Федеральное агентство связи

Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования

ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ

ЭЛЕКТРОННАЯ БИБЛИОТЕЧНАЯ СИСТЕМА

Самара

Федеральное государственное образовательное бюджетное учреждение
высшего профессионального образования
«Поволжский государственный университет телекоммуникаций
и информатики»

Стефанов А. М., Солодов А.Г.

МЕТОДИЧЕСКАЯ РАЗРАБОТКА
для выполнения лабораторных работ

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕРА ПРОЦЕССОРА TMS320C6x

Самара
2014

ОГЛАВЛЕНИЕ

Введение

Методическая разработка содержит 6 лабораторных работ, направленных на освоение основных приёмов программирования на языке ассемблера сигнального процессора TMS320C62x и отладки соответствующих программ.

Методическая разработка может использоваться на лабораторных и практических занятиях по дисциплинам «Вычислительная техника и информационные технологии» и «Цифровые устройства и микропроцессоры» для студентов телекоммуникационных 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), а также указанием номера и содержимого используемой ячейки памяти данных (ЯПД) процессора в 16-ричной системе счисления:

Заголовок таблицы результатов выполнения лабораторной работы

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

Данная таблица предъявляется преподавателю до прогона программы с целью выявления возможных методических ошибок и получения указаний по адаптации программы к особенностям симулятора команд.

Исправленные в процессе отладки фрагменты исходной программы, заносятся в таблицу дополнительными строками. По завершении отладки окончательная таблица вновь предъявляется преподавателю с устными пояснениями исправлений.

Систематизация результатов выполнения работ

На любом доступном диске создать рабочую папку группы с соответствующим именем, например МС-01, в котором используются **только латинские** буквы.

Для удобства хранения результатов выполнения лабораторных работ в папке группы (например, МС-01) создать дерево папок (рис. 1), где имена папок и содержащиеся в них файлы включают номер студента в списке группы (обозначен символом N) и прописываются **только латинскими буквами.**

В дальнейшем N используется в качестве номера варианта задания.

С целью удобства работы с симулятором команд из папки C:\C6XTOOLS\CODEGEN.110\BIN скопировать в рабочую папку студента <Папка группы>\<Familiy_N> файлы ASM6x.exe и LNK6x.exe (см. рис. 1). При выполнении текущей лабораторной работы все необходимые файлы сначала размещаются в рабочей папке < Familiy_N>, а после завершения отладки **перемещаются** в папку соответствующей лабораторной работы (LB<№ работы>_N).

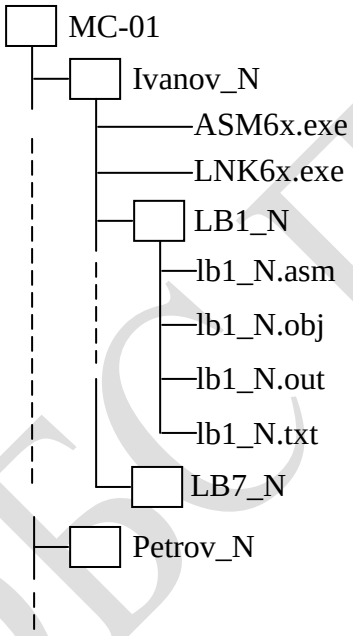


Рис. 1. Организация хранения результатов выполнения лабораторных работ

1. ЗНАКОМСТВО С СИМУЛЯТОРОМ 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.

Задание и порядок выполнения работы

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 по умолчанию дублирует первые две слева колонки окна Disassembly. Здесь также 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. Создать исполняемый программный модуль.

Исполняемый модуль непосредственно загружается в симулятор. Получается он 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 < Familiy_N> под именем lb1_N.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 свою рабочую папку;

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

– в поле *Имя файла* ввести полное имя файла (здесь lb1_N.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 открывшемся окне ввести имя ассемблируемого файла, причем расширение имени указывать не обязательно (в данном случае достаточно ввести lb1_N). После этого нажать клавишу <Enter>.

Отсутствие в рабочей папке объектного файла означает, что в исходном файле (здесь lb1_N.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_N);

– нажать клавишу <Enter>.

В результате в рабочей папке образуется файл листинга (здесь lb1_N.lst), содержащий сведения об ошибках.

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

4). Получить исполняемый файл (здесь lb1_N.out), для чего из рабочей папки запустить программу компоновщика (файл LNK6x.exe) и в открывшемся окне (рис. 2) 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_N.out), предлагаемым компоновщиком (см. рис. 2).

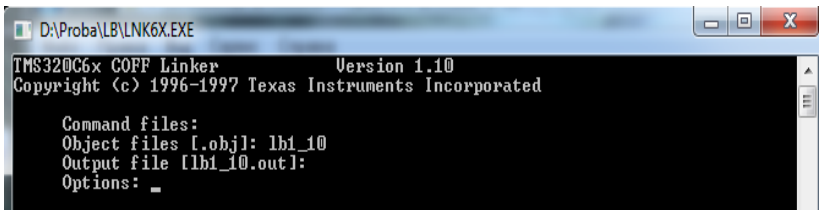


Рис. 2. Окно программы компоновщика

3. Загрузить исполняемый модуль 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_N.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.

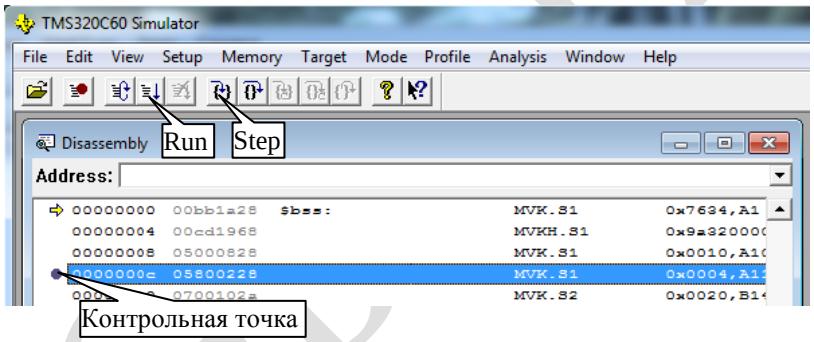


Рис. 3. Способы прогона программы

Устанавливаются контрольные точки в окне Disassembly щелчком компьютерной мыши слева от требуемой строки (см. 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_N.out, отмечая изменения в окне CPU симулятора.

5. Завершить работу 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 из них.

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.

2. ПЕРЕСЫЛКА ДАННЫХ

Пересылка данных включает операции ввода исходных данных в РОН процессора, 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, сохранения в память и загрузки из памяти 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 (между РОН, загрузи/хранения и ввода исходных данных) ассемблера TMS320C6x и особенности их выполнения.

2. Ознакомиться с методическими указаниями
3. Подготовить отчет (стр. 3 – 4).

Задание и порядок выполнения работы

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 N (№ студента в списке группы).

Таблица 1. Исходный операнд

N	Число, Hex	N	Число, Hex	N	Число, Hex	N	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

Операция 2. В регистр R2 (назначить из РОН по своему усмотрению, но с учетом последующих заданий работы и R2≠R1) ввести число 50h в качестве базового адреса памяти данных (ПД) процессора.

Операция 3. Сохранить содержимое R1 в ПД процессора в соответствии с условиями из таблицы 2.

