Федеральное агентство связи

Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования

ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ

ЭЛЕКТРОННАЯ БИБЛИОТЕЧНАЯ СИСТЕМА

Самара

ФГОБУ ВПО Поволжский государственный университет
телекоммуникаций и информатики

Стефанов А. М.

ПЕРЕСЫЛКА ДАННЫХ И АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ НА ЯЗЫКЕ АССЕМБЛЕРА ПРОЦЕССОРА TMS320C6x

Задания и методические указания к лабораторным работам

Самара
2012

ОГЛАВЛЕНИЕ

Введение

Методическая разработка содержит 3 лабораторных работы, направленных на освоение основных приёмов работы с симулятором команд сигнального процессора TMS320C62x и изучение особенностей команд пересылки данных и арифметических операций.

Методическая разработка может использоваться на лабораторных и практических занятиях по дисциплинам «Вычислительная техника и информационные технологии» и «Цифровые устройства и микропроцессоры» для студентов телекоммуникационных специальностей и направлений.

Рекомендуемая литература

1. Сперанский, В. С. Сигнальные микропроцессоры и их применение в системах телекоммуникаций и электроники: учеб. пособие для вузов/В. С. Сперанский. – М.: Горячая линия - Телеком, 2008. – 168 с.
2. Стефанов, А. М. Вычислительная техника и информационные технологии: учеб. пособие/А. М. Стефанов. – Самара: ПГАТИ, 2006. – 85 с.
3. Конспект лекций по дисциплине.

Содержание отчета

1. Название лабораторной работы.
2. Код группы, фамилия и инициалы студента.
3. Формулировка индивидуальных заданий данной лабораторной работы.
4. Блок-схема алгоритма решения задачи.
5. Таблица, содержащая структурированную программу, каждая командная строка которой сопровождается прогнозом содержимого используемых регистров РОН (регистра-приемника и регистра адреса) и ячейки памяти данных (ПД) процессора в 16-ричной системе счисления:

Заголовок таблицы результатов выполнения лабораторной работы

Командная строка	Регистры РОН командной строки		Ячейка ПД процессора, используемая в командной строке	
	Имя	Прогноз содержимого, Hex	Номер, Hex	Содержимое, Hex

Данная таблица предъявляется преподавателю до прогона программы с целью обнаружения ошибок и получения соответствующих разъяснений у преподавателя.

Систематизация результатов выполнения лабораторных работ

На любом доступном диске создать папку группы, например MS-91. В этой папке создать индивидуальную рабочую папку с уникальным именем, например Ivanov. В рабочей папке размещаются имена файлов с результатами выполнения лабораторных работ. При этом имена файлов должны включать номер соответствующей лабораторной работы, например lb1.*, lb2.*, …, lb8.*.

Для удобства работы с симулятором команд процессора TMS320C6x из папки C:\C6XTOOLS\CODEGEN.110\ BIN в рабочую папку ***скопировать*** файлы ASM6x.exe и LNK6x.exe.

1. СИМУЛЯТОР КОМАНД TMS320C6201

Симуляторы являются программами, имитирующими работу процессора на уровне его команд. Используются они для тестирования и улучшения программного кода. Симуляторы предоставляют возможность как пошагового, так и автоматического выполнения (прогона) программы.

Цель работы

Изучить основные приемы работы с симулятором команд ассемблера сигнального процессора TMS320C6x.

Подготовка к работе

По указанной выше литературе изучить:

- структуру процессора TMS320C6x;
- этапы разработки программы;
- процесс подготовки исполняемой программы;
- средства отладки прикладных программ: аппаратные эмуляторы, проверочные модули, симуляторы команд, отладчики.

