Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический университет имени П.О. Сухого»

Кафедра «Автоматизированный электропривод»

СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА КР580ВМ80А НА БАЗЕ МИКРО-ЭВМ «ЭЛЕКТРОНИКА 580»

ПРАКТИЧЕСКОЕ РУКОВОДСТВО
по выполнению лабораторных работ по курсу «Микропроцессорные средства в автоматизированном электроприводе» для студентов специальности 1-53 01 05 «Автоматизированные электроприводы» дневной и заочной форм обучения

Гомель 2006

УДК 662.61(075.8)
ББК 32.973.26-04я73
С40

Рекомендовано научно-методическим советом факультета автоматизированных и информационных систем ГГТУ им. П. О. Сухого

Автор-составитель: *В. И. Луковников, В. А. Савельев*

Рецензент: канд. техн. наук, доц., зав. каф. «Промышленная электроника» ГГТУ им. П. О. Сухого *Б. А. Верига*

С40 **Система** команд микропроцессора КР580ВМ80А на базе микро-ЭВМ «Электроника 580» : практ. рук. по выполнению лаб. работ по курсу «Микропроцессорные средства в автоматизированном электроприводе» для студентов специальности 1-53 01 05 «Автоматизированные электроприводы» днев. и заоч. форм обучения / авт.-сост.: В. И. Луковников, В. А. Савельев. – Гомель : ГГТУ им. П. О. Сухого, 2006. – 35 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: http://gstu.local/lib. – Загл. с титул. экрана.

Практическое руководство содержит краткие теоретические сведения, задания и программы лабораторных работ по разделу «Система команд микропроцессора КР580ВМ80А» дисциплины «Микропроцессорные средства в автоматизированном электроприводе» и охватывает все группы команд процессора 580-й серии.

Для студентов специальности 1-53 01 05 «Автоматизированные электроприводы» дневной и заочной форм обучения.

УДК 662.61(075.8)
ББК 32.973.26-04я73

Лабораторная работа №1
Знакомство с работой на учебной микро-ЭВМ «Электроника 580»

1.1. Цель работы

1. Изучить порядок подготовки микро-ЭВМ «Электроника 580» к работе и правила техники безопасности при ее эксплуатации.

2. Ознакомиться с назначением, составом и техническими характеристиками учебной микро-ЭВМ «Электроника 580».

3. Изучить органы управления и режимы работы микро-ЭВМ «Электроника 580».

4. Изучить принцип адресного хранения информации и организацию памяти микро-ЭВМ «Электроника 580».

6. Получить начальные навыки работы с микро-ЭВМ «Электроника 580».

1.2. Указания мер безопасности

З А П Р Е Щ А Е Т С Я

1. Включать устройство в сеть, не подключив заземление.

2. Работать с устройством при снятых кожухах.

3. Бросать шнур во избежание поломок вилки.

4. Производить смену предохранителей не отсоединив шнур питания от сетевой розетки.

1.3. Краткие сведения из теории

1.3.1. Назначение и состав микро-ЭВМ «Электроника 580»

Микро-ЭВМ «Электроника 580» предназначена для создания микропроцессорных систем различного назначения и для отладки их программного обеспечения. Кроме того, она может быть использована для обучения инженерного персонала работе с микропроцессорным комплектом КР580.

В устройстве применены восьмиразрядный микропроцессор (МП) типа КР580ВМ80А и оперативное запоминающее устройство (ОЗУ) на ИС типа КР565РУ2А общей емкостью 2 килобайта (Кбайта) с адресным пространством $8000_{16}…87FF_{16}$ (нижний индекс «16» означает, что число записано в шестнадцатеричном коде).

Для осуществления диалога с пользователем предусмотрены клавиатура и цифровой дисплей, действие которых обеспечивается *программой-монитором* (аналог базовой системы ввода-вывода (BIOS)) объемом 1 Кбайт с адресацией $0000_{16}…03FF_{16}$, записанной в постоянное запоминающее устройство (ПЗУ) типа К573РФ22 емкостью 1 Кбайта, либо К573РФ2 емкостью 2 Кбайта.

Монитор позволяет загрузить в ОЗУ программу пользователя, переписывать ее на бытовой магнитофон, считывать с магнитофона в ОЗУ, выполнить программу пользователя в режиме отладки (в пошаговом режиме

или с остановкой по заданным условиям), осуществить прогон программы пользователя.

Клавиатура содержит 25 клавиш. Верхний и правый ряды клавиш содержат командные клавиши, функции которых описаны ниже. Оставшиеся 16 клавиш используются для ввода шестнадцатеричных цифр (от «0» до «F»). С помощью клавиши «RST» формируется сигнал сброса. Опрос остальных клавиш производится с помощью ИС интерфейса типа КР580ВВ55А, которая, кроме того, может использоваться для целей ввода-вывода.

Цифровой дисплей выполнен на восьми светодиодных семисегментных индикаторах. Его действие основано на принципе прямого доступа к памяти.

1.3.2. Технические характеристики микро-ЭВМ «Электроника 580»

Тип центрального процессора – параллельный.

Разрядность параллельно обрабатываемой информации – 8 двоичных разрядов.

Максимальное число внешних устройств – 180 устройств ввода, 180 устройств вывода.

Тактовая частота – 2,0 МГц.

Количество команд – 78. Система команд соответствует системе команд МП КР580ВМ80А.

Виды адресации – непосредственная, прямая, регистровая и косвенно-регистровая.

Ввод и вывод информации осуществляется в гексадецимальном (hexadecimal) коде (шестнадцатеричная система счисления).

Режимы работы: пошаговый, прогон в автоматическом режиме, прогон с остановом по заданному адресу и числу проходов.

Возможность прерывания – до 8 векторов.

Возможность согласования по быстродействию с медленно работающими устройствами.

1.3.3. Внешний вид и назначение органов управления микро-ЭВМ «Электроника 580»

На передней панели устройства расположены:

1) клавиатура, включающая 9 командных клавиш и 16 клавиш данных;

2) индикатор адреса («АДРЕС») и данных («РЕГИСТР» и «ДАННЫЕ»), содержащий 8 разрядов;

3) индикатор («ФЛАГ») состояния признаков переноса («С») и нуля («Z»);

4) тумблеры включения-выключения («СЕТЬ») и задания режима работы («ПРОГОН» и «ОТЛАДКА»);

5) индикаторы работы с магнитофоном («МФ. ВХОД» и «МФ. ВЫХОД»).

На задней стенке находится гнездо для подключения магнитофона, на передней стенке – разъемы для подключения других внешних устройств.

1.3.4. Назначение командных клавиш.

«RST» - служит для формирования сигнала сброса (ReSTart) устройства. Для начала работы с микро-ЭВМ «Электроника 580» после включения тумблера «сеть» необходимо нажать клавишу «RST», при этом на индикаторе должно появиться сообщение «8200 --??». Здесь и в дальнейшем знаком «-» обозначен пробел, а знаком «?» какая-либо ранее записанная в ОЗУ или случайная информация.

«ADDR» - служит для перевода устройства в режим задания адреса (ADDRess) ячейки памяти. Для чтения содержимого ячейки памяти с адресом XXXX необходимо нажать клавиши:

ADDR X X X X

После этого в разрядах 1-4 индикатора отобразится заданный адрес ячейки памяти, а в разрядах 7 и 8 - ее содержимое.

«NEXT» - служит для увеличения на единицу адреса индицируемой ячейки памяти или регистра МП. Нажатие клавиши «NEXT» выведет на индикатор информацию об адресе и содержимом следующей ячейки памяти.