Таблица 2. Условия сохранения операнда

N	Условия сохранения
1	В ячейке памяти F4h с изменением содержимого R2.
2	В ячейке памяти 50h с изменением содержимого R2 до величины 4Fh.
3	В ячейке памяти 50h с изменением содержимого R2 до величины 64h.
4	В ячейке памяти 4Ch без изменения содержимого R2.
5	В ячейке памяти 60h с изменением содержимого R2.
6	В ячейке памяти 50h с изменением содержимого R2 до величины 4Ch.
7	В ячейке памяти 70h без изменения содержимого R2.
8	В ячейке памяти 50h с изменением содержимого R2 до величины F0h.
9	В ячейке памяти 50h без изменения содержимого R2.
10	В ячейке памяти 48h с изменением содержимого R2.
11	В ячейке памяти 54h без изменения содержимого R2.
12	В ячейке памяти 30h с изменением содержимого R2.
13	В ячейке памяти 50h с изменением содержимого R2 до величины 54h.
14	В ячейке памяти 34h без изменения содержимого R2.
15	В ячейке памяти 58h с изменением содержимого R2.
16	В ячейке памяти 51h.
17	В ячейке памяти 50h с изменением содержимого R2 до величины 3Ch.
18	В ячейке памяти 80h без изменения содержимого R2.
19	В ячейке памяти 50h с изменением содержимого R2 до величины 60h.
20	В ячейке памяти 38h с изменением содержимого R2.
21	В ячейке памяти 5Ch без изменения содержимого R2.
22	В ячейке памяти 20h с изменением содержимого R2.
23	В ячейке памяти 4Fh.
24	В ячейке памяти 50h с изменением содержимого R2 до величины 68h.
25	В ячейке памяти 40h без изменения содержимого R2.
26	В ячейке памяти 6Ah с изменением содержимого R2.
27	В ячейке памяти 50h с изменением содержимого R2 до величины 48h.
28	В ячейке памяти 78h без изменения содержимого R2.
29	В ячейке памяти 50h с изменением содержимого R2 до величины 90h.
30	В ячейке памяти 50h с изменением содержимого R2 до величины 51h.
31	В ячейке памяти 40h с изменением содержимого R2.
32	В ячейке памяти 58h без изменения содержимого R2.

Операция 4. Из ячейки ПД процессора, используемой в предыдущем пункте, загрузить в регистр R3 (назначить из РОН по своему усмотрению и R3≠R1, R3≠R2) число в соответствии с условиями, выбранными из таблицы 3.

Таблица 3. Условия загрузки числа в РОН

N	Условия загрузки
1	Полуслово с расширением знаком и без изменения содержимого R2**.**
2	Байт без расширения знаком и изменения содержимого R2.
3	Полуслово без расширения знаком и с изменением содержимого R2**.**
4	Байт с расширением знаком и изменением содержимого R2.
5	Полуслово с расширением знаком и изменением содержимого R2 до величины 52h.
6	Байт без расширения знаком и изменения содержимого R2.
7	Полуслово без расширения знаком и с изменением содержимого R2**.**
8	Байт без расширения знаком и с изменением содержимого R2**.**
9	Полуслово с расширением знаком и изменением содержимого R2 до величины 76h.
10	Байт с расширением знаком и без изменения содержимого R2.
11	Полуслово без расширения знаком и с изменением содержимого R2**.**
12	Байт без расширения знаком и с уменьшением содержимого R2 на 1.
13	Полуслово с расширением знаком и без изменения содержимого R2**.**
14	Байт с расширением знаком и изменением содержимого R2.
15	Полуслово без расширения знаком и с увеличением содержимого R2 на 1.
16	Байт без расширения знаком и изменения содержимого R2.
17	Полуслово с расширением знаком и без изменения содержимого R2**.**
18	Байт с расширением знаком и изменением содержимого R2.
19	Полуслово без расширения знаком и изменения содержимого R2.
20	Байт без расширения знаком и с уменьшением содержимого R2 на 1.
21	Полуслово с расширением знаком и изменением содержимого R2.
22	Байт с расширением знаком и уменьшением содержимого R2 на величину 12h.
23	Полуслово без расширения знаком и с увеличением содержимого R2 на величину Eh.
24	Байт без расширения знаком и изменения содержимого R2.
25	Полуслово с расширением знаком и изменением содержимого R2.
26	Байт с расширением знаком и без изменения содержимого R2.
27	Полуслово без расширения знаком и изменения содержимого R2.
28	Байт без расширения знаком и с изменением содержимого R2**.**
29	Полуслово с расширением знаком и изменением содержимого R2.
30	Байт с расширением знаком и без изменения содержимого R2.
31	Полуслово без расширения знаком и изменения содержимого R2.
32	Байт без расширения знаком и с изменением содержимого R2**.**

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

2. Получить исполняемый программный модуль (см. стр. 8 – 9).

3. Загрузить исполняемый модуль в симулятор (см. стр. 10).

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).

Методические указания

Операция 1. 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 эта 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 3. 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 ($A_и$) 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.

Операция 4. Операции загрузки в зависимости от объёма данных реализуются посредством команд:

LDW(H, B) *Ак,r,

где r – имя регистра-приемника операнда. При этом следует помнить:

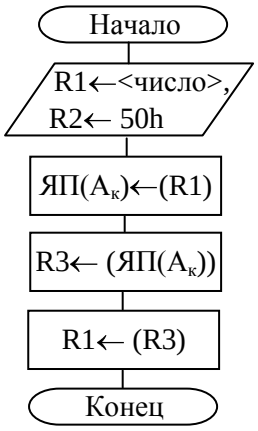
– третья буква мнемоники помимо объёма пересылаемых данных определяет закономерность изменения величины смещения;

– команды LDH и LDB выполняются с расширением знаком пересылаемой части слова. При необходимости расширения нулем следует применять коман-

ды 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.

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 и программа в обобщённом виде

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 функциональные особенности команды, заданной преподавателем.
5. Укажите ограничения на операнды команды, заданной преподавателем.
6. Определите результат выполнения команды, заданной преподавателем.
7. Назовите метод адресации, характерный практически для всех арифметических команд.

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 арифметических команд ассемблера процессора TMS320C6x.

Подготовка к работе

1. По указанной выше литературе изучить форматы и особенности выполнения арифметических команд ассемблера TMS320C6x.
2. Ознакомиться с методическими указаниями
3. Подготовить отчет (см. стр. 3 – 4).

Задание и порядок выполнения работы

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

Операция 1. Выбирается из таблицы 4 по номеру варианта N с размещением результата в регистре (регистровой паре) R1 (назначить из РОН по своему усмотрению).

Таблица 4. Вариант операции.

N	Содержание операции
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.

N	Содержание операции
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, операнды со знаком.
18	Сложение чисел 4C5A8600h и 78B05D03h без знака.
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 вариантов N содержимое R2 сложить с 16-разрядной знаковой константой (принять по своему усмотрению);

– для четных вариантов N сложить старшую и младшую половины R2 соответственно со старшей и младшей половинами числа, выбранного из таблицы 5.

Таблица 5. Второй операнд

N	Число, Hex	N	Число, Hex	N	Число, Hex
2	80A1F5C1	12	E4C6A0	22	D046E890
4	C5D0A5	14	B23495A0	24	B9B483
6	90B3E8C9	16	F7B0D3	26	E456F792
8	D9A2B3	18	C345A5B7	28	C1A293
10	A123F1C0	20	9AC580	30	F598C4E5

2. Получить исполняемый программный модуль (см. стр. 8 – 9).

3. Загрузить исполняемый модуль в симулятор (см. стр. 10).

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).

Методические указания

Примечание: при выполнении некоторых команд ассемблера происходит дополнение старших разрядов знаком и если число отрицательное в этих разрядах появятся единицы, что может привести к неверному результату.

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

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 функциональные особенности команды, заданной преподавателем.

5. Укажите ограничения на операнды команды, заданной преподавателем.

6. Определите результат выполнения команды, заданной преподавателем.

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

4. ВЕТВЛЕНИЕ С ПРОСТЫМ УСЛОВИЕМ

Ветвления позволяют реализовать альтернативные вычисления в зависимости от каких-либо условий. Наиболее просто ветвление организуется для одного условия с одним отношением, например a<b.

Цель работы

Изучить особенности реализации простых условий на языке ассемблера процессора TMS320C6x.

Подготовка к работе

1. По указанной выше литературе изучить поле условия строки ассемблера TMS320C6x, а также форматы и особенности выполнения логических и сервисных команд.

2. Уяснить особенности реализации нестрогих отношений в условии ветвления.

3. Ознакомиться с методическими указаниями.

4. Подготовить отчет (см. стр. 3 – 4).

Задание и порядок выполнения работы

1. На языке ассемблера TMS320C6x подготовить программу, реализующую алгоритм рис. 6.

