МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ

КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. И. РАЗЗАКОВА

Кафедра «Информатики и вычислительной техники»

МИКРОПРОЦЕССОРЫ И МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ

Методические указания к лабораторным работам 1-4 для студентов специальности 552801.01 «Вычислительные машины, комплексы, системы и сети»

Бишкек – 2011

“Рассморено”
на заседании кафедры
“ИВТ”
Протокол № 2 от 23.09.2010 г.

“Одобрено”
Методическим советом
ФИТ
Протокол № 2 от 25.10.2010 г.

УДК 681.3.

Составители: АЛЫМКУЛОВ С.А., ТЕНТИЕВА С.М., ТУЛЬТЕМИРОВА Г.У.

Микропроцессоры и микропроцессорные системы. Методические указания к лабораторным работам для студентов специальности 552801.01 «Вычислительные машины, комплексы, системы и сети» / КГТУ им. И.Раззакова; сост.: С.А.Алымкулов, С.М.Тентиева, Г.У.Тультемирова. – Б.: ИЦ «Текник», 2011. – 32 с.

Приведены краткие теоретические сведения. Излагается методика выполнения лабораторных работ и приведены варианты заданий.

Предназначены для студентов дневной формы обучения.

Табл.: 1. Иллюстр.: 3. Библиогр.: 8.

Рецензент к.т.н., доцент каф. «ИВТ» Шабданов М.А.

Микропроцессоры и микропроцессорные системы
Методические указания к лабораторным работам для студентов специальности 552801.01 «Вычислительные машины, комплексы, системы и сети»
Составители: ***С.А.Алымкулов, С.М.Тентиева, Г.У.Тультемирова***

Тех. редактор ***Субанбердиева Н.Е.***

Подписано к печати 23.05.2011 г. Формат бумаги 60x84 $^{1}/_{16}$.
Бумага офс. Печать офс. Объем 2 п.л. Тираж 50 экз. Заказ 206.

Бишкек, ул. Сухомлинова, 20. ИЦ “Текник” КГТУ им. И.Раззакова, т.: 54-29-43
e-mail: beknur@mail.ru

Лабораторная работа №1

Работа с программой – монитором

Цель работы: изучение принципов работы программы монитора, изучение основных команд программы- монитора, получение навыков работы с программой- монитором.

Краткие теоретические сведения

Программа- монитор- это программа, предназначенная для упрощения действий по работе с микропроцессорным комплектом SKDP16, которая позволяет снять с пользователя основную часть действий по загрузке кода программы в оперативную память для дальнейшего ее выполнения. Программа-монитор содержит стандартный набор команд, с помощью которых можно работать с оперативной памятью микропроцессорного стенда, просматривать ее содержимое в определенных ячейках памяти, менять местами блока данных в оперативной памяти, загружать шестнадцатеричные коды программ в оперативную память и запускать загруженную программу на выполнение.

К основным командам программы- монитора относятся:

?- позволяет вызвать список команд программы- монитора.

D[W] [aaaa] [eeee]- позволяет просмотреть область памяти в виде шестнадцатеричного кода байтов (команд D) или слов (команда DW) начиная с ячейки, код которой указан первым операндом [aaaa], по ячейку, код которой указан вторым операндом [eeee].Вызов этой команда без указания диапазона ячеек приводит к выдаче информации начиная с первого адреса (0000h) 16 строк по 16 байтов (команда D) или по 8 слов (команда DW) в каждой строке. Последующий вызов этой команды без параметров продолжает вывод диапазона из 256 ячеек начиная с адреса на единицу большего того, на котором закончился предыдущий вывод данных. Задание этой команды только с первым параметром [aaaa] обеспечивает вывод информации из 256 ячеек памяти начиная с заданного начального адреса в виде 16 строк по 16 байтов (команда D) или по 8 слов (команда DW) в каждой строке.

E[W] [aaaa] [=dddd]- позволяет изменить байт (команда E) или слово (команда EW) данных хранящееся в ячейке данных с адресом указанным в

первом операнде [aaaa] на данные хранящиеся во втором операнде [=dddd]. Замена данных приходит установкой начального адреса и нажатием клавиш «ENTER», после чего на экран выводится значение текущего адреса, содержимое ячейки памяти по текущему адресу и знак «=» после которого следует ввести необходимое значение. При необходимости ввода диапазона значений в команде задается первая ячейка диапазона , вводится ее значение а затем нажимается «ENTER» в результате чего происходит переход на следующий адрес и появляется возможность модифицировать данные по этому адресу. Адреса доступные для модификации начинаются с адреса 4000h, так как с этого адреса начинается адресное пространство оперативной памяти.

F[W] [aaaa] [eeee] [dddd] – позволяет заполнить диапозон адресов начиная с адреса [aaaa] по адрес [eeee] значениями указанными в операнде[dddd] (для команды FW) или dd (для команды F). Адреса доступные для заполнения начинаются с ячейки с номером 4000h, так как с этого адреса начинается адресное пространство оперативной памяти.

L- позволяет загрузить файл с шестнадцатеричными кодами программы в оперативную память для дальнейшего выполнения.

G [aaaa]- позволяет запустить программу начиная с адреса указанного в первом операнде [aaaa].

Правильность вызова команд программы монитора приведена в примерах:

Пример 1.

Для вывода ячеек памяти заданного начального по заданной конечный адрес следует вызвать команду D или DW следующим образом:

> d100 150

где 100h- начальный адрес диапазона, 150h- конечный адрес диапазона.

В результате будет выведен массив данных:

0100: F2 0702 EC CF 4F F 06 – 02 EC CF 4F F2 05 02 EC
0110: CF 4F F2 04 02 EC CF 4F – F2 03 02 EC CF 4F F2 02
0120: 06 00 FE EF FF FF DA B4 – 00 10 00 E3 F8 F3 31 00
0130: 0D E3 F8 F3 31 00 02 E3 – F8 F3 33 00 94 E3 F8 F3
0140: 33 01 BF B3 00 10 F0 EA- 10 FA 1C B4 00 10 32 EA
0150: 0E

При задании только первого параметра в команде DW произойдет следующее:

> dw 30

Результат вызова команды будет:

0030: EC 02 5FCF 05F3 EC02 – 5FCF 06F3 EC02 5FCF
0040: 07F3 EC02 5FCF 08F3- EC02 5FCF 09F3 EC02
0050: 5FCF 0AF3 EC02 5FCF- 0BF3 EC02 5FCF 0CF3
0060: EC02 5FCF EE07 1EDD- 2EDE CE27 1000 EA9A
0070: FA10 B4CF 1000 EAAC- FA10 B4CB 1000 0904
0080: B547 1000 B2DA 1000- EAB4 FA10 B4C2 1000
0090: 0924 B53E 1000 B2D1- 1000 EABC FA10 B4B9
00A0: 1000 0934 B535 1000- B3E3 1000 EFFE FFFF
00B0: EC5E FC01 0C08 1000- ED81 FD31 100E 0ED0
00C0: ED0F 1ED9 100D ED0A- 79DA DA05 1000 100A
00D0: EA07 4A99 100D ED30- 4D99 B4BE 1000 EC02
00E0: 4FCF 0CF2 EC02 4FCF- 0BF2 EC02 4FCF 0AF2
00F0: EC02 4FCF 09F2 EC02- 4FCF 08F2 EC02 4FCF
0100: 07F2 EC02 4FCF 06F2- EC02 4FCF 05F2 EC02
0110: 4FCF 04F2 EC02 4FCF- 03F2 EC02 4FCF 02F2
0120: 0006 EFFE FFFF B4DA- 1000 E300 F3F8 0031

Задание команды без операндов приведет к следующему:

