Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра электронных вычислительных средств

В. Б. Клюс, М. В. Качинский, А.Б. Давыдов

ПРОГРАММИРОВАНИЕ ПРОБЛЕМНО-ОРИЕНТИРОВАННЫХ ВЫЧИСЛИТЕЛЬНЫХ СРЕДСТВ РЕАЛЬНОГО ВРЕМЕНИ

Лабораторный практикум

для студентов специальности I-40 02 02
«Электронные вычислительные средства»
дневной формы обучения

В 2-х частях

Часть 1

Минск 2008

УДК 681.3
ББК 32.973
К 52

Рецензент
и.о. зав. кафедрой радиоэлектронных средств БГУИР,
доцент А.М. Ткачук

Клюс, В. Б.

К 52 Программирование проблемно-ориентированных вычислительных средств реального времени: лабораторный практикум для студ. спец. I-40 02 02 «Электронные вычислительные средства» днев. формы обуч. В 2 ч. Ч. 1 / В. Б. Клюс, М. В. Качинский, А. Б. Давыдов. – Минск : БГУИР, 2008. − 34 с. : ил.

ISBN 978-985-488-287-1 (ч. 1)

Данный лабораторный практикум содержит описание четырех лабораторных работ, посвященных различным алгоритмам формирования синусоидальных сигналов и цифровой фильтрации. Приведены примеры реализации некоторых алгоритмов и использования цифровых процессоров сигналов для формирования и анализа различных сигналов. Лабораторные работы ориентированы на использование отладочного модуля на базе процессора TMS320VC5402, выполняются на ПЭВМ.

УДК 681.3
ББК 32.973

ISBN 978-985-488-287-1 (ч. 1)
ISBN 978-985-488-286-4

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА №1
ПРОГРАММИРОВАНИЕ СИНУСОИДАЛЬНЫХ СИГНАЛОВ МЕТОДОМ СУММЫ ДВУХ УГЛОВ

1.1. ЦЕЛЬ РАБОТЫ

Знакомство с различными алгоритмами формирования синусоидальных сигналов, разработка и отладка программы генератора синусоидальных сигналов на основе рекуррентно-аналитического метода суммы двух углов на ассемблере процессора TMS320VC5402 для лабораторного макета TMS320VC5402 DSP Starter Kit (DSK).

1.2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.2.1. Формулы суммы двух углов

Существует несколько аналитических (вычислительных) методов формирования синусоидальных сигналов. Один из таких методов основан на формулах суммы двух углов:

$$\begin{aligned} \sin(\alpha+\beta) &= \sin(\alpha)\cdot\cos(\beta)+\cos(\alpha)\cdot\sin(\beta), \\ \cos(\alpha+\beta) &= \cos(\alpha)\cdot\cos(\beta)-\sin(\alpha)\cdot\sin(\beta). \end{aligned} \qquad (1.1)$$

Этот метод применим для последовательной генерации отсчетов одной синусоиды и позволяет вычислить очередной отсчет функции, например **sin(an)=sin(a(n-1)+a)** через предыдущий **sin(a(n-1))**.

Использовав **a(n-1)** в качестве **α** и **a** в качестве **β** и обозначив **S1=sin(a)**, **C1=cos(a)**, **Sn(n)=sin(an)**, **Cn(n)=cos(an)**, получим следующие выражения для вычисления очередных значений синуса и косинуса:

$$\begin{aligned} Sn(n) &= C1\cdot Sn(n-1)+S1\cdot Cn(n-1), \\ Cn(n) &= C1\cdot Cn(n-1)-S1\cdot Sn(n-1). \end{aligned} \qquad (1.2)$$

Исходными данными для вычисления первой гармоники (с минимальной возможной частотой) являются следующие значения: **Sn(0)=0**, **Cn(0)=1** – значения синуса и косинуса для нулевого аргумента; **N** – длина выборки (число отсчетов в первой гармонике); **a=2П/N** – минимальное приращение аргумента и **S1=sin(2П/N)**, **C1=cos(2П/N)** – значения синуса и косинуса для минимального аргумента.

На рис. 1.1 показан пример синусоиды с периодом 8 отсчетов (N=8).

Учитывая, что исходными значениями для **k**-й гармоники являются начальный угол **β=ak**, **Sk=sin(ak)** и **Ck=cos(ak)**, для их вычисления можно использовать те же самые формулы:

$$\begin{aligned} Sk(k) &= C1\cdot Sk(k-1)+S1\cdot Ck(k-1), \\ Ck(k) &= C1\cdot Ck(k-1)-S1\cdot Sk(k-1). \end{aligned} \qquad (1.3)$$

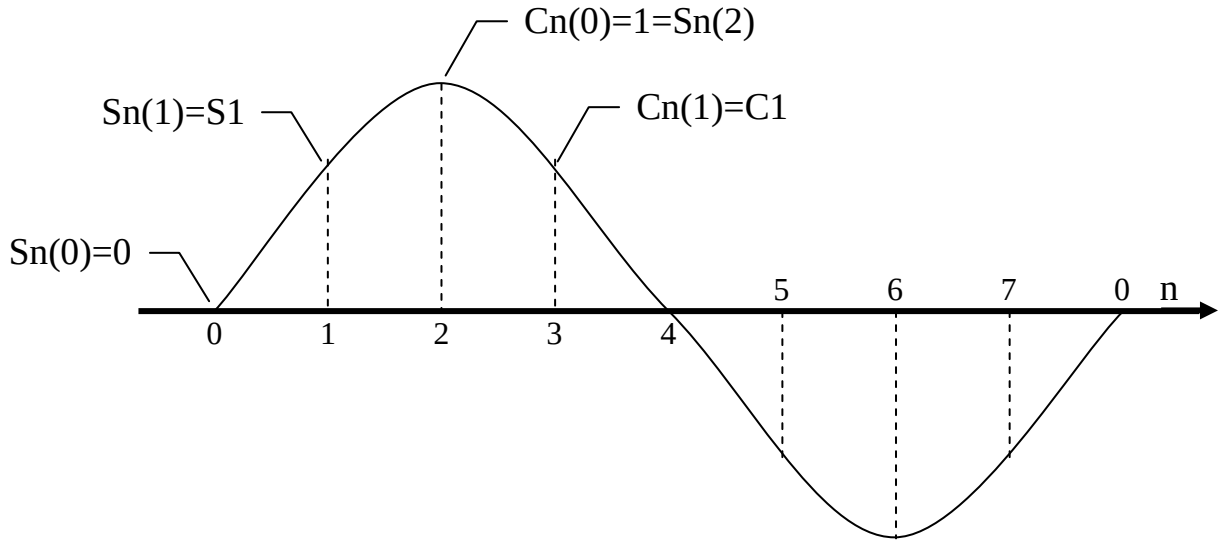

Рис. 1.1. Синусоида с периодом 8 отсчетов

Исходными данными для вычисления, как и в выражении (1.2), являются значения **Sk(0)=0** и **Ck(0)=1**.

Расположив данные в памяти следующим образом: **Cn**, **Sn**, **Ck**, **Sk**, **C1**, **S1**, можно использовать одну и ту же подпрограмму с косвенной адресацией для вычисления как отсчетов очередной гармоники **Sn** и **Cn**, так и начальных значений **Sk** и **Ck**.

Для чего вначале устанавливаем указатель (адрес во вспомогательном регистре) на **Ck** и вычисляем очередное значение **Ck** и **Sk**, а затем на **Cn** и вычисляем нужное число отсчетов очередной гармоники **Sn** и **Cn**:

$$\begin{aligned} Sn(n) &= Ck \cdot Sn(n-1) + Sk \cdot Cn(n-1), \\ Cn(n) &= Ck \cdot Cn(n-1) - Sk \cdot Sn(n-1). \end{aligned} \qquad (1.4)$$

1.2.2. Замечания по программированию

При программировании синусоиды данным методом необходимо учесть следующие замечания:

- используя два вспомогательных регистра и двухоперандные команды (Xmem, Ymem), вычисление синуса (косинуса) можно выполнить тремя командами;
- поскольку ассемблер TMS320VC5402 не имеет директив для задания дробных чисел, то дробные числа (формат Q15) переводятся в целые путем умножения на 32768 и используется директива .word;
- поскольку не существует кода для положительной единицы, то для задания **cos(0)** используется приближенное значение 7FFFh = $1-2^{-15}$;
- для того чтобы формируемая синусоида (косинусоида) не затухала, начальное значение **C1=cos(2П/N)** увеличивается на одну или две единицы младшего разряда и вычисления выполняются в старшей части аккумулятора при установленном режиме коррекции переполнения.

1.2.3. Алгоритм формирования последовательности синусоид

Для исследования цифровых фильтров требуется генератор последовательности синусоид с возрастающей частотой. Схема алгоритма формирования последовательности синусоид на рис. 1.2.

