Федеральное государственное автономное образовательное учреждение высшего образования "Российский университет транспорта"

Российская открытая академия транспорта (РОАТ)

Методические указания для курсовой работы (проекта) по дисциплине:

«Основы микропроцессорной техники и прикладное программирование»

На тему: «Программирование микропроцессора серии Intel 8080 (К580)»

Общие указания и задание на курсовую работу (проект)

Курсовая работа (проект) посвящен программированию микропроцессорной системы серии К580 (аналог i8080).

В курсовой работе (проекте) требуется:

- дать ответы на вопросы из Приложения 2
- изучить заданный алгоритм работы микропроцессорного устройства управления объектом и дать его описание;
- составить и описать машинный алгоритм функционирования микропроцессорного устройства
- по составленному машинному алгоритму написать в операторах языка Ассемблер программу с метками и комментариями, используя систему команд из Приложения 1.
- определить адресное пространство программы, используемые адреса ячеек ОЗУ, адреса портов вывода
- полученную программу записать в машинных кодах с использованием шестнадцатеричной системы счисления.

Для выполнения курсовой работы (проекта) необходимо:

- изучить лекционный и практический материал, рекомендуемую литературу;
- определить свой вариант задания (Приложение 2, 3, 4);
- привести краткие сведения по изучаемому устройству и ответы на конкретные вопросы по варианту задания.

Вариант задания выбирается из Приложений 2 – 4 к данным методическим указаниям. Теоретические вопросы, на которые следует дать ответ в курсовой работе выбираются из приложения 2 по предпоследней цифре номера зачетной книжки обучающегося.

Алгоритм работы микропроцессорного устройства задается последней цифрой номера зачетной книжки обучающегося и находится в Приложении 3. В этом же приложении находится условие подпрограммы для выбранного варианта. Все параметры (par1, par2, dop) должны быть заданы в ячейках памяти в конце программы. Начальная ячейка подпрограммы выбирается произвольно. Номера портов ввода/вывода выбираются из Приложения 4 по предпоследней цифре номера зачетной книжки обучающегося.

Последние три цифры номера зачетной книжки обучающегося, представленные в шестнадцатеричной системе счисления, определяют адрес ячейки памяти, с которой начинается программа.

Две последние цифры номера зачетной книжки обучающегося, представленные в шестнадцатеричной системе счисления, определяют значения константы допуска dop, используемой в алгоритме.

Значения остальных параметров (par1, par2, par3 и др.) при необходимости их использования, выбираются произвольно.

В курсовой работе (проекте) должны быть выполнены все пункты задания.

Пояснительная записка должна содержать:

- исходные данные, выбранные по варианту задания, и перевод параметров в шестнадцатеричную систему счисления
- выбранные произвольно значения параметров (при необходимости их применения)
- краткие теоретические сведения по изучаемому устройству
- ответы на вопросы, выбранные по варианту задания
- схема заданного алгоритма и ее описание
- схема машинного алгоритма и ее описание
- отдельный машинный алгоритм подпрограммы и его описание
- программа, составленная с учетом исходных данных на языке Ассемблер с комментариями и метками
- подпрограмма, составленная с учетом исходных данных на языке Ассемблер с комментариями и метками
- определить адресное пространство написанной программы и подпрограммы микропроцессорного устройства.

Общие требования по оформлению курсовой работы (проекта)

Работа (проект) должна быть выполнена в печатном виде на листах формата А4 и прошита.

Титульный лист работы (проекта) обязательно должен содержать название дисциплины и тему курсовой работы (проекта), фамилию, инициалы и номер зачетной книжки студента (полностью!).

В работе обязательно нумеруются страницы, кроме титульной, а также должны быть обязательно следующие элементы:

- содержание
- введение
- исходные данные
- основная часть
- заключение
- список литературы

Требования к тексту:

- шрифт Times New Roman, 14
- основной текст работы форматируется по ширине с наличием абзацев
- заголовки оформляются по середине и могут иметь нумерацию
- все приведенные схемы вставляются в пояснительную записку после той страницы, на которой имеется первая ссылка на них.

Методические указания по выполнению основной части курсовой работы (проекта)

Задание приведем в укороченном виде без ответа на теоретические вопросы из Приложения 2:

- Выбрать исходные данные и произвести перевод параметров в шестнадцатеричную систему счисления;
- Изучить заданный алгоритм работы микропроцессорного устройства управления объектом и дать его описание;
- Составить машинный алгоритм работы микропроцессорного устройства и дать его описание;
- Написать текст программы на языке Ассемблер с комментариями;
- Представить текст программы в машинных кодах используемого микропроцессора в шестнадцатеричной системе счисления;
- Определить адресное пространство программы, используемые адреса ячеек памяти, адреса портов вывода.

Рассмотрим решение представленной задачи.

Предположим, что номер зачетной книжки обучающегося 2010-СДс-1195. согласно представленному номеру зачетной книжки выбирается алгоритм работы микропроцессорного устройства из Приложения 3 по последней цифре приведенного номера. Представленные далее данные алгоритма не относятся к какому-либо определенному варианту задания из Приложений и взяты для примера выполнения работы (проекта).

Найдем оставшиеся исходные данные.

Номер зачетной книжки обучающегося заканчивается на «95», следовательно переведя данное число в шестнадцатеричную систему счисления будет получен параметр допуска dop. Перевод представлен ниже.

Шестнадцатеричная система счисления – это позиционная целочисленная система счисления с основанием 16. Является одной из самых популярных в информатике, наряду с двоичной, восьмеричной и десятичной.

Шестнадцатеричная система, как и восьмеричная активно применяется в компьютерных технологиях. При этом, запись чисел гораздо компактнее. В отличии от восьмеричной, которая за годы развития информатики – устарела, шестнадцатеричная – применяется **в следующих областях:**

1. Низкоуровневое программирование (к примеру, ассемблер).
2. Стандарт Юникод.
3. Шестнадцатеричный цвет (RGB).
4. Запись кодов ошибок.
5. Представление данных в малоразрядных ЭВМ.

Шестнадцатеричная система — это традиционная система счисления **с основанием 16.** Алфавит состоит их цифр **от 0 до 9** и латинских букв **от A до F**. Латинские буквы представляют собой десятичные числа от 10 до 15.

Развернутая форма записи числа будет выглядеть **следующим образом:**

$$A_{16} = a_{n-1} \cdot 16^{n-1} + a_{n-2} \cdot 16^{n-2} + \cdots + a_0 \cdot 16^0 + a_{-1} \cdot 16^{-1} + \cdots + a_{-m} \cdot 16^{-m}$$

Например:

$$3BD_{16} = 3 \cdot 16^2 + B \cdot 16^1 + D \cdot 16^0 = 3 \cdot 256 + 11 \cdot 16 + 13 \cdot 1$$
$$= 768 + 176 + 13 = 957_{10}$$

Для перевода в шестнадцатеричную систему счисления можно воспользоваться стандартным делением на основание системы счисления, в нашем случае основание «16».

Переводим целую часть 95_{10} в 16-ую систему последовательным делением на 16:

95/16 = 5, остаток: 15, 15 = **F**

5/16 = 0, остаток: **5**

95_{10} = **5F**$_{16}$

Очень часто при обозначении шестнадцатеричной системы счисления вместо нижнего индекса «16» в качестве указателя системы счисления используется написание латинской буквы «h» («Н»), что далее будет показано при упоминании значений параметров.

Также найдем значение ячейки памяти, с которой начнется адресное пространство программы в нашем варианте задания.

Последние три цифры номера зачетной книжки обучающегося «195», поэтому именно их переведем в шестнадцатеричную систему счисления.

