МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

Государственный университет информатики и искусственного интеллекта

Факультет современных компьютерных информационных технологий

Кафедра программного обеспечения интеллектуальных систем

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ по курсу "Системное программирование"

Утверждено:
заседание кафедры
«Программного обеспечения
интеллектуальных систем»

заседание Ученого совета

2010

Разработчики:
А.И.Ольшевский
Р.А.Сорокин
Д.М.Бочаров

1 ЛАБОРАТОРНАЯ РАБОТА № 1

Тема: Разработка структуры микропроцессора

1.1 Задание на лабораторную работу

Ознакомиться с типовыми схемами микропроцессоров. Согласно варианту задания, составить и обосновать структурно-логическую схему своего микропроцессора. Информационные связи между устройствами должны соответствовать заданным режимам адресации.

Исходные данные (основные технические характеристики) для разработки структурно-логической схемы микропроцессора приведены в таблице 1.1. Режимы адресации, которые необходимы в микроЭВМ:

1) прямая;
2) прямая регистровая;
3) косвенная;
4) косвенная регистровая;
5) непосредственная;
6) относительная;
7) индексная;
8) базовая;
9) адресация с автоувеличением и автоуменьшением;
10) стековая.

1.2 Разработка структурно – логической схемы микропроцессора

Требования к структуре микропроцессора.

Полнота. В микропроцессор должны быть включены все устройства, необходимые для приема из памяти, хранения и выполнения команд с заданными согласно варианту режимами адресации. Разрядность таких устройств, как регистры, шины и т. п. должны быть достаточными для хранения соответствующих данных, номеров регистров, адресов и других частей команд.

Минимальность. Не следует включать в микропроцессор не используемые ни в одной из команд микроЭВМ устройства. Разрядность устройств не должна превышать требуемую.

Требование к информационным связям.

Информационные связи между устройствами должны обеспечивать возможность направленного обмена информацией между устройствами, участвующими в выполнении каждой команды микроЭВМ с заданными режимами адресации. Вместе с тем в структурно – логической схеме микропроцессора не должны присутствовать «лишние» (не используемые ни в одной из команд) связи.

В соответствии с требованиями полноты микропроцессор обычно включает следующие устройства:

- адресную шину;
- шину данных;

- регистр команд (IR);
- дешифратор команд;
- блок управления и синхронизации (БУС);
- программный счетчик (PC);
- устройство для выполнения арифметических и логических преобразований (АЛУ);
- аккумулятор (ы) и (или) регистры.

Адресная шина должна иметь разрядность, достаточную для передачи адреса заданной длины, согласно варианту задания.

Разрядность шины данных обычно совпадает с длиной байта. В исключительных случаях, когда все команды имеют длину, совпадающую с разрядностью основного слова, последняя определяет и разрядность шины данных. Во всяком случае, разрядность шины данных кратна длине байта. Разрядность регистра команд совпадает с длиной байта.

Если максимальная длина команды больше байта, то в микропроцессор следует включить регистр адреса и данных (DAR). Регистр адреса обычно используется для хранения адресной части команды.

Разрядность DAR должна быть достаточной для приема максимальной адресной части команды, определяющей один операнд.

Разрядность программного счетчика (счетчика адреса команд), очевидно, совпадает с длиной адреса.

Наличие и разрядности аккумуляторов и регистров указано в варианте задания. Следует помнить, что аккумулятор является по существу «особым» регистром. Если он единственный, то, естественно, не требует адресации. Если же в команде присутствует номер аккумулятора, это можно считать реализацией прямой регистровой адресации. Обычно, аккумулятор содержит данные, полученные из АЛУ.

Индексные регистры предназначены для хранения адреса или его части. Регистры общего назначения (РОНы) могут использоваться как для хранения адреса или его части, так и для хранения данных.

Для каждого вида регистров – аккумуляторов, если их несколько, РОНов, индексных регистров рекомендуется использовать свою отдельную нумерацию (отдельный селектор). Нумерацию рекомендуется начать с нуля, это обеспечит возможность выделения под номер регистра в команде поля минимальной длины.

Информационные связи между составляющими частями микроЭВМ зависят от заданных режимов адресации. Например:

- для обеспечения прямой адресации необходима передача из DAR на шину адреса;
- при индексной адресации исполнительный адрес вычисляется в АЛУ путем сложения переданных в АЛУ содержимого регистра и смещения из DAR, а из АЛУ результат посылается на шину адреса.

