALL PROC2** | **0x0021** | **1**  **2** | **0x002B**  **0x001F** |
| **0х0021** | **Команда14** | **0х0022** | **1**  **2** | **0x002B**  **0x001F** |
| **0х0022** | **Команда15** | **0х0023** | **1**  **2** | **0x002B**  **0x001F** |
| **0х0023** | **ВОЗВРАТ ИЗ ПРОЦЕДУРЫ (PROC2) (RTS)** | **0x001F** | **1**  **2** | **0x002B**  **0xXXXX** |
| **0х001F** | **Команда12** | **0х0020** | **1**  **2** | **0x002B**  **0xXXXX** |
| **0х0020** | **ВОЗВРАТ ИЗ ПРОЦЕДУРЫ PROC1 (RTS)** | **0x002B** | **1**  **2** | **0xXXXX**  **0xXXXX** |
| **0х002В** | **Команда8** | **. . .** | **1**  **2** | **0xXXXX**  **0xXXXX** |

**АППАРАТНАЯ ПОДДЕРЖКА ЦИКЛИЧЕСКИХ ВЫЧИСЛЕНИЙ**

При организации цикла DO <loop> UNTIL <условие> МП ЦОС использует следующие аппаратные ресурсы:

* Счетчик циклов (CNTR)
* Аппаратные Стеки:

1. стек счетчика (Counter Stack)
2. стек циклов (Loop Stack)
3. стек счетчика команд (PC Stack)

* Компаратор циклов (Loop Comparator)

**Счетчик циклов (CNTR)**

* инструкция DO UNTIL обеспечивает циклическое исполнение требуемой последовательности инструкций
* Счетчик циклов (CNTR) с пост-декрементированием служит для аппаратной поддержки выполнения инструкции DO <loop> UNTIL (выполнять цикл, пока не выполняется условие)

* CNTR осуществляет подсчет числа итераций цикла. В качестве условия завершения цикла используется условие CE (счетчик пуст)
* CNTR проверяется в начале каждого цикла и декрементируется после выполнения последней команды цикла

**стек счетчика (Counter Stack)**

* Стек счетчика циклов позволяет организовать вложенные циклы за счет временного сохранения в стеке счетчика номеров итераций вложенных циклов
* При загрузке нового значения в счетчик циклов прежнее значение CNTR автоматически помещается в стек счетчика

* Каждый раз при выполнении условия CE (счетчик пуст) значение из стека счетчика автоматически извлекается и загружается в Счетчик циклов (CNTR)

* При перезагрузке счетчика циклов - возврат к выполнению внешнего цикла (если такой есть). Максимальная глубина стека счетчика циклов равна 4

**ПРОГРАММА С ВЛОЖЕННЫМИ ЦИКЛАМИ**

**0х001С команда1;**

**{-----------внешний цикл-------------}**

**0х001D CNTR=4;**

**0x001E DO cycle1 UNTIL CE;**

**0x001F команда4;**

**0x0020 команда5;**

**{------------внутренний цикл--------}**

**0x0021 CNTR=2;**

**0x0022 DO cycle2 UNTIL CE;**

**0x0023 команда8;**

**0x0024 cycle2: команда9;**

**{--------------------------------------------}**

**0x0025 команда10;**

**0x0026 cycle1: команда11;**

**{--------------------------------------------}**

**0x0027 команда12;**

В данном примере показана организация двух вложенных циклов DO UNTIL.

Внешний цикл cycle1 выполняется 4 раза при логическом условии CE (счетчик пуст),

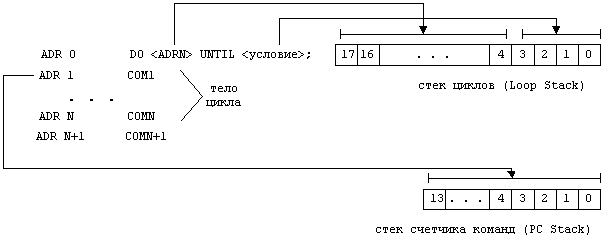
а внутренний цикл cycle2 выполняется 2 раза.

