Тема 14 Прикладне програмування

0бработка структуры данных

Под структурами данных понимают наборы некоторым образом организованных данных. Программисту следует знать наиболее распространенные структуры данных, способы хранения их в памяти МП-систем и эффективного использования в прикладных программах. Наиболее простой структурой данных является одномерный массив, представляющий собой набор элементов данных одинаковой длины, размещённых в области смежных ячеек памяти с начальным / базовым / адресом, например, BASE/. Число элементов в массиве называется его длиной. Положение любого элемента в массиве характеризуется его порядковым номером, называемым индексом. Адрес элемента равен сумме базового адреса и индекса, умноженного на длину элемента в байтах. Целесообразно применять массивы, длина элементов которых равна степени двух. Тогда при вычислении адреса элемента операция умножения заменяется сдвигами. Формирование и обработка массивов осуществляется циклическими программами. Примером обработки массива является программа поиска максимального числа в массиве однобайтовых целых чисел, рис.14.1.

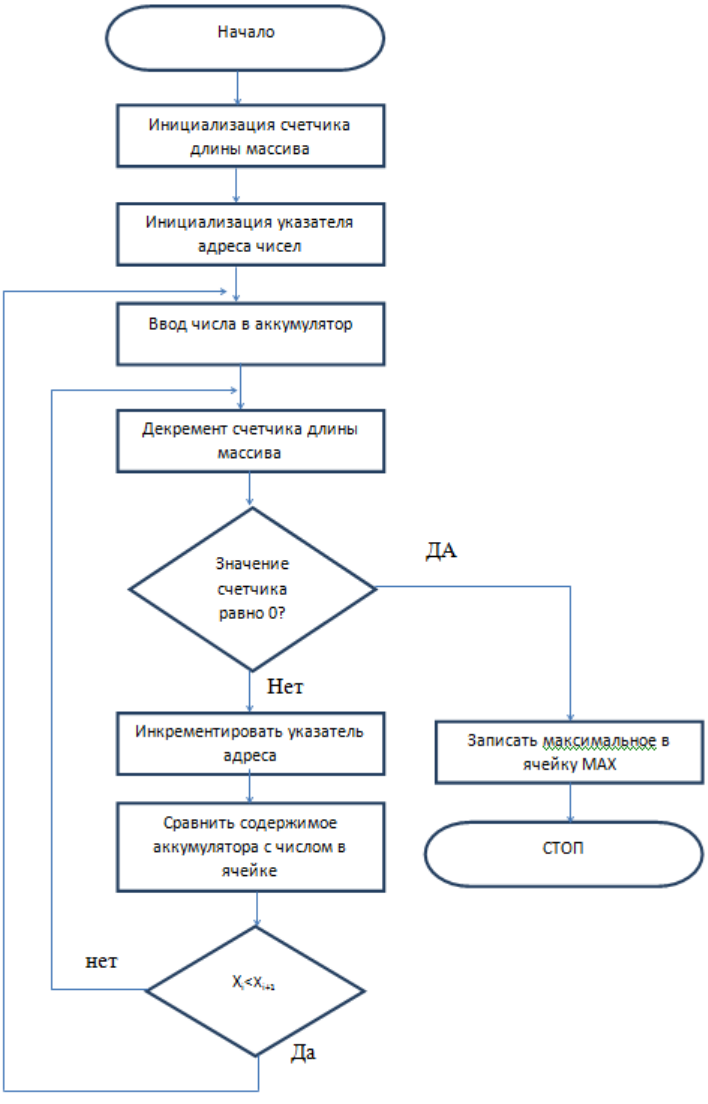

Рис.14.1 . Структурная схема поиска максимального числа.

Длина массива хранится в ячейке с условным адресом LENGT, адрес первого элемента - BASE, в качестве счетчика используется регистр B. Максимальное число направляется в ячейку памяти с адресом MAX.

Метка	Мнемоника	Операнд	Комментарий
	LDA	LENGT;	Инициализация счетчика длины массива
	MOV	B,A;	
	LXI	H.BASE;	Инициализация указателя адреса чисел
NEWMX:	MOV	A,M;	
NEXT	DCR	B;	Загрузка массива
			Проверка окончания цикла
	JZ	DONE;	Переход на конец программы, если флаг Z = 1
	INX	H;	Переход к следующему элементу
	CMP	M;	Сравнение его с максимумом
	JC	NEWMX ;	Новый максимум
	JMP	NEXT;	Следующий элемент меньше
	STA	MAX;	Запись максимума ячейку
	HLT		Останов

Сначала первый элемент массива принимается в качестве максимального, а затем каждый последующий элемент сравнивается с ним. Если текущий элемент больше ранее найденного максимума , он замещает его в аккумуляторе. Таким образом, просматривается в цикле весь массив данных.