Переводим целую часть 195_{10} в 16-ую систему последовательным делением на 16:

195/16 = 12, остаток: **3**

12/16 = 0, остаток: 12, 12 = **C**

195_{10} = **C3**H

В микропроцессорных устройствах программа хранится в оперативном запоминающем устройстве (ОЗУ). Нумерация ячеек ОЗУ в программе указывается в шестнадцатеричной системе и состоит из 4-х знаков. Диапазон значений ячеек ОЗУ в МК К580 составляет $0000 \div FFFF$. Таким образом к полученному числу добавим слева два нуля, которые технически не меняют само число, но позволяют заполнить нужные разряды.

Получим следующее значение: **00C3**H

Алгоритм работы устройства управления объектом приведен на рис.1.

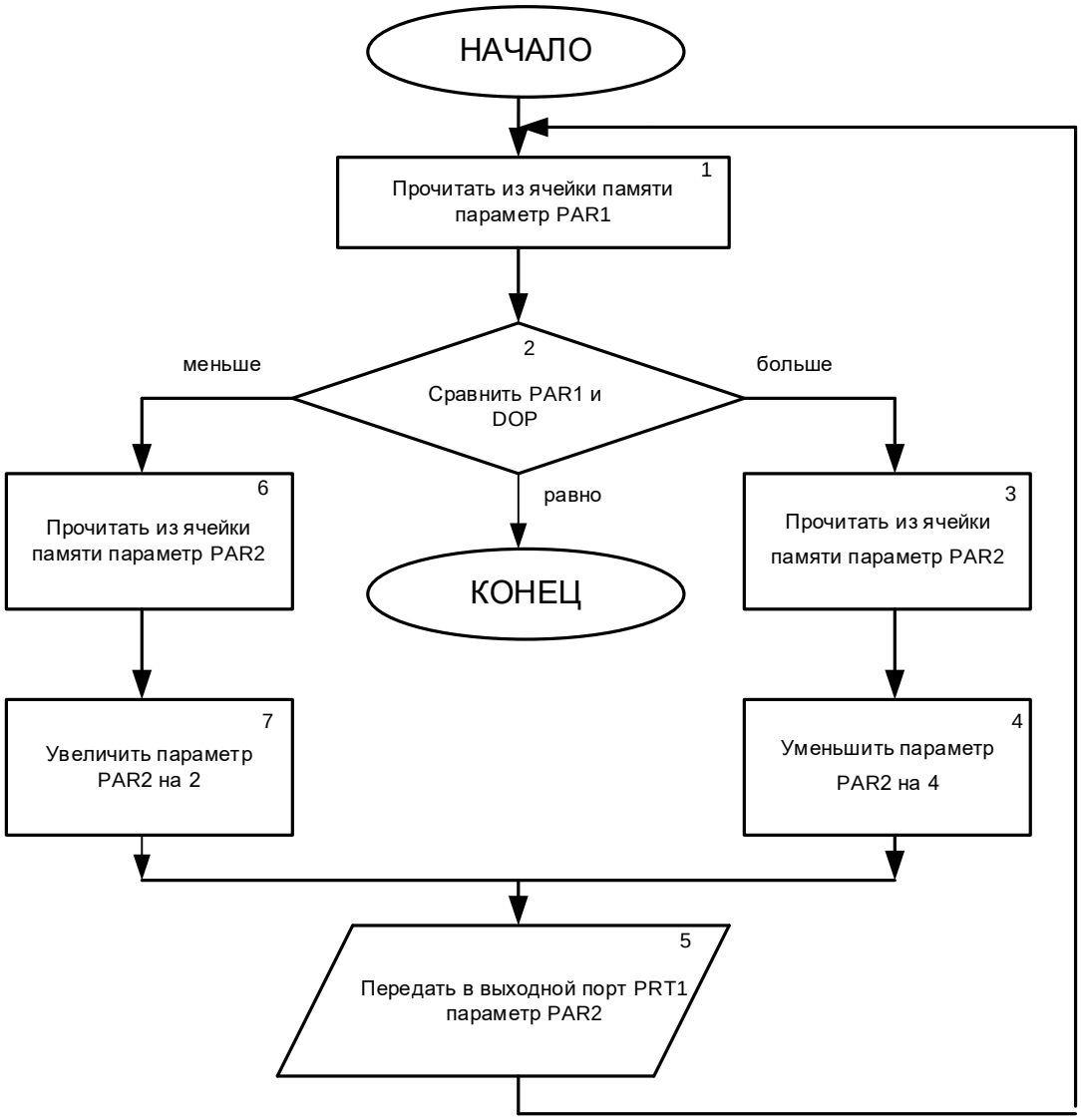


Рис. 1. Алгоритм работы устройства управления

Изучение предложенного алгоритма позволяет представить микропроцессорное устройство управления (МУУ) следующими функциями.

МУУ считывает из памяти параметр PAR1 (блок 1) и сравнивает его с допуском DOP (блок 2), значение которого задано.

Если PAR1 больше DOP, то алгоритмом функционирования предусмотрен переход к блокам 3 и 4 последовательно, где МУУ считывает из памяти параметр PAR2 (блок 3) и уменьшает его на 4 (блок 4). После чего результат передается в выходной порт PRT1 (блок 5). Далее алгоритмом предусмотрен возврат к блоку 1.

Если PAR1 меньше DOP, то МУУ считывает из памяти параметр PAR2 (блок 6), увеличивает его значение на 2 (блок 7) и результат выдается в порт PRT1 на объект управления (блок 5). Далее алгоритмом предусмотрен возврат к блоку 1.

Процесс управления заканчивается при достижении ситуации, когда PAR1 равен DOP.

Далее опишем назначение каждого блока структурной схемы и, на основании принципов функционирования МУУ и системы команд, составим машинный алгоритм работы устройства управления.

Машинный алгоритм представлен на рис.2.

