

СОДЕРЖАНИЕ

Введение	4
1. Команды передачи управления	5
2. Организация циклов	8
3. Организация временных задержек.	12
3.1 Разработка программы временной задержки	14
3.2 Программная реализация временных задержек	16
3.3 Расчет величины задержки времени	18
4. Задание к лабораторной работе.	18
5. Содержание отчета.	18
6. Вопросы.	19
Список использованной литературы	19

ВВЕДЕНИЕ

В настоящее время одним из основных направлений научно-технического прогресса является широкая автоматизация технологических процессов.

Внедрение МПС в контрольно-измерительную аппаратуру позволяет повысить точность измерений, надежность, расширить функциональные возможности приборов и обеспечивает выполнение следующих функций: калибровка, коррекция и температурная компенсация, контроль и управление измерительным комплексом, принятие решений и обработка данных, диагностика неисправностей, индикация, испытание и проверка приборов.

Использование микропроцессоров и устройств, созданных на их основе, позволяет значительно повысить уровень автоматизации.

Поэтому знание устройства, порядка функционирования и программирования работы микропроцессоров является необходимым для специалиста в области автоматизации .

Цель работы: изучение команд передачи управления микропроцессора КР580ВМ80А. Изучение команд условного и безусловного переходов. Организация циклов в программе.

1. КОМАНДЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ

Команды управления, часто называемые командами перехода, позволяют выполнять различные действия в соответствии со значением внешних сигналов или выработанных внутри системы условий. Все команды управления делятся на команды безусловного и условного перехода.

К окончанию текущей команды, а точнее — после выборки команды из памяти в программном счетчике РС сформирован адрес следующей по порядку команды. При естественном порядке выполнения команд, соответствующем простейшим линейным программам, производятся выборка и исполнение этой следующей команды, формируется адрес следующей команды и т. д.

Однако линейные прикладные программы на практике не встречаются. В разветвляющихся и циклических программах и при использовании подпрограмм приходится выполнять не следующую по порядку команду, а команду, находящуюся в другой ячейке программной памяти. Для этого достаточно загрузить в РС адрес новой ячейки, называемый адресом перехода. Такая процедура называется передачей управления, а специальные команды, которыми она реализуется, называются командами передачи управления, или командами управления программой. В системе команд микропроцессора К580 предусмотрен обширный набор команд передачи управления.

Команда безусловной передачи управления без возврата JMP, называемая также безусловным переходом или просто переходом, состоит из байта кода операции и двух байтов полного 16-битного адреса перехода. При ее исполнении адрес перехода загружается в РС, а текущее содержимое РС теряется: $PC \leftarrow \langle B3 \rangle, \langle B2 \rangle$. Данная команда, позволяет передать управление любой ячейке адресного пространства.

Современная методика создания структурированных программ рекомендует использовать минимум команд JMP.

Трехбайтные команды условной передачи управления без возврата, называемые также условными переходами или разветвлениями, осуществляют передачу управления только при удовлетворении некоторого условия, заданного в коде операции.

Если условие не удовлетворяется, то передачи управления не происходит, а выполняется следующая по порядку команда, т. е. в этой ситуации команда разветвления эквивалентна холостой команде.

Проверяемым условием является текущее значение одного из индикаторов, указываемых в коде операции.

Для удобства программирования предусмотрены команды разветвлений, осуществляющие передачу управления по единичному и нулевому значению каждого из индикаторов, кроме индикатора АС.

Например, команда JZ передает управление, т. е. осуществляет загрузку $PC \leftarrow \langle B3 \rangle, \langle B2 \rangle$, если флаг $Z=1$, а команда JNZ — если флаг $Z = 0$. Аналогичные парные команды имеются для флагов CY(JC, JNC), S(JM, JP) и P(JPE, JPO). Всего, таким образом, получается восемь команд разветвлений.

В таблице 1.1 приведены команды условных переходов, их машинный код, формат записи и действия, которые они производят.

При работе с двумя последними командами из таблицы 1 следует помнить, что процессор предполагает что эти числа записаны в дополнительном коде, т.е. являются числами со знаком. Поэтому при работе с числами без знака от этих команд стоит воздержаться.

Таким образом, команды условных переходов позволяют строить ветвящиеся алгоритмы, аналогичные представленному на рисунке 1.

JMP addr (Jump). Переход или ветвление (PC) \leftarrow addr. Управление передается команде, адрес которой установлен в байте 2 и 3 текущей команды.

1100 0011
Младший байт адреса
Старший байт адреса

Циклов – 3; периодов T – 10; адресация – непосредственная; индикаторы – нет.

J condition addr. Условное ветвление. Если условие выполняется, то управление передается команде, адрес которой установлен в байте 2 и 3 команды; если нет – управление продолжается последовательно.

11 CCC 010
Младший байт адреса
Старший байт адреса

Циклов – 3; периодов T – 10; адресация – непосредственная.

CALL addr. Вызов $((SP) - 1) \leftarrow (PCH)$; $((SP) - 2) \leftarrow (PCL)$; $(SP) \leftarrow ((SP) - 2)$; $(PC) \leftarrow$ addr. В стек помещается адрес возврата, содержимое SP

дважды декрементируется. Управление передается команде, адрес которой указан в байте 2 и 3 текущей команды.