> d 101

Результат вызова команды будет:
0100: 07 02 EC CF 4F F2 06- 02 EC CF 4F F2 05 02 EC
0110:CF 4F F2 04 02 EC CF 4F – F2 03 02 EC CF 4F F2 02
0120: 06 00 FE EF FF FF DA B4 – 00 10 00 E3 F8 F3 31 00
0130: 0D E3 F8 F3 31 00 02 E3- F8 F3 33 00 04 E3 F8 F3
0140: 33 01 BF B3 00 10 F0 EA- 10 FA 1C B4 00 10 32 EA
0150: 0E FA 5F B4 00 10 56 EA- 0E FA 5B B4 00 10 00 EA
0160: 10 F4 04 05 45 35 FF EA- 7F F4 55 14 55 00 00 10
0170: 00 10 00 10 7C EC 01 FC – 08 0C 00 10 8C B3 00 10
0180: 3F B3 00 10 75 B3 00 10 – 0D E3 F8 F3 30 04 F7 C4
0190: 00 10 0E EA F8 FA A0 09- 01 EC CA 4A 8E B2 00 10
01A0: 44 E3 00 F3 93 23 1F C3 – 00 10 45 E3 00 F3 93 23

01B0: 1E C3 00 10 46 E3 00 F3- 93 23 1D C3 00 10 47 E3
01C0: 00 F3 93 23 1C C3 00 10 – 4C E3 00 F3 93 23 1B C3
01D0: 00 10 3F E3 00 F3 93 23 – 1A C3 00 10 1E B0 00 10
01E0: CE C0 00 10 93 B1 00 10 – CA C0 00 10 23 B1 00 10
01F0:C6 C0 00 10 D2 B0 00 10 – C2 C0 00 10 1A B0 00 10

Пример 2.

Для изменения содержимого ячеек памяти начиная с заданного адреса используется команда E или EW следующим образом:

>e 4000= 66

Изменяем 4000 ячейку на значение 66

проверка занесения значения

>d 4000 4000
4000:66

При изменении диапазона ячеек после ввода начального следует нажать « Enter».

>ew 4000

4000: 5555= ffff
4002:5555= ffff
4004:0000= 6666
4006:0000= 6666
4008:000=

Завершение изменения диапазона – нажатие клавиши «Enter» без ввода значения в ячейку

проверка занесения значения

> dw 4000
4000: FFFF FFFF 6666 6666- 0000 0000 0000 0000

Пример 3.

Вызов заполнения диапазона ячеек оперативной памяти константой производится следующим образом:

Применение команды F

> f 4000 4006 = 00

проверка занесения значения

> d 4000

4000: 00 00 00 00 00 00 00 66- 00 00 00 00 00 00 00 00

применение команды FW

> fw 4000 4010= 8765

проверка занесения значения

> dw 4000

4000: 8765 8765 8765 8765 – 8765 8765 8765 8765

Пример 4.

Для загрузки кода программы в оперативную память необходимо сначала создать ресурсный файл с шестнадцатеричными кодами программы. Для этого в командную строку вводится команда L, нажимается клавиши «Enter, в пункте меню Transfer программы Гипертерминалы выбирается команда Send Text File, открывается файл с расширением S19. при корректном срабатывании программы на экран выводится сообщение «Load complete», которое говорит о том, что программа загружена успешно. Если выдается сообщение «load error», то следует настроить программу Гипертерминала по следующему принципу: вызвать пункт меню «File», команду « Properties», вкладку «Setting», нажать на кнопку «ASCII Setup» и установить в строке «Line delay» значение 10 миллисекунд. После того как программа будет корректно загружена можно запускать ее на выполнение.

Пример 5.

Загрузка программы на выполнение происходит при помощи команды G следующим образом:

>g4000

При корректном выполнении программы должны выполниться запрограммированы действия.

Действия по загрузке программы Гипертерминала и инициализации микропроцессорного комплекта:

Для активизации работы микропроцессора необходимо включить блок питания при помощи переключателя находящего на задней панели блока питания микропроцессорного стенда, на блоке загорается зеленый индикатор,

говорящий о том, что питание включено. Тюнер напряжения должен быть установлен в 5 вольт, тюнер ток а в 1 ампер. На плите микропроцессорного комплекта должен загореться красный индикатор, говорящий о том, что питание к плате подключено. На жидкокристаллическом мониторе должна появиться надпись «SDK P16 monitor program». Далее следует запустить программу Гипертерминала с помощью ярлыка Sdkp16 находящегося на рабочем столе. Если подключение стенда правильно, то на экрана будет выведено окно программы Гипертирминала, в которой можно вызвать команды программы- монитора. Для инициализации программы монитора на лабораторном стенде следует нажать кнопку Reset, после чего на экран будет выведен список команд программы- монитора, и будет сформирована командная строка для ввода команд.

Разработка программ для запуска на микропроцессорном стенде

Для того чтобы запустить программу с помощью микропроцессорного стенда необходимо написать программу инициализации соответствующей схемы микропроцессорного комплекта, сохранив ассемблерный текст программы в папке Source, находящейся в папке ASMP16 в файле с расширением p16. проверить программу на правильность, откомпилировав ее с помощью программы P16.exe. код программы при этом вместе с ошибками (если они есть) будет помещен в файл с расширением lst. Если программа написана ьез ошибок, то при компиляции будет создаваться файл с двоичными кодами и расширением bin. Полученный файл необходимо конвертировать в шестнадцатеричный формат при помощи файла L.bat, который в свою очередь создаст файл с расширением s19. только при наличии этого файла возможность дальнейшее преобразование программы. Для загрузка с помощью команды-монитора служить файл с расширением S19.

Полученный в результате конвертирования файл может быть загружен в программу Гипертерминала при помощи команды L.

Порядок выполнения работы

1) Включить микропроцессорный стенд
2) Поверить правильность работы стенда.
3) Вызвать программу Гипертерминала.

4) Нажать кнопку «Reset» на микропроцессорном стенде и убедиться в том, что сработала программа- монитор, то есть, был ли выведен на экран список команд программы монитора.
5) Выполнить задание лабораторной работы согласно варианту.

Варианты заданий

1) Просмотреть область данных, начиная с адреса 4000 по 4050 пословно, заменить первые 10 байт из этой область на коды 12 13 13 14 16 09 98 87 76 44, проверить произведены ли изменения, заполнить диапазон ячеек с адреса 4500 по 4570 значением 3333, проверить произведены ли изменения. Откомпилировать программу теста жидкокристаллического дисплея (файл lcdtest) и запустить ее на выполнение.
2) Просмотреть область данных начиная с адреса 5000 по 5100 побайтно, заменить первые 8 байт из этой область на коды 00 01 02 03 04 05 06 07, проверить произведены ли изменения, заполнить диапазон ячеек с адреса 4300 по 4330 значением ff, проверить произведены ли изменения. Откомпилировать программу теста жидкокристаллического дисплея (файл lcdl) и запустить ее на выполнение.
3) Просмотреть область данных начиная с адреса 4500 по 4550 пословно, заменить первые 12 байт из этой области на коды 6666 8888 9999 3333 1111 5555, проверить произведены ли изменения, заполнить диапазон ячеек с адреса 5500 по 5570 значением ffff, проверить произведены ли изменения. Откомпилировать программу теста жидкокристаллического дисплея (файл lcdtest) и запустить ее на выполнение.
4) Просмотреть область данных начиная с адреса 5000 по 5100 побайтно, заменить первые 11 байт из этой области на коды 00 01 02 03 04 05 06 07 ff ff ff , проверить произведены ли изменения, заполнить диапазон ячеек с адреса 5000 по 5330 значением cc, проверить произведены ли изменения. Откомпилировать программу теста жидкокристаллического дисплея (файл lcdl) и запустить ее на выполнение.
5) Просмотреть область данных начиная с адреса 4012 по 4022 пословно, заменить первые 16 байт из этой область на коды ffff fffe eeef cccc 0000 aaaa dddd bbbb, проверить произведены ли изменения, заполнить диапазон ячеек с адреса 5011 по 5035 значением feef, проверить произведены ли изменения. Откомпилировать программу теста жидкокристаллического дисплея (файл lcdl) и запустить ее на выполнение.