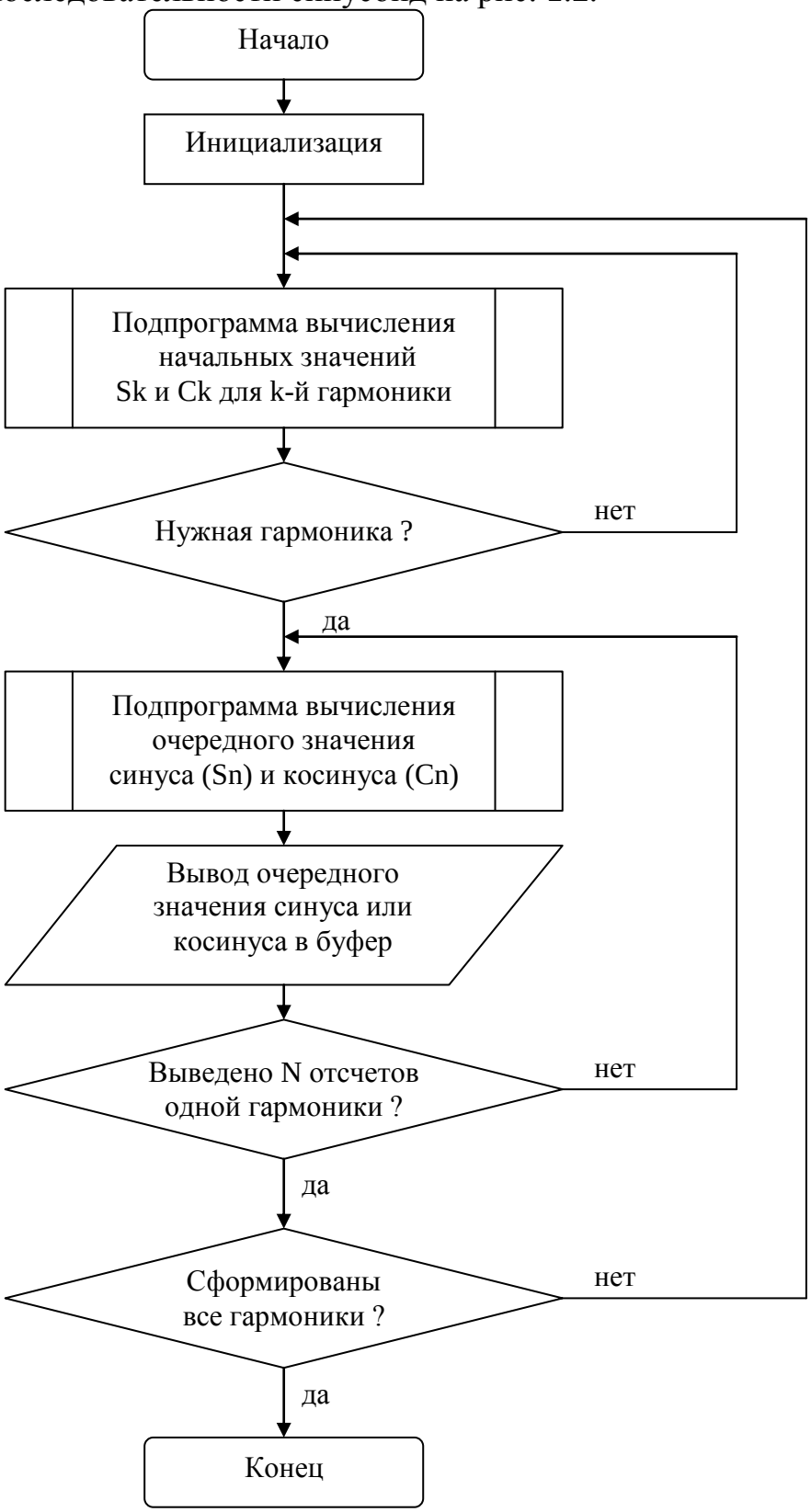

Рис. 1.2. Схема алгоритма формирования последовательности синусоид

1.3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические сведения по теме лабораторной работы по подразд. 1.2.
2. Получить у преподавателя задание для выполнения практической части работы.
3. Согласно заданию написать, оттранслировать и выполнить программу.
4. Продемонстрировать результат трансляции и работы программы преподавателю.
5. Оформить и защитить отчет по лабораторной работе.

1.4. СОДЕРЖАНИЕ ОТЧЕТА

1. Цель работы.
2. Описание алгоритма работы программы.
3. Листинг программы с комментариями.
4. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №2
ПРОГРАММИРОВАНИЕ СИНУСОИДАЛЬНЫХ СИГНАЛОВ МЕТОДОМ ДВОЙНОГО УГЛА

2.1. ЦЕЛЬ РАБОТЫ

Знакомство с различными алгоритмами формирования синусоидальных сигналов, разработка и отладка программы генератора синусоидальных сигналов на основе рекуррентно-аналитического метода двойного угла на ассемблере процессора TMS320VC5402 для лабораторного макета TMS320VC5402 DSP Starter Kit (DSK).

2.2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.2.1. Формулы двойного угла

Другой аналитический метод основан на формулах двойного угла:

$$\sin(2\alpha) = 2\cdot \sin(\alpha)\cdot\cos(\alpha),$$
$$\cos(2\alpha) = \cos^2(\alpha)-\sin^2(\alpha)=1-2\cdot \sin^2(\alpha)=2\cdot \cos^2(\alpha)-1. \quad (2.1)$$

Для вычисления косинуса целесообразно использовать второе выражение, т.к. первое требует возведения в квадрат двух значений, а третье – дает не точный результат из-за того, что косинус малого угла близок к единице.

Данный метод позволяет вычислить синус и косинус любого угла, но требует больше затрат для вычисления последовательных отсчетов синуса и косинуса.

Суть метода заключается в вычислении очередного значения аргумента синуса (косинуса) **a(n)=a(n-1)+s(k)**, где **s(k)** – шаг приращения аргумента (угла) для **k**-й гармоники. При этом данный шаг вычисляется по аналогичной формуле **s(k)=s(k-1)+a**, где **a=2П/N** – приращение аргумента первой гармоники.

Для вычисления функции аргумент **a(n)** делится на два **m** раз до тех пор, пока не будет выполняться с заданной точностью соотношение $\sin(\alpha)\approx\alpha$, где $\alpha=a/2^m$.

После чего, применив **m** раз вышеприведенные формулы (2.1) и использовав в качестве первого приближения $\sin(\alpha) = \alpha$ и $\cos^2(\alpha) = 1 - \sin^2(\alpha) = 1 - \alpha^2$, получим значения синуса и косинуса угла **a(n)**.

Например, для угла **0.0625** рад (**0.0625×32768=2048=800h**) точное значение синуса равно **0.0624593** (**2046.7≈7FFh**), а косинуса – **0.9980475** (**32704≈7FC0h**); для угла **0.03125** рад (**1024=400h**) значение синуса равно **0.0312449** (**1023.8≈400h**), а косинуса – **0.9995117** (**32752≈7FF0h**), и для угла **0.015625** рад (**512=200h**) значение синуса равно **0.0156244** (**511.9≈200h**), а косинуса – **0.9998779** (**32764≈7FFCh**).

Поскольку точные значения косинуса малого угла отличаются от единицы, то для повышения точности расчета следовало бы для его первого приближения находить значение из выражения $\mathbf{\cos^2(\alpha) = 1 - \sin^2(\alpha)}$. Но так как вычисление квадратного корня трудоемкая операция, то с достаточно высокой точностью можно получить значение косинуса по формуле $\mathbf{\cos(\alpha) = 1 - \alpha^2/2}$, что для малых углов дает очень хорошее приближение. По этой формуле для трех приведенных выше углов получим следующие значения: **0.9980469** (**32704≈7FC0h**), **0.9995117** (**32752≈7FF0h**) и **0.9998779** (**32764≈7FFCh**). Из приведенных выше расчетов видно, что с точностью ½ младшего разряда мы получим результат при углах, меньших **0.03125** рад или **1,8°** (**1024=400h**).

Для нахождения **m** и для деления на $\mathbf{2^m}$ можно использовать специализированные команды цифрового процессора сигналов (ЦПС) **EXP** и **NORM**. Так, например, если порог равен **400h**, а аргумент равен **0.09473** рад (**0C20h**), то его необходимо разделить на **2** два раза (**m=2**), чтобы получить значение **308h**, меньшее порога. Исходное число после загрузки в старшую часть 40-разрядного аккумулятор содержит слева **3** незначащих двоичных нуля (**000C200000h**), а максимальное число, меньшее порога, – **5** (**0003FF0000h**), т.к. незначащие нули считаются без 8-ми защитных разрядов. Выполнив команду **EXP**, мы получим в регистре **T** значение **3**, и, вычтя его из **5,** мы получим **2**, что равно **m**, для организации цикла итерационных вычислений по формуле (2.1) нужно **m-1** (т.е. **4-T**), которое загружается во вспомогательный регистр. А для деления на $\mathbf{2^m}$ нужно число **-m**, которое получается вычитанием из регистра **T** значения **5**, и, использовав команду **NORM,** получим в старшей части аккумулятора значение **0C20h**, деленное на $\mathbf{2^m}$, т.е. **308h.** Если в результате выполнения команды **EXP** в регистре **T** получается значение **≥5**, то значит аргумент меньше порога и итерации выполнять не нужно, а в качестве синуса берется $\mathbf{\sin(\alpha) = \alpha}$.

2.2.2. Замечания по программированию

При программировании синусоиды данным методом необходимо учесть следующие замечания:

- поскольку аргумент и шаг могут быть >1 (максимальное значение аргумента 6.28=2П), то для исключения переполнения необходимо промасштабировать все переменные (разделить на 8);
- значение **m-1=7-T** (у порога 8 нулей), а для команды **NORM** из регистра **T** нужно вычитать по прежнему **5**, а не **8**, чтобы восстановить аргумент (/$\mathbf{2^m}$ и ***8)**;
- вычисления выполняются без коррекции переполнения (увеличение аргумента), а для коррекции переполнения (при записи) используется бит **SST** в **PMST**;
- необходима проверка для малых углов (меньше порога), иначе получится отрицательное число переполнений.

2.2.3. Алгоритм формирования последовательности синусоид

Схема алгоритма формирования последовательности синусоид с возрастающей частотой методом двойного угла представлена на рис. 2.1 и отличается от схемы алгоритма из первой лабораторной работы только формированием начального значения аргумента для k-й гармоники.

Рис. 2.1. Схема алгоритма формирования последовательности синусоид

2.3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические сведения по теме лабораторной работы по подразд. 2.2.
2. Получить у преподавателя задание для выполнения практической части работы.
3. Согласно заданию написать и отладить программу.
4. Продемонстрировать результат отладки и работы программы преподавателю.
5. Оформить и защитить отчет по лабораторной работе.