«MEM» - служит для перевода устройства в режим записи данных в ячейку памяти (MEMory). Для записи данных в ячейку памяти с адресом XXXX необходимо нажать последовательность клавиш:

ADDR X X X X MEM

После этого в четырех левых разрядах дисплея высветится адрес ячейки памяти, в двух правых разрядах - ее содержимое и появится десятичная точка. При отсутствии на дисплее десятичной точки данные в память записаны не будут. Дальнейшее нажатие одной или двух цифровых клавиш изменят содержимое данной ячейки. В двух правых разрядах появятся новые данные. Если в набранных данных имеется ошибка, то ее можно исправить нажатием требуемых цифровых клавиш. Сколько бы клавиш не было нажато, в любом случае, в двух правых разрядах высвечиваются значения двух последних нажатых клавиш.

Для перехода к адресу следующей ячейки необходимо нажать клавишу «NEXT». При этом нет необходимости повторного нажатия клавиши

«MEM». Повторное нажатие клавиши «MEM» уменьшает на единицу адрес ячейки памяти.

При попытке ввести данные без предварительного нажатия клавиши «MEM», а так же, если на дисплее установлен адрес области ПЗУ, либо адрес, физически отсутствующий в ОЗУ, на дисплее высветится надпись «Err». В этом случае для восстановления предыдущего адреса и разрешения ввода данных в память необходимо нажать клавишу «MEM».

«CLR» - (CLeaR) служит для восстановления начального значения адреса или данных, если после их ввода не нажималось других командных клавиш.

«REG» - служит для отображения содержимого восьмиразрядного регистра (REGister) МП. Для чтения содержимого одного из регистров необходимо нажать следующие клавиши:

REG X

где X - клавиша с наименованием соответствующего регистра:

A - аккумулятор;

B, C, D, E, H, L - регистры общего назначения;

F - регистр признаков (флаговый);

SP – указатель стека.

После нажатия указанных клавиш в разряде 5 индикатора отобразится имя регистра, а в разрядах 7 и 8 его содержимое. Для записи данных в регистр после нажатия указанных клавиш необходимо нажать одну или две шестнадцатиричных клавиш.

«STEP» - служит для выполнения очередной команды МП.

«BRK» - (BReaK) служит для задания адреса контрольной точки в программе.

Можно выполнять программы пользователя с введением контрольных точек, т.е. адресов на которых необходимо прервать выполнение программы для проверки промежуточных результатов.

«RUN» - служит для запуска программы на выполнение с остановом на введенной контрольной точке или команде останова МП.

1.3.5. Организация памяти микро-ЭВМ «Электроника 580»

Во время работы микропроцессора команды и данные необходимо хранить и выбирать по мере необходимости. Для этой цели служат запоминающие устройства (ЗУ). Различают ЗУ двух типов: постоянные запоминающие устройства и оперативные запоминающие устройства.

ПЗУ используются для хранения команд и констант, т.е. такой информации, которая остается неизменной при работе микропроцессорной системы. В англоязычной технической литературе аббревиатуре ПЗУ соответствует ROM (Read Only Memory, т.е. память только для чтения).

ПЗУ делятся на три класса:

- масочные ПЗУ, программируемые изготовителем (ROM);
- однократно программируемые (пользователем) ПЗУ (PROM);
- многократно программируемые ПЗУ (EPROM).

ПЗУ микро-ЭВМ «Электроника 580» имеет максимально возможную емкость 8 Кбайт (при наличии дополнительных ИС ПЗУ) и может содержать до восьми ИС типа К573РФ22 с организацией 1024х8 бит, либо до четырех ИС типа К573РФ2 с организацией 2048х8 бит. Указанный тип ИС представляет собой многократно программируемое ПЗУ со стиранием информации ультрафиолетовым облучением.

Как минимум одна ИС К573РФ22 всегда установлена и пространство с адресами с 0000_{16} по $03FF_{16}$ (1 Кбайт) занято под программу-монитор. Из них 768 ячеек с адресами с 0000_{16} по $02FF_{16}$ занимает программа-монитор. Следующие 256 ячеек с адресами от 0300_{16} по $03FF_{16}$ отводятся для дополнительной области монитора, которую пользователь может запрограммировать в случае необходимости расширить функциональные возможности программы-монитора.

Для использования адресного пространства с 0400_{16} по $0FFF_{16}$ (дополнительные 3 Кбайта) необходимо установить дополнительные ИС. Для использования адресного пространства с 1000_{16} по $1FFF_{16}$ необходимо установить дополнительную плату ЗУ.

ОЗУ используются для хранения данных, изменяющихся в процессе работы системы, например, исходных данных, промежуточных результатов расчетов. Аббревиатуре ОЗУ соответствует англоязычная RAM (Random Access Memory, т.е. память произвольного доступа). Информация может быть записана в ОЗУ и считана из него в любом порядке адресов.

Различают два типа ОЗУ:

- статические (SRAM);
- динамические (DRAM).

Динамические ОЗУ эффективны при построении памяти относительно большого объема. Память же малого объема обычно реализуют на более быстродействующих статических элементах.

ОЗУ микро-ЭВМ «Электроника 580» построено на базе ИС КР565РУ2А. Эта ИС представляет собой статическое ОЗУ и имеет объем 1 Кбит с организацией 1024х1 бит. Для получения восьмиразрядного слова используют 8 таких ИС.

Общая емкость ОЗУ составляет 2 Кбайта, а при подключении дополнительной платы емкость ОЗУ возрастает до 4 Кбайт. В адресном пространстве микро-ЭВМ «Электроника 580» под встроенное ОЗУ отведены 2048 адресов начиная с 8000_{16} и заканчивая $87FF_{16}$, а также 2048 адресов с 8800_{16} по $8FFF_{16}$ зарезервированы под дополнительную плату ЗУ.

Шина адреса микропроцессора КР580ВМ80А, который применен в микро-ЭВМ «Электроника 580», имеет 16 разрядов. Т.е. при ее помощи можно адресовать до $2^{16} = 65536$ ячеек памяти (64 Кбайта). Как было отме-

чено выше, память микро-ЭВМ включает значительно меньшее количество ячеек. Оставшееся адресное пространство находится в распоряжении пользователя и может стать доступными при установке дополнительных ИС памяти и разработке дешифратора вновь занимаемых адресов.

1.4. Задания к лабораторной работе

Задание №1.

В ячейку памяти 8210_{16} записать число $1A_{16}$. Просмотреть содержимое двух последующих ячеек и вернуться к исходной ячейке. Записать в таблицу 1.1 порядок нажатия клавиш, информацию на дисплее при нажатии каждой клавиши и комментарий выполняемых действий.

Таблица 1.1

Клавиши	Информация на индикаторе		Комментарий
	1 2 3 4	5 6 7 8	

Задание №2.

В один из регистров общего назначения записать число $A1_{16}$. Записать в таблицу 1.1 порядок нажатия клавиш, информацию на дисплее при нажатии каждой клавиши и комментарий выполняемых действий.

Задание №3.

Ввести в устройство программу сложения двух чисел 57_{16} и $B5_{16}$:

Адрес	Код	Мнемокод	Примечание
8200	3E	MVI A	Запись в аккумулятор числа 57_{16}
8201	57	57	
8202	06	MVI B	Запись в регистр B числа $B5_{16}$
8203	B5	B5	
8204	80	ADDB	Сложение чисел 57_{16} и $B5_{16}$
8205	76	HLT	Останов

Выполняя программу в пошаговом режиме, после выполнения каждого шага, записывать в таблицу 1.2 содержимое регистров A и B.