Контрольные вопросы

1. Поясните структуру строки ассемблера процессора TMS320C6x.
2. Чем отличаются директива и команда ассемблера?
3. Назовите основные этапы преобразования исходной программы в исполняемый программный модуль.
4. Дайте краткую характеристику этапа редактирования текста исходной программы и соответствующим инструментальным средствам.
5. Дайте краткую характеристику этапа трансляции исходной программы.
6. Дайте краткую характеристику этапа загрузки объектных модулей в оперативную память.
7. Дайте краткую характеристику этапа компоновки объектных модулей.
8. Поясните назначение файла листинга исходной программы и способ его получения с помощью инструментов симулятора команд процессора TMS320C6x.
9. Поясните назначение и содержание окна Disassembly симулятора команд процессора TMS320C6x.

10. Поясните назначение и содержание окна CPU симулятора команд процессора TMS320C6x.

11. Поясните процесс загрузки исполняемого модуля в симулятор команд процессора TMS320C6x.

12. Поясните способы прогона программы в симуляторе команд процессора TMS320C6x.

Задания и методические указания к их выполнению

1. Запустить симулятор (C:\C6XTOOLS\CODEGEN. 110\BIN\SIM62x.exe) и изучить окна его интерфейса.

В режиме отладки автоматически создаются четыре окна с отображением чисел в 16-ричной системе счисления:

– окно Disassembly отображает команды дизассемблера – ассемблера, восстановленного по объектному коду, содержащемуся в программной памяти симулятора. В первой слева колонке этого окна отображаются адреса ячеек программной памяти симулятора, во второй – их содержимое (объектный код), в третьей – мнемоники команд и имена исполняющих их модулей процессора, в четвертой – поле операндов командной строки ассемблера;

– окно CPU показывает содержимое регистров процессора. При необходимости его можно задавать принудительно. Для этого двойным щелчком компьютерной мыши по имени регистра выделяется его содержимое, вводится требуемое 16-ричное число и нажимается клавиша <Enter>;

– окно Memory дублирует вторую слева колонку окна Disassembly. Однако здесь можно изменить содержимое ячеек программной памяти посредством двойного щелчка компьютерной мыши по нужным разрядам выбранной ячейки памяти;

– окно Command включает область ввода команд управления симулятором и область отображения сообщений об ошибке загрузки программы, результате выполнения введенной команды управления симулятором и служебной информации.

2. Создание исполняемого программного модуля.

Исполняемый модуль непосредственно загружается в симулятор. Необходимый файл получается в результате следующей последовательности действий.

1). В текстовом редакторе «Блокнот» сформировать текст исходной программы на языке ассемблера.

В данной работе ввести текст:

```
k  .set  2            ; присвоение символу k значения 2
   mvk k,a2           ; ввод значения k в РОН a2
   mv   a2,b2         ;копирование содержимого a2 в РОН b2
   add  a2,b2,a2  ;сложение содержимого a2 и b2 с
*размещением результата в a2.
```

Обратите внимание:

– каждая командная строка начинается как минимум с одного пробела,

поскольку предыдущие поля не используются;

– символ «;» открывает текст комментария (на машинный язык не переводится) при его размещении в данной командной строке, а символ «*» – если он начинается с первого поля строки ассемблера.

2). Сохранить текст исходной программы в рабочей папке под именем lb1.asm.

С этой целью в пункте «Файл» оконного меню редактора «Блокнот» выбрать команду *Сохранить как …* В открывшемся окне диалога выполнить следующее:

– указать место хранения, то есть на дереве папок найти и открыть рабочую папку;

– в списке *Тип файла* выбрать *Все файлы*;

– в поле *Имя файла* ввести полное имя файла (здесь lb1.asm);

– компьютерной мышью «щелкнуть» кнопку *Сохранить*.

3). Получить объектный файл (здесь lb1.obj), для чего из рабочей папки запустить программу ассемблера (файл ASM6x. exe) и в её окне ввести имя ассемблируемого файла, причем расширение имени указывать не обязательно (в данном случае достаточно ввести lb1). После этого нажать клавишу <Enter>.

Отсутствие в рабочей папке объектного файла означает, что в исходном файле (здесь lb1.asm) имеются ошибки. Определить их удобно с помощью файла листинга программы, который получается следующим образом:

– из командной строки (Пуск\Программы\Стандартные\Выполнить) запустить программу Ассемблера: <путь к рабочей папке>\<имя рабочей папки>\ASM6x.exe -l (опция *-l* отделяется от имени файла пробелом);

– в окне программы ассемблера, как и ранее, ввести имя ассемблируемого файла (здесь lb1);

– нажать клавишу <Enter>.