2.4. СОДЕРЖАНИЕ ОТЧЕТА

1. Цель работы.
2. Описание алгоритма работы программы.
3. Текст программы с комментариями.
4. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №3
ПРОГРАММИРОВАНИЕ ЦИФРОВОГО РЕКУРСИВНОГО ФИЛЬТРА

3.1. ЦЕЛЬ РАБОТЫ

Изучение алгоритмов программирования цифровых фильтров на процессоре цифровой обработки сигналов, разработка и отладка программы цифрового рекурсивного фильтра на базе лабораторного макета TMS320VC5402 DSP Starter Kit (DSK).

3.2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

3.2.1. Цифровые рекурсивные и нерекурсивные фильтры

Как известно, уравнение цифрового фильтра (ЦФ) во временной области описывается выражением (прямая форма 1):

$$y(n) = \sum_{k=0}^{N-1} a_k \cdot x(n-k) - \sum_{k=1}^{M-1} b_k \cdot y(n-k). \qquad (3.1)$$

Если все коэффициенты b_k равны нулю, мы получаем уравнение цифрового нерекурсивного фильтра (ЦНФ), или фильтра с конечной импульсной характеристикой (КИХ или FIR – Finite Impulse Response):

$$y(n) = \sum_{k=0}^{N-1} h(k) \cdot x(n-k), \qquad (3.2)$$

где коэффициенты a_k из предыдущей формулы обозначены как h(k) (коэффициенты импульсной характеристики фильтра).

Такой фильтр N-го порядка может быть представлен следующей схемой (рис. 3.1):

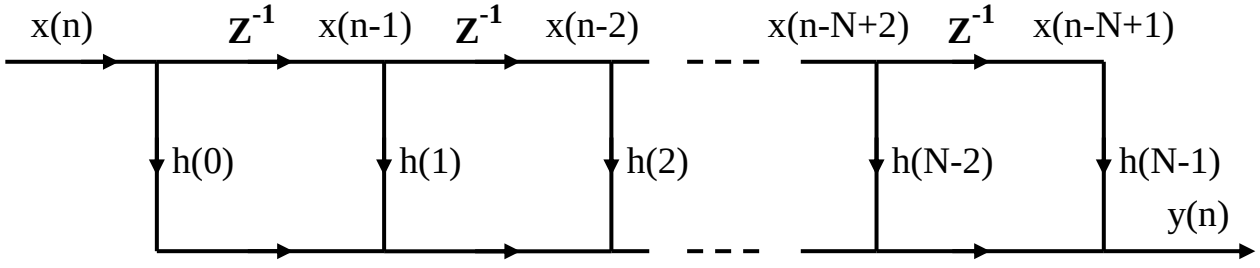

Рис. 3.1. Граф-схема цифрового нерекурсивного фильтра

Передаточная характеристика этого фильтра описывается уравнением

$$H(z) = \frac{y(z)}{x(z)} = \sum_{k=0}^{N-1} a_k \cdot z^{-k} = \sum_{k=0}^{N-1} h(k) \cdot z^{-k}. \qquad (3.3)$$

Передаточная функция цифрового рекурсивного фильтра (ЦРФ) или фильтра с бесконечной импульсной характеристикой (БИХ или IIR – Infinite Impulse Response) в Z-области записывается уравнением

$$H(z) = \frac{y(z)}{x(z)} = \frac{\sum_{k=0}^{N-1} a_k \cdot z^{-k}}{1 + \sum_{k=1}^{M-1} b_k \cdot z^{-k}}. \qquad (3.4)$$

Кроме прямой формы (3.1) имеется и другая форма (называемая канонической) уравнения ЦРФ с использованием промежуточных переменных d:

$$\begin{aligned} d(n) &= x(n) - \sum_{k=1}^{M-1} b_k \cdot d(n-k), \\ y(n) &= \sum_{k=0}^{N-1} a_k \cdot d(n-k). \end{aligned} \qquad (3.5)$$

Обе эти формы (3.1) и (3.5) для случая N = M могут быть представлены следующими схемами (рис. 3.2):

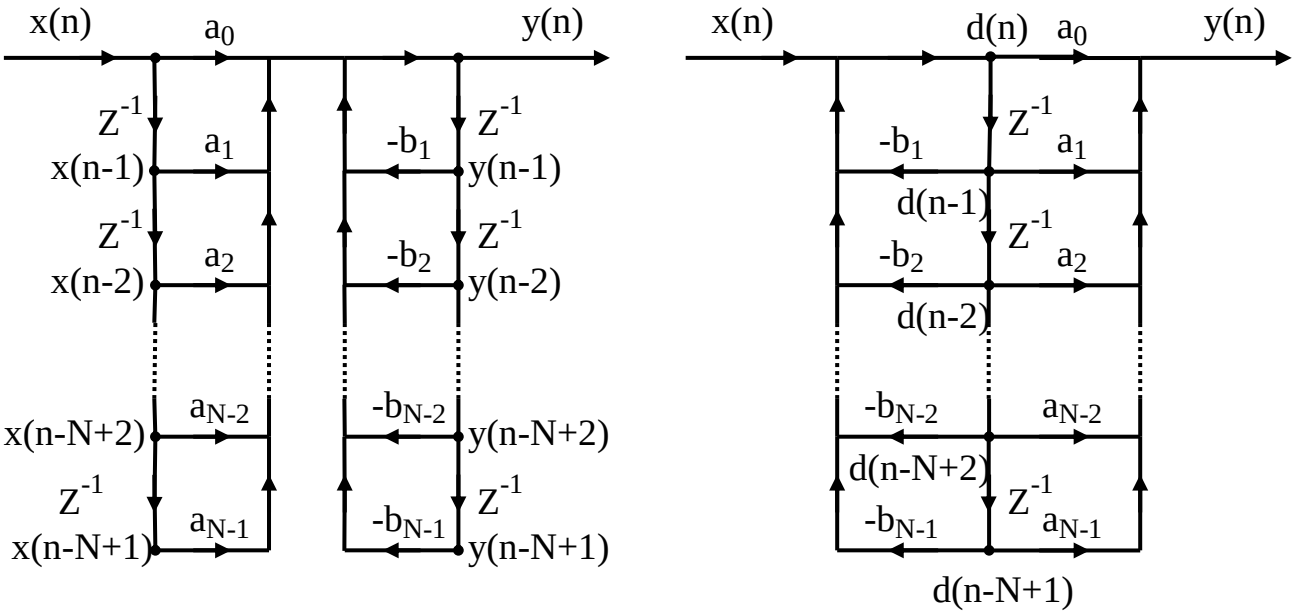

Рис. 3.2. Граф-схемы цифровых рекурсивных фильтров

Прямая форма требует 2N+1 коэффициентов и 2N+1 рабочих ячеек для переменных X, Y (переменные x(n), y(n) могут храниться в одной ячейке памяти). При канонической реализации число переменных в два раза меньше (N+2).

При реализации ЦФ на ЦПС важным является оптимальное распределение в памяти процессора констант и данных.

3.2.2. Каскадное соединение цифровых рекурсивных фильтров

Сложный фильтр высокого порядка (рис. 3.3) может быть реализован с помощью комбинации простых полосовых или режекторных фильтров (обычно 2-го порядка). При этом режекторные фильтры соединяются последовательно и каждый следующий вычитает из сигнала свою составляющую в заданной спектральной полосе (рис. 3.4).

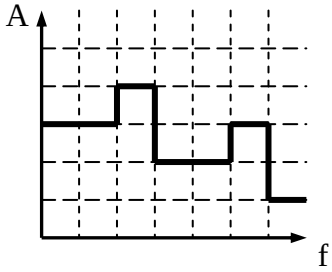

Рис. 3.3. Передаточная характеристика цифрового фильтра

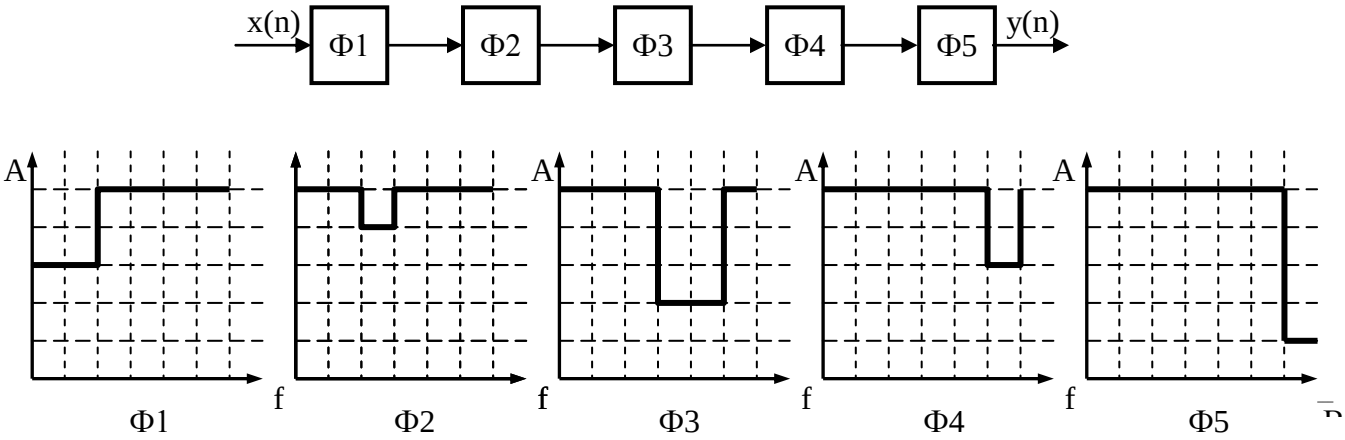

Рис. 3.4. Последовательное соединение фильтров

Передаточная функция ЦРФ при последовательном соединении описывается уравнением

$$H(z) = \prod_{k=1}^{(M-1)/2} \frac{\alpha_{0k} + \alpha_{1k} \cdot z^{-1} + \alpha_{2k} \cdot z^{-2}}{1 + \beta_{1k} \cdot z^{-1} + \beta_{2k} \cdot z^{-2}}. \qquad (3.6)$$

Уравнение одного фильтра во временной области в канонической форме записывается как

$$\begin{aligned} d_k(n) &= y_{k-1}(n) - \beta_{1k}\, d_k(n-1) - \beta_{2k}\, d_k(n-2)\,, \\ y_k(n) &= \alpha_{0k}\, d_k(n) + \alpha_{1k}\, d_k(n-1) + \alpha_{2k}\, d_k(n-2)\,, \end{aligned} \qquad (3.7)$$

где k изменяется от 1 до (M-1)/2. Входное значение $y_0(n) = x(n)$, выходное значение $y_{(M-1)/2}(n) = y(n)$, а структура такого фильтра для двух каскадов представлена в следующем виде (рис. 3.5).