1100 1101
Младший байт адреса
Старший байт адреса

Циклов – 5; периодов Т – 17; адресация – непосредственная, косвенная регистровая.

Таблица 1.1 Команды условных переходов.

операция	мнемоника	код	формат	символика
Перейти по адресу addr, если результат 1	JZ addr	CAH	CA мл.байт ст.байт адреса	Если Z=1 (PC) \leftarrow addr.
перейти по адресу addr , если результат не 0	JNZ addr	C2H	C2 мл.байт ст.байт адреса	Если Z=0,то (PC) \leftarrow addr.
перейти по адресу addr , если результат 1	JC addr	DAH	DA мл.байт ст.байт адреса	Если CY=1,то (PC) \leftarrow addr.
перейти по адресу addr , если результат 0	JNC addr	D2H	D2 мл.байт ст.байт адреса	Если CY=0,то (PC) \leftarrow addr.
перейти по адресу addr , если результат 0	JPO addr	E2H	E2 мл.байт ст.байт адреса	Если P=0.то (PC) \leftarrow addr.
перейти по адресу addr , если результат 1	JPE addr	EAH	EAH мл.байт ст.байт адреса	Если P=1,то (PC) \leftarrow addr.
перейти по адресу addr , если результат 0	JP addr	F2H	F2 мл.байт ст.байт адреса	Если S=0,то (PC) \leftarrow addr.
перейти по адресу addr , если результат 1	JM addr	FAH	FA мл.байт ст.байт адреса	Если S=1,то (PC) \leftarrow addr.

Регулирующая логика

Регулирующая логика, как правило, часть цикла, состоящая из команд, изменяющих величину счетчика циклов и определяющих число уже выполненных повторений либо число повторений, которое осталось выполнить. Счетчиком циклов может быть регистр или ячейка памяти. Если для него используются 8 бит, то можно организовать до 256 повторений цикла. Если счетчик 16-разрядный, то цикл может быть повторен до 65536 раз.

2.ОРГАНИЗАЦИЯ ЦИКЛОВ

Циклом компьютерной программы называется группа команд, которые повторяют необходимое число раз выполнение одного и того же задания. Число повторений цикла обычно задается специальным счетчиком циклов. При анализе структуры цикла в последнем можно выделить четыре функциональные части: подготовку, тело цикла, регулируемую логику и проверку окончания цикла. Все эти части взаимосвязаны.

Тело цикла

Данная часть цикла состоит из машинных команд, выполняющих задание, с повторением исполнения. Тело может содержать несколько команд или даже целую программу, включающую команды ввода-вывода для связи с периферийными устройствами; арифметические команды, производящие вычисления; логические команды, осуществляющие тестирование или изменяющие определенные условия и состояния центрального процессора; команды CALL для включения в работу других процедур и другие команды, неоднократное повторение которых требуется.

Все другие функциональные части цикла предназначены для повторения тела цикла нужное число раз и с необходимыми данными.

Структура программы, в которых команды располагаются в следующих друг за другом ячейках памяти и не имеют переходов и ветвлений, называется линейной.

Но линейную структуру имеет очень ограниченное число программ. В основном программы имеют ветвления и переходы.

Рассмотрим следующий пример. Пусть в ячейках ОЗУ расположены два однобайтовых числа. Необходимо сравнить их и большее записать в аккумулятор.

Если они равны, то в аккумулятор записать 00H.

Числа располагаются в ОЗУ по адресам 900H и 901H.

Для решения этой задачи можно предложить следующий алгоритм(рисунок 1).

1.Записать в пару HL значение адреса первого числа (900H).

2.Поместить в аккумулятор второе число.

3.Произвести операцию сравнения содержимого аккумулятора и ячейки памяти, адрес которой находится в паре HL.

4.Проверить бит признака нуля (флаг Z) флагового регистра F. Если он установлен в 1,то числа равны друг другу и в аккумулятор запишем 00.

Завершим выполнение программы.

5.Проверить бит переноса (флаг C) флагового регистра F. Если он установлен в 1,следовательно число по адресу 900H больше, чем число по адресу 901H.Тогда записываем в аккумулятор число из ячейки памяти с адресом 900H и завершаем программу.

6.Если флаг нуля не был установлен в 1,и флаг переноса также не был установлен в 1,следовательно число по адресу 901H больше. Запишем его в аккумулятор и завершим программу.

Данная программа уже не является линейной. В ней появились блоки передачи управления по условию, ветвления. Команды ветвления или перехода являются средством изменения содержимого счетчика команд PC и следовательно изменения нормальной последовательности выполнения программы. Блок-схема алгоритма представлена на рисунке 1.