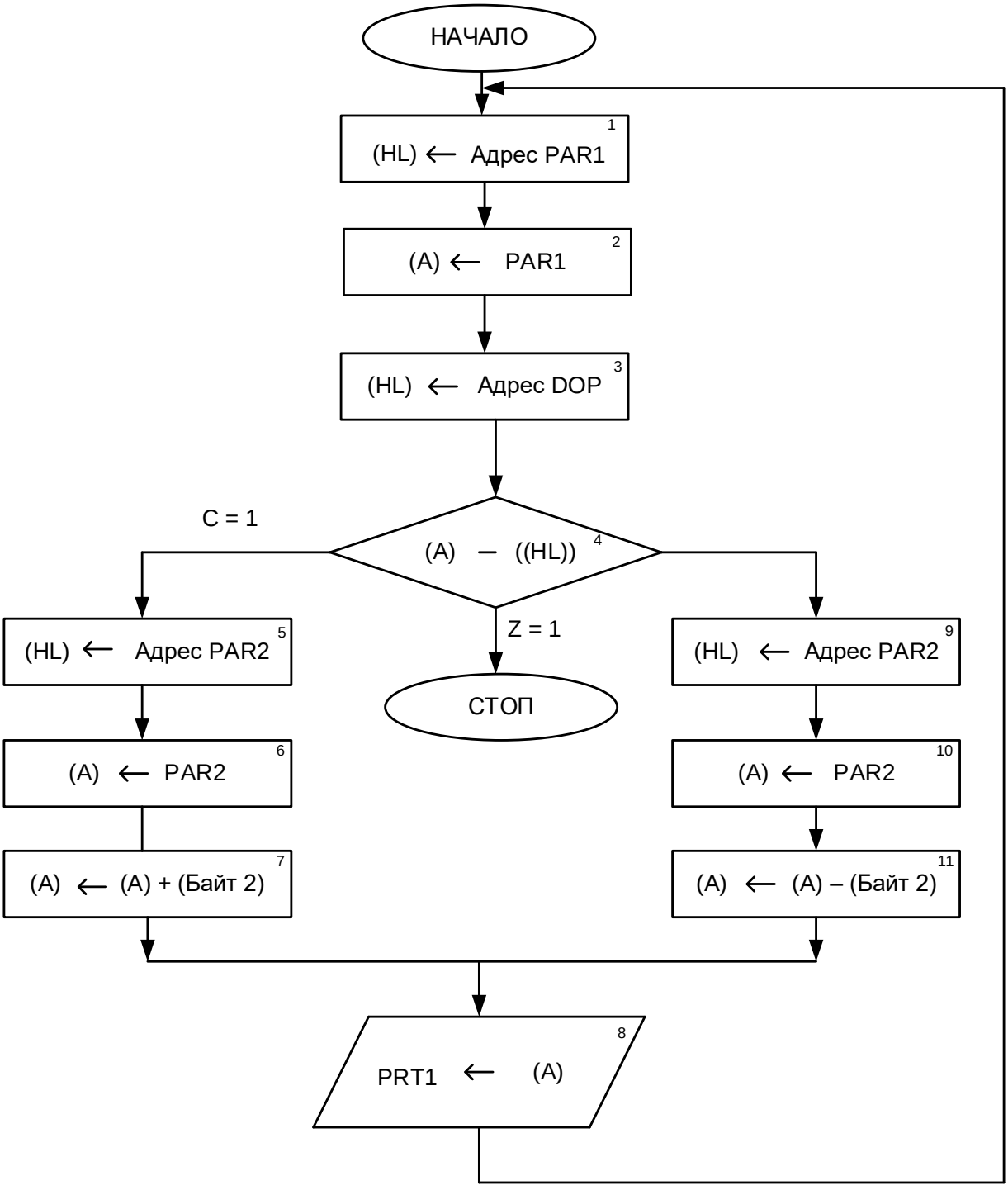


Рис. 2. Машинный алгоритм функционирования МУУ

Блоком 1 алгоритма осуществляется загрузка пары регистров HL адресом ячейки памяти, в которой хранится параметр PAR1. Блок 2 пересылает содержимое ячейки памяти (параметр PAR1), адрес которой указан в паре

регистров HL, в регистр аккумулятор. Блок 3 осуществляет загрузку пары регистров HL адресом ячейки памяти, где находится параметр DOP.

Блок 7 осуществляет сравнение содержимого регистра аккумулятора A и содержимого ячейки памяти M, адресуемой парой регистров HL. В ячейке памяти хранится параметр DOP, а в регистре аккумулятора – параметр PAR1. В результате сравнения содержимого аккумулятора с содержимым ячейки памяти устанавливаются флаги C и Z.

В первую очередь проверяется выполнение условия установки флага Z. Если (A) равно (M), то разность (A) – (M) = 0, устанавливается флаг Z = 1 и осуществляется переход к блоку «СТОП». В результате выполнения операции сравнения содержимое регистра аккумулятора, где хранится параметр PAR1, не изменяется.

Далее будет проверено условие установки флага C. Если $(A) < (M)$, то разность $(A) – (M) < 0$, устанавливается флаг C = 1 и осуществляется переход к блоку 5. В результате выполнения операции сравнения содержимое регистра аккумулятора, где хранится параметр PAR1, не изменяется.

Блоком 5 алгоритма осуществляется загрузка пары регистров HL адресом ячейки памяти, в котором хранится параметр PAR2. Блок 6 пересылает содержимое ячейки памяти (параметр PAR2), адрес которой указан в паре регистров HL, в регистр аккумулятор. Блок 7 увеличивает содержимое регистра аккумулятора на десятичную константу 2, и результат заносится в аккумулятор. Далее блоком 8 содержимое аккумулятора передается в выходной порт PRT1, и управление передается на блок 1 алгоритма.

В случае, когда ни один из флагов не устанавливается, должно быть выполнено условие, когда $(A) – (M) > 0$.

Блоком 9 алгоритма осуществляется загрузка пары регистров HL адресом ячейки памяти, в которой хранится параметр PAR2. Блок 10 пересылает содержимое ячейки памяти (параметр PAR2), адрес которой указан в паре регистров HL, в регистр аккумулятор. Блок 11 уменьшает содержимое регистра аккумулятора на десятичную константу 4, и результат заносится в аккумулятор. Далее блоком 8 содержимое аккумулятора передается в выходной порт PRT1, и управление передается на блок 1 алгоритма.

В соответствии с рассмотренным алгоритмом составим программу для МУУ (таб.1).

Таблица 1

Программа функционирования МУУ

Машинные коды		Ассемблер		
Адрес	Код	Метка	Команда	Комментарий
00C3H	21H	BEGIN	LXI H, 01B2H	; загрузить в пару регистров HL адрес ячейки памяти, в которой хранится параметр PAR1.
00C4H	B2H			
00C5H	01H			
00C6H	7EH		MOV A, M	; переслать содержимое ячейки памяти, адрес которой находится в паре HL, в регистр аккумулятор.
00C7H	23H		INX H	; инкремент содержимого пары регистров HL.
00C8H	BEH		CMP M	; сравнение содержимого регистра аккумулятора с содержимым ячейки памяти, адрес которой находится в паре регистров HL.
00C9H	CAH		JZ STOP	; переход по адресу STOP при равенстве сравниваемых величин.
00CAH	B1H			
00CBH	01H			
00CCH	DAH		JC LET1	; переход по адресу LET1 при отрицательном результате сравнения.
00CDH	AAH			
00CEH	01H			
00CFH	23H		INX H	; инкремент содержимого пары регистров HL.
00D0H	7EH		MOV A, M	; переслать содержимое ячейки памяти, адрес которой находится в паре HL, в регистр аккумулятор.
00D1H	D6H		SUI 4H	; уменьшение параметра PAR2 на 4
00D2H	04H			
00D3H	D3H	LET2	OUT F0	; вывод параметра PAR2 в порт PRT1.
00D4H	F0H			
00D5H	C3H		JMP BEGIN	; передача управления команде по адресу BEGIN.
00D6H	95H			
00D7H	01H			
00D8H	23H	LET1	INX H	; инкремент содержимого пары регистров HL
00D9H	7EH		MOV A, M	; переслать содержимое ячейки памяти, адрес которой находится в паре HL, в регистр аккумулятор
00DAH	C6H		ADI 2H	; увеличение параметра PAR2 на 2
00DBH	02H			
00DCH	C3H		JMP LET2	; переход по адресу LET2
00DDH	A5H			
00DEH	01H			
00DFH	76H	STOP	HLT	; останов.
00E0H	64H	PAR1		параметр PAR1
00E1H	5FH	DOP		параметр DOP
00E2H	40H	PAR2		параметр PAR2

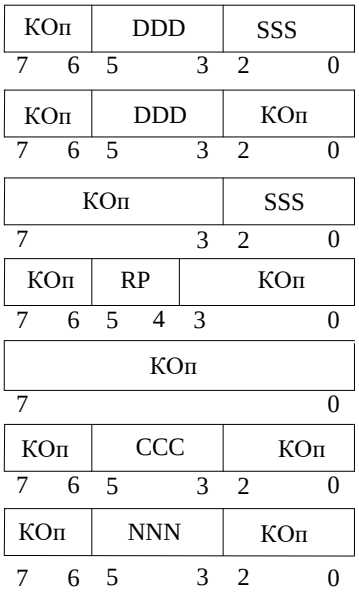
Таким образом, программа функционирования МУУ размещается в 32 ячейках памяти. Ячейка памяти с адресом 00E0H используется для хранения параметра PAR1, равного 64H, ячейка с адресом 00E1H – для хранения допуска DOP, равного 5FH, ячейка с адресом 00E2H – для хранения параметра PAR2, равного 40H. Значения параметров PAR1 и PAR2 выбраны произвольно в шестнадцатеричной системе счисления. Адресное пространство памяти, занимаемое программой, определяется адресами 00C3H ÷ 00E2H. Порту PRT1 присвоен адрес F0H.