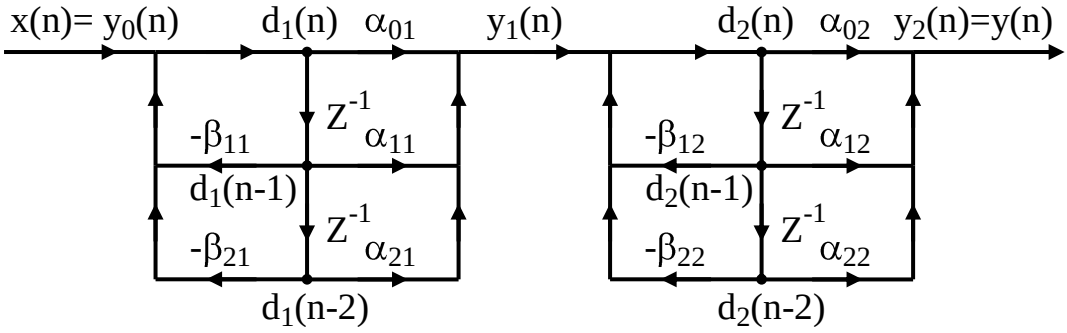

Рис. 3.5. Последовательное соединение цифровых рекурсивных фильтров

Для прямой формы реализации структура этого же фильтра выглядит следующим образом (рис. 3.6):

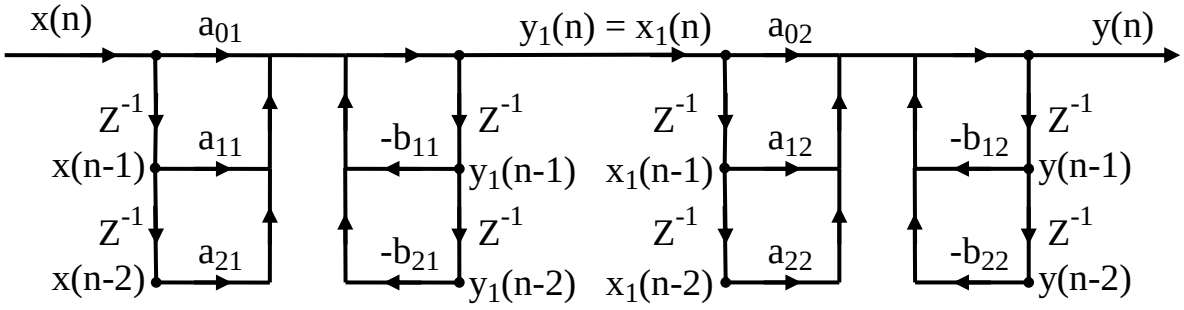

Рис. 3.6. Последовательное соединение фильтров (прямая форма)

Учитывая, что данные предыстории между каскадами совпадают ($y_1(n-k)=x_1(n-k)$), структура фильтра может быть преобразована к следующему виду (рис. 3.7).

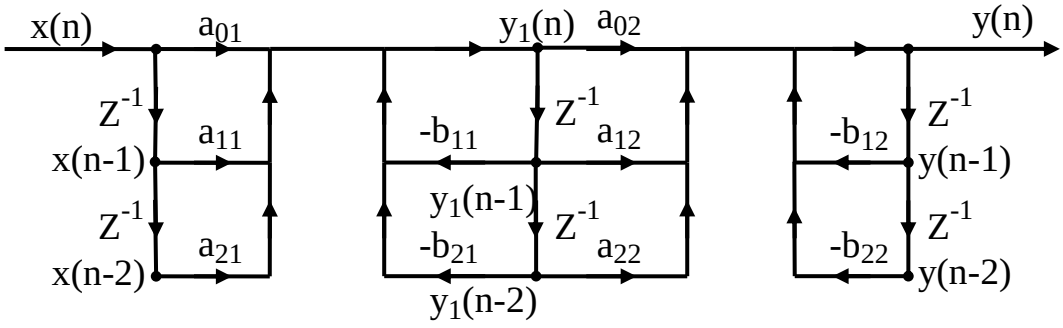

Рис. 3.7. Последовательное соединение цифровых рекурсивных фильтров (модифицированная прямая форма)

При такой реализации сразу исчезает недостаток прямой формы, связанный с двойным количеством ячеек предыстории. В данной реализации общее число ячеек предыстории при прямой форме только на 3 больше, чем при канонической. При программной реализации такого фильтра необходимо учитывать то, что сдвигать по памяти нужно только отсчеты предыстории входных значений, поскольку выходные значения будут сдвинуты при работе следующего каскада (в котором они становятся входными) и только после последнего каскада необходимо сдвинуть два значения y(n-1) и y(n).

Рекомендуемое расположение промежуточных данных в памяти показано на рис. 3.8.

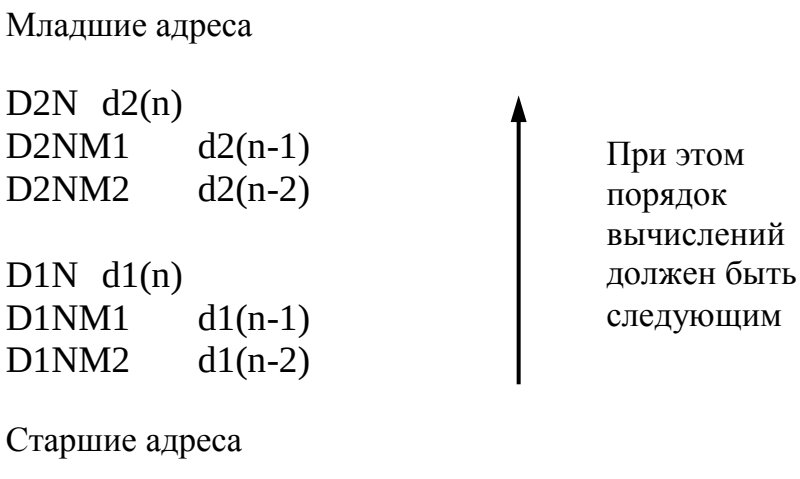

Рис. 3.8. Расположение промежуточных данных в памяти

3.2.3. Программная реализация цифровых рекурсивных фильтров

```
; Линейная программа ЦРФ (каноническая форма) 4-го порядка
; с последовательным соединением 2-х каскадов

        .data
D2N     .WORD  0
        .  .  .
D1NM2   .WORD  0
A01     .WORD  0
A11     .WORD  0
A21     .WORD  0
B11     .WORD  0
B21     .WORD  0
A02     .WORD  0
        .  .  .
B22     .WORD  0
XN      .WORD  0

        .text
START   LD     #XN, DP      ; текущая страница с XN
        SSBX   FRCT         ; режим умножения дробных чисел
        RSBX   OVM          ; нет коррекции переполн. в АЛУ
        ORM    #1, PMST     ; коррекция перепол. при записи
        STM    #D2NM1, AR2  ; AR2:= адрес d(n-(N-1))
        RPTZ   A, #4        ; A:=0 и повторить 5 раз
        STL    A, *AR2+     ; обнуление предыстории di(n-k)
        RPT    #10-1        ; повторить 10 раз,AR2:=адр.a01
        MVPD   CA01, *AR2+  ; пересылка коэффициентов
WAIT    BC     WAIT, BIO    ; ожидание готовности
        PORTR  PA2, XN      ; ввод из порта x(n)
```

```
        ; вычисление d1(n)
        LD      XN, 16, A   ; x(n) в старшую часть акк.A
        LD      D1NM1, T    ; T:=d1(n-1)
        MAS     B11, A      ; A:=A-T*b11=A-b11*d1(n-1)
        LD      D1NM2, T    ; T:=d1(n-2)
        MASR    B21, A      ; A:=A-T*b21=A-b21*d1(n-2)
        STH     A, D1N      ; d1(n)
        ; вычисление y1(n)
        MPY     A21, A      ; A:=T*a21=a21*d1(n-2)
        LTD     D1NM1       ; T:=d1(n-1), d1(n-1)→d1(n-2)
        MAC     A11, A      ; A:=A+T*a11=A+a11*d1(n-1)
        LTD     D1N         ; T:=d1(n), d1(n)→d1(n-1)
        MAC     A01, A      ; A:=A+T*a01=A+a01*d1(n)=y1(n)
        ; вычисление d2(n)
        LD      D2NM1, T    ; T:=d2(n-1)
        MAS     B12, A      ; A:=A-T*b12=A-b12*d2(n-1)
        LD      D2NM2, T    ; T:=d1(n-2)
        MASR    B22, A      ; A:=A-T*b22=A-b22*d2(n-2)
        STH     A, D2N      ; d2(n)
        ; вычисление y(n)
        MPY     A22, A      ; A:=T*a22=a22*d2(n-2)
        LTD     D2NM1       ; T:=d2(n-1), d2(n-1)→d2(n-2)
        MAC     A12, A      ; A:=A+T*a12=A+a12*d2(n-1)
        LTD     D2N         ; T:=d2(n), d2(n)→d2(n-1)
        MAC     A02, A      ; A:=A+T*a02=A+a02*d2(n)=y(n)
        STH     B, XN       ; y(n)→x(n)
        BD      WAIT        ; задержанный переход на WAIT
        PORTW   XN, PA2     ;2w вывод в порт y(n)

; коэффициенты в памяти программ
CA01    .WORD   01F05h      ; 0.242342
CA11    .WORD   02B75h      ; 0.339521
CA21    .WORD   01EFDh      ; 0.242117
CB11    .WORD   0394Dh      ; 0.447687
CB21    .WORD   0D889h      ;-0.308310
CA02    .WORD   062F1h      ; 0.772990
CA12    .WORD   026DBh      ; 0.303581
CA22    .WORD   062EDh      ; 0.772887
CB12    .WORD   0FEF7h      ;-0.008080
CB22    .WORD   090EEh      ;-0.867723
```