Вместе с тем, следует отметить, что необходимость установки информационной связи между устройствами может быть обусловлена назначением команд. Например, если прямая адресация реализована в командах загрузки, записи или арифметико – логической обработки данных, то адрес из DAR передается прямо на адресную шину, а если связь - для команд перехода, то адрес из DAR

может попасть в PC, а затем уже на адресную шину. Приблизительный вид структурно-логической схемы гипотетической ЭВМ приведен на рисунке 1.1.

Краткий перечень и описание элементов структурной схемы:

PC (IP) – счетчик команд. Содержит адрес текущей выполняемой команды, автоматически увеличивается на длину текущей команды (при переходе на следующую).

IR – регистр команд. Предназначен для приема из памяти и хранения кода команды.

DAR – регистр данных и адреса. При наличии операндов в команде после считывания кода операции в этот регистр считываются операнды (адрес, данные для загрузки регистров, а также номера регистров).

Acc0,Acc1 – регистры-аккумуляторы. Для определенных команд (например, для арифметических) один из операндов находится в одном аккумуляторе, результат помещается в первый и второй аккумуляторы.

R0...R7 – 8 регистров общего назначения. Предназначены для хранения данных. Обращение к регистрам производится при помощи 4-битной кодировки.

ПРИМЕЧАНИЕ: для некоторых команд номер регистра включен в код команды.

Регистр флагов содержит биты признаков состояния микропроцессора. Данные признаки изменяют свое состояние в результате операций над данными (в основном, арифметических). Также они используются для организации условных переходов в программе.

Z – признак нуля. Устанавливается в том случае, если результат арифметической или логической операции равен 0.

C – перенос из старшего разряда.

S – знак. Отражает состояние старшего бита результата.

P – Паритет (чётность). Устанавливается при чётном числе единиц в двоичном коде результата.

O - Переполнение.

БУС - Блок управления и синхронизации. Служит для выработки управляющих сигналов и синхронизации устройств, необходимых для выполнения команды в соответствии с сигналами дешифратора команд.

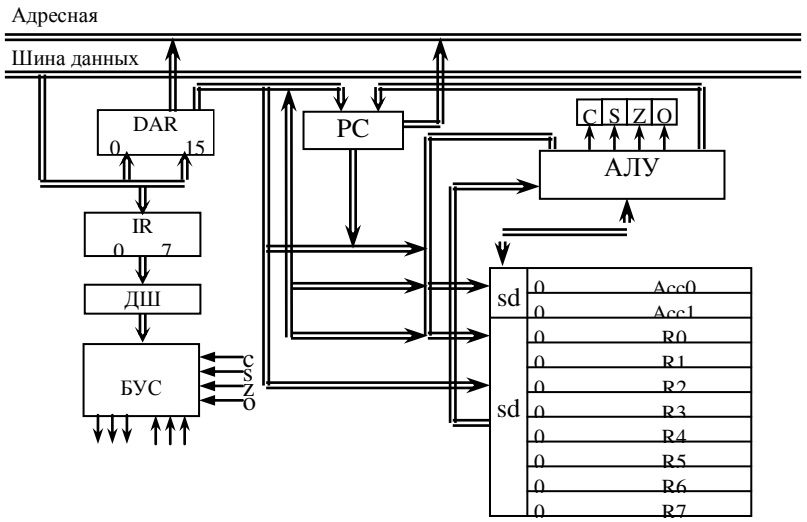

Рисунок 1.1 - Структурно-логическая схема гипотетической ЭВМ

1.3 Пример выполнения лабораторной работы №1.

Исходные данные:
Размер байта: 8 бит.
Разрядность основного слова: 8 бит.
Аккумуляторы: количество: 1; разрядность: 16 бит.
РОНы: количество: 8; разрядность: 16 бит.
Индексные регистры: отсутствуют.
Длина адреса: 16 бит.
Макс. длина команды: 24 бита.
Длина регистра условия: 4 бита.
Режимы адресации: прямая; прямая регистровая; косвенная регистровая; непосредственная; стековая.

Схема гипотетического МП представлена на рисунке 1.2.

Разрядность шины данных выбирается 16 бит, а не 8 так как РОНы и аккумулятор по условию 16-битные (удобнее осуществлять запись за 1 порцию данных). Шины от PC к ША и от DAR к ША используются при прямой адресации. Шина, соединяющая DAR и Sd используется при прямой регистровой, косвенной регистровой и непосредственной адресациях. Для реализации стековой адресации вводится регистр ST, который связан с ША.