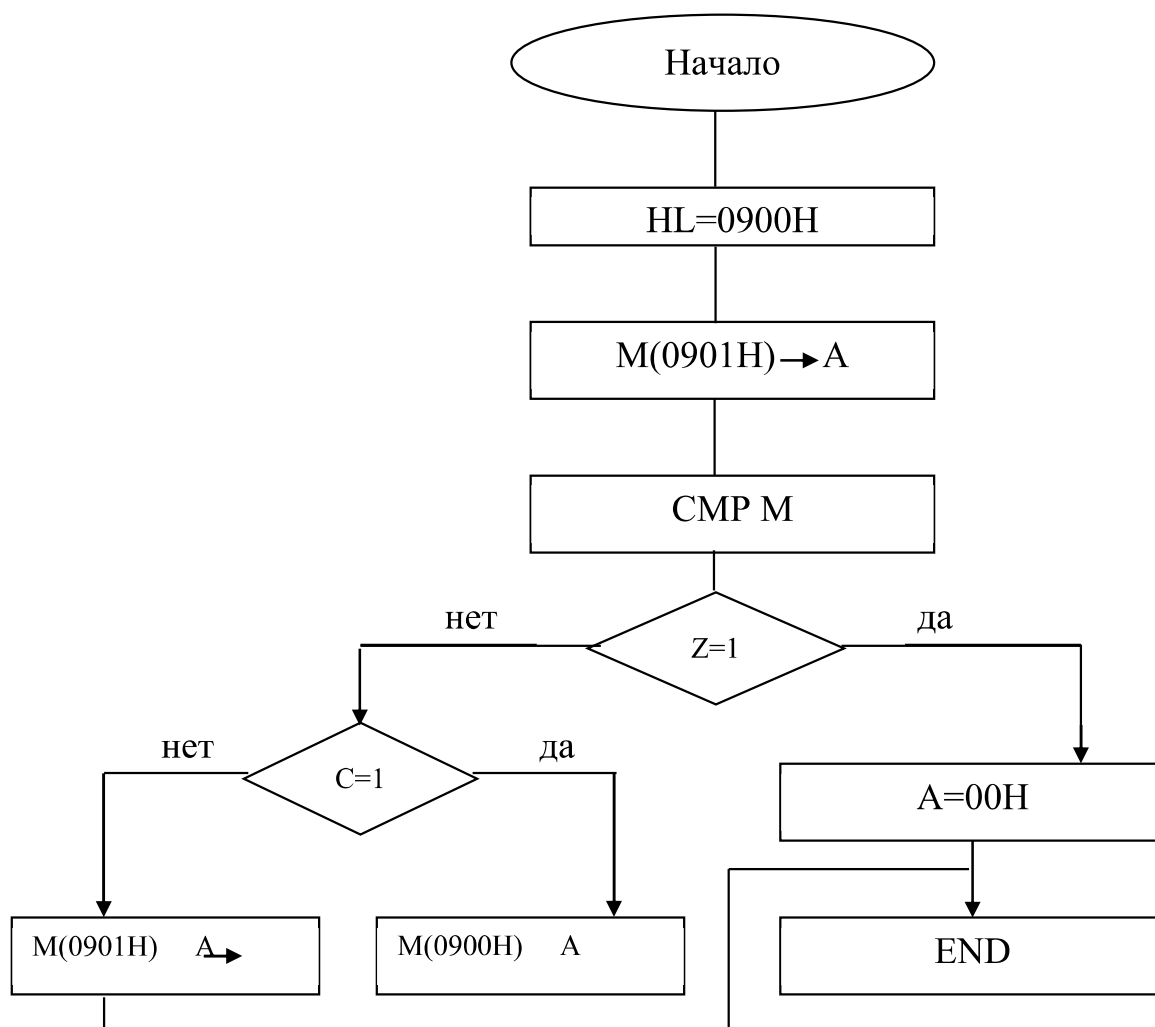


Рисунок 1- Блок-схема

Циклы применяются в программах для многократного повторения какого-либо действия или последовательности действий. С их помощью можно организовывать временные задержки и т.п .

Рассмотрим следующую задачу. Пусть необходимо обнулить область памяти ОЗУ начиная с адреса 900H и заканчивая адресом 950H(всего 80 ячеек).Возможны два варианта решения. Первый-организация программы, имеющую линейную структуру:

```
MVI A,00H
STA 900H
STA 901H
.....
.....
.....
STA 950H
HLT
```

Но такая программа привела бы к большому расходу памяти, т.к. она занимает место в памяти $80 \times 3 + 2 = 242$ байта.

Второй путь -организация циклической программы, блок-схема которой приведена на рисунке 2.

В начале происходит запись в регистр D длины массива 50H (80D), т.к. надо обнулить $950H - 900H = 50H$ ячеек памяти. Затем в регистровую пару HL записываем начальный адрес массива, а в аккумулятор число 00H.

Потом выводим в ячейку ОЗУ, адрес которой записан в паре HL, содержимое аккумулятора. Таким образом ячейка обнуляется. После того как программа обнулила первую ячейку, необходимо уменьшить счетчик цикла на 1.

Счетчик цикла находится в регистре D и 0 в этом регистре условие выхода из цикла. Если число в регистре D не равно 0, то значит не все ячейки обнулены и программа подготавливает адрес следующей ячейки массива, увеличивая содержимое пары HL на 1.Затем передает управление в точку 1 по блок-схеме на рисунке 2 . И весь цикл повторится снова.