3.3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические сведения по теме лабораторной работы по подразд. 3.2.
2. Получить у преподавателя задание для выполнения практической части работы.
3. В соответствии с заданием рассчитать с помощью программы SKIF коэффициенты фильтра.
4. Согласно выражениям (3.1), (3.6) написать и отладить программу цифрового рекурсивного фильтра.
5. Показать результат работы программы преподавателю.
6. Оформить и защитить отчет по лабораторной работе.

3.4. СОДЕРЖАНИЕ ОТЧЕТА

1. Цель работы.
2. Описание алгоритма работы программы.
3. Текст программы с комментариями.
4. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №4
ИЗМЕРЕНИЕ АМПЛИТУДНО-ЧАСТОТНОЙ ХАРАКТЕРИСТИКИ (АЧХ) ЦИФРОВОГО РЕКУРСИВНОГО ФИЛЬТРА

4.1. ЦЕЛЬ РАБОТЫ

Разработка алгоритма и написание программы для измерения АЧХ цифрового рекурсивного фильтра на базе лабораторного макета TMS320VC5402 DSP Starter Kit (DSK) и её отладка на лабораторном макете.

4.2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

4.2.1. Алгоритм измерения АЧХ фильтра

Для измерения АЧХ фильтра может быть использован генератор синусоидальных сигналов, разработанный в лабораторных работах №1 и №2. На вход фильтра в течение заданных интервалов времени последовательно подается синусоидальный сигнал с увеличивающейся частотой.

По истечении времени, необходимого на установку сигнала на выходе фильтра, необходимо измерить мощность сигнала, для чего можно воспользоваться выражением

$$E = \frac{1}{N}\sum_{n=0}^{N-1} y^2(n), \qquad (4.1)$$

где E – энергия сигнала, y(n) – значение на выходе фильтра, N – интервал усреднения, состоящий из целого числа периодов сигнала.

Для нахождения АЧХ фильтра необходимо найти отношение амплитуды синусоидального сигнала на выходе фильтра к амплитуде на входе фильтра на каждой частоте. При этом если взять единичную амплитуду входного сигнала, амплитуды выходных сигналов будут представлять собой АЧХ фильтра. Измерив мощность сигнала, можно найти амплитуду выходного сигнала из следующего соотношения:

$$E = \frac{A^2}{2}, \qquad (4.2)$$

где E – энергия сигнала, A – амплитуда синусоидального сигнала.

4.2.2. Программа измерения АЧХ рекурсивного фильтра

Для измерения АЧХ фильтра, запрограммированного в лабораторной работе №3, используются программы генераторов из лабораторных работ №1 и №2. После начала формирования очередной гармоники на вход фильтра необходимо подать определенное число отсчетов синуса (не менее одного периода) для достижения установившегося режима на выходе фильтра.

После достижения установившегося режима необходимо в течение N тактов (целое число периодов) накаливать сумму квадратов амплитуд. При N=256 после завершения суммирования в старшей части аккумулятора получится значение энергии (деление на 256 за счет 8-ми защитных разрядов).

Схема алгоритма программы измерения АЧХ фильтра представлена на рис. 4.1.

Рис. 4.1. Схема алгоритма измерения АЧХ фильтра

4.3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические сведения по теме лабораторной работы по подразд. 4.2.
2. Получить у преподавателя задание для выполнения практической части работы.
3. Согласно заданию написать и отладить программу.
4. Продемонстрировать результат работы программы в пошаговом режиме преподавателю.
5. Оформить и защитить отчет по лабораторной работе.

4.4. СОДЕРЖАНИЕ ОТЧЕТА

1. Цель работы.
2. Описание алгоритма работы программы.
3. Листинг программы с комментариями.
4. Выводы по работе.

ЛИТЕРАТУРА

1. Лабораторный практикум по курсу «Проектирование проблемно-ориентированных вычислительных средств» для студ. спец. «Электронные вычислительные средства». В 2-х ч. / А. А. Петровский [и др.]. – Минск : БГУИР : Ч. 1, 1999. – 51 с. ; Ч. 2, 2001. – 46 с.
2. Петровский, А. А. Программирование цифровых фильтров на процессорах для работы в реальном времени: учебно-метод. пособие по курсам «Проектирование проблемно-ориентированных вычислительных систем» и «Программирование проблемно-ориентированных ЭВС реального времени» для студ. спец. Т 08 02 00 "Проектирование и технология электронных вычислительных средств" / А. А. Петровский, В. Б. Клюс, В. В. Серков. – Минск : БГУИР, 1997. – 40 с.
3. TMS320C54x DSP. Reference Set. Volume 2 / Texas Instruments, 1990.
4. TMS320C54x DSKplus. User's Guide / Texas Instruments, 1996.
5. Digital Signal Processing Applications with the TMS320 Family. Theory, Algorithms, and Implementations. Volume 2 : Mnemonic Instruction Set / Texas Instruments, 2001.
6. Code Composer Studio. Getting Started Guide / Texas Instruments, 2001.

ПРИЛОЖЕНИЕ 1

Обозначения и сокращения в описаниях команд

Символ	Значение
A	Аккумулятор A
ALU	Арифметическо-логическое устройство
AR	Вспомогательный регистр
ARx	Определяет указанный вспомогательный регистр (0…7)
ARP	Указатель вспомогательного регистра (3-битное поле в ST0), указывает текущий AR
ASM	5-битное поле режима сдвига аккумулятора в ST1 (-16…15)
B	Аккумулятор B
BRAF	Флажок активности повторения блока в ST1
BRC	Счетчик повторения блока
BITC	4-битное значение, определяющее бит ячейки памяти, проверяемый командой BITC (0…15)
C16	Бит режима двухсловной/двойной точности арифметики в ST1
C	Бит переноса в ST0
CC	2-битный код условия (0…3)
CMPT	Бит режима совместимости в ST1
CPL	Бит режима трансляции в ST1
cond	Операнд, представляющий условие, используемое условными командами
[D]	Режим задержки
DAB	Шина адреса D
DAR	Регистр адреса D
dmad	Непосредственный 16-битный адрес памяти данных (0…65535)
Dmem	Операнд в памяти данных
DP	9-битное поле указателя страницы памяти данных в ST0 (0…511)
dst	Аккумулятор-приемник (A или B)
dst _	Противоположный аккумулятор: если dst=A, то dst_=B; если dst=B, то dst_=A
EAB	Шина адреса E
EAR	Регистр адреса E
extpmad	Непосредственный 23-битный адрес памяти программ
FRCT	Бит режима дробных чисел в ST1
hi (A)	Старшая часть аккумулятора (биты 31-16)
HM	Бит режима захвата (HOLD-режима) в ST1
IFR	Регистр флагов прерывания
INTM	Бит режима прерывания в ST1
K	Короткое непосредственное значение (константа) меньше, чем 9 бит
K3	3-битное непосредственное значение (0…7)
K5	5-битное непосредственное значение (-16…15)
K9	9-битное непосредственное значение (0…511)
lk	16-битное длинное непосредственное значение
Lmem	32-битный операнд памяти данных, использующий адресацию длинного слова
mmr, MMR	Регистр, отображаемый на память
MMRx, MMRy	Регистр, отображаемый на память, AR0-AR7 или SP
n	Число слов после команды XC; n = 1 или 2

N	Определяет регистр состояния, изменяемый в командах RSBX, SSBX и XC: 0 – ST0; 1 – ST1
OVA	Флаг переполнения аккумулятора A в ST0
OVB	Флаг переполнения аккумулятора B в ST0
OVdst	Флаг переполнения аккумулятора приемника (A или B)
OVdst _	Флаг переполнения аккумулятора, противоположного приемнику (B или A)
OVsrc	Флаг переполнения аккумулятора источника (A или B)
OVM	Бит режима переполнения в ST1
PA	Непосредственный 16-битный адрес порта (0…65 535)
PAR	Регистр адреса программы
PC	Программный счетчик
pmad	Непосредственный 16-битный адрес памяти программ (0…65 535)
Pmem	Операнд в памяти программ
PMST	Регистр состояния режима процессора
prog	Операнд в памяти программ
[R]	Режим округления
RC	Счетчик повторения
REA	Регистр конечного адреса повторяемого блока
rnd	Округление
RSA	Регистр начального адреса повторяемого блока
RTN	Регистр быстрого возврата, используемый в команде RETF[D]
SBIT	4-битное значение, которое определяет номер разряда регистра состояния, изменяемого в командах RSBX, SSBX и XC (0…15)
SHFT	4-битное значение сдвига (0…15)
SHIFT	5-битное значение сдвига (-16…15)
Sind	Операнд в памяти данных одиночного доступа, использующий косвенную адресацию
Smem	16-битный операнд в памяти данных одиночного доступа
SP	Указатель стека
src	Аккумулятор-источник (A или B)
ST0	Регистр состояния 0
ST1	Регистр состояния 1
SXM	Бит режима расширения знака в ST1
T	Временный регистр
TC	Флаг проверки/управления в ST0
TOS	Вершина стека
TRN	Переходный регистр
TS	Значение сдвига, указанное битами 5-0 регистра T (-16…31)
uns	Без знака
XF	Бит внешнего флага в регистре состояния ST1
XPC	Регистр расширения программного счетчика
Xmem	16-битный операнд в памяти данных двойного доступа, используемый в двухоперандных и некоторых однооперандных командах
Ymem	16-битный операнд в памяти данных двойного доступа, используемый в двухоперандных командах
– – SP	Значение указателя стека уменьшается на 1
++ SP	Значение указателя стека увеличивается на 1
++ PC	Значение программного счетчика увеличивается на 1