6) Просмотреть область данных начиная с адреса 3001 по 3037 побайтно, заменить 8 байт начиная с адреса 4057 на коды 00 01 02 03 04 05 06 07, проверить произведены ли изменения, заполнить диапазон ячеек с адреса 4047 по 4077 значением EC, проверить произведены ли изменения. Откомпилировать программу теста жидкокристаллического дисплея (файл lcdl) и запустить ее на выполнение.
7) Просмотреть область данных начиная с адреса 100 по 1022 пословно, заменить 12 байт начиная с адреса 4666 на коды 12 13 14 15 16 09 98 87 76 44 ff bf, проверить произведены ли изменения, заполнить диапазон ячеек с адреса 4543 по 4567 значением 3333, проверить произведении ли изменения. Откомпилировать программу теста жидкокристаллического дисплея (файл lcdtest) и запустить ее на выполнение.
8) просмотреть область данных начиная с адреса 3000 по 3147 побайтно, заменить 13 байт из этой области на коды fe ff fe ff fe ff fe ff fe ff fe ff cf, проверить произведены ли изменения, заполнит диапазон ячеек с адреса 4016 по 4112 значением 0b, проверить произведены ли изменения. Откомпилировать программу теста жидкокристаллического дисплея (файл lcdl) и запустить ее на выполнение.

Контрольные вопросы

1) Какой командой можно загрузить программу в оперативную память для выполнения.
2) Как настроить программу «Гипертерминалы»
3) В файле с каким расширением хранится листинг программы и коды ошибок.
4) Как конвертировать код программы из двоичного в шестнадцатеричный код.
5) Какую информацию выведет команда DW без параметров.
6) Как заполнить несколько подряд идущих ячеек разной информацией.
7) Для чего служит программа- монитор.
8) Для чего служит программа «Гипертерминал»

Лабораторная работа № 2
Работа с системой команд

Цель работы: Изучение системы команд микропроцессора SKDP 16 и принципов работы с командами микропроцессора, разработка простейших программ.

Краткие теоретические сведения

Микропроцессор SKDP 16 имеет свою систему команд в которую входят команды перемещения данных, контроля флагов, обращения к процедурами возврата из них, целочисленной арифметики, логические команды и команды сдвигов.

Команды перемещения данных

LB - загрузка байта.

LB Ri, Rp Rp<-(Ri)

Загружает байт данных находящийся по указанному адресу в регистр приемник. Адрес должен содержаться в регистре источнике. Байт данных перед передачей в шестнадцатиразрядный регистр дополняется в старшей части значением бита старшего разряда младшего байта.

LW –загрузка слова

LW Ri, Rp Rp<-(Ri)

Загружает слово данных находящееся по указанному адресу в регистр приемник. Адрес памяти должен содержаться в шестнадцатиразрядном регистре источнике.

SB- запись байта.

SB Ri, Rp Rp<-(Ri)

Записывает байт находящийся в регистр Ri в ячейку памяти, указанную в регистр Rp

SW- запись байта.

SW Ri, Rp Rp<-(Ri)

Записывает слово находящееся в регистре Ri в ячейку памяти, указанную в регистре Rp

MLO- загрузка младшего байта.

MLO Is, Rp Rp(7-00)<-Is Rp(15-8)<-0

Производит загрузку младшего байта регистра приемника явно указанной информацией, например константой (прямая адресация)

MHI – загрузка старшего байта.

MHI Is, RP Rp(15-8)<-Is

Производит загрузку старшего байта регистра приемника явно указанной информацией, например константой (прямая адресация)

Команды контроля флагов (команды переходов)

JMP-безусловный переход.

JMP RM Pc<-Rm

Производится безусловный переход шестнадцатеричный адрес находится в регистре Rm.

BEQ0- переход если ноль

BEQ0 Rs, D8 if (Rs=o) PC<-PC+(D8<<1)

Если регистр Rs содержит нулевое значение, то происходит переход на метку в программе, путем изменения содержимого счетчика команд на величину смещения адреса метки.

BNEO-переход если ноль.

BNEO Rs, D8 if (Rs<>0) PC <-PC +(D8>>1)

Если регистр Rs содержит ненулевое значение, то происходит переход на метку в программе, путем изменения содержимого счетчика команд на величину смещения адреса метки.

BSR-вызов подпрограммы

BSR D12 R2<-PC+2 PC<-PC+(D12<<2)

Происходит переход на метку программы, которая определяет начало выполняемой процедуры. При этом текущее содержимое указателя команд увеличенное на 2 помещается в регистр R2 блока регистров общего назначения.

RTS - возврат из процедуры.

Помещается в конце описания процедуры.

JSR- безусловный переход на метку.

JSR Rs R2<-PC =2 PC<-Rs

Происходит переход на указанную метку, при этом текущее содержимое указателя команд увеличенное на 2 помещается в регистр R2 блока регистров общего назначения.

Команды целочисленной арифметики

ADD- целочисленное сложение

ADD =Rsl , Rs2, Rp Rp<-Rsl+Rs2

Производит сложение целых чисел находящихся в регистрах Rs1 и Rs2 и помещает полученный результат в регистр приемник Rp.

SUB- целочисленное вычитание

SUB- Rsl, Rs2, Rp Rp<-Rsl-Rs2

Производит сложение целых чисел находящихся в регистрах Rsl и Rs2 и помещает полученный результат в регистр приемник Rp.

ADD- целочисленное сложение

ADD =Rsl , Rs2, Rp Rp<-Rsl+Rs2

Производит сложение целых чисел находящихся в регистрах Rs1 и Rs2 и помещает полученный результат в регистр приемник Rp.

SLO- проверка на меньше без учета знаков операндов.

SLO Rsl, Rs2, Rd if (Rsl<Rs2) Rd<-1 else Rd <-0

Сравнивает значения содержимого регистров Rsl и Rs2 не учитывая знаки и если первое число меньше второго, то помещает в регистр- приемник Rd значение 1, в противном случае 0.

SLT- проверка на меньше с учетом знаков операндов.

SLT Rsl, Rs2, Rd if (Rsl<Rs2) Rd<-1 else Rd <-0

Сравнивает значения содержимого регистров Rsl и Rs2 учитывая знаки и если первое число меньше второго, то помещает в регистр приемник Rd значение 1, в противном случае 0.

Команды сдвигов

SRL- сдвиг вправо логический.

SRL

Rsl, Rs2, Rd RdБ- Rsl>> Rs2, заполнение 0

Происходит сдвиг числа указанного в регистре Rsl вправо на число разрядов указанных в младших четырех разрядах регистра Rs2,

освободившиеся разряды заполняются нулями. Результат помещается в регистр приемник Rd.

SLL- сдвиг влево логический.

SLL Rsl, Rs2, Rd Rd <-Rsl<< Rs2, заполнение 0

Происходит сдвиг числа указанного в регистре Rsl влево на число разрядов указанных в младших четырех разрядах регистра Rs2, освободившиеся разряды заполняются нулями. Результат помещается в регистр приемник Rd.

SRA- сдвиг вправо арифметический

SRA Rsl, Rs2, Rd Rd <-Rsl<< Rs2, заполнение битом знака.

Происходит сдвиг числа указанного в регистре Rsl вправо на число разрядов указанных в младших четырех разрядах регистра Rs2, освободившиеся разряды заполняются битом знака. Результат помещается в регистр приемник Rd.