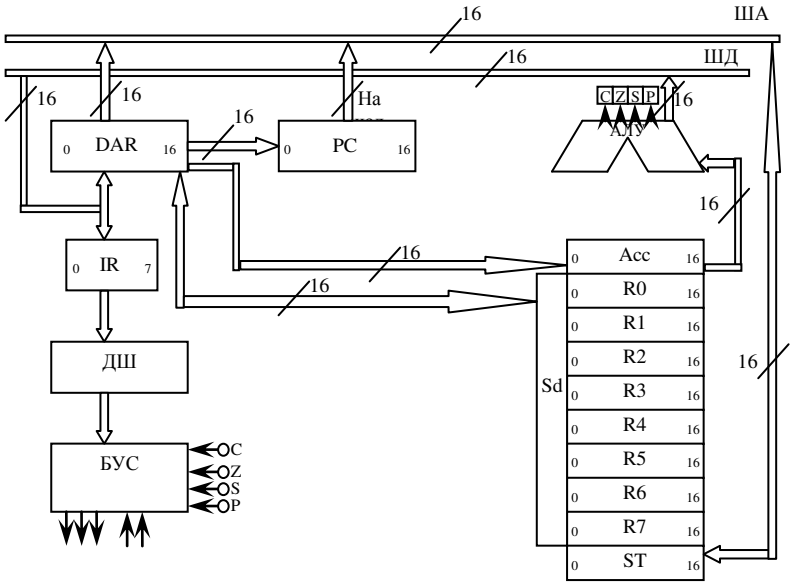

Рисунок 1.2 – Структурно-логическая схема гипотетического МП

2 ЛАБОРАТОРНАЯ РАБОТА № 2

Тема: Разработка системы команд микропроцессора

2.1 Требования к системе команд

а) ***функциональная полнота:*** система команд должна предоставлять максимально удобные (в условиях ограниченных ресурсов микроЭВМ) средства для программирования различных операций по обработке данных и алгоритмов;

б) ***минимальность:*** количество и длина команд должны быть минимальными при условии удовлетворения требования функциональной полноты с учетом варианта задания.

В соответствии с требованием функциональной полноты система команд обычно включает следующие операции:

- загрузка регистров и запись в память;
- сложение и вычитание целых чисел;
- поразрядные логические операции И, ИЛИ, НЕ (или две из них);
- сдвиг;
- условные переходы;
- изменение содержимого разрядов регистра кода условия (флажков);
- ввод и вывод;
- останов.

Система команд должна обеспечивать с помощью вышеприведенных или дополнительных команд организацию циклов, модификацию адресов массивов, переход к программе и возврат. Модификация адресов обеспечивается наличием команд загрузки, записи в память, арифметических и логических с индексной, косвенной или косвенной регистровой адресацией, и команд, изменяющих содержимое индексных регистров или регистров общего назначения. Если микропроцессор имеет аккумуляторы (при их отсутствии – регистры общего назначения) с разрядностью, вдвое большей, чем разрядность основного слова, в систему команд необходимо включить операции умножения и деления целых чисел.

Необходимость наличия флажка переноса, команд сложения и вычитания с переносом (заемов) определяется разрядностью основного слова: если разрядность меньше 16, перенос необходим, если 16 – по желанию автора, если больше 16 – не нужен. Важно понимать, что наличие флажка переноса обеспечивает сравнительно простую реализацию операций с длинными числами.

Проектирование системы команд в значительной степени зависит от заданных режимов адресации. Наиболее характерное использование режимов адресации для различных групп команд:

- прямая: команды переходов;
- прямая регистровая: загрузка, запись в память, арифметические и логические операции, сдвиги;
- косвенная: загрузка, запись в память, переход к подпрограмме, возврат из подпрограммы;

- косвенная регистровая: загрузка, запись в память, переход к подпрограмме и возврат, арифметические и логические операции;
- непосредственная: загрузка непосредственная, сдвиги на несколько разрядов, поразрядные логические операции;
- индексная: аналогично косвенной регистровой;
- относительная: команды переходов.

Это рекомендации, в конкретных случаях могут быть приняты и другие решения.

Важным этапом проектирования системы команд является выбор форматов. Желательно, чтобы количество форматов не превышало 3 – 4.

При выборе форматов необходимо учитывать следующие особенности:

1. В коротких командах часто используется переменная длина кода операции. Это позволяет плотнее упаковывать информацию и сократить число форматов команд и их длину.
2. В командах с индексной адресацией нужно обеспечить возможно большую величину смещения (не превышающую, однако, размер оперативной памяти). Подходящие значения смещения находятся обычно в пределах 0…4096 байт (соответственно на смещение отводится 8 ….12 разрядов команды).
3. В командах с непосредственной адресацией длина непосредственного операнда равна длине минимально адресуемой единицы памяти (1 байт).
4. В командах должны быть, как правило, унифицированы местоположения 1 и 2 операндов, операнда – источника и операнда – приемника, признаков и т. п.
5. Некоторые команды могут состоять только из кода операции, например, останов, сброс и установка флажков, сброс аккумулятора (если он один) и другие.