ПРИЛОЖЕНИЕ 2

Методы адресации и модификации косвенного адреса

Поле MOD	Синтаксис операнда	Функции	Описание
Одинарный операнд (Smem)			
0000 (0)	*ARx	адрес = ARx	ARx содержит адрес памяти данных
0001 (1)	*ARx-	адрес = ARx ARx = ARx - 1	После доступа к памяти адрес в ARx уменьшается на 1
0010 (2)	*ARx+	адрес = ARx ARx = ARx + 1	После доступа к памяти адрес в ARx увеличивается на 1
0011 (3)	*+ARx	адрес = ARx + 1 ARx = ARx + 1	Перед использованием адрес в ARx увеличивается на 1, и этот новый адрес используется для адресации операнда
0100 (4)	*ARx-0B	адрес = ARx ARx = B(ARx-AR0)	После доступа AR0 вычитается из ARx с обратным переносом (rc)
0101 (5)	*ARx-0	адрес = ARx ARx = ARx - AR0	После доступа AR0 вычитается из ARx
0110 (6)	*ARx+0	адрес = ARx ARx = ARx + AR0	После доступа AR0 прибавляется к ARx
0111 (7)	*ARx+0B	адрес = ARx ARx = B(ARx+AR0)	После доступа AR0 прибавляется к ARx с обратным переносом (rc)
1000 (8)	*ARx-%	адрес = ARx ARx = circ (ARx-1)	После доступа адрес в ARx уменьшается с использованием циклической адресации
1001 (9)	*ARx-0%	адрес = ARx ARx=circ(ARx-AR0)	После доступа AR0 вычитается из ARx с использованием циклической адресации
1010 (10)	*ARx +%	адрес = ARx ARx = circ (ARx+1)	После доступа адрес в ARx увеличивается с использованием циклической адресации
1011 (11)	*ARx+0%	адрес = ARx ARx=circ(ARx+AR0)	После доступа AR0 прибавляется к ARx с использованием циклической адресации
1100 (12)	*ARx(lk)	адрес = ARx + lk ARx = ARx	Сумма ARx и смещения длиной 16 битов (lk) используется как адрес памяти данных. ARx не модифицируется
1101 (13)	*+ARx(lk)	адрес = ARx + lk ARx = ARx + lk	Перед использованием смещение длиной 16 битов (lk) прибавляется к ARx, и эта сумма заменяет предыдущее содержимое ARx; эта сумма используется для адресации операнда памяти данных
1110 (14)	*+ARx(lk)%	addr=circ (ARx+lk) ARx =circ (ARx+lk)	Перед использованием смещение длиной 16 битов (lk) прибавляется к ARx с использованием циклической адресации и эта сумма заменяет предыдущее содержимое ARx; эта сумма используется для адресации операнда памяти данных
1111 (15)	* (lk)	адрес = lk	Смещение длиной 16 битов без знака (lk) используется как абсолютный адрес памяти данных (абсолютная адресация)
Двойной операнд (Xmem или Ymem) – используются только регистры AR2, AR3, AR4, AR5			
00 (0)	*ARx	адрес = ARx	ARx содержит адрес памяти данных
01 (1)	*ARx-	адрес = ARx ARx = ARx - 1	После доступа к памяти адрес в ARx уменьшается на 1
10 (2)	*ARx+	адрес = ARx ARx = ARx + 1	После доступа к памяти адрес в ARx увеличивается на 1
11 (3)	*ARx+0%	адрес = ARx ARx=circ(ARx+AR0)	После доступа AR0 прибавляется к ARx с использованием циклической адресации

ПРИЛОЖЕНИЕ 3

Система команд процессора TMS320VC5420 и условия переходов

Синтаксис	Выражение (операция)	С†	Ц†
Арифметические операции			
Команды сложения			
ADD Smem, src	src = src + Smem {src += Smem}	1	1
ADD Smem, TS, src	src = src + Smem << TS {}	1	1
ADD Smem, 16, src [, dst]	dst = src + Smem << 16 {}	1	1
ADD Smem [,SHIFT], src [,dst]	dst = src + Smem << SHIFT {}	2	2
ADD Xmem, SHFT, src	src = src + Xmem <<_SHFT {}	1	1
ADD Xmem, Ymem, dst	dst = Xmem << 16 + Ymem << 16	1	1
ADD #lk [,SHFT], src [,dst]	dst = src + #lk << SHFT {}	2	2
ADD #lk, 16, src [, dst]	dst = src + #lk << 16 {}	2	2
ADD src [, SHIFT] [, dst]	dst = dst + src << SHIFT {}	1	1
ADD src, ASM [, dst]	dst = dst + src << ASM {}	1	1
ADDC Smem, src	src = src + Smem + C (CARRY) {}	1	1
ADDM #lk, Smem	Smem = Smem + #lk {}	2	2
ADDS Smem, src	src = src + uns(Smem) {}	1	1
Команды вычитания			
SUB Smem, src	src = src – Smem {src -= Smem}	1	1
SUB Smem, TS, src	src = src – Smem << TS {}	1	1
SUB Smem, 16, src [, dst]	dst = src – Smem << 16 {}	1	1
SUB Smem [,SHIFT], src [,dst]	dst = src – Smem << SHIFT {}	2	2
SUB Xmem, SHFT, src	src = src – Xmem << SHFT {}	1	1
SUB Xmem, Ymem, dst	dst = Xmem << 16 – Ymem << 16	1	1
SUB #lk [, SHFT], src [, dst]	dst = src – #lk << SHFT {}	2	2
SUB #lk, 16, src [, dst]	dst = src – #lk <<16 {}	2	2
SUB src[, SHIFT] [, dst]	dst = dst – src << SHIFT {}	1	1
SUB src, ASM [, dst]	dst = dst – src << ASM {}	1	1
SUBB Smem, src	src = src – Smem – C (BORROW) {}	1	1
SUBC Smem, src (SUBC (Smem, src))	Если (src – Smem << 15) > 0 src = (src – Smem<<15) << 1 + 1 иначе src = src << 1	1	1
SUBS Smem, src	src = src – uns(Smem) {}	1	1
Команды умножения			
MPY Smem, dst	dst = T * Smem	1	1
MPYR Smem, dst	dst = rnd(T * Smem)	1	1
MPY Xmem, Ymem, dst	dst = Xmem * Ymem, T = Xmem	1	1
MPY Smem, #lk, dst	dst = Smem * #lk , T = Smem	2	2
MPY #lk, dst	dst = T * #lk	2	2
MPYA dst	dst = T * hi(A)	1	1
MPYA Smem	B = Smem * hi(A), T = Smem	1	1

Синтаксис	Выражение (операция)	С†	Ц†
MPYU Smem, dst	dst = uns(T) * uns(Smem) (dst = T * uns(Smem))	1	1
SQUR Smem, dst	dst = Smem * Smem, T = Smem	1	1
SQUR A, dst	dst = hi(A) * hi(A)	1	1
Команды умножения со сложением и вычитанием			
MAC Smem, src	src = src + T * Smem {src += T…}	1	1
MAC Xmem, Ymem, src [, dst]	dst = src + Xmem * Ymem, {} T = Xmem	1	1
MAC #lk, src [, dst]	dst = src + T * #lk {}	2	2

MAC Smem, #lk, src [, dst]	dst = src + Smem * #lk, {}
	T = Smem
		2	2

MACR Smem, src	src = rnd(src + T * Smem)	1	1
MACR Xmem, Ymem, src [, dst]	dst = rnd(src + Xmem * Ymem), T = Xmem	1	1
MACA Smem [, B]	B = B + Smem * hi(A), {} T = Smem	1	1
MACA T, src [, dst]	dst = src + T * hi(A) {}	1	1
MACAR Smem [, B]	B = rnd(B + Smem * hi(A), T = Smem	1	1
MACAR T, src [, dst]	dst = rnd(src + T * hi(A)	1	1
MACD Smem, pmad, src (MACD (Smem, pmad, src))	src = src + Smem * pmad, T = Smem, (Smem + 1) = Smem	2	3
MACP Smem, pmad, src (MACP (Smem, pmad, src))	src = src + Smem * pmad, T = Smem	2	3
MACSU Xmem, Ymem, src	src = src + uns(Xmem) * Ymem, {} T = Xmem	1	1
MAS Smem, src	src = src – T * Smem {src -= T…}	1	1
MASR Smem, src	src = rnd(src – T * Smem)	1	1
MAS Xmem, Ymem, src [, dst]	dst = src – Xmem * Ymem, {} T = Xmem	1	1
MASR Xmem, Ymem, src [, dst]	dst = rnd(src – Xmem * Ymem), T = Xmem	1	1
MASA Smem [, B]	B = B – Smem * hi(A), {} T = Smem	1	1
MASA T, src [, dst]	dst = src – T * hi(A) {}	1	1
MASAR T, src [, dst]	dst = rnd(src – T * hi(A))	1	1
SQURA Smem, src	src = src + Smem * Smem, {…+=…} T = Smem	1	1
SQURS Smem, src	src = src – Smem * Smem, {…-=…} T = Smem	1	1
Команды с двойным операндом			
DADD Lmem, src [, dst] (dst = src + dbl(Lmem)){+=} (dst = src + dual(Lmem))	Если C16 = 0 dst = Lmem + src Если C16 = 1 dst(39–16)=Lmem(31–16)+src(31–16) dst(15–0) =Lmem(15–0) +src(15–0)	1	1