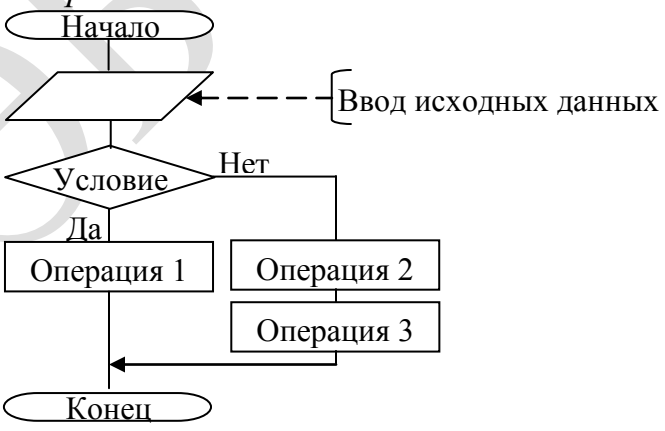


Рис. 6. Алгоритмическая структура «ветвление»

При этом:

– операция 1 заключается в вычислении абсолютной величины числа, взятого из таблицы 6.

Таблица 6. Операнд

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

–операцию 2 с размещением результата в регистре (регистровой паре) R1 (назначить из РОН по своему усмотрению) выбрать из таблицы 7.

Таблица 7. Операция

N	Содержание и условия операции
1	Вычитание знаковых чисел 9C5A8600h - B8B05D03h.
2	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где первый операнд – число без знака, второй – со знаком.
3	Сложение чисел EC5A8600h + D8B05D03h без знака.
4	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где операнды числа без знака.
5	Вычитание знаковых чисел AC5A8600h - 38B05D03h.
6	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где операнды знаковые числа.
7	Вычитание знаковых чисел 3C5A8600h - A8B05D03h.
8	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где первый операнд – число со знаком, второй – без знака.
9	Вычитание знаковых чисел 7C5A8600h - 48B05D03h.
10	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где первый операнд – число без знака, второй – со знаком.
11	Сложение чисел 4C5A8600h и 78B05D03h без знака.
12	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где операнды числа без знака.
13	Вычитание знаковых чисел BC5A8600h - 98B05D03h.
14	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где операнды знаковые числа.
15	Сложение 5C5A8600h с 5-разрядной константой без знака (выбрать по своему усмотрению).
16	Умножение 16×16 старших бит чисел 9100A000h и 8100C000h, где первый операнд – число со знаком, второй – без знака.
17	Вычитание знаковых чисел 9C5A8600h - 78B05D03h.
18	Умножение 16×16 старших бит чисел 9100A000h и 8100C000h, где первый операнд – число без знака, второй – со знаком.

N	Содержание и условия операции
19	Сложение 9C5A8600h с 5-разрядной положительной константой (выбрать по своему усмотрению).
20	Умножение 16×16 старших бит чисел 7100A000h и 8100C000h, где операнды числа без знака.
21	Вычитание знаковых чисел 78B05D03h - 9C5A8600h.
22	Умножение 16×16 старших бит чисел 7100A000h и 8100C000h, где операнды знаковые числа.
23	Сложение знаковых чисел 9C5A8600h и A8B05D03h.
24	Умножение 16×16 младших бит чисел 9A000h и 7C000h, где первый операнд – число со знаком, второй – без знака.
25	Вычитание чисел без знака 78B05D03h - 9C5A8600h.
26	Умножение 16×16 младших бит чисел 9A000h и 7C000h, где первый операнд – число без знака, второй – со знаком.
27	Сложение знаковых чисел 9C5A8600h и 78B05D03h.
28	Умножение 16×16 младших бит чисел 109A00h и C0081h, где операнды числа без знака.
29	Умножение 16×16 младших бит чисел 109A00h и C0081h, где операнды знаковые числа.
30	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где первый операнд – число со знаком, второй – без знака.

–операция 3 заключается в вычислении абсолютной величины содержимого R1 (для регистровой пары использовать только четный регистр) с размещением результата в регистре R2 (назначить из РОН по своему усмотрению);

–условие и способ формирования содержимого регистра условия R_{yc} (назначить из РОН по своему усмотрению) выбрать из таблицы 8.

Таблица 8. Условие

N	Условие ветвления	Способ формирования содержимого регистра условия R_{yc}
1	A8 ≥ A9	Сравнение чисел без знака
2	0 в пяти старших разрядах B8	Сброс области бит
3	A8 ≥ 0	Арифметический сдвиг вправо
4	B8 четное	Операция конъюнкции
5	A8 < A9	Сравнение знаковых чисел
6	B8 = B9	Операция «неравнозначность»
7	A8 ≥ 0	Выделение старшего бита без расширения знаком
8	0 в пяти старших разрядах B8	Логический сдвиг вправо
9	A8 > A9	Сравнение чисел без знака
10	B8 нечетное	Сброс области бит
11	A8 ≥ 0	Сравнение знаковых чисел
12	B8 ≠ B9	Операция «неравнозначность»
13	A8 ≤ A9	Сравнение знаковых чисел

N	Условие ветвления	Способ формирования содержимого регистра условия R_{yc}
14	0 в пяти старших разрядах B8	Выделение области бит с расширением знаком
15	A8 ≥ 0	Сброс области бит
16	B8 четное	Сдвиг влево
17	A8 ≥ A9	Сравнение знаковых чисел
18	В пяти старших разрядах B8 хотя бы одна 1	Выделение области бит без расширения знаком
19	A8 ≥ 0	Выделение области бит с расширением знаком
20	0 в пяти старших разрядах B8	Операция конъюнкции
21	A8 < A9	Сравнение чисел без знака
22	B8 нечетное	Выделение области бит без расширения знаком
23	A8 ≥ 0	Логический сдвиг вправо
24	B8 = B9	Сравнение чисел
25	A8 > A9	Сравнение знаковых чисел
26	1 в пяти старших разрядах B8	Сравнение чисел без знака
27	A8 ≥ 0	Операция конъюнкции
28	B8 четное	Выделение области бит с расширением знаком
29	A8 ≤ A9	Сравнение чисел без знака
30	B8 ≠ B9	Сравнение чисел

В таблице имена регистров РОН совпадают с их содержимым.

2. Получить исполняемый программный модуль (см. стр. 8 – 9).

3. Загрузить исполняемый модуль в симулятор (см. стр. 10).

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).

Методические указания

Алгоритмическая структура, приведенная на рис. 6, используется при программировании на языках высокого уровня. Однако в языке ассемблера, **во-первых**, возможны только два признака ветвления – содержимое регистра условия R_{yc} нулевое (выполняются операции одной ветви) или ненулевое (выполняются операции другой ветви). Для формирования этих признаков требуются дополнительные логические и/или сервисные операции, что должно быть отражено в алгоритмической структуре (рис. 7).

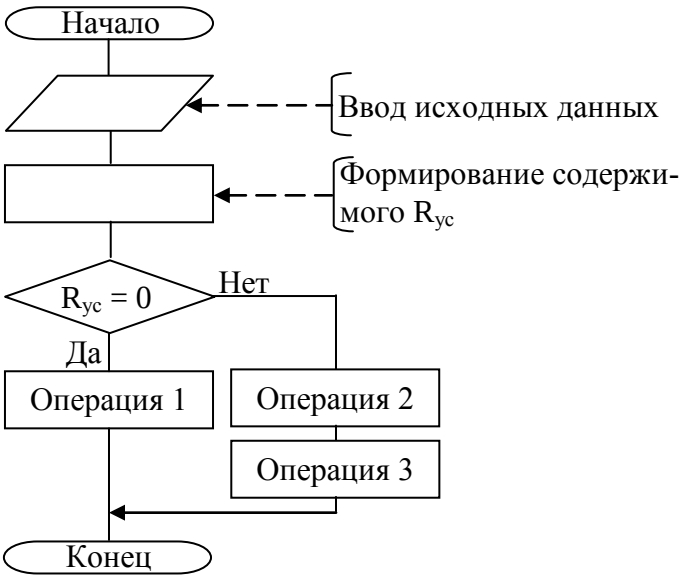


Рис. 7. Обобщенная структура ветвления, приведенная к ассемблеру

При этом для одного и того же исходного условия признаки ветвления могут быть получены разными способами. Например:

1. В случае отношений $a \geq 0$ либо $a < 0$ помимо команд сравнения (обсуждаются ниже) можно использовать знак числа (значение 31-го двоичного разряда). Для его выделения пригодны конъюнкция (команда AND) с числом 80000000h, выделение области из одного старшего бита (команда EXT или EXTU), сдвиг вправо на 31 разряд (команда SHR или SHRU), обнуление 31 младших разрядов (команда CLR).