Приведем примеры наиболее используемых команд для различных форматов, режимов адресации, наборов регистров микропроцессора (табл.2.1). Отдельные команды из примеров не составляют единую систему команд и никак не связаны друг с другом.

Обозначения данных, используемых в таблице 2.1:

R,R1,R2 - регистры общего назначения – РОН (индекс указывает номер операнда);
Rинд. - индексный регистр;
S - адрес памяти;
Rk - регистр общего назначения, косвенная регистровая адресация;
Sk - адрес памяти, косвенная адресация;
D - смещение (относительно адреса в индексном регистре);
Dot - смещение при относительной адресации (относительно счетчика команд);
I - непосредственный операнд;
(M[адрес])- содержимое памяти по указанному адресу.

Таблица 2.1 - Пример команд микроЭВМ

Наименование и выполнение команд	Формат	Примечание характеристики процессора
Загрузка аккумулятора (M [Rk])->Acc	КОП Rk 0 3 4 7	РОНов -16 Индексных регистров -нет
Загрузка аккумулятора (M[S])->Acc	КОП старш.разряды младш.разряды 0 7 8 15 16 23	Прямая адресация
Загрузка регистра (R1)->R2	КОП R1 R2 0 1 2 4 5 7	РОНов -8; Прямая регистровая адресация
Загрузка регистра (M [(Rинд.)+D])->R	КОП R Rинд. D 0 1 2 4 5 7 8 15	РОНов-4;Инд.регистров-8 Индексная и прямая регистр-ая
Загрузка непосредственная I->Acc	КОП A I 0 4 5 6 7 11	Аккумуляторов - 2 Прямая регистровая и непосред-ная адр
Сложение акк-ра и регистра (Acc)+(R)-> Acc	КОП R 0 4 5 7	РОНов -8; Прямая регистровая адресация
Сложение регистров и переноса (R 1)+(R2)+(C)-> Acc	КОП R1 R2 0 1 2 4 5 7	РОНов -8; Прямая регистровая адресация
Сложение аккумулятора и содержимого памяти (Acc)+ (M[S])-> Acc	КОП A S 0 6 7 8 9 15	Аккумуляторов - 2 Прямая регистровая и прямая адресация
Логическое умножение (Acc) /\ (R)-> Acc	КОП ///// R 0 3 4 5 6 7	РОНов -4; Прямая регистр. Разряды4,5 не используем
Сдвиг аккумулятора на 1 разряд влево	КОП 0 7	Без адресации
Условный переход по нулю если (Acc) =0, то (Rинд.)+D-> PC	КОП D Rинд. 0 7 8 13 14 15	Индексных регистров-4 Индексная адресация
Переход к подпрограмме(с запоминанием адреса возврата в памяти) PC-> M[S1]; S2 -> PC	КОП S1 S2 0 4 5 11 12 18	Прямая адресация

2.2 Содержимое отчета

1. Форматы данных.
2. Форматы команд.
3. Таблица команд в виде:

Мнемон. код операции	Наименование	Формат	Описание
Команды преобразования данных			
ADD	Сложение с регистром	КОП R 1000 XXXX	(A)+(R)->A
ADC	Сложение с регистром и переносом	КОП R 1001 XXXX	(A)+(R)+C->A

2.3 Пример выполнения

Исходные данные:
Размер байта: 8 бит.
Разрядность основного слова: 8 бит.
Аккумуляторы: количество: 1; разрядность: 16 бит.
РОНы: количество: 8; разрядность: 16 бит.
Индексные регистры: отсутствуют.
Длина адреса: 16 бит.
Макс. длина команды: 24 бита.
Длина регистра условия: 4 бита.
Режимы адресации: прямая; прямая регистровая; косвенная регистровая; непосредственная; стековая.

2.3.1 Форматы данных

При выполнении микропроцессором операций, используются различные форматы данных:

- в РОНах и аккумуляторе могут храниться данные максимального размера 2 байта;
- непосредственно в команде может указываться операнд размером в 1 байт;
- в памяти хранятся операнды размером в 1 байт;
- размер элемента массива – 1 байт;
- адресное пространство составляет 64К;
- адреса имеют размер 2 байта.

2.3.2 Режимы адресации

Согласно варианта задания, разрабатываемый МП поддерживает следующие виды адресации:

- прямая – в команде указывается адрес, по которому в памяти хранится операнд;
- прямая регистровая – указывается номер регистра, в котором находится операнд;
- косвенная регистровая – указывается номер регистра, в котором хранится адрес, по которому в памяти находится операнд;
- непосредственная – в команде указывается непосредственно значение операнда;
- стековая – используется при сохранении в стек значений регистров, адресов возврата из подпрограмм.