Таблица 1.2

№ шага	Рег. A	Рег. B	Комментарий
0	*XX*	*XX*	*Случайные данные*
1			

1.5. Содержание отчета

В отчете привести наименование и цель работы, а также заполненные в соответствии с заданиями 1-3 таблицы 1.1 и 1.2 с подробными комментариями.

1.6. Контрольные вопросы

1. Объяснить отличие терминов «микро-ЭВМ», «микропроцессор», «центральный процессор».
2. Что такое оперативная память микро-ЭВМ?
3. Чем отличается постоянное запоминающее устройство от оперативного?
4. Назначение программы-монитора?
5. Описать назначение функциональных клавиш микро-ЭВМ «Электроника 580».
6. Каким образом просмотреть содержимое ячейки памяти, регистра, регистровой пары, указателя стека?
7. Каким образом можно осуществлять переход от одной ячейки памяти к другой?
8. Как запустить программу на выполнение в пошаговом режиме?
9. Что представляет собой запись программы в мнемокодах и в машинных кодах?

Лабораторная работа №2
Изучение команд пересылки данных

2.1. Цель работы

1. Ознакомиться со способами адресации микропроцессора.
2. Изучить действие команд пересылки данных.
3. Получить практические навыки составления простейших программ.

2.2. Краткие сведения из теории

2.2.1. Способы адресации микропроцессора

Непосредственная адресация (непосредственная загрузка регистра)**.** При таком способе адресации код операции команды размещается в первом байте. Сразу же за кодом операции следуют данные, занимающие один или два байта. Эти данные берутся не из памяти, а предоставляются машине программистом при записи команды.

Прямая адресация. Команды с прямой адресацией могут иметь два или три байта. Первый байт команды предназначен для кода операции, второй и, если имеется, третий - для адреса. Адрес указывает область памяти, в которой находятся подлежащие обработке данные.

Регистровая адресация (неявная). Не смотря на сказанное выше, операции по пересылке данных можно производить и при помощи однобайтовых команд, т.е. без явного указания адреса данных. Это возможно за счет того, что 1-байтовые команды адресуются к данным, расположенным

не в памяти, а к данным, загруженным либо в регистр, либо в регистровую пару.

Косвенно-регистровая адресация (косвенная). Команды с косвенно-регистровой адресацией также являются однобайтовыми. Но в отличии от команд с регистровой адресации в регистровой паре, определяемой кодом команды, содержатся не сами данные, а полный 16-разрядный адрес ячейки памяти, в которой эти данные находятся.

2.2.3. Команды пересылки данных

К командам с непосредственной адресацией относятся:

Непосредственная загрузка регистра	MVI r, данные
Непосредственная загрузка регистровой пары	LXI rp, данные
Непосредственная загрузка ячейки памяти	MVI M,данные

К командам с прямой адресацией относятся:

Прямая загрузка аккумулятора	LDA, адрес
Прямая загрузка аккумулятора из порта ввода	IN, адрес
Прямая запись аккумулятора в память	STA, адрес
Прямая запись аккумулятора в порт вывода	OUT, адрес
Прямая загрузка регистровой пары HL	LHLD, адрес
Прямая запись регистровой пары HL в память	SHLD, адрес

К командам с регистровой адресацией относятся:

Пересылка из регистра в регистр	MOV r1, r2
Обмен данными между регистровыми парами HL и DE	XCHG
Обмен данными между двумя верхними ячейками стека и регистровой парой HL	XTHL
Передача в указатель стека содержимого регистровой пары HL	SPHL

К командам с косвенно-регистровой адресацией относятся:

Косвенная загрузка аккумулятора	LDAX rp
Косвенная запись аккумулятора в память	STAX rp
Пересылка из памяти в регистр	MOV r, M
Пересылка из регистра в память	MOV M, r

2.3. Задания к лабораторной работе

Задание №1.

1. Ввести в память микро-ЭВМ следующую программу:

Мнемокод	Операнд
LDA	813F
ADD A[1]	
STA	8140
HLT[2]	

2. Записать в ячейку памяти по адресу $813F_{16}$ любое число, например 12_{10}, а по адресу 8140_{16} и в аккумулятор - число 00.

3. Выполняя программу в пошаговом режиме, после выполнения каждого шага, записывать в таблицу 2.1 содержимое аккумулятора и ячеек памяти с адресами $813F_{16}$ и 8140_{16}.

Таблица 2.1

№ шага	Рег. A	813F	8140	Комментарий
0	*00*	*12*	*00*	*Исходные данные*
1				
2				

4. Прокомментировать действия микро-ЭВМ на каждом шаге.

Задание №2.

1. Ввести в память микро-ЭВМ следующую программу:

Мнемокод	Операнд
LXI H	813F
MOV A, M	
ADD A	
INX H[3]	
MOV M, A	
HLT	

2. Записать в ячейку памяти по адресу $813F_{16}$ любое число, например 12_{10}, а по адресу 8140_{16} и в регистры A, H, L - число 00.

3. Выполняя программу в пошаговом режиме, после выполнения каждого шага, записывать в таблицу 2.2 содержимое регистров A, H, L и ячеек памяти с адресами $813F_{16}$ и 8140_{16}.

4. Прокомментировать действия микро-ЭВМ на каждом шаге.

---

[1] Команда **ADD A** производит сложение содержимого аккумулятора с самим собой.

[2] Команда **HLT** осуществляет останов программы.

[3] Команда **INX H** производит увеличение на 1 содержимого регистровой пары HL.

Таблица 2.2

№ шага	Рег. A	Рег. H	Рег. L	813F	8140	Комментарий
0	*00*	*00*	*00*	*12*	*00*	*Исходные данные*
1						
2						

2.4. Содержание отчета

В отчете привести наименование и цель работы, тексты программ задач №1 и №2, заполненные таблицы 2.1 и 2.2 с подробными комментариями.

2.5. Контрольные вопросы

1. Прокомментируйте выполнение следующих команд пересылки данных

а) с непосредственной адресацией;

б) с прямой адресацией;

в) с регистровой адресацией;

г) с косвенно-регистровой адресацией.

2. Какие способы адресации использованы в программе задания №2.

3. Придумайте другие примеры использования команд пересылки данных с различными способами адресации.

Лабораторная работа №3
Изучение арифметических команд (часть 1)

3.1. Цель работы

1. Изучить действие арифметических команд сложения и десятичной коррекции.

2. Получить практические навыки составления программ.

3.2. Краткие сведения из теории

3.2.1. Команды сложения

Все рассматриваемые в данном разделе команды, кроме последней, складывают данные, на которые имеется указание в команде, с данными, находящимися в аккумуляторе. Результат выполнения команды «сложение» (сумма) помещается в аккумулятор. Исходное содержимое аккумулятора теряется. При этом могут изменяться разряды (флаги) регистра состояния.

К командам «обычного» сложения относятся:

Сложение с регистром ADD r

Косвенное сложение с памятью ADD M

Сложение с непосредственными данными ADI, данные
Сложение содержимого регистровых пар DAD rp

3.2.2. Команды сложения с переносом

Кроме обычных команд сложения, рассмотренных выше, существует еще одна разновидность этих команд. Отличаются они тем, что при их выполнении в сложении участвует содержимое разряда (флага) переноса регистра состояния. Таким образом, содержимое области памяти прибавляется к содержимому аккумулятора, а затем к полученной сумме прибавляется значение разряда переноса регистра состояния, полученное на предыдущем шаге вычисления.