Логические команды

AND- логические умножение

AND Rsl, Rs2, Rd Rp<-Rsl*Rs2

Производит операцию поразрядного логического умножения с соответствующими битами регистров Rsl и Rs2, результат помещается в регистр Rd.

OR – логические сложение.

OR Rsl, Rs2, Rd Rp<-Rsl+Rs2

Производит операцию поразрядного логического сложения с соответствующими битам регистров Rsl и Rs2, результат помещается в регистр Rd.

XOR - сложение по модулю 2

XOR Rsl, Rs2, Rd Rp<-Rsl │Rs2

Производит операцию поразрядного логического сложения с по модулю 2 с соответствующими битам регистров Rsl и Rs2, результат помещается в регистр Rd.

Принципы построения программ

1.Программы для микропроцессора SKDP 16 пишутся в формате com, причем директива Org 4000h позволяет задать начальный адрес в оперативной память, с которого будет располагается программа.

2. До начала описания программы следует описать все константы, которые могут использоваться в программе, применяя директиву Equ. Так же до начала текста исходной программы могут располагаться макроопределения задаваемые с помощью директив:

```
Имя  macro параметры
{тело макроопределения}
End
```

3. Перед запуском программы следует сохранить значение стека в регистре R15 причем начало стека расположено по адресу fffeh.

4. Объявление процедур происходит в конце программы, после того как представлен основной текст. Ставится метка, на которую произойдет переход в результате вызова процедур, а затем приводится процедуры, которое завершается командой Rts.

5. После описания процедур может быть сформирован раздел переменных, в котором значения переменных определяются с помощью директивы Db.

6. Завершается программа служебным словом End.

7. Для выполнения программы, в конце текста основной программы следует поставить переход на стартовый адрес программы.

Пример написания программы

Задание

Получить из ячейки памяти с адресом 4300H байт информации, увеличить его на 5 и занести в ячейку памяти с адресом 4302h.

Текст программы:

```
;-------------------------------------------------------------------------
Stack       egu             f ffeh :начальный адрес стека
-------------------------------------------------------------------------
            org             4000h
start       mlo             stack,r 15;загрузка указателя стека
            mli             stack, r15
            mlo             00h,r4 ; загрузка адреса ячейки памяти
```

```
        mhi     43h,r4
        lb      r4,r5   ;чтение байта из ячейки памяти
        mhi     00h,r5  ; обнуление старшего байта регистра
        mlo     05h,r6  ; загрузка числа 5
        mhi     00h,r6  ;
        add     r5,r6,r7 ;сложение полученного из памяти байта с
                числом 5
        mlo     02h,r4 ; загрузка адреса ячейки памяти
        mhi     43h,r4
        sb      r5,r4 ; вывод байта в заданную ячейку памяти
        mlo     start, r12 ;загрузка стартового адреса
        mhi     start, r12
        jmp     r12 ;  переход на стартовый адрес
end
```

Порядок выполнения работы

1) Изучить методическое руководство к выполнению работы.
2) Пользуясь представленными командами написать программу, выполняющую заданные действия.
3) Откомпилировать программу, преобразовав ее в файл с расширением S19.
4) Загрузить программу в оперативную память, пользуясь программой Гипертерминала.
5) Занести в ячейку памяти указанные в задании некоторую исходную информацию.
6) Запустить программу на выполнение.
7) Проверить результаты выполнения программы, проанализировав результирующие ячейки памяти.
8) Защитить лабораторную работу, представив по ней отчет (текст программы) и ответив на контрольные вопросы.

Варианты заданий

1) Написать программу, которая модифицирует значения находящиеся в ячейках памяти с адреса 4500h по 4510h путем сдвига влево значений расположенных по четным адресам и вправо, расположенных по нечетным адресам. Переместить программно модифицируемые ячейки в область памяти начиная с адреса 4530h

2) Написать программу, которая модифицирует значения находящиеся в ячейках памяти с адреса 4510h по 4520h путем сдвига влево значений расположенных по четным адресам и увеличения на 3 значений, расположенных по нечетным адресам. Переместить программно модифицируемые ячейки в область памяти начиная с адреса 4530h.
3) Написать программу, которая модифицирует значения находящиеся в ячейках памяти с адреса 4300h по 4320h следующим образом. Яч= (Яч*4) – 11h. Переместить программно модифицируемые ячейки в область памяти начиная с адреса 4330h.
4) Написать программу, которая модифицирует значения находящиеся в ячейках памяти с адреса 4500h по 4510h путем умножения 4 значений расположенных по четным адресам и деления на 8 значений, расположенных по нечетным адресам. Переместить программно модифицируемые ячейки в область памяти начиная с адреса 4330h.
5) Написать программу, которая модифицирует значения находящиеся в ячейках памяти с адреса 4297h по 4307h путем сдвига влево значений расположенных по четным адресам и уменьшения на 6 значений, расположенных по нечетным адресам. Переместить программно модифицируемые ячейки в область памяти начиная с адреса 4330h.
6) Написать программу, которая модифицирует значения находящиеся в ячейках памяти с адреса 4297h по 4307h путем сдвига влево значений расположенных по нечетным адресам и увеличения на 3 значений, расположенных по четным адресам. Переместить программно модифицируемые ячейки в область памяти начиная с адреса 4330h.
7) Написать программу, которая модифицирует значения находящиеся в ячейках памяти с адреса 4307h по 4327h путем сдвига влево значений на 2 разряда и увеличения полученных значений на 5, для ячеек расположенных по четным адресам, значения по нечетным адресам оставить неизменными. Переместить программно модифицируемые ячейки в область памяти начиная с адреса 4330h.
8) Написать программу, которая модифицирует значения находящиеся в ячейках памяти с адреса 4250h по 4565h путем сдвига вправо значений расположенных по нечетным адресам и уменьшения их на 6,значения расположенные по четным адресам оставить неизменными. Переместить программно модифицируемые ячейки в область памяти начиная с адреса 4330h.

Контрольные вопросы

1. Почему загрузка программы начинается с адреса 4000h?
2. Какое значение помещается в старшие 8 битов регистра при чтении из памяти младшего байта?
3. На сколько разрядов производится арифметический и логический сдвиги?
4. Какие команды относятся к арифметическим операциям?
5. Какие команды относятся к командам загрузки?
6. В каких командах используется косвенная адресация?
7. В каких командах используется непосредственная адресация?
8. Какие могут быть команды переходов?
9. В каких командах производится операция сравнения?
10. какой регистр служит для хранения указателя стека?

Лабораторная работа №3
Программирование жидкокристаллического дисплея

Цель работы: получение студентам навыков работы программирования периферийных устройств входящих в состав микропроцессорной системы (на примере жидкокристаллического дисплея)

Краткие теоретические сведения

Жидкокристаллический дисплей

Микросхема HD44780- это жидкокристаллический дисплейный модуль, который отображает две строки символов. Каждый символ может использовать 5*10 точек матрицы, из которых 5*7 разрешается выбирать программно. Дисплей имеет набор инструкций по выполнению различных функций.

Для ввода/вывода применяется восьмиразрядная двунаправленная шина данных. Кроме того, при организации ввода/вывода необходимо управлять следующими входами микросхемы: контакт 4 (выбор регистра, RS), контакт 5 (чтение/запись R/W), и контакт 6 (разрешение, ENABLE).

Линия RS отображает то, что загружается в модуль дисплея- инструкции или данные:

RS=0: передаются инструкции;

RS=1: передаются данные.

Линия R/W показывает, какая операция проводится в данный момент чтение или запись:

R/W=1: чтение.

R/W=0: запись.