Синтаксис	Выражение (операция)	С†	Ц†
DADST Lmem, dst (dst = dadst (Lmem, T))	Если C16 = 0 dst = Lmem+(T<<16+T) Если C16 = 1 dst(39–16)= Lmem(31–16)+ T dst(15–0) = Lmem(15–0) – T	1	1
DRSUB Lmem, src (dst = dbl(Lmem) – src) (dst = dual(Lmem) – src)	Если C16 = 0 src = Lmem – src Если C16 = 1 src(39–16)=Lmem(31–16)–src(31–16) src(15–0) =Lmem(15–0) –src(15–0)	1	1
DSADT Lmem, dst (dst = dsadt (Lmem, T))	Если C16 = 0 dst = Lmem–(T<<16+T) Если C16 = 1 dst(39–16)= Lmem(31–16)– T dst(15–0) = Lmem(15–0) + T	1	1
DSUB Lmem, src (src = src - dbl(Lmem)){-=} (src = src - dual(Lmem))	Если C16 = 0 src = src – Lmem Если C16 = 1 src(39–16)=src(31–16)–Lmem(31–16) src(15–0) =src(15–0) –Lmem(15–0)	1	1
DSUBT Lmem, dst (dst = dbl(Lmem) – T) (dst = dual(Lmem) – T)	Если C16 = 0 dst = Lmem–(T<<16+T) Если C16 = 1 dst(39–16)= Lmem(31–16)– T dst(15–0) = Lmem(15–0) – T	1	1
Проблемно-ориентированные команды			
ABDST Xmem, Ymem (ABDST (Xmem, Ymem))	B = B + |hi(A)| A = (Xmem – Ymem) << 16	1	1
ABS src [, dst]	dst = |src|	1	1
CMPL src [, dst]	dst = ~src	1	1
DELAY Smem (DELAY (Smem))	(Smem + 1) = Smem	1	1
EXP src (T = exp(src))	T = number of sign bits (src) – 8	1	1
FIRS Xmem, Ymem, pmad (FIRS (Xmem, Ymem, pmad))	B = B + A * pmad A = (Xmem + Ymem) << 16	2	3
LMS Xmem, Ymem (LMS (Xmem, Ymem))	B = B + Xmem * Ymem A = A + Xmem << 16 + 2^{15}	1	1
MAX dst	dst = max(A, B)	1	1
MIN dst	dst = min(A, B)	1	1
NEG src [, dst]	dst = –src	1	1
NORM src [, dst]	dst = src << TS { dst = norm(src, TS) }	1	1
POLY Smem (POLY (Smem))	B = Smem << 16 A = rnd(hi(A) * T + B)	1	1
RND src [, dst]	dst = src + 2^{15} (dst = rnd(src))	1	1
SAT src	saturate(src) [to 32 bits]	1	1
SQDST Xmem, Ymem (SQDST (Xmem, Ymem))	B = B + hi(A) * hi(A) A = (Xmem – Ymem) << 16	1	1
Логические операции			
Команды И			
AND Smem, src	src = src & Smem {src &= Smem}	1	1
AND #lk [, SHFT], src [, dst]	dst = src & #lk << SHFT {}	2	2

Синтаксис	Выражение (операция)	С†	Ц†
AND #lk, 16, src [, dst]	dst = src & #lk << 16 {}	2	2
AND src [, SHIFT] [, dst]	dst = dst & src << SHIFT {}	1	1
ANDM #lk, Smem	Smem = Smem & #lk {}	2	2
Команды ИЛИ			
OR Smem, src	src = src | Smem {src |= Smem}	1	1
OR #lk [, SHFT], src [, dst]	dst = src | #lk << SHFT {}	2	2
OR #lk, 16, src [, dst]	dst = src | #lk << 16 {}	2	2
OR src [, SHIFT] [, dst]	dst = dst | src << SHIFT {}	1	1
ORM #lk, Smem	Smem = Smem | #lk {}	2	2
Команды Исключающее ИЛИ			
XOR Smem, src	src = src ^ Smem {src ^= Smem}	1	1
XOR #lk [, SHFT], src [,dst]	dst = src ^ #lk << SHFT {}	2	2
XOR #lk, 16, src [, dst]	dst = src ^ #lk << 16 {}	2	2
XOR src [, SHIFT] [, dst]	dst = dst ^ src << SHIFT {}	1	1
XORM #lk, Smem	Smem = Smem ^ #lk {}	2	2
Команды сравнения			
BIT Xmem, BITC (TC = bit (Xmem, bitc))	TC = Xmem(15 – BITC)	1	1
BITF Smem, #lk (TC = bitf (Smem, #lk))	TC = (Smem & #lk)	2	2
BITT Smem (TC = bitt(Smem))	TC = Smem(15 – T(3–0))	1	1
CMPM Smem, #lk	TC = (Smem == #lk)	2	2
CMPR CC, ARx (TC=(AR0 ? ARx)) CC= EQ,LT,GT,NEQ; ?= ==, >, <,!=	Сравнивает ARx с AR0	1	1
Команды сдвига			
ROL src (src = src\\CARRY)	Циклический сдвиг влево через C	1	1
ROLTC src (roltc (src))	Циклический сдвиг влево через TC	1	1
ROR src (src = src//CARRY)	Циклический сдвиг вправо через C	1	1
SFTA src, SHIFT [, dst] (src = src <<C SHIFT)	dst = src<<SHIFT{арифметический}	1	1
SFTC src (shiftc (src))	если src(31)=src(30) то src=src<<1 TC=0 иначе(или src=0) TC=1	1	1
SFTL src, SHIFT [, dst] (src = src <<< SHIFT)	dst = src << SHIFT {логический}	1	1
Команды управления ходом выполнения программы			
Команды перехода			
B[D] pmad ([d]goto pmad)	PC = pmad(15–0)	2	4/ 2¶
BACC[D] src ([d]goto src)	PC = src(15–0)	1	6/ 4¶
BANZ[D] pmad, Sind (if(Sind!=0) [d]goto pmad)	Если (Sind≠0) то PC = pmad(15–0)	2	4‡/ 2§/ 2¶

Синтаксис	Выражение (операция)	С†	Ц†
BC[D] pmad,cond[,cond[,cond]] (if (cond(s)) [d]goto pmad)	Если условия (cond(s)) выполняются, то PC = pmad(15-0)	2	5‡/ 3§/ 3¶
FB[D] extpmad (far [d]goto extpmad)	PC = pmad(15-0), XPC = pmad(22-16)	2	4/ 2¶
FBACC[D] src (far [d]goto src)	PC = src(15-0), XPC = src(22-16)	1	6/ 4¶
Команды вызова			
CALA[D] src ([d]call src)	--SP, TOS = PC + 1[3¶], PC = src(15-0)	1	6/ 4¶
CALL[D] pmad ([d]call pmad)	--SP, TOS = PC + 2[4¶], PC = pmad(15-0)	2	4/ 2§
CC[D] pmad,cond[,cond[,cond]] (if (cond(s)) [d]call pmad)	Если (cond(s)) то --SP, TOS = PC + 2[4¶], PC = pmad(15-0)	2	5‡/ 3§/ 3¶
FCALA[D] src (far [d]call src)	--SP, TOS = PC + 1[3¶], PC = src(15-0),XPC = src(22-16)	1	6/ 4¶
FCALL[D] extpmad (far [d]call extpmad)	--SP, TOS = PC + 2[4¶], PC = pmad(15-0),XPC = pmad(22-16)	2	4/ 2¶
Команды возврата			
FRET[D] (far [d]return)	XPC = TOS, ++SP, PC = TOS, ++SP	1	6/ 4¶
FRETE[D] (far [d]return_enable)	XPC = TOS, ++SP, PC = TOS, ++SP, INTM = 0	1	6/ 4¶
RC[D] cond [, cond [, cond]] (if (cond(s)) [d]return)	Если условия (cond(s)) выполняются, то PC = TOS, ++SP	1	5‡/ 3§/ 3¶
RET[D] ([d]return)	PC = TOS, ++SP	1	5/ 3¶
RETE[D] ([d]return_enable)	PC = TOS, ++SP, INTM = 0	1	5/ 3¶
RETF[D] ([d]return_fast)	PC = RTN, ++SP, INTM = 0	1	3/ 1¶
Команды прерывания			
INTR K (int(k))	--SP, TOS = ++PC, PC = IPTR(15-7) + K<<2, INTM = 1	1	3
TRAP K (trap(k))	--SP, TOS = ++PC, PC = IPTR(15-7) + K<<2	1	3
Команды повторения			
RPT Smem (repeat(Smem))	Repeat single, RC = Smem	1	3
RPT #K (repeat(#k))	Repeat single, RC = #K	1	1
RPT #lk (repeat(#lk))	Repeat single, RC = #lk	2	2
RPTB[D] pmad ([d]blockrepeat(pmad))	Repeat block, RSA = PC + 2[4¶], REA = pmad, BRAF = 1	2	4/ 2¶
RPTZ dst, #lk (repeat(#lk), dst=0)	Repeat single, RC = #lk, dst = 0	2	2