К данной группе команд относятся:

Сложение с учетом переноса с регистром ADC r
Сложение с учетом переноса с памятью косвенное ADC M
Сложение с учетом переноса с непосредственными данными ACI, данные

3.2.3. Команда десятичной коррекции

Наличие команды «десятичная коррекция» позволяет микропроцессору выполнять некоторые виды обработки двоично-десятичных чисел.

Команда «десятичная коррекция» служит для исправления результатов двоично-десятичной обработки, представляющих собой числа, большие, чем 1001_2.

Десятичная коррекция аккумулятора DAA

3.3. Задания к лабораторной работе

Задание №1.

Произвести сложение двух чисел 51120_{10} и 12460_{10}, используя косвенно-регистровую адресацию.

Для этого необходимо:

а) указанные числа перевести из десятичного формата в шестнадцатеричный;

б) составить программу вычисления в соответствии со следующим алгоритмом:

1. Загрузить младший байт (МБ) первого слагаемого в аккумулятор.
2. Сложить младший байт второго слагаемого с аккумулятором.
3. Результат (младший байт суммы) записать в память.
4. Загрузить старший байт (СБ) первого слагаемого в аккумулятор.

5. Сложить старший байт второго слагаемого с аккумулятором с учетом бита переноса.

6. Результат (старший байт суммы) записать в память.

7. Останов.

Текст программы и исходные данные желательно расположить в адресном пространстве ОЗУ в соответствии с таблицей 3.1.

Таблица 3.1

Адрес	Содержимое
8000 … 81FF	Текст программы
8200	МБ 1-го слагаемого
8201	СБ 1-го слагаемого
8202	МБ 2-го слагаемого
8203	СБ 2-го слагаемого
8204	МБ суммы
8205	СБ суммы

в) ввести исходные данные и выполнить программу в пошаговом режиме;

г) после выполнения каждого шага программы заносить в таблицу 3.2 содержимое регистров A, H, L и ячеек памяти с адресами 8204_{16} и 8205_{16};

Таблица 3.2

№ шага	Рег. A	Рег. H	Рег. L	8204	8205	Комментарий
0	*00*	*00*	*00*	*00*	*00*	*Исходные данные*
1						
2						

д) перевести результат из шестнадцатеричного формата в десятичный.

Задание №2.

Произвести сложение двух чисел 89_{10} и 76_{10}, в двоично-десятичном формате, используя косвенно-регистровую адресацию.

Для этого необходимо:

а) указанные числа перевести из десятичного формата в двоично-десятичный, а затем в шестнадцатеричный;

б) составить программу вычисления в соответствии со следующим алгоритмом:

1. Загрузить младший байт первого слагаемого в аккумулятор.

2. Сложить младший байт второго слагаемого с аккумулятором.

3. Произвести десятичную коррекцию результата.

4. Результат (младший байт суммы) записать в память.

5. Загрузить старший байт первого слагаемого в аккумулятор.

6. Сложить старший байт второго слагаемого с аккумулятором с учетом бита переноса.

7. Произвести десятичную коррекцию результата.

8. Результат (старший байт суммы) записать в память.

9. Останов.

Текст программы и исходные данные желательно расположить в адресном пространстве ОЗУ в соответствии с таблицей 3.3.

Таблица 3.3

Адрес	Содержимое
8000 … 81FF	Текст программы
8200	МБ 1-го слагаемого
8201	СБ 1-го слагаемого
8202	МБ 2-го слагаемого
8203	СБ 2-го слагаемого
8204	МБ суммы
8205	СБ суммы

в) ввести исходные данные и выполнить программу в пошаговом режиме;

г) после выполнения каждого шага программы заносить в таблицу 3.4 содержимое регистров A, H, L и ячеек памяти с адресами 8204 и 8205;

Таблица 3.4

№ шага	Рег. A	Рег. H	Рег. L	8204	8205	Комментарий
0	*00*	*00*	*00*	*00*	*00*	*Исходные данные*
1						
2						

д) перевести результат из шестнадцатеричного формата в десятичный.

3.4. Содержание отчета

В отчете привести наименование и цель работы, тексты программ задач №1 и №2, записанные в мнемокодах и машинных кодах, заполненные таблицы 3.2 и 3.4 с подробными комментариями.

3.5. Контрольные вопросы

1. Прокомментируйте выполнение следующих команд сложения:
   а) с непосредственной адресацией;
   б) с регистровой адресацией;
   в) с косвенно-регистровой адресацией.
2. Прокомментируйте выполнение следующих команд сложения с переносом:
   а) с непосредственной адресацией;
   б) с регистровой адресацией;
   в) с косвенно-регистровой адресацией.
3. Прокомментируйте выполнение команды десятичной коррекции аккумулятора.
4. Запишите тексты программ заданий №1 и №2 с использованием различных способов адресации.
5. Как изменятся значения флагов регистра состояния и содержимое аккумулятора в результате выполнения операций:
   а) сложения;
   б) сложения с переносом;
   в) десятичной коррекции.
6. Придумайте другие примеры использования команд сложения, сложения с переносом, десятичной коррекции.

Лабораторная работа №4
Изучение арифметических команд (часть 2)

4.1. Цель работы

1. Изучить действие арифметических команд вычитания и приращения.
2. Получить практические навыки составления программ.

4.2. Краткие сведения из теории

4.2.1. Команды вычитания

Команды «вычитания» похожи на команды «сложения». Все рассматриваемые в данном разделе команды вычитают данные, на которые имеется указание в команде, из данных, находящихся в аккумуляторе. Другими словами, в аккумуляторе всегда размещается уменьшаемое, а имеющийся в составе команды «вычитания» адрес, задает местоположение вычитаемого.

Результат выполнения команды «вычитание» (разность) помещается в аккумулятор. Исходное значение уменьшаемого теряется. При этом соответствующим образом устанавливаются разряды регистра состояния.

Во всех командах «вычитания» флаг переноса устанавливается при наличии заема в старший разряд, в противном случае сбрасывается. Флаг

дополнительного переноса устанавливается при наличии заема из старших четырех разрядов в младшие, в противном случае сбрасывается.

Для выполнения вычитания служат следующие команды:

Вычитание с регистром	SUB r
Косвенное вычитание с памятью	SUB M
Вычитание с непосредственными данными	SUI, данные

4.2.2. Команды вычитания с заемом

Аналогично тому, как имеются команды «сложение с переносом», существуют команды «вычитание с заемом». Заем происходит, когда вычитаемое оказывается больше уменьшаемого. При этом устанавливается в 1 разряд переноса (заема).

К командам вычитания с заемом относятся:

Вычитание с регистром и заемом	SBB r
Косвенное вычитание с памятью и заемом	SBB M
Вычитание с непосредственными данными и заемом	SBI, данные

4.2.3. Команды положительного и отрицательного приращения

Команды «приращение положительное» и «приращение отрицательное» являются в сущности специальными арифметическими командами. При выполнении команды «приращение положительное» к текущему содержимому регистра прибавляется 1. По команде «приращение отрицательное» из текущего содержимого регистра вычитается 1.