**ТРАССИРОВКА ПРОГРАММЫ**

|  |  |  |
| --- | --- | --- |
| **КОМАНДА** | **Счетчик циклов**  **(CNTR)** | **Стек счетчика циклов**  **(Counter Stack)** |
| Команда1 | 0Ххххх | 0хХХХХ |
| CNTR=4 (внешний цикл) | 0x0004 | 0хХХХХ |
| DO cycle1 UNTIL CE | 0x0004 | 0хХХХХ |
| Команда4 | 0x0004 | 0хХХХХ |
| Команда5 | 0x0004 | 0хХХХХ |
| CNTR=2 (внутренний цикл) | 0x0002 | 0x0004 |
| DO cycle2 UNTIL CE | 0x0002 | 0x0004 |
| Команда8 | 0x0002 | 0x0004 |
| cycle2: команда9 | 0x0001 (декремент счетчика внутр. цикла) | 0x0004 |
| Команда8 | 0x0001 | 0x0004 |
| cycle2: команда9 | 0x0004 – из стека счетчика | 0хХХХХ |
| Команда10 | 0x0004 | 0хХХХХ |
| cycle1: команда11 | 0x0003 (декремент счетчика внешн. цикла) | 0хХХХХ |

Комментарий: 1-ый раз цикл выполняется при значении CNTR=N, а последний раз при CNTR=1 (фактически проверяется значение (CNTR-1)=0? ) После выполнения последней команды цикла при CNTR=1 CNTR не декрементируется, а в него передается содержимое стека

**стек циклов (Loop Stack)**

****

1. В стеке счетчика команд (PC Stack) содержится адрес первой команды

Тела цикла DO UNTIL (adr1).

2. В стеке циклов (Loop Stack) содержится адрес последней команды (adr n)

тела цикла и код условия завершения .

**КОМПАРАТОР ЦИКЛОВ (Loop comparator)**

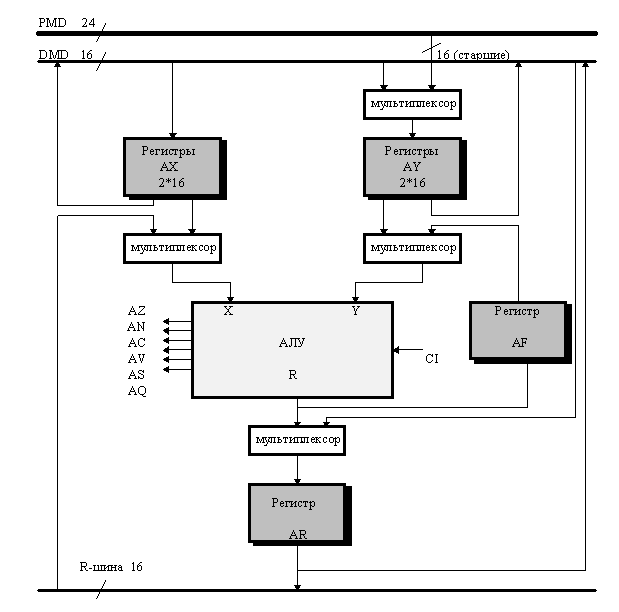
* В каждом процессорном цикле компаратор (Loop Comparator) сравнивает адресную часть содержимого стека циклов (Loop Stack), в котором содержится адрес последней команды цикла (ADR N) и адрес следующей команды (задача – фиксация последней команды цикла)
* Если эти два адреса совпали (ADR N), то в следующем процессорном цикле выполняется последняя команда тела цикла DO UNTIL, т.е. команда COMN по адресу ADR N

* После выполнения последней команды цикла DO UNTIL с условием «CE» происходит декремент счетчика CNTR и вычисляется условие «се»
* В зависимости от значения «CE», генератор адресов инструкций (PS) определяет адрес следующей за командой COMN команды:

ADR 1, если CE=0 – загрузка счетчика команд из стека счетчика команд (PC Stack) для повторения цикла

ADR N+1, если СЕ=1 – инкремент счетчика команд и выход из цикла

**CТРУКТУРА АЛУ**

****

1. АЛУ обеспечивает выполнение логических и

арифметических операций за один процессорный цикл

2. АЛУ имеет 16-ти разрядные порты - два для ввода операндов - X и Y, и один для вывода результата

3. АЛУ принимает бит переноса «CI» (флаг в регистре состояния вычислительных устройств). Бит переноса «CI» позволяет обрабатывать в АЛУ числа, разрядность которых больше, чем 16.