Чтением и запись данных управляет линия ENABLE. При загрузке данных информация подается на входы RS, R/W, DB0-DB7, а затем по отрицательному фронту на входе ENABLE (рис. 1) загружается в дисплей.

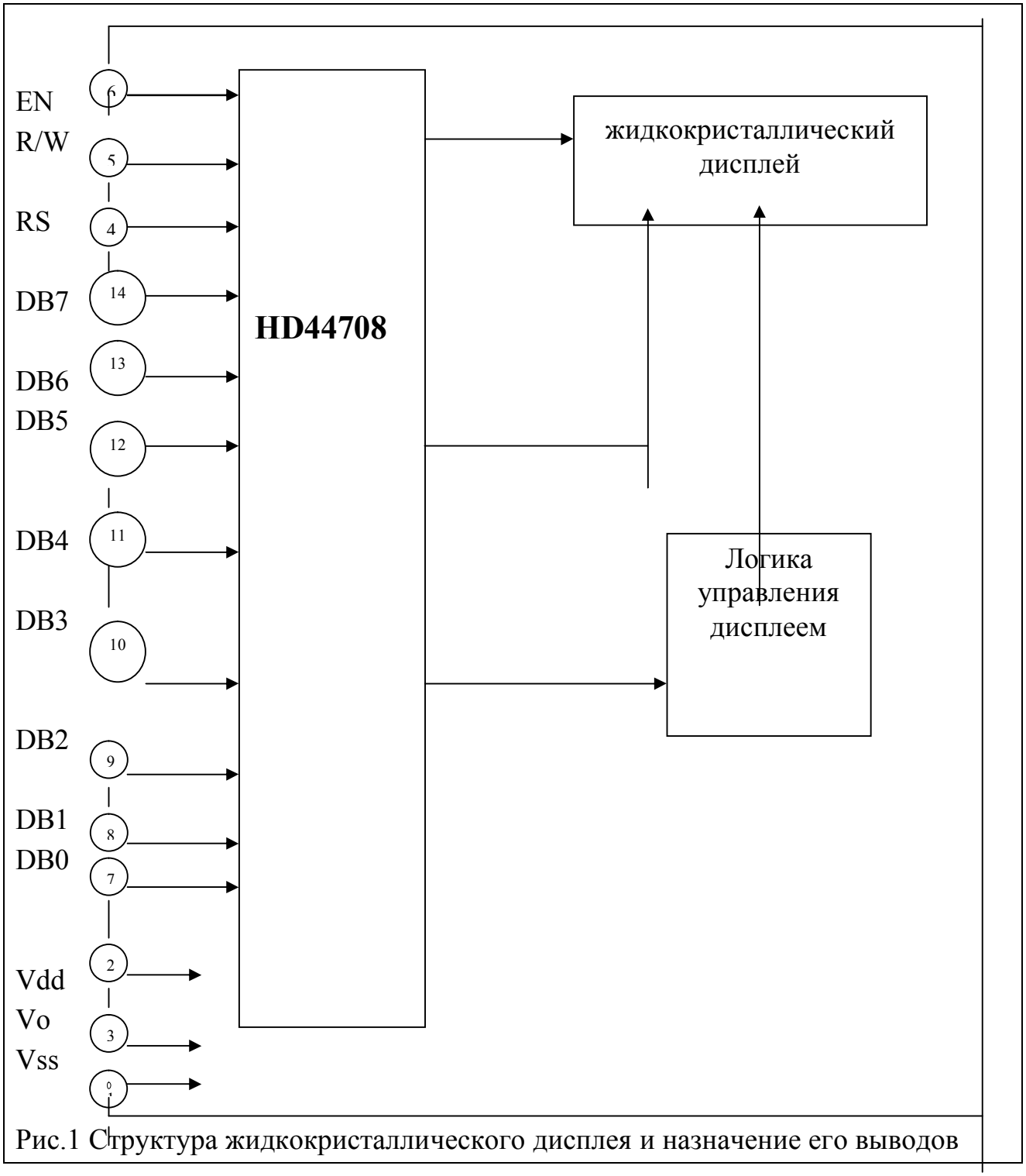

Рис.1 Структура жидкокристаллического дисплея и назначение его выводов

Таблица 1.

Набор инструкций

Инструкция	**RS**	**R/W**	**DB7**	**DB6**	**DB5**	**DB4**	**DB3**	**DB2**	**DB1**	**DB0**	**Описание**
Очистить дисплей	0	0	0	0	0	1	0	0	0	1	Очистка дисплея, возврат курсора в начальную позицию
Возврат начало	0	0	1	0	0	0	0	0	1	0	Возврат курсора в начальную позицию; ОЗУ не изменяется
Установка режима	0	0	0	0	0	0	0	1	I/D	S	Установка режима дисплея
Включение/ выключение дисплея	0	0	0	0	0	0	0	D	C	B	Включение/ Выключение: D-дисплея, C-курсора, B-мигающего курсора
Сдвиг курсора	0	0	0	0	0	1	S/C	R/L	0	0	Перемещение курсора и скроллинг; ОЗУ не изменяется
Установка функций 0	0	0	0	0	1	DL	N	F	0	0	Настройка интерфейса ввода/вывода
Установка CGRAM адреса -	0	0	0	1	ACG						Установка CGRAM- адреса, после этого передача или прием данных
Установка DDRAM-адреса	0	0	1	ADD							Установка DDRAM- адреса, после этого передача или прием данных
Чтение адреса и флаги занятости	0	1	BF	AC							Чтение флага занятости и счетчика адреса
Запись данных	1	0	Запись данных								Запись в ОЗУ
Чтение	1	1	Чтение данных								Чтение из ОЗУ
Назначе-ние битов инструкций	I/D=0: увеличение; I/D=1: уменьшение S=1:относительный скроллинг S/C=0: скроллинг S/C=1: перемещение курсора R/L=1: сдвиг вправо R/L=0: сдвиг влево DL=1:8 бит: DL=0:4 бита N=1:2 строки; N=0:1 строка F=0: матрица 5x10; F=1: матрица 5x7 BF=0: работа по внутренним инструкциям BF=1: прием внешних инструкций										DDRAM: вывод содержимого ОЗУ CGRAM: ОЗУ генератора ACG: адрес CGRAM ADD:адрес DDRAM, адрес курсора, AC: счетчик адреса, используемый для DDRAM и CGRAM

Для того чтобы обеспечить определенную работу жидкокристаллического дисплея необходимо ввести в него определенную информацию, то есть

инструкции (последовательности битов), обеспечивающие определенный режим работы.

Набор основных инструкции приведен в таблице 1.

Как любое периферийное устройство в составе микропроцессорной системы жидкокристаллический дисплей в начале работы системы должен программироваться на выполнение оперделенных видов работы. Для дисплея строк или одной строки дисплея, установка режима вывода двух информации, установка определенных режимов курсора.

Регистр статуса жидкокристаллического дисплея расположен по адресу **310h**, регистр данных по адресу **3103h.**

При помощи записи определенных инструкций в регистр статуса жидкокристаллического дисплея можно запрограммировать его на выполнение определенных действий, то есть инициализировать дисплей и выбрать определенные режимы работы. После инициализации схема жидкокристаллического дисплея становится доступна для вывода на него символьный информации. Выводимая информации может хранится как в памяти, так и приходить через схемы параллельного и последовательного портов, таймер и схемы кодирования включенные в состав микропроцессорной системы.