В результате в рабочей папке образуется файл листинга (здесь lb1.lst), содержащий сведения об ошибках.

После коррекции текста исходного файла сохранить его (команда *Сохранить*) и повторить ассемблирование.

4). Получить исполняемый файл (здесь lb1.out), для чего из рабочей папки запустить программу компоновщика (файл LNK6x.exe) и в её окне (рис. 1) в режиме диалога последовательно и построчно ввести сведения о компонуемых файлах. При этом достаточно ограничиться лишь именем объектного файла, причем без расширения, согласившись тем самым с именем исполняемого файла (здесь lb1.out), предлагаемым компоновщиком (рис. 1).

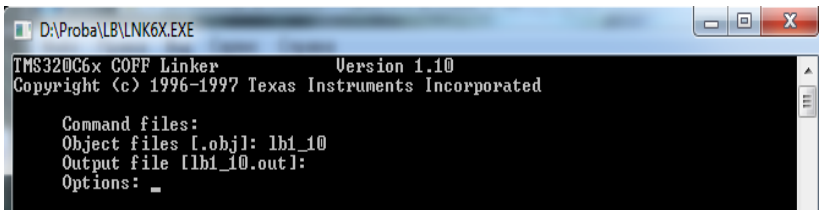


Рис. 1. Окно программы компоновщика

3. Загрузка исполняемого модуля в симулятор:

– в пункте File оконного меню симулятора выбрать команду *Load Program…*;

– в одноимённом окне диалога открыть список поля *Папка* и на дереве папок найти и открыть рабочую папку;

– в поле *Имя* того же окна диалога выделить имя исполняемого модуля (здесь lb1.out), после чего с помощью компьютерной мыши нажать кнопку *Открыть*.

В окне Command симулятора появиться сообщение о факте загрузки отлаживаемого файла.

4. Прогон программы.

В режиме отладки программ симулятор обеспечивает два основных способа их прогона – по контрольным точкам (точкам останова выполнения программы) и пошаговый.

В первом случае выполнение каждого участка программы (между двумя соседними контрольными точками) инициируется командой *Run* из пункта Target оконного меню либо одноимённой кнопкой панели инструментов окна симулятора (рис. 3), либо клавишей <F5> клавиатуры.

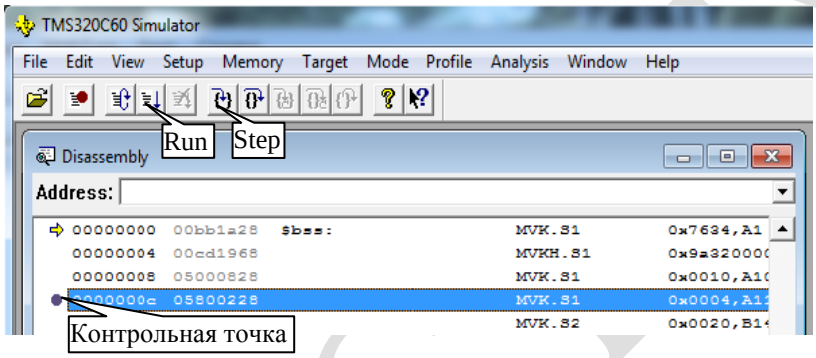


Рис. 3. Способы прогона программы

Устанавливаются контрольные точки в окне Disassembly щелчком компьютерной мыши слева от требуемой строки (рис. 3). Снимается контрольная точка щелчком мыши по её изображению.

Для апробации данного режима прогона установите контрольную точку на строке с адресом 00000004 и инициируйте команду *Run*. Снимите контрольную точку и инициируйте команду *Restart* (найти и компьютерной мышью «щелкнуть» соответствующую кнопку панели инструментов окна симулятора).