Так будет до тех пор пока число в регистре D не станет равно 0 и все ячейки не будут обнулены.

Программа на Ассемблере и в машинных кодах представлена в таблице 2.1.

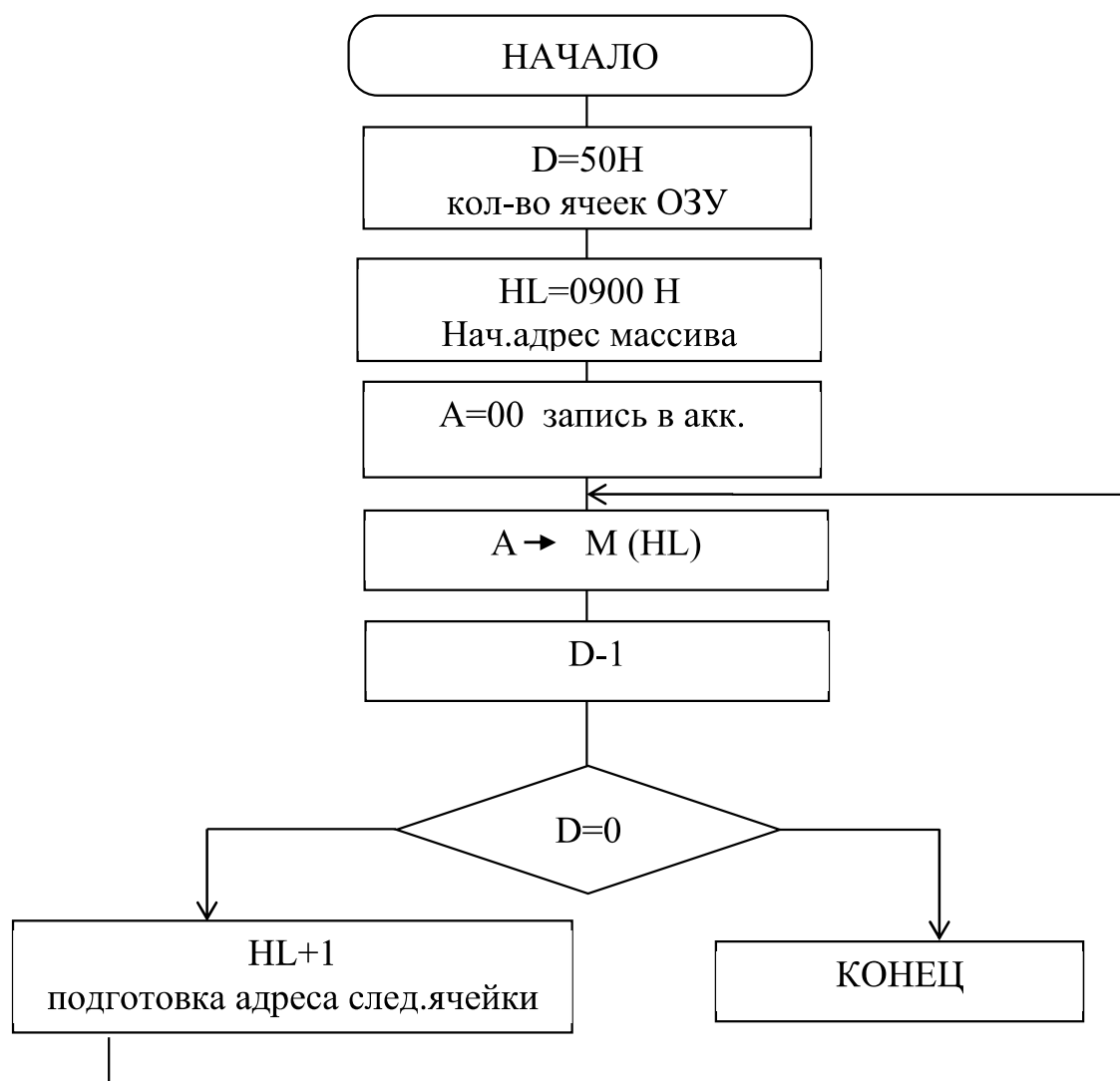


Рисунок 2 -Блок-схема программы обнуления массива.

При составлении программы на Ассемблере адреса переходов еще не определены и они заменяются символьными именами, так называемыми метками (метой может быть любая последовательность символов, начинающаяся с буквы).

В программе таблице 2.1 метками являются М1 и М2. Их адреса становятся определены после того как машинные коды будут расставлены по адресам в памяти.

В нашем случае адрес метки М1 будет равен 0807H (начало программы). Адрес метки М2 будет 0810H (адрес конца программы).

Таким образом величина циклической программы составит всего 16 байт (сравните с 242 байтами в первом случае).

По этой причине программы, имеющую циклическую структуру более предпочтительнее, т.к. они экономят память микропроцессорной системы (будь то УМК или IBM PC); самое дорогое в компьютере это микросхемы памяти и от размеров ОЗУ зависит общая стоимость системы.