2. Для определения четности числа можно использовать его конъюнкцию с числом 00000001h, выделение области из одного младшего бита, сдвиг влево на 31 разряд, обнуление 31 старших разрядов.

Указанная множественность решений отражена в разделе «Способ формирования содержимого регистра условия $R_{ус}$» таблицы задания, где для одного и того же условия в зависимости от варианта предлагаются различные команды его реализации.

Во-вторых, команды сравнения ассемблера реализуют только строгие отношения: $a < b$, $a > b$ и $a = b$. Поэтому в случае условий $a \leq b$, $a \geq b$ и $a \neq b$ следует использовать альтернативные условия: $a > b$, $a < b$ и $a = b$, соответственно. При этом признаки выполнения ветвей «Да» и «Нет» блока ветвления меняются местами. Например, некоторая операция должна выполняться, только если $a \leq b$ (признак «Да»). При использовании условия $a > b$ заданная операция должна выполняться только в случае его нарушения, то есть признак «Да» заменяется признаком «Нет» и наоборот.

В-третьих, все команды, соответствующие операциям той или иной ветви, должны быть условными. В противном случае команды одной из ветвей будут выполняться всегда, то есть вне зависимости от условия ветвления.

Обобщенное решение представленного задания:

Пример обобщенных алгоритма и программы (без указания конкретных дан-

ных) приведен на рис. 8.

При отладке требуется два прогона программы. В первом прогоне задается такое содержимое РОН, которое соответствует выполнению условия. Во втором прогоне содержимое одного из этих РОН изменяется так, чтобы условие нарушалось.

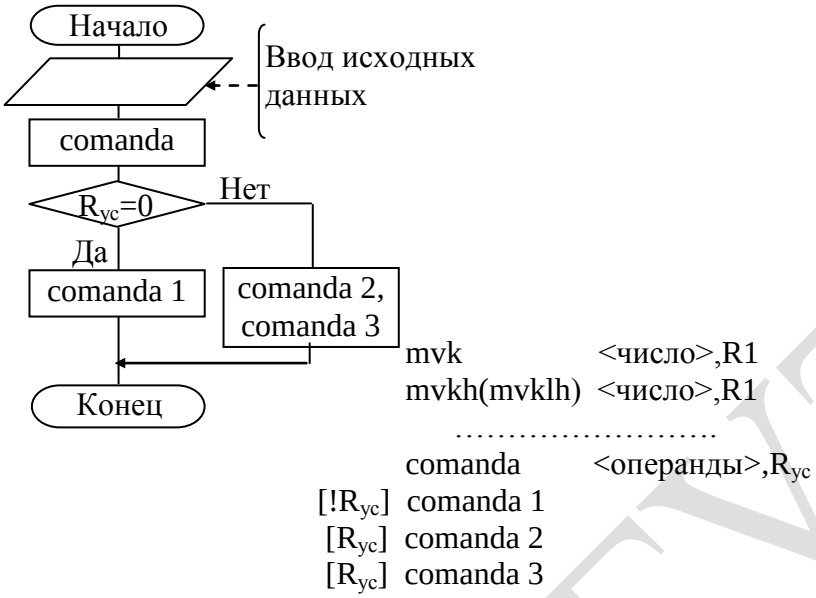


Рис. 8. Алгоритм и программа в обобщенном виде

Примечание: при выполнении некоторых команд ассемблера происходит дополнение старших разрядов знаком и если число отрицательное в этих разрядах появятся единицы, что может привести к неверному результату.

Контрольные вопросы

1. Укажите регистры РОН, допустимые для использования в качестве регистра условия.
2. Сформулируйте особенности реализации нестрогих отношений в условии ветвления.
3. Поясните общие принципы организации ветвлений при программировании на языке ассемблера.
4. С пояснениями приведите примеры поля условия строки ассемблера.
5. Приведите формат логической команды, заданной преподавателем.
6. Приведите формат сервисной команды, заданной преподавателем.
7. Определите результат выполнения логической команды, заданной преподавателем.
8. Определите результат выполнения сервисной команды, заданной преподавателем.
9. Приведите программную реализацию ветвления по любому другому варианту.

5. ВЕТВЛЕНИЕ СО СЛОЖНЫМ УСЛОВИЕМ

Сложное условие состоит из нескольких условий, объединенных логической связью: <условие1> **и** <условие2>, <условие1> **или** <условие2>, где каждое из условий может быть как простым, так и сложным.

Цель работы

Изучить особенности реализации сложных условий на языке ассемблера процессора TMS320C6x.

Подготовка к работе

1. По указанной выше литературе изучить принципы реализации сложных условий.

2. Ознакомиться с методическими указаниями.

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

Задание и порядок выполнения работы

1. На языке ассемблера TMS320C6x подготовить программу, реализующую алгоритм рис. 6.

При этом:

– операция 1 заключается в вычислении абсолютной величины числа, взятого из таблицы 9.

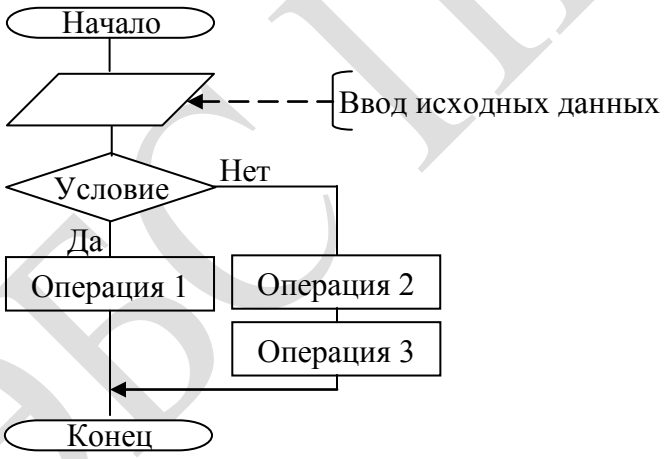


Рис. 6. Алгоритмическая структура «ветвление»

Таблица 9. Операнд

N	Число, Hex	N	Число, Hex	N	Число, Hex	N	Число, Hex
1	9F6B287	9	F67C3E4	17	D89E5F0	24	D9A2B3
2	F598C4E5	10	D046E890	18	B23495A0	25	B30B5A8
3	890A7	11	EC7A6	19	CD3A0	26	90B3E8C9
4	C1A293	12	9AC580	20	E4C6A0	27	AF9C5
5	8E2C795	13	E34F5B1	21	C70A4B2	28	C5D0A5
6	E456F792	14	C345A5B7	22	A123F1C0	29	A90C7D5

N	Число, Hex	N	Число, Hex	N	Число, Hex	N	Число, Hex
7	FB1D0	15	DA0B9	23	BE7C4	30	80A1F5C1
8	B9B483	16	F7B0D3				

–операцию 2 с размещением результата в регистре (регистровой паре) R1 (назначить из РОН по своему усмотрению) выбрать из таблицы 10.

Таблица 10. Операция

N	Содержание и условия операции
1	Вычитание знаковых чисел 9C5A8600h - B8B05D03h.
2	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где первый операнд – число без знака, второй – со знаком.
3	Сложение чисел EC5A8600h + D8B05D03h без знака.
4	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где операнды числа без знака.
5	Вычитание знаковых чисел AC5A8600h - 38B05D03h.
6	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где операнды знаковые числа.
7	Вычитание знаковых чисел 3C5A8600h - A8B05D03h.
8	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где первый операнд – число со знаком, второй – без знака.
9	Вычитание знаковых чисел 7C5A8600h - 48B05D03h.
10	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где первый операнд – число без знака, второй – со знаком.
11	Сложение чисел 4C5A8600h и 78B05D03h без знака.
12	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где операнды числа без знака.
13	Вычитание знаковых чисел BC5A8600h - 98B05D03h.
14	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где операнды знаковые числа.
15	Сложение 5C5A8600h с 5-разрядной константой без знака (выбрать по своему усмотрению).
16	Умножение 16×16 старших бит чисел 9100A000h и 8100C000h, где первый операнд – число со знаком, второй – без знака.
17	Вычитание знаковых чисел 9C5A8600h - 78B05D03h.
18	Умножение 16×16 старших бит чисел 9100A000h и 8100C000h, где первый операнд – число без знака, второй – со знаком.
19	Сложение 9C5A8600h с 5-разрядной положительной константой (выбрать по своему усмотрению).
20	Умножение 16×16 старших бит чисел 7100A000h и 8100C000h, где операнды числа без знака.
21	Вычитание знаковых чисел 78B05D03h - 9C5A8600h.