Во втором случае выполнение каждой командной строки инициируется командой *Step* из пункта Target оконного меню либо одноимённой кнопкой панели инструментов окна симулятора (см. рис. 3), либо клавишей <F8> клавиатуры.

В процессе отладки контролируются следующие окна:

Disassembly, где исполняемая на следующем шаге командная строка автоматически отмечается слева стрелкой;

CPU, содержимое регистров которого сравнивается с прогнозом выполнения текущей командной строки.

Для апробации выполните в пошаговом режиме модуль lb1.out, отмечая изменения в окне CPU симулятора.

5. Завершение работы с симулятором.

Закрыть симулятор можно любым из трех способов:

− с помощью кнопки «×», расположенной в правом верхнем углу строки заголовка окна симулятора;

− ввести в командную строку окна Command команду quit;

− в пункте оконного меню File выбрать команду *Exit*.

Опробовать все способы и выбрать для себя наиболее удобный способ.

2. ОПЕРАЦИИ ПЕРЕСЫЛКИ ДАННЫХ

Пересылка данных включает операции ввода в процессор (а именно, в его РОН) исходных данных, обмена данными между РОН (копирование содержимого одного регистра в другой), а также между РОН и памятью данных процессора.

Цель работы

Изучить особенности команд пересылки данных ассемблера процессора TMS320C6x.

Подготовка к работе

1. По указанной выше литературе изучить:

− структуру командной строки ассемблера процессора TMS320C6x;

− процесс выполнения программы процессором TMS320C6x;

− методы адресации операндов;

− форматы команд пересылки данных (загрузи/хранения, ввода исходных данных и межрегистровой пересылки) ассемблера TMS320C6x и особенности их выполнения.

2. Подготовить отчет (стр. 3 – 4) в соответствии с первым заданием работы.

Контрольные вопросы

1. По порядку слева направо назовите поля командной строки ассемблера процессора TMS320C6x.
2. Сформулируйте правило заполнения поля модуля строки ассемблера TMS320C6x.
3. Каким символом разделяются объекты поля операндов строки ассемблера TMS320C6x?
4. Покажите формат поля операндов команды пересылки данных, заданной преподавателем.
5. Сформулируйте правило заполнения поля модуля строки ассемблера TMS320C6x.
6. Назовите все объекты, допустимые к указанию в поле команды строки ассемблера TMS320C6x.

7. Назовите значения третьей буквы в мнемонике команд загрузки/хранения ассемблера TMS320C6x.

8. Назовите значение четвёртой буквы в мнемонике команд загрузки языка ассемблера TMS320C6x.

9. Сформулируйте понятие адресации, адресного кода и исполнительного адреса.

10. Назовите методы адресации, поддерживаемые командой пересылки данных, заданной преподавателем.

11. Сформулируйте особенности методов базирования и индексации.

12. Сформулируйте ограничения на использование модулей .D процессора TMS320C6x.

13. Сформулируйте правила выполнения команды ввода исходных данных, заданной преподавателем.

14. Сформулируйте правило ввода в РОН константы, превышающей полуслово.

15. Как представляются знаковые константы в поле операндов команд ввода исходных данных ассемблера TMS320C6x?

16. Приведите пример команды загрузки/хранения ассемблера TMS320C6x с использованием метода адресации, заданного преподавателем.

17. Сформулируйте правило размещения результата выполнения команды пересылки данных, заданной преподавателем.

18. Сформулируйте ограничения, связанные со смещением в командах загрузки/хранения ассемблера TMS320C6x.

Задания и методические указания к их выполнению

1. На языке ассемблера TMS320C6x подготовить программу, соответствующую следующей последовательности операций.

Операция 1. В регистр R1 (назначить из РОН по своему усмотрению) ввести число, выбранное из табл. 1 в соответствии с номером варианта V, равному номеру N студента в списке группы.