Приложение 1

Система команд микропроцессорного устройства серии К580

Однобайтная команда

Варианты однобайтных команд



Коп - поле кода операции

DDD - трехбитный номер регистра-приемника

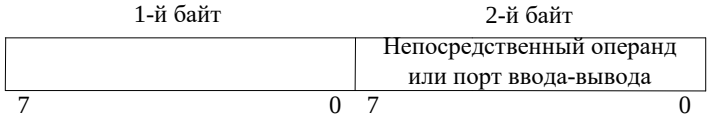
SSS - трехбитный номер регистра-источника

RP - двухразрядный указатель, адресующий пару регистров

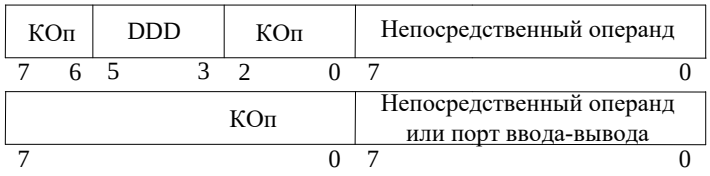
CCC - трехразрядный указатель условия передачи управления

NNN - трехразрядный код, определяющий адрес вектора прерывания

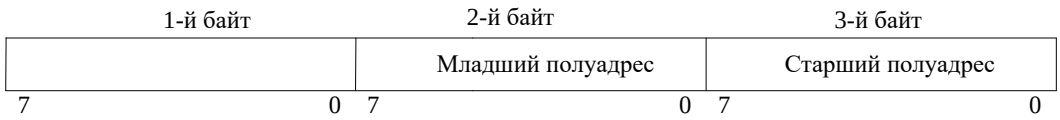
Двухбайтная команда



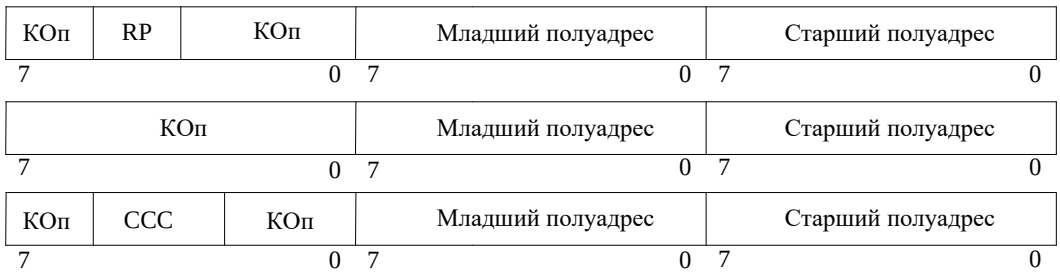
Варианты двухбайтных команд



Трехбайтная команда



Варианты трехбайтных команд



АДРЕСА РЕГИСТРОВ

Адрес (R)	Регистр
111	A
000	B
001	C
010	D
011	E
100	H
101	L

АДРЕСА ПАРЫ РЕГИСТРОВ

Адрес (RP)	Пара регистров
00	BC
01	DE
10	HL
11	SP

КОДЫ УСЛОВИЙ (cnd), ИСПОЛЬЗУЕМЫХ В КОМАНДАХ УСЛОВНОГО ПЕРЕХОДА, ВЫЗОВА ПОДПРОГРАММ И ВОЗВРАТА ИЗ НИХ

Коды	Мнемоника	Условия
000	NZ	Не ноль (Z = 0)
001	Z	Ноль (Z = 1)
010	NC	Нет переноса (C = 0)
011	C	Есть перенос (C = 1)
100	P0	Нечетный результат (P = 0)
101	PE	Четный результат (P = 1)
110	P	Результат положительный (S = 0)
111	M	Результат отрицательный (S = 1)

Система команд микропроцессора серии К580ВМ80А (i8080)

Наименование	Число тактов/циклов	Двоичный код	Количество байтов в команде	Название и описание
Команды передачи данных				
MOV R1, R2	5/1	01DDDSSS	1	Передача между регистрами. (R1) ← (R2). Содержимое регистра R2 передается в регистр R1. Содержимоерегистра R2 не меняется.
MOV R, M	7/1	01DDD110	1	Передача из памяти. (R) ← M(HL). Содержимое ячейки памяти, адрес которой находится в паре регистров HL,передается в регистр R.
MOV M, R	7/1	01110SSS	1	Передача в память. M(HL) ← (R). Содержимое регистра R передается в ячейку памяти, адрес которой находится впаре регистров HL.
MVI R, data	7/2	00DDD110	2	Непосредственная передача в регистр. (R) ← (байт2). Содержимое байта 2 команды передается в регистр R.
MVI M, data	10/2	00110110	2	Непосредственная передача в память. M(HL) ← (байт2). Содержимое байта 2 команды передается в ячейку памяти, адрес которой указан в паре регистров HL.
Загрузка				
LXI RP, addr	10/3	00RP0001	3	Непосредственная загрузка пары регистров. (RH) ← (байт3); (RL) ← (байт2). Байт 3 команды передается в старший регистр (RH) регистровой пары RP, а байт 2 в младший регистр (RL) этой же пары.
LDA addr	13/3	00111010	3	Прямая загрузка аккумулятора. (A) ← ((байт3)(байт2)). Содержимое ячейки памяти, адрес которой определенбайтами 2 и 3 команды, передается в аккумулятор.

LHLD addr	16/3	00101010	3	Прямая загрузка пары регистров. L ← ((байт3)(байт2)); H ← ((байт3)(байт2)+1). Содержимое ячейки памяти, адрес которой определяется байтами 2 и 3 команды передается в регистр L. Содержимое следующей ячейки памяти передается в регистр H.
LDAX RP	7/1	00RP1010	1	Косвенная загрузка аккумулятора. (A) ← (M). Содержимое ячейки памяти, адрес которой определяется парой регистров RP, передается в аккумулятор. Могут использоваться только пары RP = B (регистры B и C) или RP = D (регистры D и E).
XCHG	4/1	11101011	1	Обмен. (H) ↔ (D); (L) ↔ (E). Содержимое регистров H и L обмениваются с содержимым регистров D и E.
STA addr	13/3	00110010	3	Прямое размещение содержимого аккумулятора. ((байт3)(байт2)) ← (A). Содержимое аккумуляторапередается в ячейку памяти, адрес которой указан байтами 2 и 3 команды.
SHLD addr	16/3	00110010	3	Прямое размещение содержимого регистров. ((байт3)(байт2)) ← (L); ((байт3)(байт2)+1) ← (H). Содержимое регистра L передается в ячейку памяти, адрес которой определяется байтами 2 и 3 команды, одержимое регистра H передается в следующую ячейку памяти.
STAX RP	7/1	00RP0010	1	Косвенная загрузка аккумулятора. M(RP) ← (A). Содержимое аккумулятора передается в ячейку памяти,адрес которой содержится в паре регистров RP. Могут использоваться только пары RP = B (регистры B и C) или RP = D (регистры D и E).