К данной группе команд относятся:

Положительное приращение регистра	INR r
Положительное приращение памяти	INR M
Отрицательное приращение регистра	DCR r
Отрицательное приращение памяти	DCR M
Положительное приращение регистровой пары	INX rp
Отрицательное приращение регистровой пары	DCX rp

4.3. Задания к лабораторной работе

Задание №1.

Произвести вычитание двух чисел $30\ 812_{10}$ и $12\ 460_{10}$, используя косвенно-регистровую адресацию.

Для этого необходимо:

а) указанные числа перевести из десятичного формата в шестнадцатеричный;

б) составить программу[4] вычисления в соответствии со следующим алгоритмом:

1. Загрузить младший байт (МБ) уменьшаемого в аккумулятор.
2. Вычесть младший байт вычитаемого из аккумулятора.
3. Результат (младший байт разности) записать в память.

Таблица 4.1

Адрес	Содержимое
8000 … 81FF	Текст программы
8200	МБ уменьшаемого
8201	СБ уменьшаемого
8202	МБ вычитаемого
8203	СБ вычитаемого
8204	МБ разности
8205	СБ разности

4. Загрузить старший байт (СБ) уменьшаемого в аккумулятор.
5. Вычесть старший байт вычитаемого из аккумулятора с учетом бита переноса.
6. Результат (старший байт разности) записать в память.
7. Останов.

в) ввести исходные данные и выполнить программу в пошаговом режиме;

г) после выполнения каждого шага программы заносить в таблицу 4.2 содержимое регистров A, H, L и ячеек памяти с адресами 8204 и 8205;

д) перевести результат из шестнадцатеричного формата в десятичный.

Таблица 4.2

№ шага	Рег. A	Рег. H	Рег. L	8204	8205	Комментарий
0	*00*	*00*	*00*	*00*	*00*	*Исходные данные*
1						
2						

Задание №2.

Произвести умножение двух чисел 12_{10} и 3_{10}, используя косвенно-регистровую адресацию.

Для этого необходимо:

[4] Текст программы и исходные данные желательно расположить в адресном пространстве ОЗУ в соответствии с таблицей 4.1.

а) указанные числа перевести из десятичного формата в шестнадцатеричный;

б) составить программу[5] вычисления в соответствии со следующим алгоритмом:

1. Загрузить первый сомножитель в регистр B.
2. Загрузить второй сомножитель в регистр C.
3. Загрузить содержимое регистра B в аккумулятор.
4. Уменьшить на 1 содержимое регистра C.

Таблица 4.3

Адрес	Содержимое
8000 … 81FF	Текст программы
8200	1-й сомножитель
8201	2-й сомножитель
8202	Произведение

5. Сложить содержимое аккумулятора с содержимым регистра B;
6. Уменьшить на 1 содержимое регистра C.
7. Если содержимое регистра C не равно 0, перейти к п.5[6];
8. Результат (произведение) записать в память.
9. Останов.

в) ввести исходные данные и выполнить программу в пошаговом режиме;

г) после выполнения каждого шага программы заносить в таблицу 4.4 содержимое регистров A, B, C, H, L, ячейки памяти с адресом 8202 и состояние флага нулевого результата;

Таблица 4.4

№ шага	Регистры					8202	Флаг нуля	Комментарий
	A	B	C	H	L			
0	*00*	*00*	*00*	*00*	*00*	*00*	-	*Исходные данные*
1								
2								

д) перевести результат из шестнадцатеричного формата в десятичный.

[5] Текст программы и исходные данные желательно расположить в адресном пространстве ОЗУ в соответствии с таблицей 4.3.

[6] Команда **JNZ, адрес** осуществляет переход к выполнению команды, записанной по указанному адресу, при отсутствии нулевого результата (флаг нуля не установлен)

4.4. Содержание отчета

В отчете привести наименование и цель работы, тексты программ задач №1 и №2, записанные в мнемокодах и машинных кодах, заполненные таблицы 4.2 и 4.4 с подробными комментариями.

4.5. Контрольные вопросы

1. Прокомментируйте выполнение следующих команд вычитания:
   а) с непосредственной адресацией;
   б) с регистровой адресацией;
   в) с косвенно-регистровой адресацией.
2. Прокомментируйте выполнение следующих команд вычитания с заемом:
   а) с непосредственной адресацией;
   б) с регистровой адресацией;
   в) с косвенно-регистровой адресацией.
3. Прокомментируйте выполнение команд положительного и отрицательного приращения
   а) регистра;
   б) ячейки памяти;
   в) регистровой пары.
4. Запишите тексты программ заданий №1 и №2 с использованием различных способов адресации.
5. Как изменятся значения флагов регистра состояния и содержимое аккумулятора в результате выполнения операций:
   а) вычитания;
   б) вычитания с заемом;
   в) положительного и отрицательного приращения.
6. Придумайте другие примеры использования команд сложения, сложения с переносом, десятичной коррекции.

Лабораторная работа №5
Изучение логических команд

5.1. Цель работы

1. Изучить действие логических команд.
2. Получить практические навыки составления программ.

5.2. Краткие сведения из теории

5.2.1. Логические команды

Команды этой группы предназначены для выполнения логических, или *булевых*, операций над данными, содержащимися в регистрах, ячейках памяти, а также над флагами условий.

Во-первых, это четыре команды, с помощью которых могут быть реализованы основные логические операции: И, ИЛИ, исключающее ИЛИ и НЕ.

Во-вторых, команда «сравнение».

В-третьих, команды циклического сдвига.

5.2.2. Команды И, ИЛИ, исключающее ИЛИ и НЕ

Рассматриваемые логические операции являются побитовыми. Они всегда выполняются над содержимым аккумулятора и каким-то другим словом из регистра или из памяти. По окончании логической операции результат загружается в аккумулятор. Исходное содержимое аккумулятора при этом теряется. При этом может измениться состояние флагов регистра состояния. По результатам этих операций флаги переноса и дополнительного переноса сбрасываются (принимают значение 0).

Команда И может быть следующих трех видов:

И над регистром и аккумулятором	ANA r
И над косвенно адресуемой памятью и аккумулятором	ANA M
И над непосредственными данными и аккумулятором	ANI, данные

Команда ИЛИ также может быть трех видов:

ИЛИ над регистром и аккумулятором	ORA r
ИЛИ над косвенно адресуемой памятью и аккумулятором	ORA M
ИЛИ над непосредственными данными и аккумулятором	ORI, данные

Команда исключающее ИЛИ может быть следующих видов:

Исключающее ИЛИ над регистром и аккумулятором	XRA r
Исключающее ИЛИ над косвенно адресуемой памятью и аккумулятором	XRA M
Исключающее ИЛИ над непосредственными данными и аккумулятором	XRI, данные

Команда НЕ имеет следующий вид:

Инверсия аккумулятора	CMA

При выполнении команды «инверсия» над аккумулятором осуществляется поразрядное инвертирование содержимого аккумулятора, т. е. каждый бит, имевший значение логической 1, принимает значение логического 0 и наоборот, при этом формируется *обратный код* (дополнение до 1) слова данных.

5.2.3. Команды сравнения

Выполнение команды «сравнение» аналогично выполнению команды «вычитание», однако ее результат не загружается в аккумулятор. Хотя по окончании операции исходные данные в аккумуляторе не изменяются, разрядам слова регистра состояния, присваиваются значения, соответствующие особенностям полученного результата:

- флаг нуля (Z) устанавливается, если содержимое регистра или байта данных совпадает с содержимым аккумулятора;
- флаги знака (S) и переноса (C) устанавливаются, если содержимое регистра или байта данных больше содержимого аккумулятора;
- флаг вспомогательного переноса (AC) устанавливается, если содержимое младших четырех разрядов регистра или байта данных больше содержимого соответствующих разрядов аккумулятора;
- флаг четности (P) устанавливается если байт разности между содержимым аккумулятора и содержимым регистра или байта данных содержит четное число единиц.