2.3.3 Форматы команд

Форматы команд отображены в таблице 2.2.

Таблица 2.2 – Форматы команд микропроцессора

Мнемоника команды	Количество операндов	Длина команды (байт)	Пояснения
КОП	0	1	Однобайтовая команда, не имеет операндов, используется, как правило, для установки/сброса флагов, возврата из подпрограмм.
КОП R, R	2	3	Производится операция над операндами хранящимися в РОНах
КОП R, OP	2	3	Один из операндов хранится в РОНе, второй – задан непосредственно.
КОП R	1	2	Операция производится над операндом хранящимся в РОНе.
КОП MEM	1	3	Операнд находится в памяти, по указанному в команде адресу. Адрес занимает 2 байта.
КОП R, RK	2	3	В RK хранится адрес, по которому в памяти находится операнд.
КОП RK, R	2	3	

Обозначения, использующиеся в таблице 2.2:
КОП – код операции.
R – РОН.
MEM – адрес в памяти.
RK – один из РОНов, использующийся при косвенной адресации.

2.3.4 Система команд

Согласно варианта задания, максимальная длина команды составляет 3 байта, из них КОП занимает 1 байт и 2 байта отводятся под адрес.

Кодировка РОНов представлена в таблице 2.3.

Таблица 2.3 – Кодировка РОНов

R0	00000000
R1	00000001
R2	00000010
R3	00000011
R4	00000100
R5	00000101
R6	00000110
R7	00000111

Система команд МП представлена в таблицах 2.4-2.9.

Таблица 2.4 – Команды пересылки

Мнемокод операции	Наименование	Формат	Описание
movRA	Посылка операнда из регистра в аккумул-р	КОП R 00000000 XXXXXXXX	R ← Acc
movRM	Посылка операнда из памяти в регистр R0	КОП mem 00000001 XXXXXXXXXXXXXXXX	R0 ← mem
movRR	Посылка операнда из регистра в регистр	КОП R R 00000010 XXXXXXXX XXXXXXXX	R ← R
movRO	Посылка операнда в регистр	КОП R op 00000011 XXXXXXXX XXXXXXX	R ← op
movMR	Посылка операнда из регистра R0 в память	КОП mem 00000100 XXXXXXXXXXXXXXXX	mem ← R0
movRRK	Посылка операнда в первый регистр из памяти по адресу, хранящемуся во втором регистре	КОП R_1 R_2 00000101 XXXXXXXX XXXXXXX	$R_1 \leftarrow [R_2]$

Таблица 2.5 – Команды пересылки

Мнемокод операции	Наименование	Формат	Описание
movRA	Посылка операнда из регистра в аккумул-р	КОП R 00000000 XXXXXXXX	R ← Acc

Продолжение таблицы 2.5

Мнемокод операции	Наименование	Формат	Описание
movRM	Посылка операнда из памяти в регистр R0	КОП mem 00000001 XXXXXXXXXXXXXXXX	R0 ← mem
movRR	Посылка операнда из регистра в регистр	КОП R R 00000010 XXXXXXXX XXXXXXXX	R ← R
movRO	Посылка операнда в регистр	КОП R op 00000011 XXXXXXXX XXXXXXX	R ← op
movMR	Посылка операнда из регистра R0 в память	КОП mem 00000100 XXXXXXXXXXXXXXXX	mem ← R0
movRRK	Посылка операнда в первый регистр из памяти по адресу, хранящемуся во втором регистре	КОП R_1 R_2 00000101 XXXXXXXX XXXXXXX	$R_1 \leftarrow [R_2]$

Таблица 2.6 – Команды переходов

Команды переходов			
Jmp	Безусловный переход по метке	КОП mem 00001011 XXXXXXXXXXXXXXXX	ip ← ip±offset
jg	Переход если первый операнд больше второго	КОП mem 00001100 XXXXXXXXXXXXXXXX	ip ← ip±offset

Таблица 2.7 – Команды сравнения

Cmp	Сравнение операндов	КОП R_1 R_2 00010000 XXXXXXXX XXXXXXXX	R_1–R_2

Таблица 2.8 – Команды работы со стеком

push	Сохранение содержимого регистра в стек	КОП R 00011111 XXXXXXXX	R → [ST]
pop	Извлечение из стека верхней записи	КОП R 00100000 XXXXXXXX	[ST] → R