Таблица 2.1 Программа обнуления массива

АДРЕС	КОД	МЕТКИ МНЕМОНИКА	КОММЕНТАРИЙ АССЕМБЛЕРА
800	16	MVI D,50H	инициализация счетчика цикла
801	50		
802	21	LXI H,0900H	начальный адрес массива
803	00		Мл. байт адреса
804	09		Ст. байт адреса
805	3E	MVI A,00H	запись константы в аккумулятор
806	00		байт
807 M1	77	MOV M,A	обнуление ячейки M(HL)
808	15	DCR D	уменьшение счетчика на 1
809 M2	CA	JZ	если D=0, то выйти из цикла
80A	10		Мл. байт адреса
80B	08		Ст. байт адреса
80C	23	INX H	D < 0, продолжить цикл, подготовив адрес следующей ячейки
80D	C3	JMP M1	Безусловный переход (организация цикла)
80E	07		Мл. байт адреса
80F	08		Ст. байт адреса
810	76	HLT M2	конец программы

3. ОРГАНИЗАЦИЯ ВРЕМЕННЫХ ЗАДЕРЖЕК.

В начале каждого машинного цикла процессор формирует слово состояние процессора (ССП), которое выдается в систему через шину данных и запоминается во внешнем регистре на все время выполнения машинного цикла. Рассмотрим временные диаграммы (ВД) основных машинных циклов (рисунок 3). Машинные циклы состоят из машинных тактов (МТ).

Частота следования синхроимпульсов:

$$f_{c1,c2} = 2 \times 10^6 \text{ Гц} = 2 \text{ МГц}$$

Длительность машинного такта: $MT = 0.5 \text{ мкс}$ ВД всех типов машинных циклов начинаются одинаково.

В первом такте T1 по переднему фронту F2 на выходе SYNC микропроцессора появляется сигнал высокого уровня, индицируя первый такт машинного цикла (рис.1). Кроме того, на шине данных в этом такте появляется слово состояния, у которого в разрядах D1, D5, D7 записаны единицы. По переднему фронту F2 в этом такте на адресной шине устанавливается адрес ячейки памяти, который поступил в буфер адреса из счетчика команд микропроцессора (т.е. поступает адрес ячейки, из которой извлекается команда).

Во втором такте T2 на выходе микропроцессора DBIN появляется сигнал высокого уровня длительностью в один такт, по которому обычно

происходит чтение памяти или внешних устройств. В этом такте МП опрашивает сигналы на входах READY, HLD и HLDA. В зависимости от значения сигналов на этих выводах МП переходит в различные состояния: ожидания, захвата, останова.

В такте T3 может выполняться или завершаться чтение памяти, после чего шина данных DB переходит в высокоимпедансное состояние. Обычно до четвертого такта T4 уже изменяется значение счетчика команд PC и таким образом, в нем находится адрес новой команды, который поступает на адресную шину в первом такте следующего машинного цикла.

В четвертом такте T4 код команды, поступивший в регистр команд дешифрируется - определяется, сколько циклов и тактов требуется для выполнения команды, которая затем обрабатывается микропроцессором в течении данного или последующего пятого такта T5.

По переднему фронту F2 на выводах адресной шины в такте T4 появляется неопределенное значение, а в такте T5 эти выводы переходят в высокоимпедансное состояние.