Существует три вида команд сравнения:

Сравнение регистра с аккумулятором	CMP r
Сравнение косвенно адресуемой памяти с аккумулятором	CMP M
Сравнение непосредственных данных с аккумулятором	CPI, данные

5.2.4. Команды циклического сдвига

Существует четыре вида команд циклического сдвига:

Циклический сдвиг аккумулятора влево	RLC
Циклический сдвиг аккумулятора вправо	RRC
Циклический сдвиг аккумулятора влево через бит переноса	RAL
Циклический сдвиг аккумулятора вправо через бит переноса	RAR

При выполнении команды «циклический сдвиг аккумулятора влево» содержимое старшего разряда аккумулятора передается в его младший разряд и в разряд переноса (C) регистра состояния. Содержимое каждого разряда аккумулятора перемещается в соседний, старший разряд.

Аналогично выполняется команда «циклический сдвиг аккумулятора вправо».

Командой «циклический сдвиг аккумулятора влево через бит переноса» все данные, находящиеся в аккумуляторе и разряде переноса (C) регистра состояния, сдвигаются на одну позицию влево. При этом содержимое флага переноса перемешается в младший разряд аккумулятора, а со-

держимое старшего разряда аккумулятора перемещается в разряд переноса (С) регистра состояния.

Команда «циклический сдвиг аккумулятора вправо» выполняет аналогичное действие, при этом данные перемещаются по кольцу в противоположном направлении.

5.3. Задания к лабораторной работе

Задание №1.

В заданном массиве (10x1) среди чисел, содержащих во втором и пятом разрядах двоичного кода единицу, найти наибольшее.

Для этого необходимо:

а) составить программу[7] обработки массива в соответствии со следующим алгоритмом:

1. Загрузить в один из регистров общего назначения (РОН) количество элементов массива.
2. Загрузить в регистровую пару HL начальный адрес массива.
3. Обнулить регистр результата.
4. Сравнить текущий элемент массива с «маской».
5. Если второй и пятый разряды двоичного кода текущего элемента не содержат единицы, перейти к п.9.
6. Сравнить текущий элемент с числом, находящемся в регистре результата.
7. Если текущий элемент меньше содержимого регистра результата, перейти к п.9.
8. Записать текущий элемент в регистр результата.
9. Увеличить на 1 текущий адрес массива.
10. Уменьшить на 1 число элементов массива.
11. Если число элементов массива не равно 0, то перейти к п.4.
12. Останов.

б) ввести исходный массив (при вводе массива предусмотреть не менее трех различных элементов, содержащих единицу во втором и пятом разрядах двоичного кода) и выполнить программу в пошаговом режиме;

в) после выполнения каждого шага программы заносить в таблицу 5.2 содержимое регистров A, B, C, H, L, а также состояние флагов нулевого результата и переноса;

Задание №2.

Произвести вычитание двух чисел методом сложения с дополнительным кодом (см. лабораторную работу №4).

Для этого необходимо:

а) составить программу[8] в соответствии со следующим алгоритмом:

---

[7] Текст программы и исходные данные желательно расположить в адресном пространстве ОЗУ в соответствии с таблицей 5.1.

Таблица 5.1

Адрес	Содержимое
8000… 81FF	Текст программы
8200	Массив
…	
820A	

Таблица 5.2

№ шага	Регистры					Флаг нуля	Флаг переноса	Комментарий
	A	B	C	H	L			
0	*00*	*00*	*00*	*00*	*00*	*X*	*X*	*Исходные данные*
1								
2								

1. Загрузить вычитаемое в аккумулятор.
2. Инвертировать содержимое аккумулятора.
3. Увеличить полученное число на единицу.
4. К полученному дополнительному коду вычитаемого прибавить уменьшаемое.
5. Сохранить полученный результат как разность.
6. Сдвинуть полученный результат влево через флаг переноса.
7. Если содержимое флага переноса равно нулю, перейти к п.10.
8. Проинвертировать результат полученный в п.4 с последующим прибавлением к нему единицы младшего разряда.
9. Сохранить полученный результат как разность.
10. Останов.

Таблица 5.3

Адрес	Содержимое
8000 … 81FF	Текст программы
Рег. B	Уменьшаемое
Рег. C	Вычитаемое
Рег. E	Разность

[8] Текст программы и исходные данные желательно расположить в адресном пространстве ОЗУ в соответствии с таблицей 5.3.

в) ввести исходные данные и выполнить программу в пошаговом режиме;

г) после выполнения каждого шага программы заносить в таблицу 5.4 содержимое регистров A и E и состояние флага переноса;

Таблица 5.4

№ шага	Регистры		Флаг переноса	Комментарий
	A	E		
0	*00*	*00*	-	*Исходные данные*
1				
2				

д) перевести результат из шестнадцатеричного формата в десятичный.

5.4. Содержание отчета

В отчете привести наименование и цель работы, текст программы задач №1 и №2, заполненные таблицы 5.2 и 5.4 с подробными комментариями.

5.6. Контрольные вопросы

1. Прокомментируйте выполнение команд логического сложения с различными способами адресации.
2. Прокомментируйте выполнение команд логического умножения с различными способами адресации.
3. Прокомментируйте выполнение команд «исключающее ИЛИ» с различными способами адресации.
4. Что представляет собой маскирование данных? Как выполнить маскирование данных с использованием команд:
   а) логического сложения;
   б) логического умножения;
   в) исключающее ИЛИ.
5. Как сравнить два числа, используя следующие команды:
   а) исключающее ИЛИ;
   б) вычитание;
   в) сравнение.
6. Прокомментируйте выполнение команд циклического сдвига влево и вправо:
   а) без учета флага переноса;
   б) с учетом флага переноса.
7. Приведите примеры использования логических команд.

Лабораторная работа №6
Изучение команд перехода и вызова подпрограмм

6.1.Цель работы

1. Изучить действие команд перехода и вызова подпрограмм.
2. Получить практические навыки составления программ.

6.2. Краткие сведения из теории

6.2.1. Команды перехода

Команды перехода позволяют изменять последовательность выполнения команд программы. Существуют два способа изменения этой последовательности.

Первый из них называется *безусловным*. Согласно этому способу, последовательность выполнения программы подвергается изменению всякий раз, когда реализуется команда «безусловный переход».

В соответствии со вторым (*условным*) способом последовательность выполнения программы определяется некоторыми условиями, т.е. изменяется в том случае, когда значение проверяемого условия совпадает с заданным.

Существуют следующие виды команд перехода:

Безусловный переход	JMP, адрес
Переход если нуль	JZ, адрес
Переход если не нуль	JNZ, адрес
Переход если перенос	JC, адрес
Переход если нет переноса	JNC, адрес
Переход если плюс	JP, адрес
Переход если минус	JM, адрес
Переход если чётный	JPE, адрес
Переход если нечётный	JPO, адрес

6.2.2. Команды вызова подпрограмм

Существует множество ситуаций, когда необходимо осуществить выход из главной программы, но впоследствии опять вернуться в нее. При этом возврат необходимо произвести именно в то место, из которого произошел переход.