N	Содержание и условия операции
22	Умножение 16×16 старших бит чисел 7100A000h и 8100C000h, где операнды знаковые числа.
23	Сложение знаковых чисел 9C5A8600h и A8B05D03h.
24	Умножение 16×16 младших бит чисел 9A000h и 7C000h, где первый операнд – число со знаком, второй – без знака.
25	Вычитание чисел без знака 78B05D03h - 9C5A8600h.
26	Умножение 16×16 младших бит чисел 9A000h и 7C000h, где первый операнд – число без знака, второй – со знаком.
27	Сложение знаковых чисел 9C5A8600h и 78B05D03h.
28	Умножение 16×16 младших бит чисел 109A00h и C0081h, где операнды числа без знака.
29	Умножение 16×16 младших бит чисел 109A00h и C0081h, где операнды знаковые числа.
30	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где первый операнд – число со знаком, второй – без знака.

–операция 3 заключается в вычислении абсолютной величины содержимого R1 (для регистровой пары использовать только четный регистр) с размещением результата в регистре R2 (назначить из РОН по своему усмотрению);

–условие и способ формирования содержимого регистра условия R_{yc} (назначить из РОН по своему усмотрению) выбрать из таблицы 11.

Таблица 11. Условие формирования

N	Условие ветвления	Способ формирования содержимого регистра условия R_{yc}
1	A8 ≥ A9	Сравнение чисел без знака
2	0 в пяти старших разрядах B8	Сброс области бит
3	A8 ≥ 0	Арифметический сдвиг вправо
4	B8 четное	Операция конъюнкции
5	A8 < A9	Сравнение знаковых чисел
6	B8 = B9	Операция «неравнозначность»
7	A8 ≥ 0	Выделение старшего бита без расширения знаком
8	0 в пяти старших разрядах B8	Логический сдвиг вправо
9	A8 > A9	Сравнение чисел без знака
10	B8 нечетное	Сброс области бит
11	A8 ≥ 0	Сравнение знаковых чисел
12	B8 ≠ B9	Операция «неравнозначность»
13	A8 ≤ A9	Сравнение знаковых чисел
14	0 в пяти старших разрядах B8	Выделение области бит с расширением знаком

N	Условие ветвления	Способ формирования содержимого регистра условия R_{vc}
15	A8 ≥ 0	Сброс области бит
16	B8 четное	Сдвиг влево
17	A8 ≥ A9	Сравнение знаковых чисел
18	В пяти старших разрядах B8 хотя бы одна 1	Выделение области бит без расширения знаком
19	A8 ≥ 0	Выделение области бит с расширением знаком
20	0 в пяти старших разрядах B8	Операция конъюнкции
21	A8 < A9	Сравнение чисел без знака
22	B8 нечетное	Выделение области бит без расширения знаком
23	A8 ≥ 0	Логический сдвиг вправо
24	B8 = B9	Сравнение чисел
25	A8 > A9	Сравнение знаковых чисел
26	1 в пяти старших разрядах B8	Сравнение чисел без знака
27	A8 ≥ 0	Операция конъюнкции
28	B8 четное	Выделение области бит с расширением знаком
29	A8 ≤ A9	Сравнение чисел без знака
30	B8 ≠ B9	Сравнение чисел

В таблице имена регистров РОН совпадают с их содержимым.

2. Получить исполняемый программный модуль (см. стр. 8 – 9).

3. Загрузить исполняемый модуль в симулятор (см. стр. 10).

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).

Методические указания

Для решения задачи предположим, что сложное условие состоит из простых условий. Принципиального значения это не имеет, но облегчает разъяснение рассматриваемого вопроса.

В зависимости от простых условий сформировать признаки ветвления можно двумя способами:

− сначала с помощью однотипной логической операции над всеми простыми условиями получить признак ветвления в одном из разрядов результата, а затем выделить его в регистре условия (варианты 3, 5 – 13, 17, 19, 23 – 25, 27, 29, 30). В этом случае все соответствующие команды могут быть безусловны;

– последовательная проверка простых условий (варианты 1, 2, 4, 14 – 16, 18,20 – 22, 26, 28). При этом, во-первых, во всех проверках используется один и

тот же регистр условия. В противном случае возможно несанкционированное выполнение операции данной ветви. Во-вторых, соответствующие команды подбираются так, чтобы для одной и той же ветви результат их выполнения был одинаков. В-третьих, команды, реализующие проверку простых условий, должны быть условными, за исключением первой.

Так, например, для логической связи «ИЛИ» и n простых условиях алгоритмическая структура ветвления в обобщенном виде показана на рис. 9, а соответствующая программа – на рис. 10. При этом предполагается, что при выполнении простого условия содержимое Rус ненулевое.

Те же структуры, но для логической связи «И» показаны на рис. 11 и рис. 12, соответственно.

Примечание: при выполнении некоторых команд ассемблера происходит дополнение старших разрядов знаком и если число отрицательное в этих разрядах появятся единицы, что может привести к неверному результату.

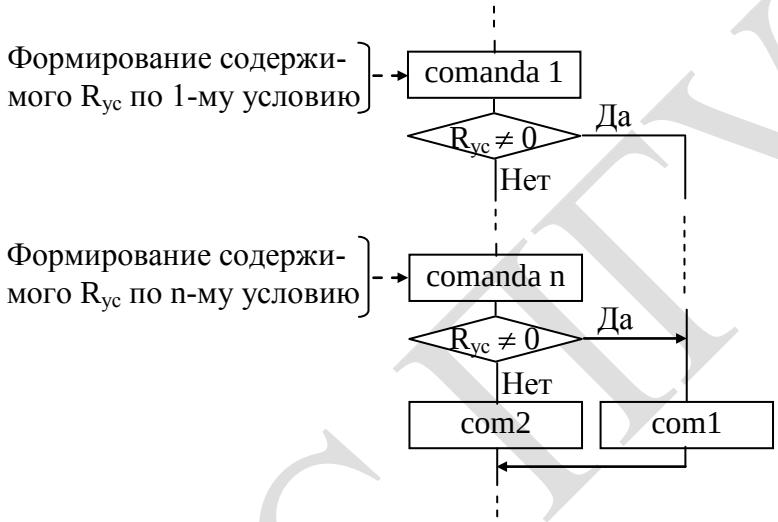


Рис. 9. Обобщенная алгоритмическая структура ветвления при логической связи «ИЛИ»

```
                    …….
          comanda 1   ; формирование Rус по 1-му условию
[!Rус]    comanda 2   ; формирование Rус по 2-му условию
                    …….
[!Rус]    comanda n   ; формирование Rус по n-му условию
 [Rус]    com1        ; операция 1 – выполнено хотя бы одно
                      ;простое условие
[!Rус]    com2        ; альтернативная операция 2 – ни одно из
                      ;простых условий не выполнено
                    …….
```

Рис. 10. Обобщенная программа ветвления при логической связи «ИЛИ»