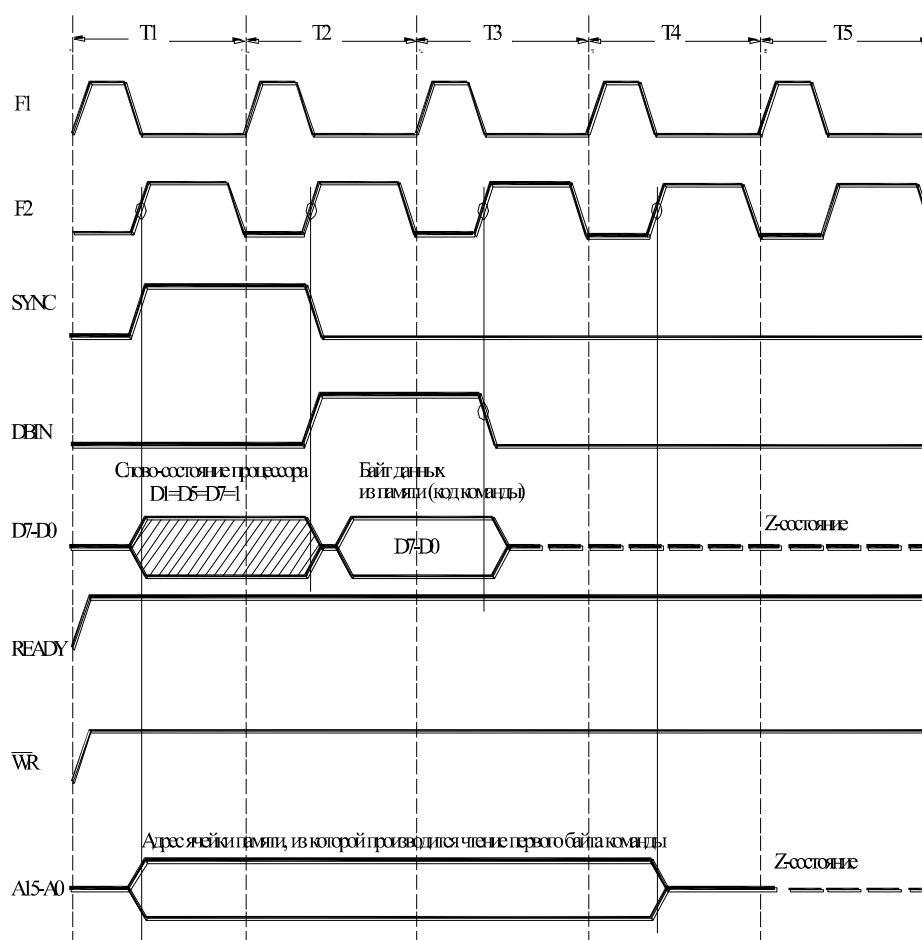


Рисунок 3- Временные диаграммы работы микропроцессора в цикле M1 (цикл извлечения команды)

3.1 Разработка программы временной задержки

В самом простейшем случае программная реализация временных задержек малой длительности представляет собой последовательность холостых команд (NOP).

Каждая такая команда обеспечивает задержку на 2 мкс. (2×10^{-6}). Таким образом, для программирования задержки времени, на 300 мкс (0,0003с) потребуется занять 150 ячеек памяти. Нерациональность такого подхода очевидна.

Поэтому для организации временной задержки используются простые программные циклы. Их сущность заключается в том, что в один из регистров РОН (например, в регистр В) записывается число, которое после прохождения каждого цикла уменьшается на единицу. Циклический процесс продолжается до тех пор, пока содержимое указанного регистра не обнулится. Это является признаком выхода из цикла. Очевидно, что длительность задержки зависит от загружаемого в регистр числа, определяющего количество отработываемых циклов, а также времени выполнения каждой команды, входящей в программу. Для удобства использования разработанной программы ее рекомендуется выполнить в виде подпрограммы. Графический алгоритм подобной подпрограммы представлен на рисунке 4.

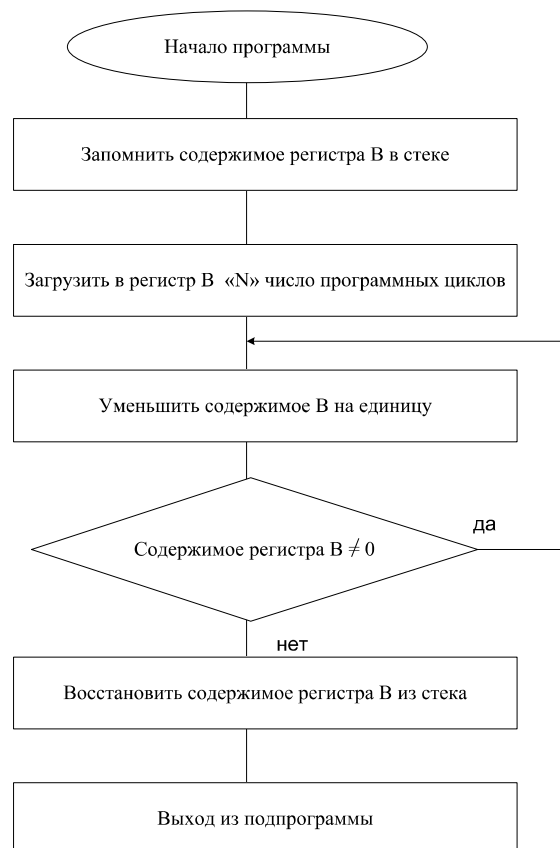


Рисунок 4 -Алгоритм временной задержки.

Текст программы, соответствующий структуре графического алгоритма (рисунок 4) можно представить в следующем виде (таблица.3.1).

Для достижения заданной временной задержки необходимо определить значение числа N повторений программных циклов, вводимого в регистр "B". Искомое число N определяется временем выполнения команд, входящих в данную программу. Если временная задержка выполнена в виде подпрограммы, то при расчёте необходимо учитывать время на обращение к подпрограмме (CALL TIME 1, $t_0=8,5$ мкс) и выхода из подпрограммы (RET, $t_6=5,0$ мкс).

Таблица 3.1 Определение числа программных циклов " N" для организации требуемой временной задержки.

Метка	Мнемокод	Время выполнения (мкс)	Комментарий
TIME 1	PUSH B	$t_1=5,5$	Содержимое регистра В запомнить в стеке
	MVI B, "N"	$t_2=3,5$	Загрузка N в регистр В числа циклов
M1	DCR B	$t_3=2,5$	Уменьшение В на единицу
	JNZ M1	$t_4=5,0$	Если не 0 возврат на метку L1
	POP B	$t_5=5,0$	Восстановить регистр В из стека
	RET	$t_6=5,0$	Возврат из подпрограммы

Согласно таблице 3.1 длительность временной задержки:

$$T_1 = t_0 + t_1 + t_2 + N \times (t_3 + t_4) + t_5 + t_6.$$

Откуда требуемое число повторений N :

$$N = \frac{T_1 - t_0 - t_1 - t_2 - t_5 - t_6}{t_3 + t_4}$$

После определения N его необходимо перевести в шестнадцатеричный код и записать в соответствующем месте подпрограммы.