Сортировка

В некоторых прикладных задачах возникает необходимость сортировки, т.е. упорядочение элементов массива данных в возрастающем или убывающем порядке. Известно несколько методов решения данной задачи. Наиболее часто применяется т.н. "пузырьковая" сортировка - меньшие элементы перемещаются вверх, а большие - вниз. Сортировка заключается в последовательном просмотре массива и сравнения значений соседних элементов. Если значение элемента в ячейке X больше значения элемента в ячейке x-1 , производится обмен этих элементов (x)←→(x+1). Если (x)<(x+1), то обмена не происходит. После этого аналогично сравниваются элементы в ячейках x+1 и x+2. Процесс сравнения и обмена продолжается до достижения конца массива. Факт наличия хотя бы одного обмена фиксируется записью в некотором регистре называемом "индикатором обмена", не нулевого кода. После окончания просмотра массива анализируется значение индикатора обмена и, если в нем зафиксирован обмен, весь массив просматривается ещё раз. Сортировка заканчивается, когда при просмотре массива не было произведено ни одного обмена.

Структурная схема программы сортировки массива 8-битовых целых без знака приведена на рис. 14.2. Программа BS0RT имеет два параметра: начальный адрес массива записывается в регистрах HL, образуется длина массива, для чего начальный адрес вычитается из конечного. После этого начальный и адрес и длина массива запоминаются в стеке для использования на последующих просмотрах. Затем производится сравнение значения двух соседних элементов массива и при необходимости осуществляется их обмен. В качестве индикатора обмена используется регистр C: в начале просмотра регистр C сбрасывается, а когда при просмотре массива производится обмен, в него загружается код FFH. После сравнения значений пары элементов выполняется декремент содержимого регистров D,E и происходит зацикливание до окончания одного просмотра. Наконец проверяется индикатор обмена - если регистр содержит, не ноль, инициируется новый просмотр массива.