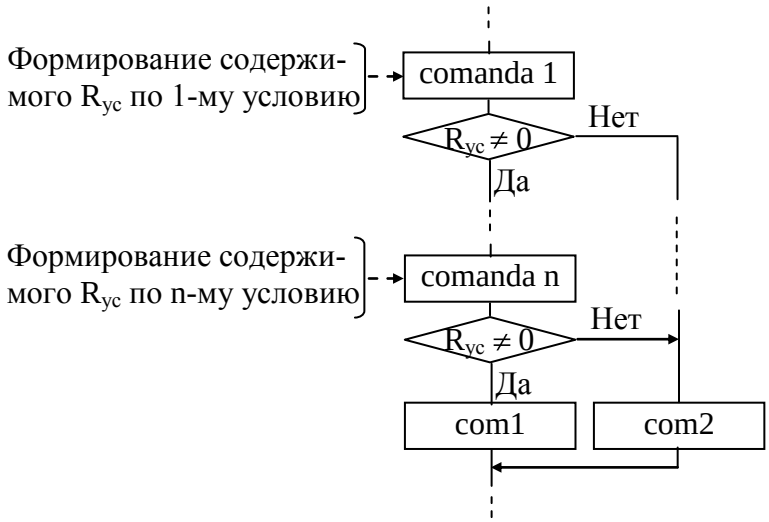


Рис. 11. Обобщенная алгоритмическая структура ветвления при логической связи «И»

```
                                    ……….
        comanda 1   ; формирование Rус по 1-му условию
[Rус]   comanda 2   ; формирование Rус по 2-му условию
                                    ……….
[Rус]   comanda n   ; формирование Rус по n-му условию
[Rус]   com1        ; операция 1 – выполнены все простые
                    ; условия
[!Rус]  com2        ; альтернативная операция 2 – хотя бы одно
                    ;из простых условий не выполнено
                                    ……….
```

Рис. 12. Обобщенная программа ветвления при логической связи «И»

Контрольные вопросы

1. Поясните принципы организации ветвлений на языке ассемблера со сложным условием и логической связью «И».
2. Поясните принципы организации ветвлений на языке ассемблера со сложным условием и логической связью «ИЛИ».
3. Приведите программную реализацию ветвления по любому другому варианту.
4. Укажите регистры РОН, допустимые для использования в качестве регистра условия.
5. Сформулируйте особенности реализации нестрогих отношений в условии ветвления.
6. Поясните общие принципы организации ветвлений при программировании на языке ассемблера.
7. С пояснениями приведите примеры поля условия строки ассемблера.
8. Приведите формат логической команды, заданной преподавателем.
9. Приведите формат сервисной команды, заданной преподавателем.
10. Определите результат выполнения логической команды, заданной преподавателем.
11. Определите результат выполнения сервисной команды, заданной преподавателем.

6. ВЕТВЛЕНИЕ С ВЛОЖЕНЫМИ УСЛОВИЯМИ

Совокупность условий, определяющая выполнение одной из более двух операций, называются вложенными. Вложенность проявляется в последовательной проверке условий. С помощью вложенных условий организуются ветвления

Цель работы

Изучить особенности реализации вложенных условий на языке ассемблера процессора TMS320C6x.

Подготовка к работе

1. По указанной выше литературе изучить принципы реализации сложных условий.

2. Ознакомиться с методическими указаниями.

3. Подготовить пункты 3 и 4 отчета (см. стр. 3 – 4) в соответствии с первым заданием работы. При этом требования пункта 4 отчета ограничить программой и прогнозами результатов выполнения ее команд.

Задание и порядок выполнения работы

1. На языке ассемблера TMS320C6x подготовить программу, реализующую ветвление рис. 13.

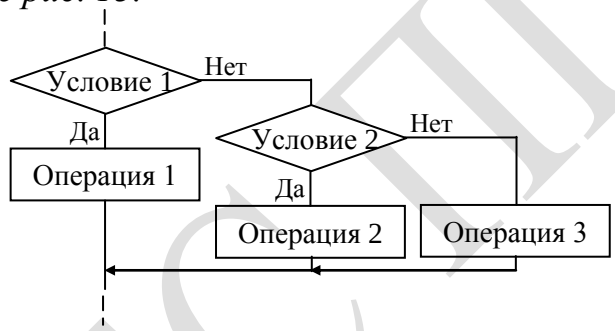


Рис. 13. Обобщенная алгоритмическая структура ветвления

При этом:

– исходное число (для всех операций одно) выбрать из таблицы 12.

Таблица 12. Исходное число

N	Число, Hex	N	Число, Hex	N	Число, Hex	N	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

Операция 1. Получить абсолютную величину исходного числа.

Операция 2. Установить «1» в области бит числа с 0 по N+5 включительно.

Операция 3. Переслать исходное число в регистр РОН противоположной стороны;

Регистр-источник операнда, регистры-приемники результатов и регистры условия назначить из РОН по своему усмотрению;

Условия выбираются из таблицы 13 по номеру варианта V = 31-N.

Таблица 13. Условие

V	**Условие ветвления**	**Способ (команды) формирования содержимого регистра условия R**
1	(B8) < 0 и (B9) < 0	2 команды CMPGT
2	(A9) = 0 или (A8) нечетно	CMPEQ и выделение в (A8) младшего бита (EXTU)
3	(B8) ≥ 0 и (B9) ≥ 0	(B8)∨(B9) и сдвиг результата вправо на 31 разряд (SHR)
4	(A9) = 0 или в A8 пять старших единиц	CMPEQ и сравнение числа F7FFFFFFh с (A8) (CMPLTU)
5	(B8) < 0 и (B9) < 0	(B8)∧(B9) и выделение старшего бита результата (EXTU)
6	(A9) ≤ 0 или (A8) четно	CMPLT и сдвиг влево (A8) на 31 разряд (SHL)
7	(B8) ≥ 0 и (B9) ≥ 0	OR и CMPGT: 0>((B8)∨(B9))
8	(A9) ≠ 0 или в A8 четыре старших единицы	MV над (A9) и сравнение (A8) с числом EFFFFFFFh (CMPLT)
9	(B8) < 0 и (B9) < 0	(B8)∧(B9) и сравнение результата с 0 (CMPGT)
10	(A9) ≠ 0 или (A8) нечетно	MV над (A9) и сброс в (A8) области бит с 1 по 31 (CLR)
11	(B8) ≥ 0 и (B9) ≥ 0	OR и SHRU (см. вариант 3)
12	(A9) = 0 или в A8 пять старших единиц	MV над (A9) и сравнение числа F8000000h с (A8) (CMPGTU)
13	(B8) < 0 и (B9) < 0	(B8)∧(B9) и сдвиг результата вправо на 31 разряд (SHR)
14	(A9) ≤ 0 или (A8) четно	CMPLT и выделение в (A8) младшего бита (EXT)
15	(B8) > 0 и (B9) > 0	2 команды CMPLT
16	(A9) ≠ 0 или в A8 четыре старших единицы	CMPEQ и сравнение (A8) с числом F0000000h (CMPGT)
17	(B8) ≥ 0 и (B9) ≥ 0	OR и выделение старшего бита результата (EXT)
18	(A9) = 0 или (A8) нечетно	CMPEQ и SHL (см. вариант 6)

V	Условие ветвления	Способ (команды) формирования содержимого регистра условия R
19	(B8) < 0 и (B9) < 0	(B8)∧(B9) и сброс области бит результата с 0 по 30 (CLR)
20	(A9) = 0 или в A8 пять старших нулей	MV над (A9) и выделение в (A8) области бит с 27 по 31 (EXT)
21	(B8) ≥ 0 и (B9) ≥ 0	2 команды CMPGT
22	(A9) ≤ 0 или (A8) четно	CMPLT и EXTU (см. вариант 2)
23	(B8) < 0 и (B9) < 0	AND и SHRU (см. вариант 13)
24	Хотя бы одно из(A9) и (A8) нечетно	OR и сброс области бит результата с 1 по 31 (CLR)
25	(B8) ≥ 0 и (B9) ≥ 0	OR и EXTU (см. вариант 17)
26	(A9) ≠ 0 или (A8) =0	CMPEQ над (A9) и MV для (A8)
27	(B8) < 0 и (B9) < 0	AND и EXT (см. вариант 5)
28	(A9) = 0 или A8 = 0	2 команды MV
29	(B8) ≥ 0 и (B9) ≥ 0	OR и сброс области бит результата с 0 по 30 (CLR)
30	Одно из (A9) и (A8) нечетно, другое четно	(A9)⊕(A8) и сброс области бит результата с 1 по 31 (CLR)

2. Получить исполняемый программный модуль (см. стр. 8 – 9).

3. Загрузить исполняемый модуль в симулятор (см. стр. 10).

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).

Методические указания

Примечание: при выполнении некоторых команд ассемблера происходит дополнение старших разрядов знаком и если число отрицательное в этих разрядах появятся единицы, что может привести к неверному результату.