Пример. Определить применительно к подпрограмме TIME 1 количество N повторений программных циклов, обеспечивающих временную задержку $T_1=300$ мкс.

$$t_0=8,5 \text{ мкс}; t_1=5,5 \text{ мкс}; t_2=3,5 \text{ мкс}; t_3=2,5 \text{ мкс}; t_4=5,0 \text{ мкс};$$

$$t_5=5,0 \text{ мкс}; t_6=5,0 \text{ мкс};$$

$$T_1=300 \text{ мкс};$$

Требуемое число повторений:

$$N = \frac{300 - 8,5 - 5,5 - 3,5 - 5,0 - 5,0}{2,5 + 5,0} = 36,3$$

Принимаемое число повторений $N=36$. Действительная выдержка времени:

$$T_{1д} = 8,5 + 5,5 + 3,5 + 36(2,5 + 5,0) + 5,0 + 5,0 = 297,5 \text{ мкс}.$$

Относительная ошибка составит:

$$T = \frac{T_1 - T_{1д}}{T_1} \times 100 \% = \frac{300 - 297,5}{300} \times 100 \% = 0,83 \%$$

Как показали расчеты ошибка составляет весьма малую величину и в большинстве случаев ошибкой до 1% можно пренебречь.

Однако в тех случаях, когда необходимо повысить точность получения требуемых временных задержек в текст программы можно вставить команду NOP ($t_7=2\text{мкс}$). Причем если эту команду вставить вне цикла, то

$$T_{1д} = t_0 + t_1 + t_2 + N \times (t_3 + t_4) + t_5 + t_6 + t_7 = 297,5 + 2 = 299,5 \text{ мкс.}$$

и в этом случае:

$$T = \frac{T_1 - T_{1д}}{T_1} \times 100 \% = \frac{300 - 299,5}{300} \times 100 \% = 0,16 \%$$

Если команду NOP вставить в тело цикла, то $T_{1д}$ увеличится на время $N \times t_7 = 36 \times 2,0 = 72 \text{ мкс}$, то есть таким образом можно корректировать грубые ошибки временных задержек.

После проверки и уточнения десятичное число $N(10) = 36$ необходимо перевести в шестнадцатиричный код $N(16) = 24$ и загрузить в соответствующее место программы.

Максимальная временная задержка для подпрограммы подобного вида определяется максимальным значением числа $N = FF(16) = 256(10)$.

В этом случае

$$T_{1 \max} = 8,5 + 5,5 + 3,5 + 256 \times (2,5 + 5,0) + 5,0 + 5,0 + 2,0 = 1949,5 (\text{мкс})$$

При необходимости получения временных задержек большей длительности необходимы программы обеспечивающие большее количество повторений программных циклов.

3.2 Программная реализация временных задержек большой длительности.

При реализации некоторых программ, особенно программ управления различными внешними устройствами, появляется необходимость в временных задержках достаточно большой длительности. Для программной реализации таких временных задержек можно использовать алгоритм (рисунок 3), но с увеличением числа N до значений, обеспечивающих требуемую временную задержку. Для этого число N определяющее количество повторений цикла необходимо записать в пару регистров, в этом случае число повторений может достигать $FF FF(16)$ или $65535(10)$.

Тогда графический алгоритм подпрограммы с использованием пары регистров BC будет выглядеть так (рисунок 5):