В процессе начальной инициализации выполняются следующие установки:

- В регистр R7 заносится маска, позволяющая определить занять ли жидкокристаллический дисплей (это выполняется установкой старшего бита маски в значение 1);
- Читается регистр статуса жидкокристаллического дисплея.
- С помощью логического умножения сверяются маска и считанные данные.
- Если дисплей занят, то результат логического умножения вернет значение неравное 1 и можно продолжать программирование (инициализацию дисплея), в противном случае организуется цикл ожидания освобождения жидкокристаллического дисплея с постоянным чтением байта статуса дисплея.
- Производится установка режима работы дисплея, а именно сколько строк дисплея будет использоваться при выводе и какой режим используется (8 бит, 4 бит).
- Проверяется статус дисплея (на занятость).
- Устанавливается режим перемещения курсора (инкремент или декремент).
-Проверяется статус дисплея (на занятость).

- Устанавливается режим контроля дисплея (включение дисплея, отображение курсора, мигание курсора).
- Проверяется статус дисплея (на занятость).
- Производится очистки дисплея.
- Проверяется статус дисплея (на занятость).

После того как дисплей инициализирован, можно осуществлять вывод информации.

Вывод символьный строки осуществляется по одному символу, причем после вывода каждого символа должен проверяться статус жидкокристаллического дисплея, так как при выводе очередного символа дисплей может быть занят выводом предыдущего значения. Код символа байтовой длины после инициализации дисплея должен выводиться в регистр динах дисплея, в результате чего он будет отображаться на дисплее.

Строки выводимых символов должна описываться в разделе данных с помощью директивы **db,** заключаться в двойные кавычки и заканчиваться двумя символами $$.

Пример:

Stroke db “slovo dlya vivoda$$”

Вывод сообщения осуществляется загрузкой кода очередного символа из области данных, отведенной под строку с последующим наращиванием этого адреса не единицу для каждого нового символа, при выборе символа из строки производится одновременная проверка его кода на совпадение с кодом символа **$**. И при совпадении символов происходит прекращение вывода текстового сообщения.

В общем случае алгоритм программирования жидкокристаллического дисплея и последующего вывода строки символов может быть представлен следующим образом (см.рис. 2).

Согласно представленному алгоритму и занося соответствующие инструкции в регистр статуса жидкокристаллического дисплея производится его программирование на выполнение выбранного вывода информации.

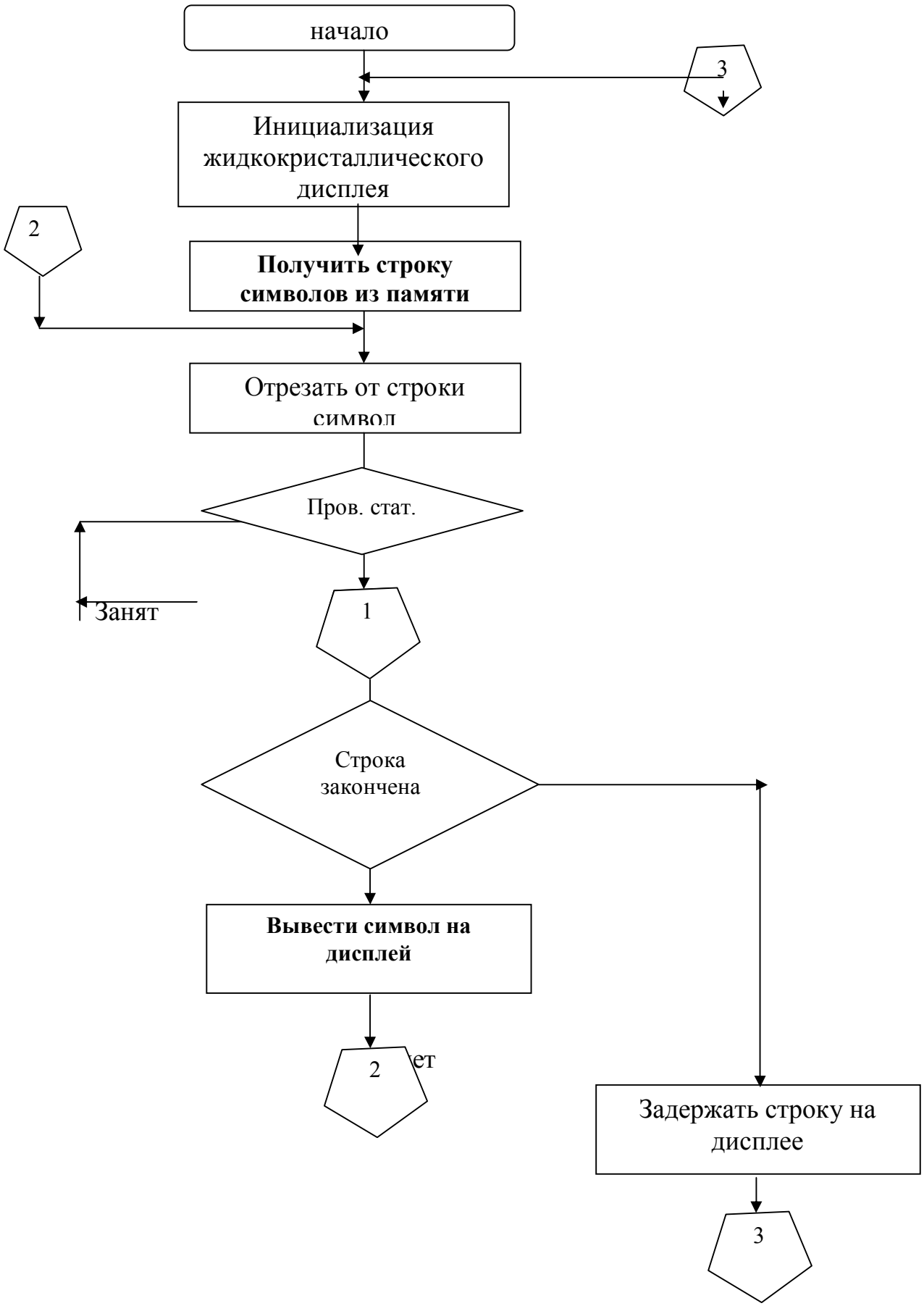

Рис. 2. Алгоритм программирования жидкокристаллического дисплея для вывода сообщения

Порядок выполнения работы

1) Изучить методическое руководство к выполнению работы.
2) Написать программу для программирования жидкокристаллического дисплея согласно варианту задания.
3) Откомпилировать программу, преобразовав ее в файл с расширением S19.
4) Загрузить программу в оперативную память, пользуясь программой Гипертерминала.
5) Запустить программу на выполнение.
6) Проверить результаты выполнения программы.
7) Защитить лабораторную работу представив по ней отчет (текст программы) и ответив на контрольные вопросы.

Варианты заданий

1) Вывести на жидкокристаллический монитор фразу “S novim godom” и заставить ее мигать.
2) Вывести на жидкокристаллический монитор фразу “Privet” и заставить ее переходить с нижней строки в верхнюю, и наоборот.
3) Вывести на жидкокристаллический монитор фразы “Privet” “Poka” которые будут сменять друг друга.
4) Вывести на жидкокристаллический монитор фразы “IVT-2010” и заставить ее перемещаться по верхней строке монитора справа налево а затем слева направо.
5) Вывести на жидкокристаллический монитор фразу “Pozdravlyau” и заставить ее мигать и перемещаться справа налево.
6) Вывести на жидкокристаллический дисплей фразы “Svet” в верхней строке и “Tma” в нижней строке и заставить их меняться местами.
7) Вывести на жидкокристаллический дисплей слово “IVT” которое будет перемещаться из левого верхнего угла в правый нижний, где и должно задержаться на 10 секунд, затем действия должны повториться.
8) Вывести на жидкокристаллический дисплей слово “IVT” которое будет перемещаться из правого нижнего угла в левый верхний, где и должно задержаться на 5 секунд, затем действия должны повторится.

Контрольные вопросы

1. Почему загрузка программы начинается с адреса 4000h?