Добиться этой цели можно с помощью команды вызова подпрограммы. Когда происходит вызов подпрограммы, то в начале своего выполнения она реализует действия по запоминанию текущего содержимого счетчика команд. Когда выполнение подпрограммы заканчивается, то сохраненное содержимое счетчика команд извлекается из памяти. По этой информации микропроцессор осуществляет возврат в прерванную последовательность команд главной программы.

При работе с подпрограммами используется *стек* микропроцессора. Стек - это область памяти, в которой временно сохраняется информация, необходимая для осуществления возврата в программу после выполнения подпрограммы. Состояние стека отображается в регистре, аналогичном регистровой паре и называемом *указателем стека* (SP). Указатель стека содержит информацию о том, какая область памяти должна быть использована при очередном обращении к стеку.

Команды «вызов подпрограммы» имеют длину 3 байта. Во всех этих командах используется прямая адресация.

Существуют следующие виды команд вызова подпрограмм:

Безусловный вызов подпрограммы	CALL, адрес
Вызов подпрограммы если нуль	CZ, адрес
Вызов подпрограммы если не нуль	CNZ, адрес
Вызов подпрограммы если перенос	CC, адрес
Вызов подпрограммы если нет переноса	CNC, адрес
Вызов подпрограммы если плюс	CP, адрес
Вызов подпрограммы если минус	CM, адрес
Вызов подпрограммы если чётный	CPE, адрес
Вызов подпрограммы если нечётный	CPO, адрес
Безусловный возврат из подпрограммы	RET
Возврат из подпрограммы если нуль	RZ
Возврат из подпрограммы если не нуль	RNZ
Возврат из подпрограммы если перенос	RC
Возврат из подпрограммы если нет переноса	RNC
Возврат из подпрограммы если плюс	RP
Возврат из подпрограммы если минус	RM
Возврат из подпрограммы если чётный	RPE
Возврат из подпрограммы если нечётный	RPO

6.3. Задания к лабораторной работе

Задание №1.

В заданном массиве чисел (10x1) найти все числа, равные 9_{10} и заменить их на 0_{10}.

Для этого необходимо:

а) составить программу[9] обработки массива в соответствии со следующим алгоритмом:

1. Загрузить в один из РОН количество элементов массива.
2. Загрузить в регистровую пару HL начальный адрес массива.
3. Сравнить текущий элемент массива с числом 9_{10}.
4. Если числа не равны перейти к пункту 6.

[9] Текст программы и исходные данные желательно расположить в адресном пространстве ОЗУ в соответствии с таблицей 6.1

5. Заменить текущий элемент массива числом 0_{10}.

6. Увеличить на 1 текущий адрес массива.

7. Уменьшить на 1 число элементов массива.

8. Если число элементов массива не равно 0, то перейти к п.3.

9. Останов.

б) ввести исходный массив (при вводе массива предусмотреть не менее трех элементов, равных 9_{10}, а также наличие элементов больших и меньших 9_{10}) и выполнить программу в пошаговом режиме;

в) после выполнения каждого шага программы заносить в таблицу 6.2 содержимое регистров A, одного из РОН, H, L, а также состояние флагов нулевого результата и переноса;

г) прокомментировать установку флагов.

Таблица 6.1

Адрес	Содержимое
8000 … 81FF	Текст программы
8200	Массив
…	
820A	

Таблица 6.2

№ шага	Регистры				Флаг нуля	Флаг переноса	Комментарий
	A	РОН	H	L			
0	*00*	*00*	*00*	*00*	*X*	*X*	*Исходные данные*
1							
2							

Задание №2

Подсчитать сумму чисел заданного массива (10x1).

Для этого необходимо:

а) составить программу[10] обработки массива в соответствии со следующим алгоритмом:

1. Загрузить в регистровую пару HL начальный адрес массива.

2. Загрузить в регистровую пару DE конечный адрес массива.

3. Загрузить в указатель стека адрес нижней границы стека.

4. Загрузить в регистр B начальное значение суммы элементов массива (00).

[10]Текст программы и исходные данные желательно расположить в адресном пространстве ОЗУ в соответствии с таблицей 6.3

5. Произвести расчет суммы элементов массива (цикл):

5.1. К начальному значению суммы прибавить содержимое очередной ячейки массива.

5.2. Результат поместить в регистр B.

5.3. Записать в регистровую пару HL адрес очередной ячейки массива.

6. Сравнить адрес текущей ячейки массива с адресом конечной ячейки массива (подпрограмма).

7. Если текущий адрес меньше конечного, то повторить вычисления с п.5.

8. Останов.

Подпрограмма[11] сравнения содержимого регистровых пар:

1. Вычесть из содержимого младшего байта конечного адреса массива содержимое младшего байта текущего адреса массива.

2. Вычесть из содержимого старшего байта конечного адреса массива содержимое старшего байта текущего адреса массива с учетом бита переноса.

3. Возврат к п.7.

Таблица 6.3

Адрес	Содержимое
8000 … 81FF	Текст программы
8200	Массив
…	
8209	
820A	Стек
…	
821A	

б) ввести исходный массив (с целью упрощения контроля результата (суммы) все элементы массива могут быть равны, например, 1_{10}) и выполнить программу в пошаговом режиме;

в) после выполнения каждого шага программы заносить в таблицу 6.4 содержимое всех используемых регистров, а также содержимое стека и флага переноса:

Таблица 6.4

[11] Поскольку команда непосредственного сравнения содержимого регистровых пар отсутствует, то сравнение можно произвести посредством операции вычитания с последующим контролем флагов нуля или переноса.

№ шага	Регистры						SP	SP-1	Флаг C	Коммента-рий
	A	B	D	E	H	L				
0	*XX*	*XX*	*XX*	*XX*	*XX*	*XX*	*XX*	*XX*	*X*	*Исх. данные*
1										
2										

г) прокомментировать содержимое стека и установку флага.

6.4. Содержание отчета

В отчете привести наименование и цель работы, тексты программ задач №1 и №2, записанные в мнемокодах и машинных кодах, заполненные таблицы 6.2 и 6.4 с подробными комментариями.

6.5. Контрольные вопросы

1. В чем состоит назначение команд перехода и вызова подпрограмм? В чем отличие указанных команд?
2. Прокомментируйте принцип выполнения команд перехода.
3. Прокомментируйте принцип выполнения команд вызова подпрограмм.
4. Что представляет собой стек? Каков его максимальный объем?
5. С помощью каких команд можно задать или переобозначить область памяти, отведенную под стек?
6. Приведите примеры использования команд перехода и вызова подпрограмм, отличные от приведенных в заданиях №1 и №2.
7. Напишите программу, сдвигающую содержимое аккумулятора на переменное число бит, определяемое содержимым регистра D.
8. Напишите программу, устанавливающую в ноль область памяти, начиная с ячейки 8300 и заканчивая ячейкой 830A.

Лабораторная работа №7
Арифметическое умножение и деление

7.1. Цель работы

1. Изучить способы организации арифметического умножения и деления.
2. Получить практические навыки составления программ.

7.2. Краткие сведения из теории

7.2.1. Арифметическое умножение

Для умножения двоичных чисел используется метод сдвига и сложения. При этом каждый раз необходимо проверять содержимое разрядов множителя, начиная со стороны его младшего разряда. Если в текущем разряде множителя записана единица, то множимое прибавляется к промежу-

точной сумме и сдвигается влево на один разряд. Если же в текущем разряде множителя записан нуль, то необходимо произвести только сдвиг множимого.