Синтаксис	Выражение (операция)	С†	Ц†
Команды управления стеком			
FRAME K	SP = SP + K	1	1
POPD Smem (Smem = pop())	Smem = TOS, ++SP	1	1
POPM MMR (MMR = pop())	MMR = TOS, ++SP {mmr(MMR)=pop()}	1	1
PSHD Smem (push(Smem))	--SP, TOS = Smem	1	1
PSHM MMR (push(MMR))	--SP, TOS = MMR {push(mmr(MMR))}	1	1
Другие команды управления программой			
IDLE K	idle(K)	1	4
MAR Smem (mar(Smem))	Если CMPT = 0, модифицируется ARx Если CMPT = 1 и ARx ≠ AR0, модифицируется ARx, ARP = x Если CMPT = 1 и ARx = AR0, модифицируется AR(ARP)	1	1
NOP	Нет операции	1	1
RESET	Программный сброс	1	3
RSBX N, SBIT (SBIT = 0)	STN (SBIT) = 0 (ST(N,SBIT) = 0)	1	1
SSBX N, SBIT (SBIT = 1)	STN (SBIT) = 1 (ST(N,SBIT) = 1)	1	1
XC n, cond [, cond[, cond]] (if (cond(s)) execute(n))	Если (cond(s)) выполняются следующие **n** команд; n = 1 or 2	1	1
Команды загрузки, сохранения и пересылки данных			
Команды загрузки			
DLD Lmem, dst	dst = Lmem (dst={**dbl**|**dual**}(Lmem))	1	1
LD Smem, dst	dst = Smem	1	1
LD Smem, TS, dst	dst = Smem << TS	1	1
LD Smem, 16 , dst	dst = Smem << 16	1	1
LD Smem [, SHIFT], dst	dst = Smem << SHIFT	2	2
LD Xmem, SHFT, dst	dst = Xmem << SHFT	1	1
LD #K, dst	dst = #K	1	1
LD #lk [, SHFT], dst	dst = #lk << SHFT	2	2
LD #lk, 16, dst	dst = #lk << 16	2	2
LD src, ASM [, dst]	dst = src << ASM	1	1
LD src [, SHIFT], dst	dst = src << SHIFT	1	1
LD Smem, T	T = Smem	1	1
LD Smem, DP	DP = Smem(8-0)	1	3
LD #k9, DP	DP = #k9	1	1
LD #k5, ASM	ASM = #k5	1	1
LD #k3, ARP	ARP = #k3	1	1
LD Smem, ASM	ASM = Smem(4-0)	1	1
LDM MMR, dst	dst = MMR { dst = mmr(MMR) }	1	1
LDR Smem, dst	dst = rnd(Smem){=Smem<<16+1<<15}	1	1

Синтаксис	Выражение (операция)	С†	Ц†
LDU Smem, dst	dst = uns(Smem)	1	1
LTD Smem (ltd(Smem))	T = Smem, (Smem + 1) = Smem	1	1
Команды сохранения			
DST src, Lmem	Lmem = src ({**dbl**|**dual**}(Lmem)=dst)	1	2
ST T, Smem	Smem = T	1	1
ST TRN, Smem	Smem = TRN	1	1
ST #lk, Smem	Smem = #lk	2	2
STH src, Smem	Smem = hi(src)	1	1
STH src, ASM, Smem	Smem = hi(src) << ASM	1	1
STH src, SHFT, Xmem	Xmem = hi(src) << SHFT	1	1
STH src [, SHIFT], Smem	Smem = hi(src) << SHIFT	2	2
STL src, Smem	Smem = src	1	1
STL src, ASM, Smem	Smem = src << ASM	1	1
STL src, SHFT, Xmem	Xmem = src << SHFT	1	1
STL src [, SHIFT], Smem	Smem = src << SHIFT	2	2
STLM src, MMR	MMR = src { mmr(MMR) = src }	1	1
STM #lk, MMR	MMR = #lk { mmr(MMR) = #lk }	2	2
Команды условного сохранения			
CMPS src, Smem (cmps(src, Smem))	Если src(31–16) > src(15–0) то Smem = src(31–16) TRN<<1 TC=0 иначе Smem = src(15–0) TRN<<1+1 TC=1	1	
SACCD src, Xmem, cond	Если (cond) Xmem = hi(src) << ASM	1	1
SRCCD Xmem, cond	Если (cond) Xmem = BRC	1	1
STRCD Xmem, cond	Если (cond) Xmem = T	1	1
Команды параллельной загрузки и сохранения			
ST src, Ymem || LD Xmem, dst	Ymem = hi(src) << ASM || dst = Xmem << 16	1	1
ST src, Ymem || LD Xmem, T	Ymem = hi(src) << ASM || T = Xmem	1	1
Команды параллельной загрузки и умножения			
LD Xmem, dst || MAC Ymem, dst_	dst = Xmem << 16 || dst_ = dst_ + T * Ymem {+=}	1	1
LD Xmem, dst || MACR Ymem, dst_	dst = Xmem << 16 || dst_ = rnd(dst_ + T * Ymem)	1	1
LD Xmem, dst || MAS Ymem, dst_	dst = Xmem << 16 || dst_ = dst_ – T * Ymem {–=}	1	1
LD Xmem, dst || MASR Ymem, dst_	dst = Xmem << 16 || dst_ = rnd(dst_ – T * Ymem)	1	1
Команды параллельного сохранения и сложения/вычитания			
ST src, Ymem || ADD Xmem, dst	Ymem = hi(src) << ASM || dst = (Xmem << 16) + dst_	1	1
ST src, Ymem || SUB Xmem, dst	Ymem = hi(src) << ASM || dst = (Xmem << 16) – dst_	1	1

Синтаксис	Выражение (операция)	С†	Ц†
Команды параллельного сохранения и умножения			
ST src, Ymem || MAC Xmem, dst	Ymem = hi(src) << ASM || dst = dst + T * Xmem { += }	1	1
ST src, Ymem || MACR Xmem, dst	Ymem = hi(src) << ASM || dst = rnd(dst + T * Xmem)	1	1
ST src, Ymem || MAS Xmem, dst	Ymem = hi(src) << ASM || dst = dst – T * Xmem { -= }	1	1
ST src, Ymem || MASR Xmem, dst	Ymem = hi(src) << ASM || dst = rnd(dst – T * Xmem)	1	1
ST src, Ymem || MPY Xmem, dst	Ymem = hi(src) << ASM || dst = T * Xmem	1	1
Другие команды загрузки и сохранения			
MVDD Xmem, Ymem	Ymem = Xmem	1	1
MVDK Smem, dmad	data(dmad) = Smem	2	2
MVDM dmad, MMR	MMR = data(dmad)	2	2
MVDP Smem, pmad	prog(pmad) = Smem	2	4
MVKD dmad, Smem	Smem = data(dmad)	2	2
MVMD MMR, dmad	data(dmad) = MMR	2	2
MVMM MMRx, MMRy	MMRy = MMRx	1	1
MVPD pmad, Smem	Smem = prog(pmad)	2	3
PORTR PA, Smem	Smem = port(PA)	2	2
PORTW Smem, PA	port(PA) = Smem	2	2
READA Smem	Smem = prog(A)	1	5
WRITA Smem	prog(A) = Smem	1	5

Условия переходов(cond операнд)

Условие	Гр.	Описания	Код условия	Условие	Гр.	Описания	Код условия
BIO	2C	/BIO низкий	0000 0011	NBIO	2C	/BIO высокий	0000 0010
C	2B	C = 1	0000 1100	NC	2B	C = 0	0000 1000
TC	2A	TC = 1	0011 0100	NTC	2A	TC = 0	0010 0000
AEQ	1A	(A) = 0	0100 0101	BEQ	1A	(B) = 0	0100 1101
ANEQ	1A	(A) ≠ 0	0100 0100	BNEQ	1A	(B) ≠ 0	0100 1100
AGT	1A	(A) > 0	0100 0110	BGT	1A	(B) > 0	0100 1110
AGEQ	1A	(A) ≥ 0	0100 0010	BGEQ	1A	(B) ≥ 0	0100 1010
ALT	1A	(A) < 0	0100 0011	BLT	1A	(B) < 0	0100 1011
ALEQ	1A	(A) ≤ 0	0100 0011	BLEQ	1A	(B) ≤ 0	0100 1111
AOV	1B	Переполн. A	0111 0000	BOV	1B	Переполн. B	0111 1000
ANOV	1B	Нет переп.A	0110 0000	BNOV	1B	Нет переп.B	0110 1000
UNC		Безусловный	0000 0000				

Примечания:
† Количество слов (С) и циклов (Ц) соответствуют использованию DARAM для данных.
‡ Условие истинно.
§ Условие ложно.
¶ Задержанная команда.

Св. план 2008, поз. 18

Учебное издание

Клюс Владимир Борисович

Качинский Михаил Вячеславович

Давыдов Александр Борисович

ПРОГРАММИРОВАНИЕ ПРОБЛЕМНО-ОРИЕНТИРОВАННЫХ ВЫЧИСЛИТЕЛЬНЫХ СРЕДСТВ РЕАЛЬНОГО ВРЕМЕНИ

Лабораторный практикум
для студентов специальности I-40 02 02
«Электронные вычислительные средства»
дневной формы обучения

В 2-х частях

Часть 1

Редактор М. В. Тезина
Корректор Е. Н. Батурчик
Компьютерная верстка Е. Г. Бабичева

Подписано в печать 18.06.2008. Формат 60x84 1/16. Бумага офсетная.
Гарнитура «Таймс». Печать ризографическая. Усл. печ. л. 2,21
Уч.-изд. л. 2,1. Тираж 100 экз. Заказ 4.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ № 02330/0056964 от 01.04. 2004. ЛП № 02330/0131666 от 30.04. 2004.
220013, Минск, П. Бровки, 6.