Таблица 1. Исходный операнд

V	Число, Hex	V	Число, Hex	V	Число, Hex	V	Число, Hex
1	80A1F5C1	9	A123F1C0	17	C345A5B7	25	E456F792
2	A90C7D5	10	C70A4B2	18	E34F5B1	26	8E2C795
3	C5D0A5	11	E4C6A0	19	9AC580	27	C1A293
4	AF9C5	12	CD3A0	20	EC7A6	28	890A7
5	90B3E8C9	13	B23495A0	21	D046E890	29	F598C4E5
6	B30B5A8	14	D89E5F0	22	F67C3E4	30	9F6B287
7	D9A2B3	15	F7B0D3	23	B9B483	31	D0E184
8	BE7C4	16	DA0B9	24	FB1D0	32	9A3CB

Для ввода исходных данных используются команды MVK, MVKH и MVKLH. При этом если:

– число не превышает полуслова (от -32767 до +32767) достаточно одной

команды MVK. При этом следует помнить, что команда выполняется с расширением знаком;

– число превышает полуслово, необходимы две команды: сначала MVK, а затем MVKH или MVKLH. Последняя команда обязательна при 16-ричном представлении операнда и буквой в старшем его разряде.

Операция 2. В регистр R2 общего назначения (задать в соответствии с последующими заданиями работы и отличающимся от R1) ввести число 50h в качестве базового адреса ячейки ПД процессора.

Операция 3. Сохранить содержимое R1 в ПД процессора в соответствии с условиями, выбранными из следующей таблицы по номеру варианта V = N:

V	**Условия сохранения**
1	**2**
1	В ячейке памяти 48h с изменением содержимого R2.
2	В ячейке памяти 50h без изменения содержимого R2.
3	В ячейке памяти 50h с изменением содержимого R2 до величины 4Ch.
4	В ячейке памяти F4h с изменением содержимого R2.
1	**2**
5	В ячейке памяти 4Ch без изменения содержимого R2.
6	В ячейке памяти 50h с изменением содержимого R2 до величины 51h.
7	В ячейке памяти 44h с изменением содержимого R2.
8	В ячейке памяти 51h.
9	В ячейке памяти 50h с изменением содержимого R2 до величины F0h.
10	В ячейке памяти 60h с изменением содержимого R2.
11	В ячейке памяти 54h без изменения содержимого R2.
12	В ячейке памяти 50h с изменением содержимого R2 до величины 64h.
13	В ячейке памяти 38h с изменением содержимого R2.
14	В ячейке памяти 51h.
15	В ячейке памяти 50h с изменением содержимого R2 до величины 4Fh.
16	В ячейке памяти 70h с изменением содержимого R2.
17	В ячейке памяти 4Fh.
18	В ячейке памяти 34h без изменения содержимого R2.
19	В ячейке памяти 58h с изменением содержимого R2.
20	В ячейке памяти 50h с изменением содержимого R2 до величины 4Fh.
21	В ячейке памяти 50h с изменением содержимого R2 до величины 54h.
22	В ячейке памяти 40h с изменением содержимого R2.
23	В ячейке памяти 51h.

24	В ячейке памяти 50h без изменения содержимого R2.
25	В ячейке памяти 54h с изменением содержимого R2.
26	В ячейке памяти 4Fh.
27	В ячейке памяти 50h с изменением содержимого R2 до величины 51h.
28	В ячейке памяти E0h с изменением содержимого R2.
29	В ячейке памяти 5Ch без изменения содержимого R2.
30	В ячейке памяти 28h с изменением содержимого R2.
1	**2**
31	В ячейке памяти 50h с изменением содержимого R2 до величины 3Ch.
32	В ячейке памяти 6Ch с изменением содержимого R2.