2. Какое значение помещается в старшие 8 битов регистра при чтении из памяти младшего байта?
3. На сколько разрядов производятся арифметический и логический сдвиги?
4. По какому адресу находится регистр управляющего слова жидкокристаллического дисплея?
5. Какие действия подразумевает инициализация жидкокристаллического дисплея?
6. Как вывести символ в центре жидкокристаллического дисплея?
7. По какому адресу размещен регистр данных жидкокристаллического дисплея?
8. Для чего необходимо проверять сигнал занятости жидкокристаллического дисплея?
9. Как перевести курсор во вторую строку жидкокристаллического дисплея?
10. Каково назначение битов регистра управляющего слова жидкокристаллического дисплея?
11. какова инструкция для очистки дисплея?
12. какова инструкция для инкремента курсора дисплея?
13. Какова инструкция вывода данных в обе строки дисплея?

Лабораторная работа № 4
Работа с портами ввода/вывода микропроцессорного стенда

Цель работы: Получение практических навыков работы с подключаемыми периферийными устройствами через параллельный порт ввода/вывода микропроцессорного стенда.

Краткие теоретические сведения

Параллельный интерфейс ввода/вывода служит для обмена данными с внешними системами. По своим функциональным возможностям интерфейс аналогичен параллельному порту персонального компьютера. Интерфейс имеет 8 линий передачи данных, что позволяет отправлять данные со скоростью 1 байт/такт. Так же интерфейс имеет 4 входящие управляющие линии. Интерфейс не предназначен для приема/передачи плавающего аналогового сигнала. Через интерфейс передаются только четко установленные биты. Есть напряжение =

бит 1, нет напряжения = бит0. Интерфейс параллельного порта микропроцессорного стенда построен на базе микросхемы Intel 8255A.

LPT - один из самых старых портов, но он является самым удобным для управления электронными устройствами, так как сигналы порта постоянные и не нуждаются в каких либо преобразованиях. На базе данного порта можно организовать автоматизированную систему с датчиками и рабочими элементами или простое управление, какими либо электронными устройствами.

LPT имеет 25 выводов (пинов). Пины 18-25 - заземлены на корпус компьютера, пины 2-9 управляющие выводы. Также через эти пины возможен прием входных сигналов. Пины 10-13 и 15 служат только для приема сигналов. Пины 1, 14, 16 и 17 служат только для вывода сигналов.

Порт организован на транзисторной логике типа TTL. Уровень напряжения на выходах при отсутствии сигнала (логический ноль) составляет долю вольта. При установленном сигнале напряжение приближено к 5В, обычно ~ 4,90 В. Чтобы подать входной сигнал, необходимо соединить нужный входной пин с массой.

Теперь перейдем непосредственно к подключению схем или устройств к порту. Одним важным недостатком порта является отсутствие выводов источника питания, поэтому для Ваших схем понадобится независимый источник питания. Аналогично можно вывести питание с одного из разъемов питания изнутри компьютера. Чтобы использовать другой источник питания необходимо соединить его минус (землю) с одним из пинов 18-25 (с массой компьютера).

Для управления устройствами, потребляющими малый ток, необходимость в схеме сопряжения отпадает, так как мощность порта может обеспечить нормальную работу таких устройств. Таковыми являются, например светодиоды, электронные наручные часы, черно-белые ЖК-дисплеи. Управляющие входы каких либо приборов. Исключением является использование устройств, которые нагружают порт до предела. Такие устройства лучше подключать через резисторы для отвода тепла и предохранения порта (основная нагрузка будет приходится именно на эти резисторы, а не на резисторы порта), при отсутствии этих резисторов Вы подвергаете опасности сам порт. Также они ограничивают потребление тока в цепи. Сопротивление резисторов - в диапазоне 470 Ом - 2kОм (в зависимости от того, что подключаете). Точных расчетов тут не требуется: просто найдите сопротивление, при котором устройство или схема будет работать нормально, а

сами резисторы не раскалялись. Рекомендуется всегда соединять схему с портом через резисторы.

Для подключения более мощных устройств вплоть до высоковольтных используются транзисторы и реле. Транзистор тут используются в режиме электронного ключа. Подобрать транзистор, который будет соединен с портом нужно так, чтобы для его отпирания было достаточно мощности порта, а также, чтобы данный транзистор мог выдержать нагрузку управляемой цепи. Предположим, что требуется с помощью LPT-порта управлять электродвигателем в 12В. Разумеется, напрямую в порт его подключать нельзя. В цепь двигателя необходимо включить транзистор. Для этого используется его коллектор и эмиттер. Если используется транзистор обратной проводимости, то коллектор следует подключать к минусу внешнего источника, а эмитор к одному из выводов двигателя. Второй вывод двигателя на плюс источника. Если используется транзистор прямой проводимости, то просто нужно поменять его положение на противоположное. База транзистора должна быть соединена через резистор с одним из выходных пинов порта. При появлении сигнала на выходе ток попадает на базу транзистора, и тот отпирает цепь с 12 вольтовым двигателем.

Для управления высоковольтными схемами требуются реле. При использовании реле все делается так же, как в приведенном выше примере, только вместо двигателя устанавливается реле, через которое проходит высокое напряжение, например 220В. Ненадежность использования реле заключается в том, что высокое напряжение при неисправности реле может попасть в порт, и тогда вы лишитесь не только порта, но и компьютера. Чтобы избежать такого случая, используются оптопары (можно сделать самодельную оптопару). К порту подключен источник света, например светодиод. В управляемую цепь через несколько транзисторных каскадов включен фотодатчик. При попадании света на фотодатчик идет отпирание цепей по каскаду вплоть до включения реле. Таким образом, при неисправности реле высокое не может попасть в LPT порт.

Есть возможность расширения порта. Для этого понадобится источник питания. Расширение заключается в увеличении мощности и количестве входов и выходов. Однако при увеличении входов и выходов уменьшается максимальная скорость работы порта. В исходном состоянии порт может работать со скоростью 9,6 кбит/с или 9600 бод. При расширении порта скорость упадет в 3-4 раза. Принцип расширения заключается в изменении функции выводов. К примеру, сделаем 21 управляющих вывода вместо 12. Для такого

расширения потребуется четырехразрядный дешифратор и 15 триггеров типа DC. Эти элементы могут быть произвольными, но должны быть совместимы с транзисторной логикой TTL. Выводы 2-5 порта напрямую соединяются с входами дешифратора. Каждый выход дешифратора соединяется с входом "C" соответствующего триггера. Вывод порта 6 соединяется с входами "D" всех пятнадцати триггеров. Вывод порта 7 также соединяется с входами "R" всех 15 триггеров. Выходы 15 триггеров будут расширенными выходами управления и плюс 6 оставшихся на порте - итого 21. Принцип работы этой схемы заключается вследующем. На вывод порта 6 подается сигнал, в который нужно установить один из 15 выходов расширения. Этот сигнал будет присутствовать на входах "D" всех 15 триггеров. Затем на четыре входа дешифратора с выводов порта 2-5 подается число в двоичной системе, которое указывает номер расширенного вывода, для которого требуется сменить состояние. При этом на выходе дешифратора, на который указывает двоичное число, возникнет сигнал, который попадает на вход "C" соответствующего ему триггера. После этого на входы дешифратора подаются нули. Сигнал с входа выбранного триггера пропадает, и по его спаду триггер запоминает состояние, которое указывается на входе "D", и сразу же передается на выход. Чтобы отключить сигналы на выходах всех триггеров, подается импульсный сигнал с вывода 7 порта, который соединен с входами "R" всех триггеров, что приведет к их обнулению. Иными словами, указывается сигнал, который нужно выдать на расширенном выходе, затем выбирается один из выходов и происходит запись. Таким образом, порт можно расширить до 256 выходов и не обязательно посредством именно этих элементов. Таким же образом происходит расширение входов. Существует много вариантов расширения порта, это зависит от интеллекта и фантазии сборщика схемы.