Арифметические команды				
ADD R	4/1	10000SSS	1	Сложение содержимого регистра. (A) ← (A)+(R). Содержимое регистра R складывается с содержимымаккумулятора. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, C, P, AC.
ADD M	7/2	10000110	1	Сложение содержимого ячейки памяти. (A) ← (A)+((H)(L)). Содержимое ячейки памяти, адрескоторой указан в паре регистров HL, складывается с содержимым аккумулятора. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, C, P, AC.
ADI data	7/2	11000110	2	Непосредственное сложение. (A) ← (A)+(байт2). Содержимое байта 2 команды складывается с содержимым аккумулятора. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, C, P, AC.
ADC R	4/1	10001SSS	1	Сложение содержимого регистра и переноса. (A) ← (A)+(R)+(C). Содержимое регистра R и флагапереноса C (бит переполнения) складывается с содержимым аккумулятора. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, C, P, AC.
ADC M	7/2	10001110	1	Сложение содержимого ячейки памяти и переноса. (A) ← (A)+((H)(L))+(C). Содержимое ячейки памяти, адрес которой указан в паре регистров HL, и флага переноса C складывается с содержимым аккумулятора. Результат помещается в аккумулятор. Устанавливаютсяфлаги – Z, S, C, P, AC.

ACI data	7/2	11001110	2	Непосредственное сложение с учетом переноса. (A) ← (A)+(байт2)+(C). Содержимое байта 2 команды ифлага переноса C складывается с содержимым аккумулятора. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, C, P, AC.
DAD RP	10/3	00RP1101	1	Сложение содержимого пары регистров с содержимымпары регистров HL. (H)(L) ← (H)(L)+(RH)(RL). Содержимое пары регистров RP складывается с содержимым пары регистров HL. Устанавливается флаг C. C=1, если есть перенос при сложении с удвоенной точностью, C=0, если нет переноса.
DAA	4/1	00100111	1	Десятичная коррекция аккумулятора: 1. Если (A_3…A_0) > 9 или AC = 1, то (A)+6; 2. Если (A_7…A_4) > 9 или C = 1, то (A_7…A_4)+6
SUB R	4/1	10010SSS	1	Вычитание содержимого регистра (A) ← (A)-(R). Содержимое регистра R вычитается из содержимого аккумулятора. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, C, P, AC.
SUB M	7/2	10010110	1	Вычитание содержимого ячейки памяти. (A) ← (A)-(M). Содержимое ячейки памяти, адрес которой указан в паре регистров HL, вычитается из содержимого аккумулятора. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, C, P, AC.
SUI data	7/2	11010110	2	Непосредственное вычитание. (A) ← (A)-(байт2). Содержимое байта 2 команды вычитается из содержимого аккумулятора. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, C, P, AC.

SBB R	4/1	10011SSS	1	Вычитание содержимого регистра и переноса (A) ← (A)-(R)-(C). Содержимое регистра R и флага переноса C вычитается из содержимого аккумулятора. Результат помещается в аккумулятор. Устанавливаютсяфлаги – Z, S, C, P, AC.
SBB M	7/2	10011110	1	Вычитание содержимого ячейки памяти и переноса. (A) ← (A)-(M)-(C). Содержимое ячейки памяти, адрес которой указан в паре регистров HL, и флага переноса Свычитается из содержимого аккумулятора. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, C, P, AC
SBI data	7/2	11011110	2	Непосредственное вычитание данных и переноса. (A) ← (A)-(байт2)-(C). Содержимое байта 2 команды ииндикатора переноса C вычитается из содержимого аккумулятора. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, C, P, AC.
INR R	5/1	00DDD100	1	Инкремент содержимого регистра. (R) ← (R)+1.Содержимое регистра увеличивается на 1. Устанавливаются флаги – Z, S, P, AC.
INR M	10/3	00110100	1	Инкремент содержимого ячейки памяти. (M) ← ((H)(L))+1. Содержимое ячейки памяти, адрес которой указан в паре регистров HL, увеличивается на 1. Устанавливаются флаги – Z, S, P, AC
INX RP	5/1	00RP0011	1	Инкремент содержимого пары регистров. (RH)(RL) ← (RH)(RL)+1. Содержимое пары регистров RP увеличивается на 1
DCR R	5/1	00DDD101	1	Декремент содержимого регистра. (R)(R)-1.Содержимое регистра уменьшается на 1. Устанавливаются флаги – Z, S, P, AC.
DCR M	10/3	00110101	1	Декремент содержимого ячейки памяти (M) ← (M)-1. Содержимое ячейки памяти, адрес которой указан в паре регистров HL, уменьшается на 1. Устанавливаются флаги – Z, S, P, AC.

DCX RP	5/1	00RP1011	1	Декремент содержимого пары регистров (RH)(RL) ← (RH)(RL)-1. Содержимое пары регистров RP уменьшается на 1.
Логические команды				
ANA R	4/1	10100SSS	1	«И» с содержимым регистра. (A) ← (A) ⋀ (R). Содержимое регистра R и содержимое аккумуляторалогически умножается. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, P, AC. C=0
ANA M	7/2	10100110	1	«И» с содержимым ячейки памяти. (A) ← (A) ⋀ (M).Содержимое ячейки памяти, адрес которой указан впаре регистров HL и содержимое аккумулятора логически умножается. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, P, AC. C = 0
ANI data	7/2	11100110	2	«И» непосредственно с данными. (A) ← (A) ⋀ (байт2).Содержимое байта 2 команды и содержимое аккумулятора логически умножается. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, P, AC. C = 0.
XRA R	4/1	10101SSS	1	«Исключающее ИЛИ» с содержимым регистра (A) ← (A) ⊕ (R). Исключающее ИЛИ выполняется с содержимым регистра R и содержимым аккумулятора. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, P. AC=0, C=0.
XRA M	7/2	10101110	1	«Исключающее ИЛИ» с содержимым ячейки памяти. (A) ← (A) ⊕ (M). Исключающее ИЛИ выполняется с содержимым ячейки памяти, адрес которой указан в паре регистров HL и содержимым аккумулятора. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, P. AC=0, C=0.

XRI data	7/2	11101110	2	«Исключающее ИЛИ» непосредственно с данными. (A) ← (A) ⊕ (байт2). Исключающее ИЛИ выполняется с содержимым байта 2 команды и содержимым аккумулятора. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, P. AC=0, C=0.
ORA R	4/1	10110SS	1	«ИЛИ» с содержимым регистра. (A) ← (A) ∨ (R). Содержимое регистра R логически складывается с содержимым аккумулятора. Результат помещается ваккумулятор. Устанавливаются флаги – Z, S, P. AC=0, C=0.
ORA M	7/2	10110110	1	«ИЛИ» с содержимым ячейки памяти. (A) ← (A) ∨ (M).Содержимое ячейки памяти, адрес которой указан в паре регистров HL логически складывается с содержимым аккумулятора. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, P. AC=0, C=0.
ORI data	7/2	11110110	2	«ИЛИ» непосредственно с данными. (A) ← (A) ∨ (байт2). Содержимое байта 2 команды логически складывается с содержимым аккумулятора. Результат помещается в аккумулятор. Устанавливаются флаги – Z, S, P. AC=0, C=0.
CMP R	4/1	10111SSS	1	Сравнить содержимое регистра. (A)-(R). Содержимое регистра R вычитается из содержимого аккумулятора. Содержимое аккумулятора не изменяется. Флаги – Z=1, если (A)=(R), C=1, если (A)<(R).
CMP M	7/2	10111110	1	Сравнить содержимое ячейки памяти. (A)-(M). Содержимое ячейки памяти, адрес которой указан паройрегистров HL, вычитается из содержимого аккумулятора. Содержимое аккумулятора не изменяется. Флаги – Z=1, если (A)=(M), C=1, если (A)<(M).
CPI data	7/2	11111110	2	Непосредственно сравнить данные. (A)-(байт2). Содержимое байта 2 команды вычитается из содержимого аккумулятора. Содержимое