Для реализации ветвления рис. 13 необходимы два регистра условия. Обозначим их в соответствии с номерами условий как R1 и R2.

Выполнение или игнорирование операций 2 и 3 зависит от комбинации двух условий. Однако в поле условия строки ассемблера допустимо указывать лишь один регистр условия. Поэтому для этих операций необходимо сформировать совокупный признак ветвления, учитывающий оба условия, который и следует размещать в регистре R2. Но тогда результат проверки условия 2, как один из операндов операции формирования совокупного признака, должен размещаться в любом другом регистре РОН. Для примера обозначим этот регистр как R и положим, что признаки ветвления формируются следующим образом:

– операция 1 выполняется, если (R1) = 1 (00000001h);

– операция 2 выполняется, если (R1) = 0 и (R) = (R2) = 1;

– операция 3 выполняется, если (R1) = 0 и (R) = (R2) = 0.

Заметим, что признаки ветвления зависят от способа формирования содержимого регистров R1 и R (т.е. от используемых команд) и в конкретной задаче могут быть другими.

На основании признаков ветвления формулируется совокупный признак ветвления. Для этого удобно составить таблицу истинности, которая для нашего примера имеет вид, представленный в таблице 14.

Таблица 14.

Содержимое R1	**Содержимое R**	**Содержимое R2**	
		для операции 2	**для операции 3**
1	0	0	1
1	1	0	1
0	0	0	0
0	1	1	1

Из таблицы истинности следует:

$R2 = \overline{R1} \wedge R$ для выполнения операции 2,

$R2 = R1 \vee R$ для выполнения операции 3

Тогда обобщенный алгоритм и соответствующая программа будут иметь вид представленный на рисунках 14 и 15.

Примечание: при выполнении некоторых команд ассемблера происходит дополнение старших разрядов знаком и если число отрицательное в этих разрядах появятся единицы, что может привести к неверному результату.

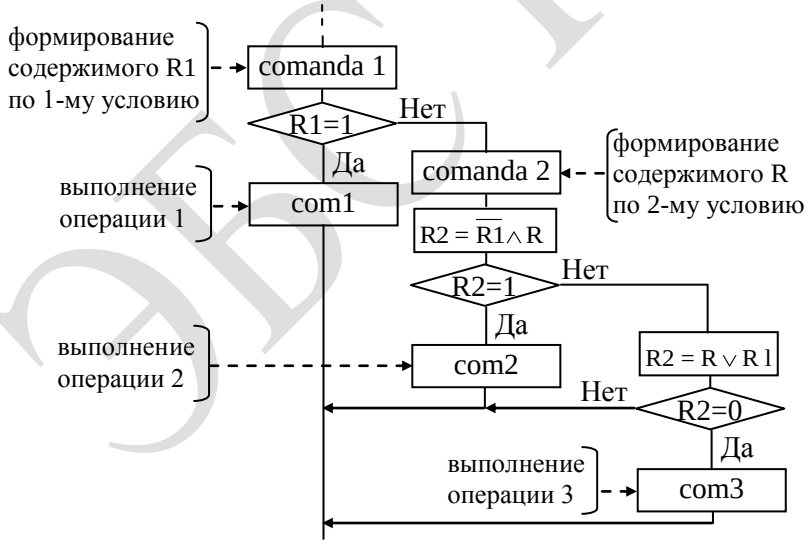


Рис.14. Итоговая алгоритмическая структура

```
        …….
          comanda 1
[R1]    com1
[!R1]  comanda 2          ; может быть и безусловной, но в таком
* виде уменьшается время выполнения программы при (R1) = 1
          NOT  R1,R3      ; эта и следующая команды должны быть
          AND  R3,R,R2    ; безусловными, т.к. иначе (R2) становит-
* ся случайным и возможно несанкционированное выполнение
* операции 2 или 3
[R2]    com2
          OR     R1,R,R2
[!R2]   com3
        …….
```

Рис. 15. Обобщенная программа для вложенного условия

Контрольные вопросы

1. Поясните структуру строки ассемблера процессора TMS320C6x.
2. Поясните принципы организации ветвлений на языке ассемблера с вложенным условием.
3. Приведите программную реализацию ветвления по любому другому варианту.
4. Укажите регистры РОН, допустимые для использования в качестве регистра условия.
5. Сформулируйте особенности реализации нестрогих отношений в условии ветвления.
6. Поясните общие принципы организации ветвлений при программировании на языке ассемблера.
7. С пояснениями приведите примеры поля условия строки ассемблера.
8. Приведите формат логической команды, заданной преподавателем.
9. Приведите формат сервисной команды, заданной преподавателем.
10. Определите результат выполнения логической команды, заданной преподавателем.
11. Определите результат выполнения сервисной команды, заданной преподавателем.

7. РЕГУЛЯРНЫЕ ЦИКЛЫ

Циклы используются с целью экономии объема оперативной памяти. В регулярных циклах число повторений (проходов) тела цикла заранее известно, что позволяет естественным образом использовать этот параметр для формирования признака выхода из цикла.

Цель работы

Изучить особенности реализации регулярных циклов на языке ассемблера процессора TMS320C6x

Подготовка к работе

1. По указанной выше литературе изучить формат и особенности выполнения команды безусловного перехода *B*.

2. Ознакомиться с методическими указаниями.

3. Подготовить пункты 3 и 4 отчета (см. стр. 3 – 4) в соответствии с первым заданием работы. При этом требования пункта 4 отчета ограничить программой и прогнозами результатов выполнения ее команд.

Задание и порядок выполнения работы

1. На языке ассемблера TMS320C6x подготовить программу ввода в память данных процессора массива из 9 чисел (рис. 16, где команда загрузки используется для контроля заполнения очередной ячейки памяти данных (ЯП)).

При этом:

– счетчик проходов тела цикла для нечетных V расположить до команды безусловного перехода, для четных V – после, где V = N (N≤10), V = N-10 (10<N≤20), V = N-20 (N>20);

– значения элементов массива изменяются с постоянным шагом h1;

– номера ЯП изменяются с постоянным шагом h2;

– регистры назначить из РОН по своему усмотрению.

Остальные исходные данные приведены в таблице 15.

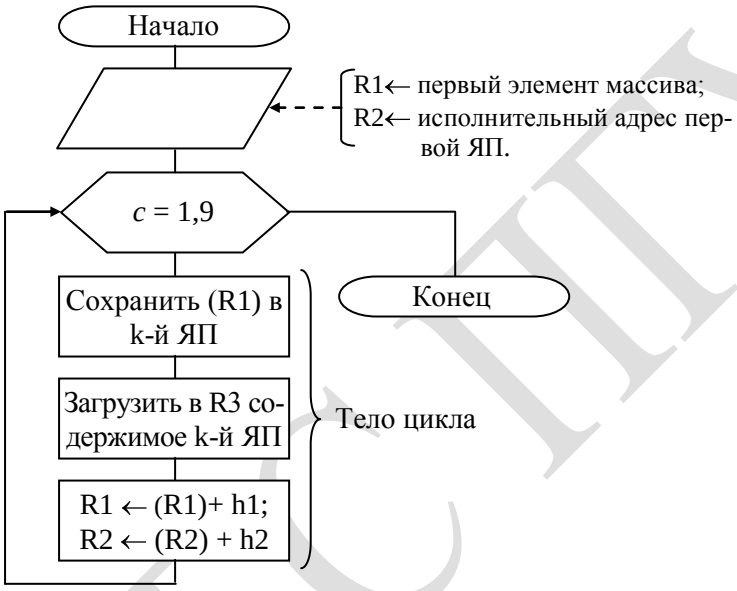


Рис. 16. Регулярная циклическая структура

Таблица 15.