Таблица 2.9 – Команды изменения содержимого регистров флагов

szf	Установка флага ZF	КОП 00010001	ZF ← 1
scf	Установка флага CF	КОП 00010010	CF ← 1
ssf	Установка флага SF	КОП 00010011	SF ← 1
sof	Установка флага OF	КОП 00010100	OF ← 1
czf	Сброс флага ZF	КОП 00010101	ZF ← 0
ccf	Сброс флага CF	КОП 00010110	CF ← 0
csf	Сброс флага SF	КОП 00010111	SF ← 0
cof	Сброс флага OF	КОП 00011000	OF ← 0

3 ЛАБОРАТОРНАЯ РАБОТА № 3

Тема: Разработка языка ассемблера и формирования баз данных ассемблера

3.1 Общее описание работы

Описание языка содержать следующие разделы:

- общие сведения;
- элементы языка.

Допускается вводить разделы:

- способы структурирования программы;
- средства обмена данными;
- встроенные элементы;
- средства отладки программы.

В зависимости от особенностей языка допускается объединять отдельные разделы или вводить новые.

В разделе “Общие сведения” должны быть указаны назначения и описания общих характеристик языка, его возможностей, основных областей применения и другие сведения.

В разделе “Элементы языка” должно быть указано описание синтаксиса и семантики базовых и составных элементов языка.

В разделе “Способы структурирования программы” должны быть указаны способы вызова процедур передачи управления и другие элементы структурирования программы.

В разделе “Средства обмена данными” должно быть приведено описание языковых средств обмена данными (например, средства ввода-вывода, внутреннего обмена данными и т.п.).

В разделе “Встроенные элементы” должны быть приведены описания встроенных в язык элементов (например, функции, классы и т.п.) и правила их использования.

В разделе “Средства отладки программы” должно быть приведено описание имеющихся в языке средств отладки программ, семантики этих средств, должны быть даны рекомендации по их применению.

При необходимости содержание разделов должно быть пояснено примерами.

В приложения к описанию языка могут быть включены дополнительные материалы (формализованные описания языковых средств, иллюстрации, таблицы, графики, формы бланков и т.п.).

3.2 Содержимое отчета

1. Общие сведения
2. Элементы языка
3. Дополнительные разделы: способы структурирования программы, средства обмена данными, средства обработки данных, встроенные элементы, средства отладки программы.

3.3 Теоретические сведения

Язык ассемблера— язык программирования низкого уровня. В отличие от языка машинных кодов, позволяет использовать более удобные для человека мнемонические (символьные) обозначения команд. При этом для перевода программы с языка ассемблера в понимаемый процессором машинный код требуется специальная программа, называемая ассемблером**.**

Команды языка ассемблера один к одному соответствуют командам процессора, фактически, они представляют собой более удобную для человека символьную форму записи (мнемокод) команд и их аргументов. При этом одной команде языка ассемблера может соответствовать несколько команд процессора.

Кроме того, язык ассемблера позволяет использовать символические метки вместо адресов ячеек памяти, которые при ассемблировании заменяются на автоматически рассчитываемые абсолютные или относительные адреса, а также так называемые директивы (команды, не переводящиеся в процессорные инструкции, а выполняемые самим ассемблером).

Директивы ассемблера позволяют, в частности, включать блоки данных, задать ассемблирование фрагмента программы по условию, задать значения меток, использовать макроопределения с параметрами.

Каждая модель (или семейство) процессоров имеет свой набор команд и соответствующий ему язык ассемблера.

Исторически можно рассматривать язык ассемблера как второе поколение языков программирования ЭВМ (если первым считать машинный код). Недостатки языка ассемблера, сложность разработки на нём больших программных комплексов привели к появлению языков третьего поколения — языков программирования высокого уровня (Фортран, Лисп, Кобол, Паскаль, Си и др.). Именно языки программирования высокого уровня и их наследники в основном используются в настоящее время в индустрии информационных технологий. Однако языки ассемблера сохраняют свою нишу, обуславливаемую их уникальными преимуществами в части эффективности и возможности полного использования специфических средств конкретной платформы.

На языке ассемблера пишутся программы или фрагменты программ, для которых критически важны:

- быстродействие (драйверы, игры);
- объем используемой памяти (загрузочные секторы, встраиваемое (англ. embedded) программное обеспечение, программы для микроконтроллеров и процессоров с ограниченными ресурсами, вирусы, программные защиты).