				аккумулятора не изменяется. Флаги – Z=1, если (A)=(байт2), C=1, если (A)<(байт2).
RLC	4/1	00000111	1	Сдвиг влево. (A) ← (2A). Содержимое аккумуляторасдвигается на один разряд влево ($A_0 \leftarrow A_7$, $C \leftarrow A_7$, $A_{n+1} \leftarrow A_n$). Устанавливается флаг C.
RRC	4/1	00001111	1	Сдвиг вправо. (A) ← (A/2). Содержимое аккумуляторасдвигается на один разряд вправо ($A_7 \leftarrow A_0$, $C \leftarrow A_0$, $A_n \leftarrow A_{n+1}$). Устанавливается флаг C.
RAL	4/1	00010111	1	Циклический сдвиг влево. Содержимое аккумуляторасдвигается влево вместе с C ($C \leftarrow A_7$, $A_0 \leftarrow C$, $A_{n+1} \leftarrow A_n$). Устанавливается флаг C.
RAR	4/1	00011111	1	Циклический сдвиг вправо Содержимое аккумуляторасдвигается вправо вместе с C ($A_7 \leftarrow C$, $C \leftarrow A_0$, $A_n \leftarrow A_{n+1}$). Устанавливается флаг C.
CMA	4/1	00101111	1	Инвертировать содержимое аккумулятора. (A) ← ($\overline{A}$). Содержимое аккумулятора инвертируется.
CMC	4/1	00111111	1	Инвертировать флаг переноса. (C) ← ($\overline{C}$). Инвертируется флаг переноса.
STC	4/1	00110111	1	Установить перенос. (C) ← 1. Флаг переноса устанавливается в 1.
Команды ветвлений и переходов				
JMP addr	10/3	11000011	3	Ветвление. (PC) ← (байт3)(байт2). Управление передается команде, адрес которой указан в байтах 3 и 2 текущей команды.
Jcnd addr	10/3	11cnd010	3	Условное ветвление. Если (cnd), то (PC) ← ((байт3)(байт2). Если условие выполняется, то управление передается команде, адрес которой указан вбайтах 2 и 3 текущей команды, иначе – выполняется следующая команда программы.

CALL addr	17/5	11001101	-	Вызов подпрограммы. (M_0) ← (PCH), (адрес (M_0)содержится в (SP)-1); (M_1) ← (PCL), (адрес (M_1) содержится в (SP)-2); (SP) ← ((SP)-2); (PC) ← (байт3)(байт2).
Ccnd addr	17/5	11cnd100	3	Условный вызов подпрограммы. Если условие выполняется, то действия те же, что и в команде CALL, иначе – выполняется следующая команда программы.
RET	10/3	11001001	1	Возврат из полпрограммы. (M_0) ← (PCL) (адрес (M_0) содержится в (SP)); (M_1) ← (PCH) (адрес (M_1) содержится в (SP)+1); (SP) ← (SP)+2.
Rcnd	11/3	11cnd000	1	Условный возврат из подпрограммы. Если условие выполняется, то действия те же, что и в RET, иначе –выполняется следующая команда программы
RST N	11/3	11NNN111	1	Рестарт. (M_0) ← (PCH), (адрес (M_0) содержится в (SP)-1); (M_1) ← (PCL), (адрес (M_1) содержится в (SP)-2); (SP) ← ((SP)-2); (PC) ← (NNN 8).
PCHL	5/1	11101001	1	Косвенный переход по адресу, указанному в паре регистров HL. (PCH) ← (H); (PCL) ← (L).
Команды ввода/вывода, управления и работы со стеком				
IN port	10/3	11011011	2	Ввести данные. (A) ← (port). Данные из порта, адрес которого указан в байте 2 команды записываются ваккумулятор.
OUT port	10/3	11010011	2	Вывести данные. (port) ← (A). Данные из аккумулятор записываются в порт, адрес которого указан в байте 2команды.

PUSH RP	11/3	11RP0101	1	Загрузить в стек содержимое пары регистров. (M_0) ← (RH), (адрес (M_0) содержится в (SP)-1); (M_1) ← (RL), (адрес (M_1) содержится в (SP)-2)); (SP) ← ((SP)-2).
PUSH PSW	11/3	11110101	1	Загрузить в стек содержимое регистра флагов. (M_0) ← (A); (адрес (M_0) содержится в (SP)-1); (M_1) ← PSW; (адрес (M_1) содержится в (SP)-2)); (SP) ← ((SP)-2).
POP RP	10/3	11RP0001	1	Считать из стека содержимое пары регистров. (RL) ← (M_0); (адрес (M_0) содержится в (SP)); (RH) ←(M_1); (адрес (M_1) содержится в (SP)+1)); (SP) ← ((SP)+2).
POP PSW	10/3	11110001	1	Считать из стека содержимое регистра флагов. (PSW) ← (M_0); (адрес (M_0) содержится в (SP)); (A) ← (M_1); (адрес (M_1) содержится в (SP)+1)); (SP) ← ((SP)+2).
XTHL	18/5	11100011	1	Обмен содержимым верхушки стека и пары регистровHL. (L) ← (M_0); (адрес (M_0) содержится в (SP)); (H) ← (M_1); (адрес (M_1) содержится в (SP)+1));
SPHL	5/1	11111001	1	Пересылка содержимого регистров HL в указатель стека. (SP) ← (HL).
EI	4/1	11111011	1	Разрешение прерываний после выполнения следующей команды.
DI	4/1	11110011	1	Запрещение прерываний после выполнения следующей команды.
HLT	7/1	01110110	1	Останов. Процессор останавливается.
NOP	4/1	00000000	1	Нет операций. Не выполняется никаких операций.

Микропроцессор работает только с двоичными кодами. Среди совокупности этих кодов имеется определенная группа кодов, каждый из которых может «заставить» микропроцессор выполнить определенные действия (операции). Такой код часто называется кодом команды и, как правило, определяет одну команду из системы команд микропроцессора. Код любой команды представляется в запоминающем устройстве двоичным восьмиразрядным числом (байтом). Всего с помощью байта можно формировать $2^8 = 256$ различных кодовых комбинаций. Почти столько же команд (244) имеет микропроцессор КР580ВМ80. Естественно, что запомнить 244 кода довольно трудно, и поэтому каждому коду ставится в соответствие мнемоническое название (мнемоника) команды, которое является сокращением от английских слов, описывающих ее действие.

Мнемонический код команд позволяет легче запомнить их функции и значительно упрощает написание программ. Такой язык написания программ называется Ассемблером**.** После того, как программа написана на Ассемблере, ее необходимо снова перевести на язык, понятный микропроцессору, т. е. перевести в последовательность двоичных восьмиразрядных чисел. Перевод в последовательность двоичных цифр может происходить автоматически с помощью специальных программ-трансляторов (такие программы носят название «кросс-ассемблер» или «ассемблер») или вручную.

Для ручной трансляции можно использовать табл.1, в которой приведены все команды микропроцессора КР580ВМ80. С помощью этой таблицы можно легко и быстро сопоставить мнемонику команды с ее кодом. Код каждой команды приведен здесь в верхней горизонтальной строке (младшие разряды) и в крайнем левом столбце (старшие разряды) в шестнадцатеричной системе счисления. Например, команда STAX D имеет код 12H, команда JZ ADR — код CAH.