На практике сдвиг множимого влево заменяют сдвигом суммы вправо.

7.2.2. Арифметическое деление

Для выполнения арифметического деления используют следующий алгоритм. Путем циклического сдвига через флаг переноса формируют так называемое *промежуточное делимое*, такое, что при каждом сдвиге содержимое старшего разряда делимого «задвигается» в младший разряд промежуточного делимого.

После каждого сдвига из промежуточного делимого пытаются вычесть делитель. Если это удается (получена положительная разность), то, во-первых, промежуточное делимое заменяют полученной разностью, а во-вторых, в младший разряд частного «задвигают» единицу.

Если вычитание не удается (получена отрицательная разность), то в младший разряд частного «задвигают» нуль, а промежуточное делимое оставляют без изменений.

Далее производится очередной сдвиг старшего разряда делимого в младший разряд промежуточного делимого. Так происходит, пока не будут просмотрены все разряды делимого, начиная с самого старшего.

7.3. Задание к лабораторной работе

Задание №1.

Составить программу умножения двух однобайтных чисел (с получением двухбайтного результата).

Исходные данные: множимое - регистр B; множитель - регистр C.

Результат: произведение - регистровая пара DE.

Вспомогательные регистры: H - «маска»; L - число разрядов множителя.

Для этого необходимо:

а) составить программу в соответствии со следующим алгоритмом:

1. Установить начальные значения регистров D, E, H, L.
2. Проверить значение очередного разряда множителя (единица или нуль) путем сравнения с «маской».
3. Загрузить в аккумулятор старший байт результата.
4. Если в очередном разряде множителя нуль, то перейти к п.6.
5. Прибавить множимое к старшему байту результата.
6. Сдвинуть полученный результат вправо через флаг переноса.
7. Сохранить полученное число как старший байт результата.
8. Загрузить в аккумулятор младший байт результата.
9. Сдвинуть число в аккумуляторе вправо через флаг переноса.
10. Сохранить полученное число как младший байт результата.

11. Уменьшить число разрядов множителя, подлежащих проверке, на единицу.

12. Если полученный результат равен нулю, то перейти к п.15.

13. Сдвинуть множитель вправо.

14. Перейти к п.2.

15. Останов.

б) ввести тестовые значения множимого и множителя;

в) после выполнения каждого шага программы заносить в таблицу 7.1 содержимое регистров A, B, C, D, E, H, L, а также состояние флагов нулевого результата и переноса;

Таблица 7.1

№ шага	Регистры							Флаг Z	Флаг C	Комментарий
	A	B	C	D	E	H	L			
0	*XX*	*XX*	*XX*	*XX*	*XX*	*XX*	*XX*	*X*	*X*	*Исх. данные*
1										
2										

г) прокомментировать установку флагов.

Задание №2.

Составить программу деления двух чисел.

Исходные данные: делимое - регистр B; делитель - регистр C.

Результат: частное - регистр D.

Вспомогательные регистры: E – промежуточное делимое; L - число разрядов делимого.

Для этого необходимо:

а) составить программу в соответствии со следующим алгоритмом:

1. Установить начальные значения регистров D, E, L.

2. Сдвинуть делимое на один разряд влево через флаг переноса.

3. Загрузить содержимое флага переноса в младший разряд промежуточного делимого.

4. Вычесть из промежуточного делимого делитель.

5. Если получен отрицательный результат, перейти к п.7.

6. Сохранить полученную разность как промежуточное делимое.

7. Проинвертировать флаг переноса.

8. Сдвинуть частное влево через флаг переноса.

9. Уменьшить число разрядов делимого, подлежащих проверке, на единицу.

10. Если полученный результат равен нулю, то перейти к п.12.

11. Перейти к п.2.

12. Останов.

б) ввести тестовые значения делимого и делителя;

в) после выполнения каждого шага программы заносить в таблицу 7.2 содержимое регистров A, B, D, E, L, а также состояние флагов нулевого результата и переноса;

Таблица 7.2

№ шага	Регистры					Флаг Z	Флаг C	Комментарий
	A	B	D	E	L			
0	*XX*	*XX*	*XX*	*XX*	*XX*	*X*	*X*	*Исх. данные*
1								
2								

г) прокомментировать установку флагов.

6.4. Содержание отчета

В отчете привести наименование и цель работы, тексты программ задач №1 и №2, записанные в мнемокодах и машинных кодах, заполненные таблицы 7.1 и 7.2 с подробными комментариями.

6.5. Контрольные вопросы

1. Объясните принцип умножения чисел путем сдвига и сложения.
2. Объясните принцип деления двух чисел путем вычитания и сдвига.
3. Приведите примеры использования рассмотренных алгоритмов умножения и деления в качестве подпрограмм.

Задания для самостоятельной работы

1. Из исходного массива составить новый, содержащий элементы исходного, которые не превышают указанного числа.
2. Из исходного массива составить новый, содержащий элементы исходного массива, расположенные на четных местах.
3. Из исходного массива составить новый, путем замены всех элементов исходного массива, равных указанному числу на 0.
4. Из исходного массива составить новый, путем замены всех элементов исходного массива, меньших указанного числа на их квадрат.
5. Из исходного массива составить новый, значения элементов которого соответствуют номерам элементов исходного массива, равных указанному числу.
6. В заданном массиве чисел, содержащих в двух указанных разрядах двоичного кода 0, найти наименьшее.
7. Из исходного массива чисел составить новый путем замены всех элементов исходного массива на 1, если сумма всех элементов исходного массива больше заданного числа.
8. Составить программу подсчета суммы чисел заданного массива.
9. Составить программу вычисления факториала заданного числа.

ЛИТЕРАТУРА

1. Гилмор Ч. Введение в микропроцессорную технику: Пер. с англ. – М.: Мир, 1984. – 334с., ил.

2. Микропроцессоры: В 3 кн. Кн.3: Средства отладки: Лаб. практикум и задачник: Учеб. для техн. вузов / Н.В. Воробьев, В.Л. Горбунов, А.В. Горячев и др.; Под ред. Л.Н. Преснухина. – Мн.: Выш. шк., 1987. – 287с.: ил.

3. МикроЭВМ: В 8 кн.: Практ. пособие / Под ред. Л.Н. Преснухина. Кн. 3. Семейство ЭВМ «Электроника К1» / А.В. Кобылинский, А.В. Горячев, Н.Г. Сабадаш, В.В. Проценко. – М.: Высш. шк., 1988. – 191с.: ил.

СОДЕРЖАНИЕ

СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА КР580ВМ80А НА БАЗЕ МИКРО-ЭВМ «ЭЛЕКТРОНИКА 580»

Практическое руководство по выполнению лабораторных работ по курсу «Микропроцессорные средства в автоматизированном электроприводе» для студентов специальности 1-53 01 05 «Автоматизированные электроприводы» дневной и заочной форм обучения

Авторы-составители: **Луковников** Вадим Иванович,
Савельев Вадим Алексеевич

Подписано в печать 19.04.06.

Формат 60x84/16. Бумага офсетная. Гарнитура Таймс.
Цифровая печать. Усл. печ. л. 2,09. Уч.-изд. л. 2,27.
Изд. № 159.
E-mail: ic@gstu.gomel.by
http://www.gstu.gomel.by

Отпечатано на МФУ XEROX WorkCentre 35 DADF
с макета оригинала авторского для внутреннего использования.
Учреждение образования «Гомельский государственный
технический университет имени П.О. Сухого».
246746, г. Гомель, пр. Октября, 48.