4. АЛУ вырабатывает сигналы состояния: AZ, AN, AC, AV, AS, AQ.

5. Два банка регистров АЛУ – сохранение контекста при обработке прерываний и вызове подпрограмм

6. Регистр af – регистр обратной связи алу. Позволяет использовать результат алу в качестве операнда «y»

**ОПЕРАЦИИ АЛУ**

|  |  |
| --- | --- |
| **ОПЕРАЦИЯ** | **ВЫПОЛНЯЕМЫЕ ДЕЙСТВИЯ** |
| R=X+Y | Сложение операндов X и Y |
| R=X+Y+CI | Сложение операндов X и Y и бита переноса |
| R=X-Y | Вычитание операнда Y из X |
| R=X-Y+CI-1 | Вычитание операнда Y из X с заемом |
| R=Y-X | Вычитание операнда X из Y |
| R=Y-X+CI-1 | Вычитание операнда X из Y с заемом |
| R=-X | Инвертирование операнда X (в дополнительном коде) |
| R=-Y | Инвертирование операнда Y (в дополнительном коде) |
| R=Y+1 | Инкремент операнда Y |
| R=Y-1 | Декремент операнда Y |
| R=PASS X | Передача операнда X на выход R |
| R=PASS Y | Передача операнда Y на выход R |
| R=0 или R= PASS 0 | Обнуление результата |
| R=ABS X | Абсолютное значение операнда X |
| R=X AND Y | Логическое И (&) над X и Y |
| R=X OR Y | Логическое ИЛИ над X и Y |
| R= X XOR Y | Исключающее ИЛИ над X и Y |
| R=NOT X | Логическое НЕ над операндом X |
| R=NOT Y | Логическое НЕ над операндом Y |

**ФЛАГИ АЛУ (СИГНАЛЫ СОСТОЯНИЯ)**

|  |  |
| --- | --- |
| **Флаг** | **Описание** |
| AZ | Ноль (нулевой результат) |
| AN | Знак результата |
| AV | Переполнение (дополнительный код) |
| AC | Перенос (перенос из старшего разряда сумматора) |
| AS | Знак (для порта Х АЛУ) |
| AQ | Бит частного (устанавливается примитивами деления) |

**УСЛОВНЫЕ ИНСТРУКЦИИ АЛУ**

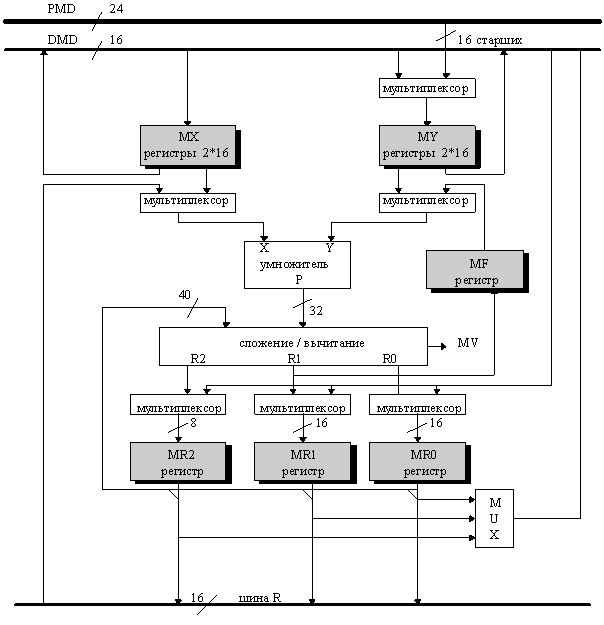
* Формат инструкций АЛУ имеет не обязательное поле условия