Таблица 1. Машинные коды команд микропроцессора К580ВМ80А

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	**NOP**				**MOV B,B**	**MOV D,B**	**MOV H,B**	**MOV M,B**	**ADD B**	**SUB B**	**ANA B**	**DRA B**	**RNZ**	**RNC**	**RPO**	**RP**	0
1	**LXI B**	**LXI D**	**LXI H**	**LXI SP**	**MOV B,C**	**MOV D,C**	**MOV H,C**	**MOV M,C**	**ADD C**	**SUB C**	**ANA C**	**DRA C**	**POP B**	**POP D**	**POP H**	**POP RSW**	1
2	**STAX B**	**STAX D**	**SHLD**	**STA**	**MOV B,D**	**MOV D,D**	**MOV H,D**	**MOV M,D**	**ADD D**	**SUB D**	**ANA D**	**DRA D**	**JNZ**	**JNC**	**JPO**	**JP**	2
3	**INX B**	**INX D**	**INX H**	**INX SP**	**MOV B,E**	**MOV D,E**	**MOV H,E**	**MOV M,E**	**ADD E**	**SUB E**	**ANA E**	**DRA E**	**JMP**	**OUT**	**XTHL**	**D1**	3
4	**INR B**	**INR D**	**INR H**	**INR M**	**MOV B,H**	**MOV D,H**	**MOV H,H**	**MOV M,H**	**ADD H**	**SUB H**	**ANA H**	**DRA H**	**CNZ**	**CNC**	**CPO**	**CP**	4
5	**DCR B**	**DCR D**	**DCR H**	**DCR M**	**MOV B,L**	**MOV D,L**	**MOV H,L**	**MOV M,L**	**ADD L**	**SUB L**	**ANA L**	**DRA L**	**PUSH B**	**PUSH D**	**PUSH H**	**PUSH PSW**	5
6	**MVI B**	**MVI D**	**MVI H**	**MVI M**	**MOV B,M**	**MOV D,M**	**MOV H,M**	**HLT**	**ADD M**	**SUB M**	**ANA M**	**DRA M**	**ADI**	**SUI**	**ANI**	**ORI**	6
7	**RLC**	**RAL**	**DAA**	**STC**	**MOV B,A**	**MOV D,A**	**MOV H,A**	**MOV M,A**	**ADD A**	**SUB A**	**ANA A**	**DRA A**	**RST 0**	**RST 16**	**RST 32**	**RST 48**	7
8					**MOV C,B**	**MOV E,B**	**MOV L,B**	**MOV A,B**	**ADC B**	**SBB B**	**XRA B**	**CMP B**	**RZ**	**RC**	**RPE**	**RM**	8
9	**DAD B**	**DAD D**	**DAD H**	**DAD SP**	**MOV C,C**	**MOV E,C**	**MOV L,C**	**MOV A,C**	**ADC C**	**SBB C**	**XRA C**	**CMP C**	**RET**		**PCHL**	**SPHL**	9
A	**LDAX B**	**LDAX D**	**LHLD**	**LDA**	**MOV C,D**	**MOV E,D**	**MOV L,D**	**MOV A,D**	**ADC D**	**SBB D**	**XRA D**	**CMP D**	**JZ**	**JC**	**JPE**	**JM**	A
B	**DCX B**	**DCX D**	**DCX H**	**DCX SP**	**MOV C,E**	**MOV E,E**	**MOV L,E**	**MOV A,E**	**ADC E**	**SBB E**	**XRA E**	**CMP E**		**IN**	**XCHG**	**EI**	B
C	**INR C**	**INR E**	**INR L**	**INR A**	**MOV C,H**	**MOV E,H**	**MOV L,H**	**MOV A,H**	**ADC H**	**SBB H**	**XRA H**	**CMP H**	**CZ**	**CC**	**CPE**	**CM**	C
D	**DCR C**	**DCR E**	**DCR L**	**DCR A**	**MOV C,L**	**MOV E,L**	**MOV L,L**	**MOV A,L**	**ADC L**	**SBB L**	**XRA L**	**CMP L**	**CALL**				D
E	**MVI C**	**MVI E**	**MVI L**	**MVI A**	**MOV C,M**	**MOV E,M**	**MOV L,M**	**MOV A,M**	**ADC M**	**SBB M**	**XRA M**	**CMP M**	**ACI**	**SBI**	**XRI**	**CPI**	E
F	**RRC**	**RAR**	**CMA**	**CMC**	**MOV C,A**	**MOV E,A**	**MOV L,A**	**MOV A,A**	**ADC A**	**SBB A**	**XRA A**	**CMP A**	**RST 8**	**RST 24**	**RST 40**	**RST 56**	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

Приложение 2

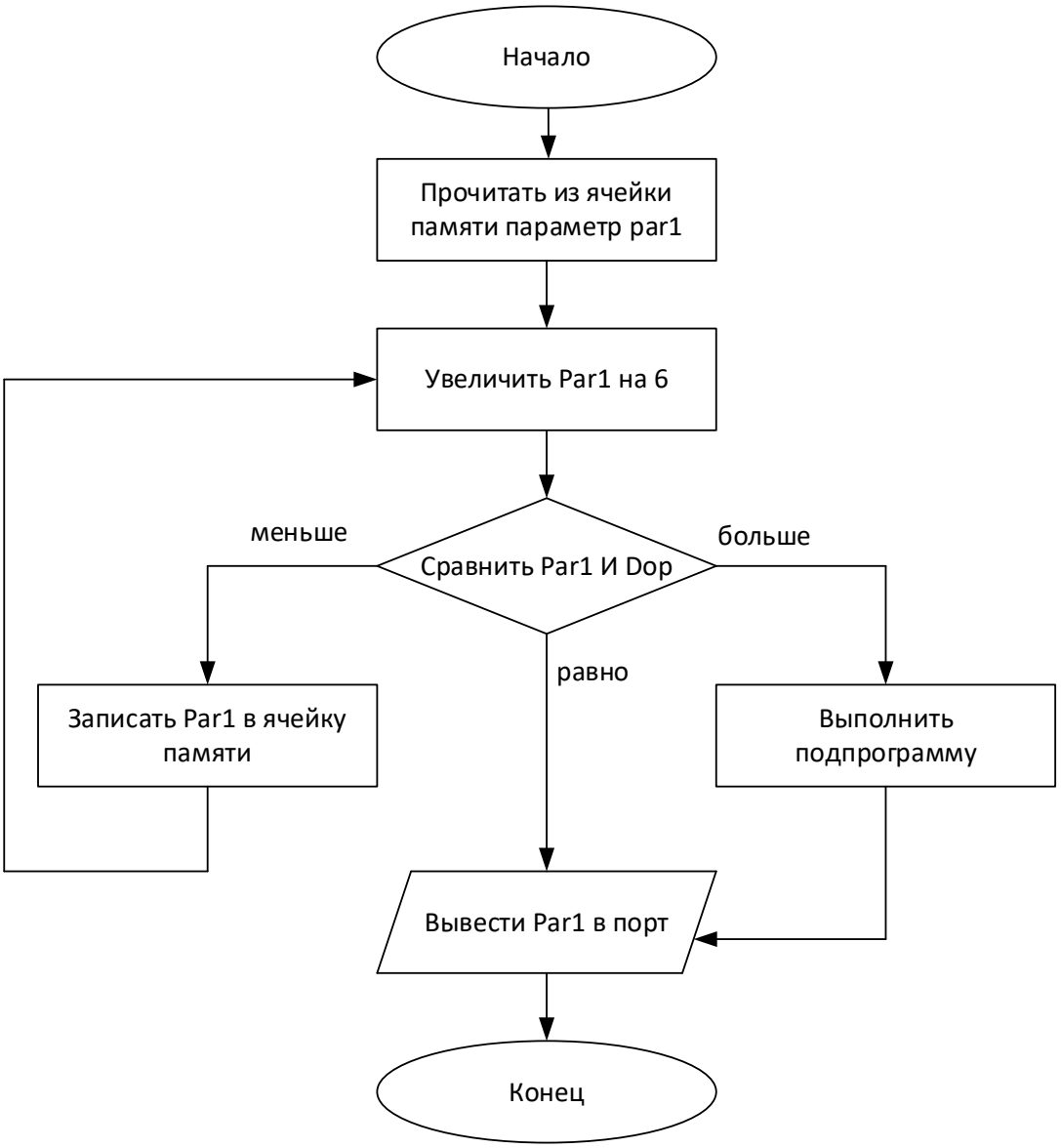
Варианты теоретических вопросов

0. Структура микропроцессорного устройства
1. Принципы построения и виды памяти микропроцессора
2. Представление данных в микропроцессорах
3. Микропроцессорный комплект серии К580
4. Система команд микропроцессора К580
5. Виды адресации и примеры команд для микропроцессора К580
6. Система прерываний микропроцессора
7. Классификация микропроцессоров. Пример построения микропроцессорного комплекта
8. Структурная схема микропроцессора К580
9. Управляющие сигналы и временные диаграммы работы микропроцессорных устройств

Приложение 3

Заданный алгоритм работы МУУ

Вариант 0



Подпрограмма

Уменьшить par1 на 1 и сравнить с dop, если par1 окажется больше dop, то уменьшать par1 до равенства параметру dop. Выйти из подпрограммы

Вариант 1

Подпрограмма.