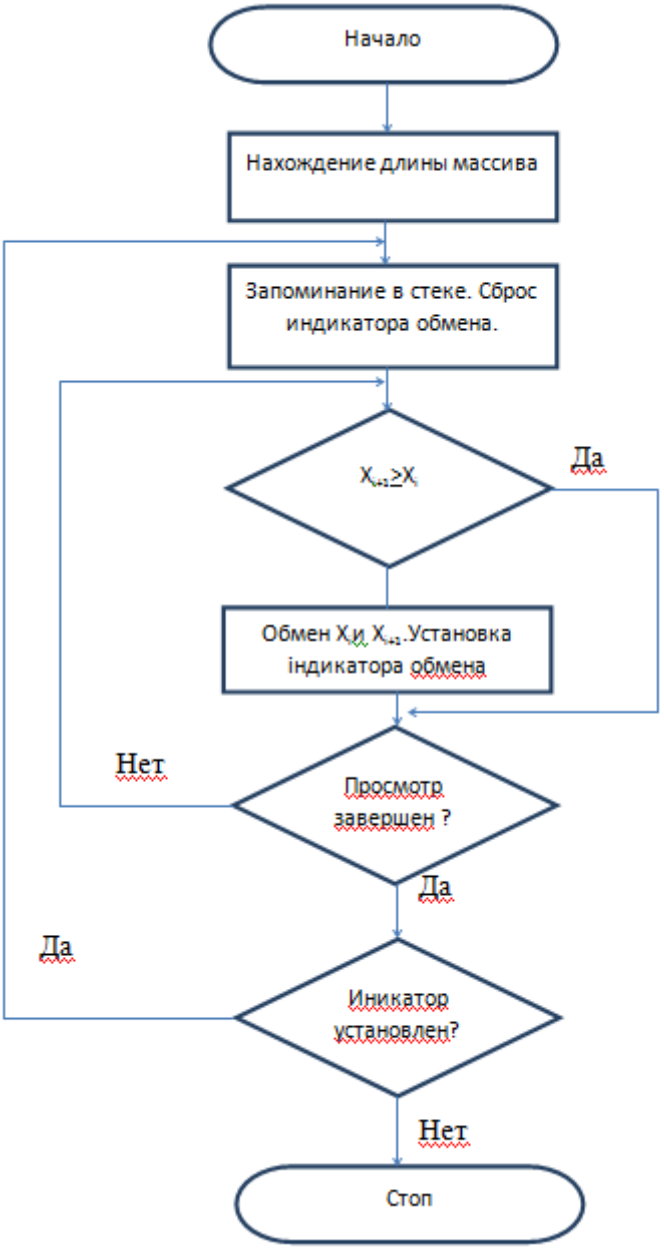

Рис. 14.2. Структурная схема программы сортировки

BSORT:	MOV	A,E	Образование в регистрах D,E длины массива
	SUB	L	
	MOV	E,A	
	MOV	A,D	
	SBB	H	
PASS:	PUSH	H	
	PUSH	D	
	MVI	C,00H	Сброс индикатора обмена
COMP:	MOV	A,M	Считывание значения
	INX	H	Указатель на X1+1
	CMP	M	Сравнение X1<X1+1
	JC	NEXT	Обмен не нужен

	JZ	NEXT	X1+1≥1
	MOV	B,M	Обмен значений элементов X1 и X1+1
	MOV	M,A	
	DCX	H	
	MOV	M,B	
	INX	H	Указатель на X1+1
	MVI	C,FFH	Установка индикатора обмена
NEXT:	DCX	D	Декремент счётчика элементов
	MOV	A,D	Анализ окончания просмотра
	ORA	E	
	JNZ	COMP	Просмотр закончен
	MOV	A,C	Да, проверка индикатора обмена
	ORA	A	
	POP	D	Восстановление начального адреса и длины
	POP	H	
	JNZ	PASS	Сортировка закончена?
	HLT		Да, стоп

Временные задержки

Организация процессов обработки информации в МП-системе осуществляется в виде последовательности простейших операций, проводимых во времени по сигналам специального синхронизирующего / тактового / генератора прямоугольных импульсов, характеризуемого частотой f или периодом синхронизации T= 1/f называемом тактом. Время, необходимое для считывания команды из памяти и ее исполнения, называется циклом команд и определяется числом тактов, необходимых для выполнения требуемой последовательности элементарных действий. Для определения времени выполнения команды нужно знать, какое число тактов содержится в цикле команд и чему равен период сигнала тактирования.

Временная задержка может быть сформирована программой, в которой некоторое множество команд не выполняет никаких операций, но занимает машинное время.

Для формирования временных задержек применяется метод, при котором в рабочий регистр загружается некоторое число и затем оно последовательно уменьшается на единицу до тех пор, пока оно не станет равным 0. Чем большую задержку необходимо получить, тем большее число нужно занести в этот регистр, рис 14.3.

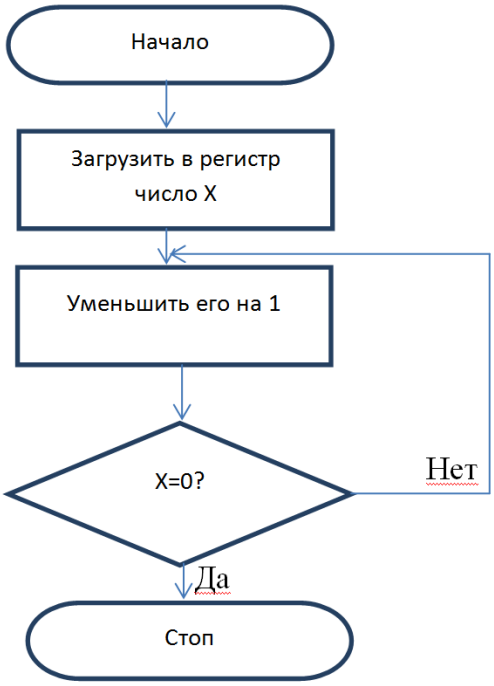

Рис. 14.3 Структурная схема задержки времени

Пример построения такой подпрограммы приведен ниже.