Поскольку речь идет о сохранении всего содержимого регистра R1, следует воспользоваться командой STW R1,*$A_к$, где $A_к$ – адресный код. При формировании $A_к$ в зависимости от целей дальнейшего использования регистра базового адреса (базы) R2 применяется один из следующих типов адресации:

– косвенная, если R2 содержит требуемый исполнительный адрес Аи и изменять содержимое R2 не следует;

– базирование, если R2 не содержит требуемый $A_и$ и изменять содержимое R2 не следует;

– преиндексация, если R2 не содержит требуемый $A_и$ и после операции пересылки необходимо заменить содержимое R2 на $A_и$;

– постиндексация, если R2 содержит требуемый $A_и$, но после операции пересылки можно или нужно заменить содержимое R2 на $A_и$;

– преавтоинкремент или преавтодекремент, если содержимое R2 на 1 меньше или, соответственно, больше требуемого $A_и$ и после операции пересылки содержимое R2 необходимо заменить на $A_и$;

– поставтоинкремент или поставтодекремент, если R2 содержит требуемый $A_и$, но после операции пересылки можно или нужно изменить содержимое R2 на 1.

Внимание! При составлении и прогоне программы учитывайте особенности симулятора:

– не поддерживает автоиндексацию. Вместо этого он автоматически устанавливает смещение, равное 1;

– если смещение превышает 31, следует предварительно ввести его в любой из свободных РОН и указать смещение именем этого регистра. При этом в качестве регистра базы можно использовать любой РОН.

Операция 4. Из ячейки ПД процессора, используемой в предыдущем пункте, загрузить в регистр R3 (назначить из РОН по своему усмотрению, но отличающимся от R1 и R2) число в соответствии с условиями, выбранными из следующей таблицы по номеру варианта V = N:

V	Условия загрузки
1	**2**
1	Полуслово с расширением знаком и без изменения содержимого R2**.**
2	Байт с расширением знаком и увеличением содержимого R2 на 1.
3	Полуслово с расширением знаком и изменением содержимого R2 до величины 52h.
4	Байт без расширения знаком и с уменьшением содержимого R2 на величину Ah.
5	Полуслово без расширения знаком и изменения содержимого R2.
6	Байт с расширением знаком и без изменения содержимого R2.
7	Полуслово без расширения знаком и с увеличением содержимого R2 на величину Eh.
8	Байт без расширения знаком и изменения содержимого R2.
9	Полуслово без расширения знаком и изменения содержимого R2.
10	Байт с расширением знаком и уменьшением содержимого R2 на величину 12h.
11	Полуслово без расширения знаком и с увеличением содержимого R2 на величину 58h.
12	Полуслово с расширением знаком и без изменения содержимого R2.
13	Байт без расширения знаком и с увеличением содержимого R2 на величину 22h.
14	Байт с расширением знаком и уменьшением содержимого R2 на величину 1Ah.
15	Полуслово без расширения знаком и изменения содержимого R2.
16	Байт с расширением знаком и без изменения содержимого R2.
17	Полуслово без расширения знаком и с увеличением содержимого R2 на величину 10h.
18	Полуслово с расширением знаком и изменением содержимого R2 до величины 76h.
19	Полуслово без расширения знаком и с увеличением содержимого R2 на величину 54h.
20	Байт без расширения знаком и изменения содержимого R2.
21	Байт с расширением знаком и изменением содержимого R2 до величины 53h.

1	2
22	Полуслово с расширением знаком и уменьшением содержимого R2 на 6.
23	Байт без расширения знаком и с увеличением содержимого R2 на величину 26h.
24	Байт с расширением знаком и увеличением содержимого R2 на величину 2Eh.
25	Полуслово без расширения знаком и изменения содержимого R2.
26	Полуслово с расширением знаком и увеличением содержимого R2 на величину 42h.
27	Байт с расширением знаком и изменением содержимого R2 до величины 58h.
28	Полуслово без расширения знаком и с уменьшением содержимого R2 на 1.
29	Байт без расширения знаком и изменения содержимого R2.
30	Полуслово с расширением знаком и увеличением содержимого R2 на 1.
1	**2**
31	Полуслово без расширения знаком и изменения содержимого R2.
32	Байт с расширением знаком и уменьшением содержимого R2 на 1.