V	Массив чисел a_1, a_2, …	Номера ЯП	Тип адресации операций хранения/загрузки
1	-4,-2,0, …	228,200, …	косвенная/постдекремент
2	7,5,3, …	4,8,12, …	косвенная/постинкремент
3	-4,-1,2, …	264,232, …	предекремент/косвенная
4	7,4,1, …	9,17,25, …	преинкремент/косвенная
5	-5,-3,-1, …	300,264, …	постдекремент/базирование

V	Массив чисел a_1, a_2, …	Номера ЯП	Тип адресации операций хранения/загрузки
6	8,6,4, …	6,18,30, …	постинкремент/базирование
7	-5,-2,1, …	336,296, …	предекремент/постдекремент
8	8,5,2, …	15,23, …	преинкремент/постинкремент
9	-6,-3,0, …	404,356, …	базирование/предекремент
10	9,6,3, …	17,33, …	базирование/преинкремент

Примечание: для V = 7, 8 смещение в адресном коде определять, исходя из половины расстояния между исполнительными адресами ячеек памяти.

2. Получить исполняемый программный модуль (см. стр. 8 – 9).

3. Загрузить исполняемый модуль в симулятор (см. стр. 10).

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).

Методические указания

В отличие от языков высокого уровня в языке ассемблера нет специальных команд по организации циклов. Поэтому при использовании ассемблера структура рис. 16 заменяется структурой ветвления (рис. 17). При этом тело цикла дополняется счетчиком числа его проходов (СП) и командой безусловного перехода.

В качестве СП используется один из регистров РОН, способный выполнять и функцию регистра условия. Этот регистр в зависимости от начального состояния работает в автоинкрементном или автодекрементном режиме.

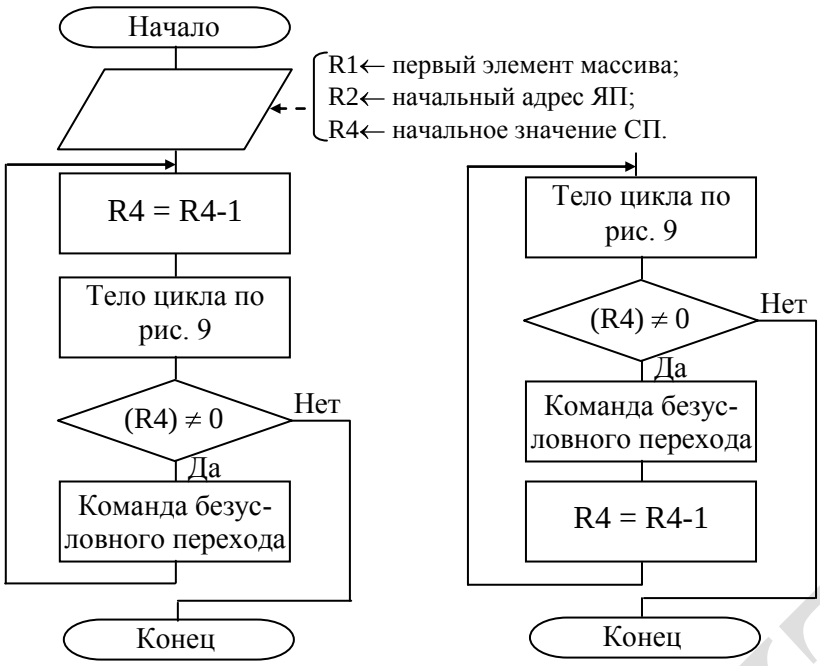


Рис. 17. Регулярные циклические структуры, приведенные к ассемблеру

В автоинкрементном случае с каждым проходом тела цикла содержимое СП увеличивается на 1. Тогда при M проходах признаком выхода из цикла является результат проверки условия СП > M (если начальное значение СП равно 1) или СП > M-1 при нулевом начальном значении СП.

В автодекрементном случае с каждым проходом тела цикла содержимое СП уменьшается на 1. При этом признаком выхода из цикла является нулевое содержимое СП, то есть естественное условие выполнения или игнорирования команд языка ассемблера. Таким образом, нет необходимости в дополнительной команде сравнения в теле цикла. Поэтому в работе используется именно этот подход.

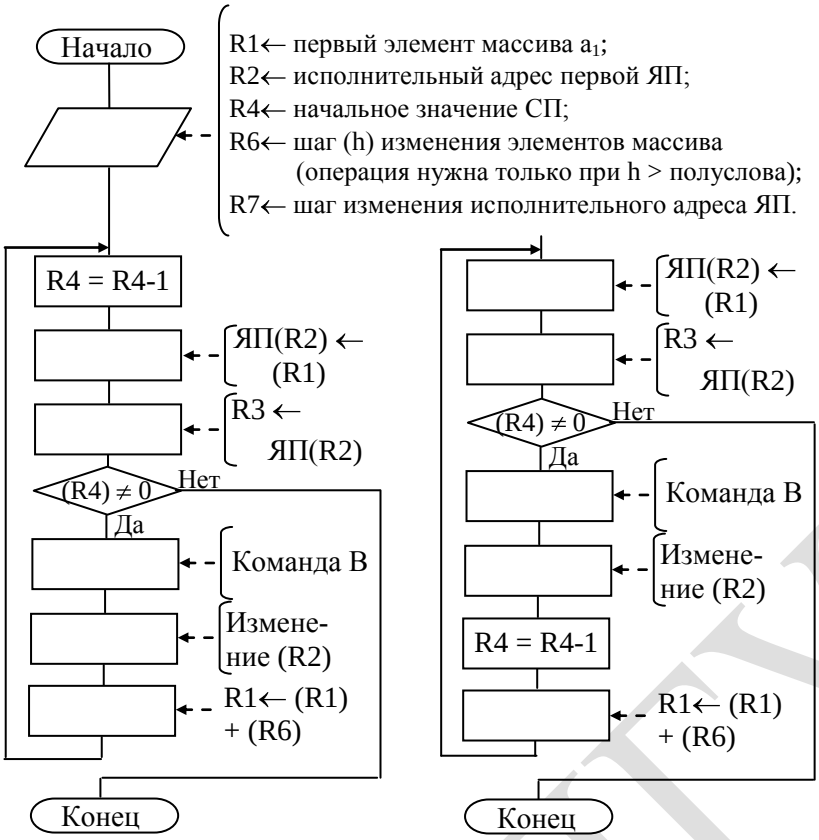
Тем не менее, приветствуется и индивидуальное творчество – формирование признака выхода из цикла по достижению адреса последней из используемых ЯП.

При организации цикла СП может размещаться как перед командой безусловного перехода (рис. 17,а), так и после нее (рис. 17,б). В первом случае начальное состояние счетчика должны быть равно требуемому числу проходов тела цикла, а во втором – на 1 меньше.

В программе команда безусловного перехода располагается в конце тела цикла, если в ее слотах задержки невозможно разместить ни одну из его команд. При этом после команды следует предусмотреть 5-тактный мультицикл NOP. В противном случае команда безусловного перехода располагается внутри тела цикла на расстоянии не более четырех тактов от его последней команды.

На рис. 18 приведен пример обобщенных алгоритмических структур и соответствующих программ, отвечающих поставленной задаче и построенных для обоих вариантов рис. 17. При этом элементы массива и исполнительные адреса

ЯП увеличиваются. Кроме того, на рис. 18 ЯП(Z) означает ячейку памяти, исполнительный адрес которой вычисляется на основе содержимого РОН Z.




```
        mvk <число>,R1
        ……….
m:      addk -1,R4
        stw   R1,*R2
        ldw   *R2++ [см.],R3
        nop    4
  [R4] b      m
  [R4] add    R1,R6,R1
  [R4] nop    3
```

а)

```
        mvk <число>,R1
        ……….
m:      stw   R1,*R2
        ldw   *R5++ [см.],R3
        nop    4
  [R4] b      m
  [R4]  addk -1,R4
  [R4] add    R1,R6,R1
  [R4] nop    2
```

б)

Рис. 18. Пример обобщенных структур регулярных циклов

Контрольные вопросы

1. Поясните структуру строки ассемблера процессора TMS320C6x.
2. Поясните принципы организации циклов на языке ассемблера.
3. Требования к счетчику проходов цикла.
4. Поясните особенности выполнения команды безусловного перехода *B*.
5. Приведите программную реализацию цикла по любому другому варианту.
6. Укажите регистры РОН, допустимые для использования в качестве регистра условия.
7. С пояснениями приведите примеры поля условия строки ассемблера.
8. Приведите формат сервисной команды, заданной преподавателем.
9. Определите результат выполнения цикла, заданного преподавателем.

10. Определите результат выполнения сервисной команды, заданной преподавателем.