С использованием программирования на языке ассемблера производятся:

- оптимизация критичных к скорости участков программ, написанных на языке высокого уровня, таком как C++. Это особенно актуально для игровых приставок, имеющих фиксированную производительность, и для мультимедийных кодеков, которые стремятся делать менее ресурсоёмкими и более популярными;
- создание операционных систем (ОС). ОС часто пишут на Си, языке, который специально был создан для написания одной из первых

версий UNIX. Аппаратно зависимые участки кода, такие как загрузчик ОС, уровень абстрагирования от аппаратного обеспечения (hardware abstraction layer) и ядро, часто пишутся на языке ассемблера. Ассемблерного кода в ядрах Windows или Linux совсем немного, поскольку авторы стремятся к переносимости и надёжности, но, тем не менее, он присутствует. Некоторые любительские ОС, такие как MenuetOS, целиком написаны на языке ассемблера. При этом MenuetOS помещается на дискету и содержит графический многооконный интерфейс;

- программирование микроконтроллеров (МК) и других встраиваемых процессоров. В МК приходится перемещать отдельные байты и биты между различными ячейками памяти. Программирование МК весьма важно, так как, в автомобиле и квартире современного цивилизованного человека в среднем содержится около 50 микроконтроллеров;
- создание драйверов. Некоторые участки драйверов, взаимодействующие с аппаратным обеспечением, программируют на языке ассемблера. Хотя в целом в настоящее время драйверы стараются писать на языках высокого уровня в связи с повышенными требованиями к надёжности. Надёжность для драйверов играет особую роль, поскольку в Windows NT и Linux драйверы работают в режиме ядра. Одна ошибка может привести к краху системы;
- создание антивирусов и других защитных программ;
- написание трансляторов языков программирования;

Инструкции языка ассемблера

Типичный формат записи команд:

[метка:] опкод [операнды] [;комментарий]

где опкод (код операции) — непосредственно мнемоника инструкции процессору. К ней могут быть добавлены префиксы (повторения, изменения типа адресации и пр.).

В качестве операндов могут выступать константы, адреса регистров, адреса в оперативной памяти и пр.

4 ЛАБОРАТОРНАЯ РАБОТА № 4

Тема: Ручная трансляция программы

4.1. Общее описание работы

Исходными данными для трансляции является текст программы на мнемокоде, которая должна включать:

– два или три модуля;

– все типы адресации, согласно варианту задания;

– определение внешних имен и внешних ссылок.

Отдельно представить формат объектной программы.

Выполнить ручную трансляцию программы из мнемокода в объектную программу с использованием таблицы машинных операций и псевдоопераций.

Выполнить операции связывания и перемещения для своего примера объектной программы и отобразить содержимое памяти.

При необходимости содержание разделов должно быть пояснено примерами, содержать иллюстрации, таблицы т.п.

4.2 Содержимое отчета

1. Текст программы на мнемокоде.
2. Объектная программа.
3. Содержимое памяти.

4.3 Пример выполнения

Входными данными для программы являются объектные файлы: текстовые файлы с расширением obj записанные в определенном формате. В объектном файле присутствуют записи заголовка, таблицы ссылок, таблицы имен и тела.

Все записи размещаются с новой строки и также имеют определенный формат.

Запись заголовка используется для хранения информации об объектном фале, данных, содержащихся в нем. Формат записи заголовка представлен в таблице 4.1.

Таблица 4.1 – Формат записи заголовка

1	2-9	10	11-14	15-18	19-23
Признак заголовка	Имя модуля	Признак основной программы	Точка входа (смещение первой команды, относительно адреса загрузки модуля)	Длина кода (тела) в байтах	Смещение 1го байта кода относительно начала объектного файла

Запись таблицы ссылок используется для хранения в объектном файле записей таблицы ссылок. В 1 байте указывается признак таблицы ссылок, который одновременно указывает тип ссылки (внешняя – E, внутренняя – L). Формат записи таблицы ссылок представлен в таблице 3.2.

Таблица 4.2 – Формат записи таблицы ссылок

1	2-9	10-13
Признак записи таблицы ссылок, тип ссылки	Имя метки, на которую ссылаются	Адрес вызова

Запись таблицы имен используется для хранения в объектном файле записей таблицы имен. В 1 байте указывается признак таблицы имен, который одновременно указывает тип имени (данные – D, метка (в т.ч. процедура) – P). Формат записи таблицы имен представлен в таблице 4.3.

Таблица 4.3 – Формат записи таблицы имен

1	2-9	10-13
Признак записи таблицы имен, тип имени	Имя метки	Адрес расположения метки в памяти

Запись тела содержит код программы: данные и команды. Формат записи тела представлен в таблице 4.4.

Таблица 4.4 – Формат записи тела

1	2-…
Признак тела	Код

Ручная трансляция программы

Для демонстрации возможностей системы команд разработанного гипотетического МП, используется программа сортировки одномерного байтового массива. Программа располагается в 2 модулях: основная часть и в отдельный модуль вынесена процедура перестановки местами элементов массива в памяти. Сортируемый массив данных содержится в модуле основной части программы. Коды модулей программы с объектными кодами команд и операндов, а также таблицы ссылок и имен модулей приведены ниже в таблицах.