Операции загрузки в зависимости от объёма данных реализуются посредством команд LDW(H, B) *$A_к$,r, где r – имя регистра-приемника операнда. При этом следует помнить:

– третья буква мнемоники помимо объёма пересылаемых данных определяет закономерность изменения величины смещения;

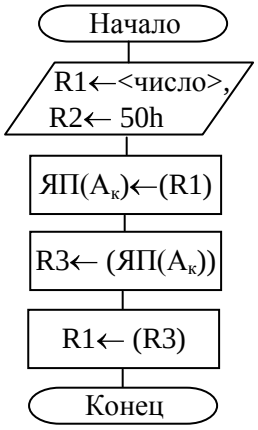
– команды LDH и LDB выполняются с расширением знаком пересылаемой части слова. При необходимости расширения нулем следует применять команды LDW(H, B)U;

– команды загрузки имеют 4 слота задержки, то есть результат доступен для использования только спустя 4 такта после объявления команды. Так, если команда объявлена в n-ом такте, результат ее выполнения сформируется в (n+4)-ом такте, а использовать его можно, начиная только с (n+5)-го такта. Таким образом, в данной работе после команды загрузки необходимо объявить 4-тактный мультицикл NOP (нет операции): NOP 4.

Операция 5. Переслать (скопировать) содержимое R3 в R1.

В соответствии с заданием обобщённые алгоритм и программа (без указания конкретных чисел, имен регистров РОН, номера ячейки ПД (ЯП) и адресного кода $A_к$) представлены на рис. 4, где (Z) – содержимое объекта Z (регистра РОН или ячейки памяти данных), а символ ← обозначает операцию

пересылки данных в требуемом направлении.




```
mvk             <число>,R1 ; ввод
mvkh(mvklh)     <число>,R1 ; исходных
……                         ; данных
stw             R1,*Aк
ldw(h,b)(u)     *Aк,R3
nop             4
mv              R3,R1
```

Рис. 4. Блок-схема алгоритма и программа в обобщённом виде

2. Получить исполняемый программный модуль (см. стр. 7 – 8).

3. Загрузить исполняемый модуль в симулятор (см. стр. 9).

4. В пошаговом режиме выполнить прогон программы (см. стр. 10), сравнивая данные прогноза с соответствующими данными окна CPU симулятора.

5. Предъявить результаты выполнения работы преподавателю, после чего завершить работу с симулятором (см. стр. 10).

3. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

Функциональные устройства (модули) ядра процессора способны вычислять суммы, разности и произведения операндов как знаковых, так и без знака.

Цель работы

Изучить особенности команд пересылки данных ассемблера процессора TMS320C6x.

Подготовка к работе

1. По указанной выше литературе изучить форматы и особенности выполнения арифметических команд ассемблера TMS320C6x.
2. Подготовить отчет (стр. 3 – 4) в соответствии с первым заданием работы.

Контрольные вопросы

1. Сформулируйте правило формирования результата при выполнении команды вычитания над знаковыми операндами.
2. Сформулируйте правило формирования результата при выполнении команды ABS.
3. Приведите формат арифметической команды, заданной преподавателем.
4. Укажите особенности команды ADD.
5. Укажите особенности команды ADDK.
6. Определите результат выполнения команды, заданной преподавателем.

7. Назовите метод адресации, характерный практически для всех арифметических команд.

8. Укажите особенности команды ADD2.

9. Укажите особенности команды ADDU.

10. Укажите особенности команд умножения.

Задания и методические указания к их выполнению

1. На языке ассемблера TMS320C6x подготовить программу, соответствующую следующей последовательности операций.

Операция 1 с размещением результата в регистре (регистровой паре) R1 (назначить из РОН по своему усмотрению) по номеру варианта V = N:

V	**Содержание и условия операции**
1	**2**
1	Умножение 16×16 младших бит чисел 109A00h и C0081h, где операнды знаковые числа.
2	Сложение знаковых чисел 9C5A8600h и 78B05D03h.
3	Умножение 16×16 младших бит чисел 109A00h и C0081h, где операнды – числа без знака.
4	Вычитание над числами без знака: 78B05D03h - 9C5A8600h.
5	Умножение 16×16 младших бит чисел 9A000h и 7C000h, где первый операнд – число без знака, второй – со знаком.
6	Сложение знаковых чисел 9C5A8600h и A8B05D03h.
7	Умножение 16×16 младших бит чисел 9A000h и 7C000h, где первый операнд – число со знаком, второй – без знака.
8	Вычитание над знаковыми числами: 78B05D03h - 9C5A8600h.
9	Умножение 16×16 старших бит чисел 7100A000h и 8100C000h, где операнды – знаковые числа.
10	Сложение 9C5A8600h с 5-разрядной положительной константой (выбрать по своему усмотрению).
11	Умножение 16×16 старших бит чисел 7100A000h и 8100C000h, где операнды – числа без знака.
12	Вычитание над знаковыми числами: 9C5A8600h - 78B05D03h.
13	Умножение 16×16 старших бит чисел 9100A000h и 8100C000h, где первый операнд – число без знака, второй – со знаком.
14	Сложение 5C5A8600h с 5-разрядной константой без знака (выбрать по своему усмотрению).
15	Умножение 16×16 старших бит чисел 9100A000h и 8100C000h, где первый операнд – число со знаком, второй – без знака.
16	Вычитание над знаковыми числами: BC5A8600h - 98B05D03h.
17	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где операнды – знаковые числа.
18	Сложение чисел 4C5A8600h и 78B05D03h без знака.

1	2
19	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где операнды – числа без знака.
20	Вычитание над знаковыми числами: 7C5A8600h - 48B05D03h.
21	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где первый операнд – число без знака, второй – со знаком.
22	Вычитание над знаковыми числами: 3C5A8600h - A8B05D03h.
23	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где первый операнд – число со знаком, второй – без знака.
24	Вычитание над знаковыми числами: AC5A8600h - 38B05D03h.
25	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где операнды – знаковые числа.
26	Сложение чисел EC5A8600h + D8B05D03h без знака
27	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где операнды – числа без знака.
28	Вычитание над знаковыми числами: 9C5A8600h - B8B05D03h.
29	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где первый операнд – число без знака, второй – со знаком.
30	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где первый операнд – число со знаком, второй – без знака.

Операция 2. Вычислить абсолютную величину содержимого R1 (для регистровой пары использовать только четный регистр) с размещением результата в регистре R2 (назначить по своему усмотрению).

Операция 3 с размещением результата в регистре R2:

– для нечетных V = N содержимое R2 сложить с 16-разрядной знаковой константой (принять по своему усмотрению);

– для четных V = N сложить старшую и младшую половины R2 с, соответственно, старшей и младшей половинами числа, выбранного из таблицы:

V	**Число, Hex**	**V**	**Число, Hex**	**V**	**Число, Hex**
1	**2**	**3**	**4**	**5**	**6**
1	A90C7D5	11	CD3A0	21	F67C3E4
2	80A1F5C1	12	E4C6A0	22	D046E890
3	AF9C5	13	D89E5F0	23	FB1D0
4	C5D0A5	14	B23495A0	24	B9B483
5	B30B5A8	15	DA0B9	25	8E2C795
6	90B3E8C9	16	F7B0D3	26	E456F792

7	BE7C4	17	E34F5B1	27	890A7
8	D9A2B3	18	C345A5B7	28	C1A293
9	C70A4B2	19	EC7A6	29	9F6B287
10	A123F1C0	20	9AC580	30	F598C4E5

При выполнении данного задания следует:

- предусмотреть регистры R3, R4, … для размещения исходных данных;
- помнить, что команды умножения имеют 1 слот задержки.

В остальном алгоритм и программа строятся аналогично предыдущей работе (рис. 5).

Рис. 5. Блок-схема алгоритма в обобщённом виде

2. Получить исполняемый программный модуль (см. стр. 7 – 8).

3. Загрузить исполняемый модуль в симулятор (см. стр. 9).

4. В пошаговом режиме выполнить прогон программы (см. стр. 10), сравнивая данные прогноза с соответствующими данными окна CPU симулятора.

5. Предъявить результаты выполнения работы преподавателю, после чего завершить работу с симулятором (см. стр. 10).