* Если данное поле используется, то инструкция выполняется при истинности содержащегося в нем условия (условное выполнение инструкции)
* Если поле условия отсутствует, то инструкция выполняется безусловно (безусловное выполнение инструкции)
* ПРИМЕР: IF GE AR=AX0-AY0;

**УСЛОВИЯ, ИСПОЛЬЗУЕМЫЕ В КОМАНДАХ**

|  |  |  |
| --- | --- | --- |
| **Разрешенные условия** | | |
| **Синтаксис** | **Условие** | **Верно, когда** |
| EQ | Равен 0 | AZ=1 |
| NE | Не равен 0 | AZ=0 |
| LT | Меньше 0 | AN xor AV=1 |
| GE | Больше или равен 0 | AN xor AV=0 |
| LE | Меньше либо равен 0 | (AN xor AV) or AZ=1 |
| GT | Больше 0 | (AN xor AV) or AZ=0 |
| AC | Перенос АЛУ | AC=1 |
| NOT AC | Нет переноса в АЛУ | AC=0 |
| AV | Переполнение АЛУ | AV=1 |
| NOT AV | Нет переполнения АЛУ | AV=0 |
| MV | Переполнение умножителя | MV=1 |
| NOT MV | Нет переполнения умножителя | MV=0 |
| NEG | X- отрицателен | AS=1 |
| POS | X –положителен | AS=0 |
| FOREVER | Бесконечный цикл | - |
| CE | Число повторений истекло | - |

**СТРУКТУРА УМНОЖИТЕЛЯ МП ЦОС (МАС)**

****

МАС имеет два 16-ти разрядных входных порта X и Y и 32-х разрядный порт вывода Результата

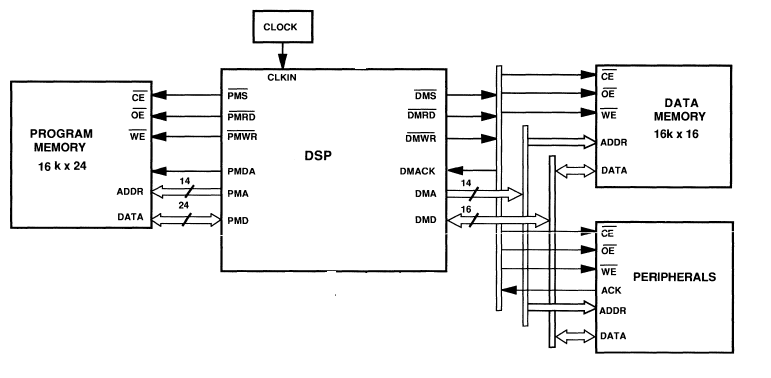
2. порт вывода результата подключен к 40-разрядному сумматору – вычитателю, который прибавляет или вычитает результат УМНОЖЕНИЯ из содержимого 40-битного регистра умножителя MR (УМНОЖЕНИЕ С НАКОПЛЕНИЕМ)

3. Сумматор - вычитатель имеет разрядность 40 бит для предотвращения переполнения В КОМАНДЕ умножения с накоплением

4. СИГНАЛ СОСТОЯНИЯ «MV» ВЫРАБАТЫВАЕТСЯ ПРИ ВЫХОДЕ РЕЗУЛЬТАТА СУММАТОРА ЗА 32-х РАЗРЯДНУЮ СЕТКУ

5. «МАС» имеет два банка регистров.

**Пример БАЗОВой СТРУКТУРы МПС ЦОС**

****

ПАМЯТЬ ДАННЫХ И ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА ИМЕЮТ РАЗДЕЛЯЕМОЕ АДРЕСНОЕ ПРОСТРАНСТВО. ВХОД DMACK ПРЕДНАЗНАЧЕН ДЛЯ «УДЛИНЕНИЯ» ЦИКЛОВ ОБМЕНА С МЕДЛЕННЫМИ ПУ.