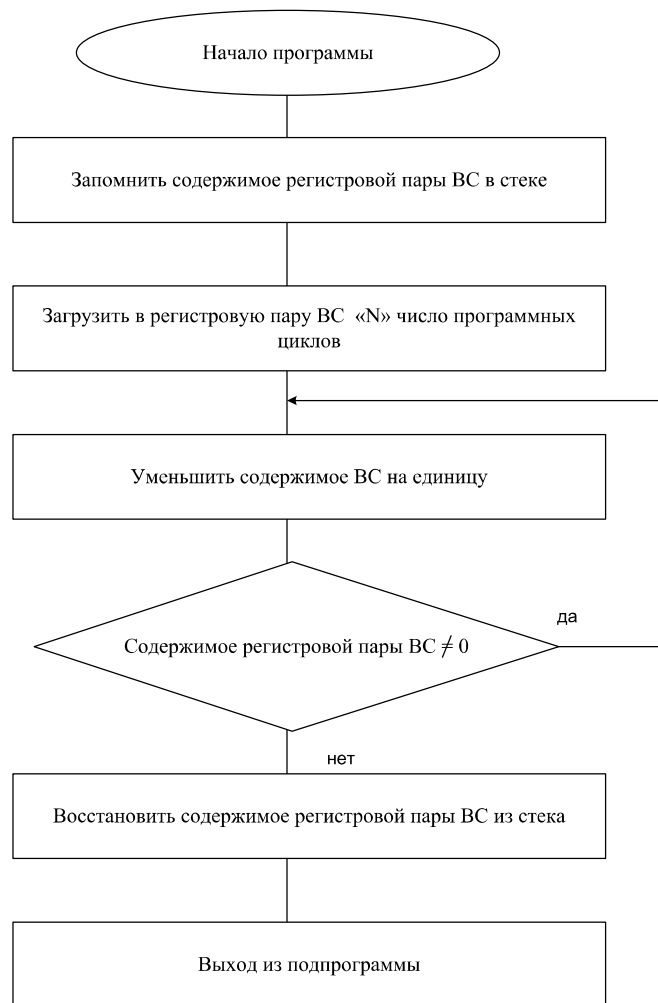


Рисунок 5 - Алгоритм подпрограммы временной задержки большой длительности

Текст программы ,соответствующий структуре графического алгоритма (рисунок 5) можно представить в следующем виде (таблица 3.2):

Таблица 3.2 Программа временной задержки большой длительности.

Метка	Мнемокод	Время выполнения,мкс	Комментарий
TIME 2	PUSH B	t1=5,5	Содержимое регистров В,С и А запомнить в стеке
	PUSH PSW	t2=5,5	
	LXI B, "N"	t3=5,0	Загрузка числа циклов N в регистры В и С
M2	DCX B	t4=2,5	Уменьшение В и С на единицу
	MOV A B	t5=2,5	Содержимое В перегрузить в А
	ORA C	t6=2,0	Логическое "ИЛИ" между А и С
	JNZ M2	t7=5,0	Если не 0 возврат на метку L2
	POP PSW	t8=5,0	Восстановить А
	POP B	t9=5,0	Восстановить регистры В и С из стека
	RET	t10=5,0	Возврат из подпрограммы

3.3 Расчет величины задержки времени

Максимальная задержка реализуемая подпрограммой TIME 2 (таблица 3.2) с учетом времени обращения к подпрограмме и числа $N=FF\ FF(16)=65535(10)$ составит :

$$T2_{max}=8,5+5,5+5,5+5,0+65535 \times (2,5+2,5+2,0+5,0)+5,0+5,0+5,0=786459,5 \\ (\text{мкс})=0,786 (\text{с}).$$

Как и в предыдущем примере увеличить длительность задержки можно добавлением одной или нескольких команд NOP внутри или вне тела цикла.

Задержки длительностью от единиц до десятков секунд можно получить последовательной реализацией нескольких подпрограмм типа TIME1 и TIME2.

4. ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ.

4.1.Изучить программы, приведенные в методическом указании. Понять принципы их функционирования назначение действий, которые они производят.

4.2.Разработать программу, заполняющую область памяти с адреса A00H и заканчивая адресом B00H, константой 98H. Просмотреть результаты выполнения программы.

4.3.Разработать программу, находящую наименьшее число, в массиве начиная с адреса 900H и заканчивая адресом 950H и записывающую его в аккумулятор. Просмотреть результаты выполнения программы.

4.4.Разработать программу, определяющую количество байт, у которых старший бит установлен в 1.Область памяти для поиска 980H-990H.

Количество байт записать в регистр D. Просмотреть результаты выполнения программы.

4.5. Разработать программу, которая производит инверсию массива из 12 однобайтовых чисел расположенных с адреса 970H.

4.6.Разработать программу временной задержки на время, указанное преподавателем. Представить полный расчет времени задержки данной программы.

5.СОДЕРЖАНИЕ ОТЧЕТА.

Отчет должен содержать - текст программы и перечень команд передачи управления микропроцессора KP580BM80A (i8080 /8085).

6.ВОПРОСЫ.

6.1.По каким условиям записывается 1 или 0 в разряды флагового регистра?

6.2.Организация ветвления в программах.

6.3.Организация циклов и циклических структур.

6.4.Какие команды микропроцессора ответственны за выполнение ветвления в программах; как они функционируют.

6.5. Для чего предназначена регулирующая логика?

6.6. Как определить требуемое количество программных циклов ?

Виды контрольных мероприятий в баллах

	Виды контрольных мероприятий	Баллы	№ недели
	Выполнение и защита лабораторных работ 2	15	4-6
	ИТОГО	15	

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Нарышкин, Александр Кириллович. Цифровые устройства и микропроцессоры [Текст]: учебное пособие для студентов вузов радиотехнических специальностей / А. К. Нарышкин. - М. : Академия, 2006. - 319 с. - (Высшее профессиональное образование. радиоэлектроника).

2. Фритч, Вольфганг. Применение микропроцессоров в системах управления [Текст] = Automatisierte Systeme mit Prozess- und Mikroprozessrechnern : пер. с нем. / В. Фритч. - М. : Мир, 2005. - 463 с. : граф., табл., рис. - Список лит. -Предм. указ.: с. 461-463. . - (в пер.) : ГРНТИ 50.09.33

3. Корнеев, Виктор Владимирович. Современные микропроцессоры [Текст] : научно-популярная литература / В.В. Корнеев, А.В. Киселев. - 3-е изд., перераб. и доп. - СПб. : БХВ-Петербург, 2003. - 440 с. : ил. - Список лит. - Предм. указ.: с.434-440. - 3000 экз. - ISBN 5-94157-385-5