Программа главного модуля MAIN приведена в таблице 4.5.

Таблица 4.5 – Программа модуля MAIN

Команды	Объектный код		Адрес команды	
	Bin	Hex	Bin	Hex
M db 1,2,3(0),6,7	00000001 00000010 00000000 00000000 00000000 00000110 00000111	01 02 00 00 00 06 07	0000000000000000	0000
movRMA M	00100101 00000000 00000000	25 00 00	0000000000000111	0007
movRR R6,R0	00000010 00000110 00000000	02 06 00	0000000000001010	000A
movRO R4,0	00000011 00000100 00000000	03 04 00	0000000000001101	000D
movRO R7,7	00000011 00000111 00000111	03 07 07	0000000000010000	0010
FORI: movRR R5,R4	00000010 00000101 00000100	02 05 04	0000000000010011	0013
push R7	00011111 00000111	1F 07	0000000000010110	0016
movRO R1,1	00000011 00000001 00000001	03 01 01	0000000000011000	0018
sub R7,R1	00001000 00000111 00000001	08 07 01	0000000000011011	001B
movRA R7	00000000 00000111	00 07	0000000000011110	001E
movRR R3,R4	00000010 00000011 00000100	02 03 04	0000000000100000	0020
add R3,R1	00000111 00000011 00000001	07 03 01	0000000000100011	0023
movRA R3	00000000 00000011	00 03	0000000000100110	0026
FORJ: movRR R1,R6	00000010 00000001 00000110	02 01 06	0000000000101000	0028
add R2,R3	00000111 00000010 00000011	07 02 03	0000000000101011	002B
movRA R2	00000000 00000010	00 02	0000000000101110	002E
movRRK R1,R2	00000101 00000001 00000010	05 01 02	0000000000110000	0030
movRR R2,R6	00000010 00000010 00000110	02 02 06	0000000000110011	0033
add R2,R5	00000111 00000010 00000101	07 02 05	0000000000110110	0036
movRA R2	00000000 00000010	00 02	0000000000111001	0039
movRRK R0,R2	00000101 00000000 00000010	05 00 02	0000000000111011	003B
cmp R1,R0	00010000 00000001 00000000	10 01 00	0000000000111110	003E
jge NC	00001101 00000000 01000111	0D 00 47	0000000001000001	0041
movRR R5,R3	00000010 00000101 00000011	02 05 03	0000000001000100	0044
NC: movRO R1,1	00000011 00000001 00000001	03 01 01	0000000001000111	0047

Таблица 4.5 – Продолжение

add R3,R1	00000111 00000011 00000001	07 03 01	0000000001001010	004A
movRA R3	00000000 00000011	00 03	0000000001001101	004D
loop FORJ	00100001 00000000 00101000	21 00 28	0000000001001111	004F
call SWAP	00100011 00000000 00000000	23 00 00	0000000001010010	0052
add R4,R1	00000111 00000100 00000001	07 04 01	0000000001010101	0055
movRA R4	00000000 00000100	00 04	0000000001011000	0058
pop R7	00100000 00000111	20 07	0000000001011010	005A
loop FORI	00100001 00000000 00010011	21 00 13	0000000001011100	005C
ret	00100100	24	0000000001011111	005F
			0000000001100000	0060

Таблица ссылок модуля MAIN приведена в таблице 4.2.

Таблица 4.2 – таблица ссылок модуля MAIN

Заголовок	Имя ссылки	Адрес ссылки
L	M	0007
L	FORI	005C
L	FORJ	004F
L	NC	0041
E	SWAP	0052

Таблица имен модуля MAIN приведена в таблице 4.3.

Таблица 4.3 – таблица имен модуля MAIN

Заголовок	Имя ссылки	Адрес ссылки
D	M	0000
P	FORI	0013
P	FORJ	0028
P	NC	0047

В результате трансляции, будет получен следующий объектный файл модуля MAIN:

```
HMAIN    100070060008C
LM       0007
```

```
      LFORI    005C
      LFORJ    004F
      LNC      0041
      ESWAP    0052
      DM       0000
      PFORI    0013
      PFORJ    0028
      PNC      0047
      T0102000000060725000002060003040003070702050 41F0703010
1080701000702030407030100030201060702030002050102020206070 2
0500020500021001000D00470205030301010703010003210028230000 0
704010004200721001324
```

Аналогично описывается модуль SWAP.

СОДЕРЖАНИЕ