Теперь следует рассмотреть работу LPT порта со стороны программного обеспечения. Начать следует с настройки самого порта. Чтобы полноценно использовать порт, в БИОСе нужно установить режим параллельного порта (parallel port mode), называемый ECP+EPP. Данный режим позволяет одновременно использовать любой из пинов 2-9 как вход и выход. Внимание, на старых компьютерах такого режима может и не быть. В этом случае устанавливать нужно режим порта EPP. Для управления LPT порта в компьютере предусмотрены три портовых адреса: 378h, 37Ah, 37Fh. Через порт 378h осуществляется доступ к выводам 2-9 (8 бит). Через порт 37Ah осуществляется доступ к остальным выводам, а также настройка порта. Порт 37Fh используется только для чтения и предназначен для получения сигналов с

пинов 2-9 (только в режиме TCP+EPP). Бит 5 порта 37Ah отвечает за направление ввода по пинам 2-9. Если данный бит установлен в 1, то при чтении с порта 378h ,будут приходить данные о входных сигналах, в обратном случае данные будут содержать информацию о сигналах, которые на данный момент выдаются портом. При чтении порта 37Ah мы будем получать информацию о направлении данных по пинам 2-9, а также состояния входов 10-13 и 15.При записи в порт 37Ah можно установить направление данных, а также установить сигналы на пинах 1, 14, 16, 17. Каждый пин LPT порта соответствует биту данных, полученных или записанных в выше описанные портовые адреса.

Предположим, что на пины 2-9 подключены светодиоды. Задача зажечь их через один. Для этого в порт 378h записываем число 10101010 в двоичном представлении или AAh в шестнадцатеричном. Каждый бит числа соответсвует пину. После записи в порт светодиоды будут светится через один.

Для ввода данных в порт представим схему из выключателей, которые одним выводом соединены с массой, вторым с пинами 2-9 соответственно. При чтении с порта 37Fh мы получим 8-разрядное число, биты которого будут соответствовать состояниям на каждом пине: 1 для разомкнутых выключателей и 0 для замкнутых.

Ниже приведен простейший алгоритм программы для работы с параллельным интерфейсом микропроцессорного стенда.

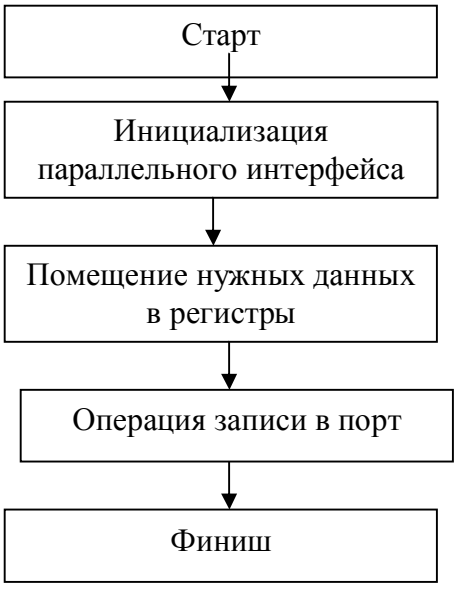

Рис.1. Алгоритм работы параллельного интерфейса

Адресные регистры для работы с параллельным портом микропроцессорного стенда:

```
ppi_pa equ 3001h ; адрес параллельного подпорта A
ppi_pb equ 3003h ; адрес параллельного подпорта A
ppi_pc equ 3005h ; адрес параллельного подпорта A
ppi_cntrl equ 3007h ; регистр контроля и статуса состояния порта.
```

Пример инициализации параллельного порта микропроцессорно стеда в операционном режиме:

```
mlo 80h, r4 ; инициализируем все порты как исходящие
ldi ppi_cntrl , r3
sb r4, r3 ; запись в регистр статуса/контроля
```

Пример программы, записывающей значение 55h в параллельный подпорт A:

```
;Определение данных и констант
ppi_cntrl equ 3007h ; регистр контроля/статуса
ppi_pa equ 3001h ; подпорт A
;инициализируем подпорт A
mlo 80h, r4 ;инициализируем все порты как исходящие
ldi ppi_cntrl , r3
sb r4, r3 ; запись а регистр контроля/статуса
mainloop:
mlo 55h, r3 ; получаем значение 55h в регистре r3
ldi ppi_pa , r4
sb r3, r4 ; запись в порт
bsr wait ; удержание данных на порту
jmpi mainloop
end
;+--------------------------------------------------------------------------+
;пример механизма Задержки
wait: mlo 2, r4
mhi ffh, r5
mlo ffh, r5
mlo 1, r6
wait1: sub r5, r6, r5
bne0 r5, wait1
decr r4
bne0 r4, wait1
;+--------------------------------------------------------------------------+
```

Порядок выполнения работы

1. Ознакомиться с теоретическими сведениями для выполнения лабораторной работы.
2. Разобрать приведенный пример программы для работы с параллельным портом ввода/вывода микропроцессорного стенда.
3. Подготовить ответы на контрольные вопросы.
4. Разработать программу для работы с параллельным портом согласно заданию.
5. Подключить микропроцессорный стенд к ПК.
6. Подключить периферийное устройство к микропроцессорному стенду согласно задания.
7. Загрузить программу в микропроцессорный стенд, запустить ее и показать, что программа взаимодействует с периферийным устройством.
8. Подготовить отчет о проделанной работе.

Варианты заданий

1. Подключить к параллельному порту устройство «Светодиодная сборка». Организовать на базе этого устройства «непрерывный бегущий огонь».
2. Подключить к параллельному порту устройство «Светодиодная сборка». Включить 1, 3 и 5 светодиоды
3. Подключить к параллельному порту устройство «Светодиодная сборка». Последовательно включить все 8 светодиодов, и затем последовательно их отключить.
4. Подключить к параллельному порту устройство «Электродвигатель». Подать сигнал на включение, а через 10 секунд на выключение электродвигателя.
5. Подключить к параллельному порту устройство «Светодиодная сборка». Вывести в цветовой индикации число 167 в бинарном виде.
6. Послать в параллельный порт число 24. Отобразить это числа на устройстве «Светодиодная сборка».

Контрольные вопросы

1. Какие интерфейсы ввода/вывода содержит микропроцессорный стенд?
2. Какой интерфейс самый быстрый?
3. Сколько исходящих линий содержит параллельный интерфейс микропроцессорного стенда?
4. На какой базе построен параллельный интерфейс микропроцессорного стенда?
5. Какие адресные регистры используются для работы с параллельным портом стенда?

Литература

1. Чжен Ю., Гибсон Т. Микропроцессоры семейство 8086/8088. – М.: Радио и связь, 1987.
2. Микропроцессорный комплект К1810/ Под ред. Ю.М.Казаринова – М.: Высшая школа, 1990.
3. Григорьев В.Л. Программирование однокристальных микропроцессоров – М.: Энергоатомиздат, 1987.
4. Микро- и миниЭВМ. Балашов Е.П., Григорьев В.П., Петров Г.А. – П.: Энергоатомиздат. 1989.
5. Проектирование радиоэлектронной аппаратуры на микропроцессорах, Алексеенко А.Т., Галицин А.А., Иванников А.Д.
6. Калабеков Б.А. Микропроцессоры и их применение в системах передачи и обработки сигналов. – М.: Радио и связь, 1988.
7. Основы проектирования микропроцессорных устройств. – М.: Энергоатомиздат, 1987.
8. Хвощ С.Т., Варианский Н.Н., Попов Е.А. Микропроцессоры и микроЭВМ в системах автоматического управления, Справочник. – Л.: Машиностроение, 1987.