Метка	Мнемоника	Операнд	Комментарий
DELAY:	MVI	B, 0XN;	Загрузить в регистр в число 0XN
FORW:	DCR	B;	Уменьшить на единицу
	JNZ	FORW;	Повторить процесс, если результат не 0
	RET		Возврат в основную программу

Подпрограмма не работоспособна до тех пор, пока не будет задано значение 0XH. Зная команды и время на их выполнение можно определить число 0XH, обеспечивающее требуемую задержку. При расчете нужно учесть, что данная подпрограмма будет вызываться командой CALL DELAY.

Учитывая, что для микропроцессора К-580 обычно T = 0,5 мкс (f = 2 МГц) время выполнения команд состави:

CALLDELAY	- 17 тактов	0,5	= 8,5 мкс
MVI B.X	- 7тактов	0,5	= 3,5 мкс
DCR B	- 5тактов	0,5	= 2,5 мкс
JNZ FORW	- 10 тактов	0,5	= 5 мкс
RET	- 10 тактов	0,5	= 5 мкс

Командные CALLDELAY, MVIB, 0X и RET используются в программе 1 раз, команды DCR B и JNZ FORW используются многократно в зависимости от значения 0XH. Чтобы получить задержку времени равную 100 мкс команды

DCR B и JNZ FORW должны повторится столько раз, чтобы этот процесс выполнялся за 100-17 = 83 мкс. Время выполнения этих команд равно 2,5+5,0=7,5 мкс. Задержку 83 мкс нельзя получить повторением этой пары, т.к. 83 не делится на 7,5 без остатка. Для корректировки времени используется команда выполняемая за 4 такта и время цикла равно 2 мкс. Окончательно подпрограмма имеет вид:

Метка	Мнемоника	Операнд	Комментарий
DELAY:	MVI	B, 10H;	Загрузить в регистр B число 10
FORW:	DCR	B;	Уменьшить содержимоеB на 1
	JNZ	FORW;	Повторить процесс, пока результат не 0
	NOOP		Доводка времени задержки до заданного значения
	NOOP		
	NOOP		
	NOOP		
	RET		Возврат в основную программу

При организации больших временных задержек используют в качестве счетчиков регистровые пары, строят программы со вложенными циклами.

Логические функции

В системах управления технологическими процессами и объектами необходимые логические связи между входными и выходными параметрами могут решаться или аппаратными средствами с помощью комбинационных логических схем или программно /гибкое или настраиваемое управление/. Считается, что применение микропроцессорных систем, по сравнении с системами с жесткой логикой обеспечивает большую гибкость, более низкую стоимость и меньше время разработки. Однако использование таких систем требует оценки их быстродействия.

Программируемая логическая система реализует заданную логическую функцию путем последовательного сравнения комбинации входных сигналов с комбинациями, заданными таблицей истинности логической функции, так называемыми масками. Результат сравнения используется для перехода в состояние, соответствующее выполнению заданного условия /операнды совладают/, либо для продолжения операций сравнения со следующими наборами /масками/. В качестве примера рассмотрим решение логической функции F= f(A,B,C) представленной следующей таблицей истинности.

Положим, что входные сигнал A.B.C представляются состояниями второго (b_2), первого(b_1) и нулевого(b_0) битов слова, поступающего с порта 01_{16}. Выходной сигнал F реализуется третьим битом управляющего слова, записанного в ячейке памяти и UPR и выдаваемого в порт вывода 05_{16}.

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0

0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Программа реализации функции F представлена ниже:

Метка	Код	Операнд	Комментарий
	IN	01H	Ввод входных сигналов
	ANI	07H	Сброс, битов b_7,b_6,b_5,b_4,b_3 в ноль
	CPI	08H	Сравнение с маской согласно 4-ой строки таблицы
	JZ	FF1	Если равенство, переход к формированию F= 1
	CPI	05H	Если нет ,продолжение сравнения
	JZ	FF1	См. выше
	CPI	06H	Сравнение с 7-ой строкой
	JZ	FF1	См. выше
	LDA	UPR	Формирование нуля в третьем бите управляющегослова
	ANI	F7H	
	JMP	WW	Переход к выводу управляющего слова
FF1:	LDA	UPR	Формирование управляющего слова
	ORI		
WW:	OUT		Вывод управляющего слова
	HLT		

Следует учитывать, что при необходимости изменения вида логической функции, коррекция в систему вносится только путём изменения значения масок. Время реализации заданной логической функции в системе с программным управлением может оказаться больше, чем в системе с жестким управлением.