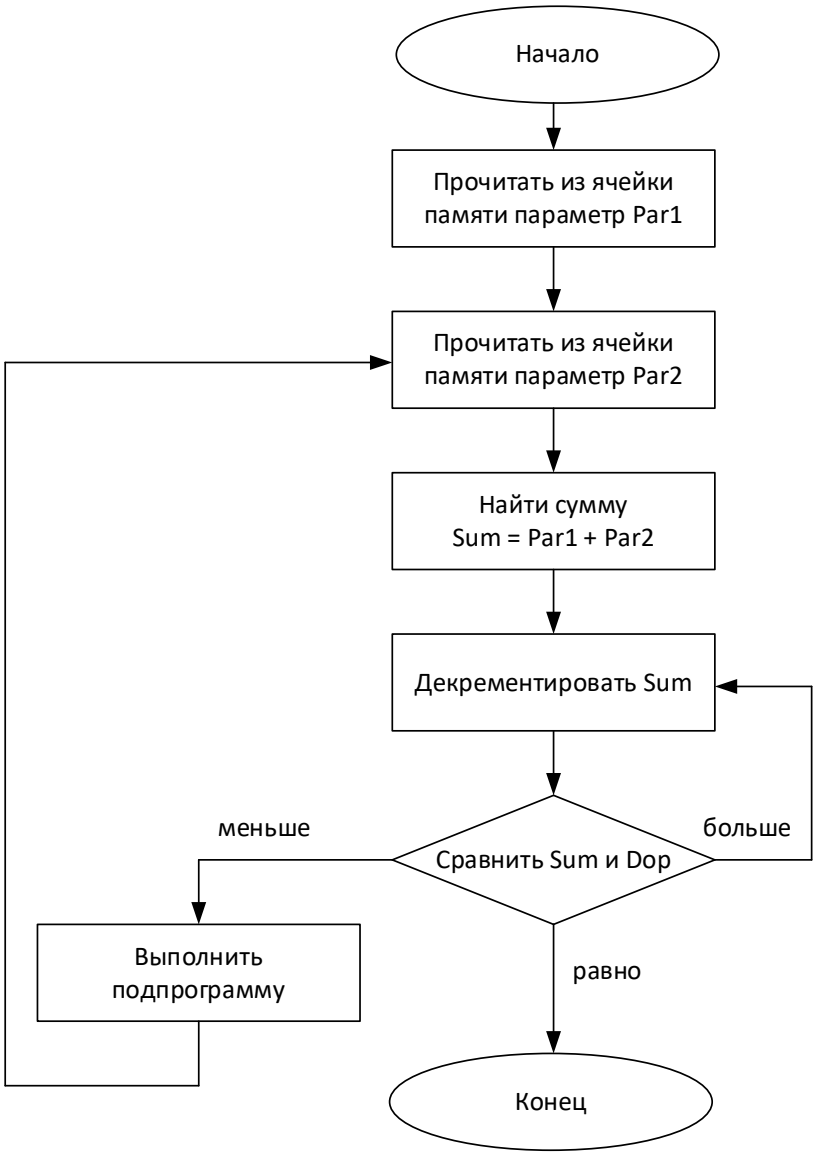
Записать последовательно в регистры B, C, D, E значение par2, уменьшая его каждый раз на 2. Выйти из подпрограммы

Вариант 2

Подпрограмма.

Декрементировать dop пока при сравнении с par1, dop не окажется равен par1. Выйти из подпрограммы

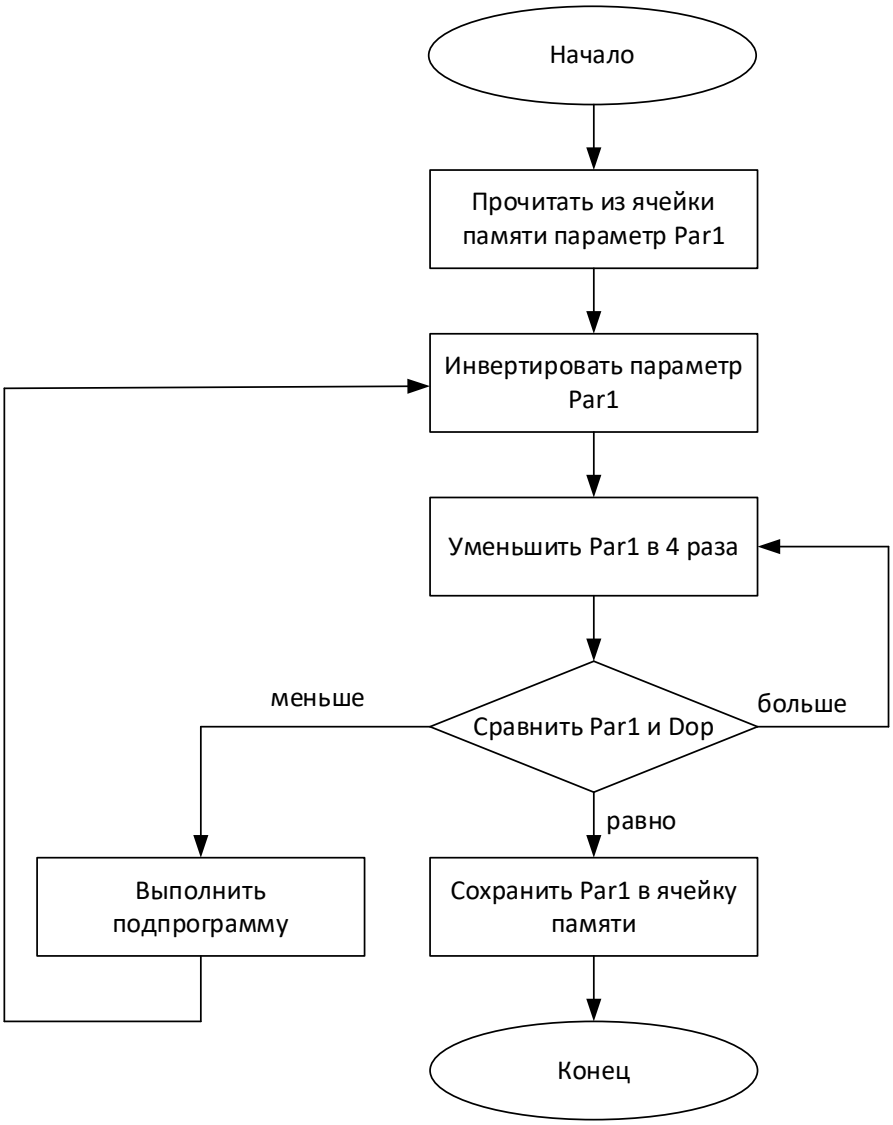
Вариант 3



Подпрограмма.

Увеличить par1 в 4 раза и инкрементировать его, затем записать результат в ячейку памяти, где хранится par1. Выйти из подпрограммы

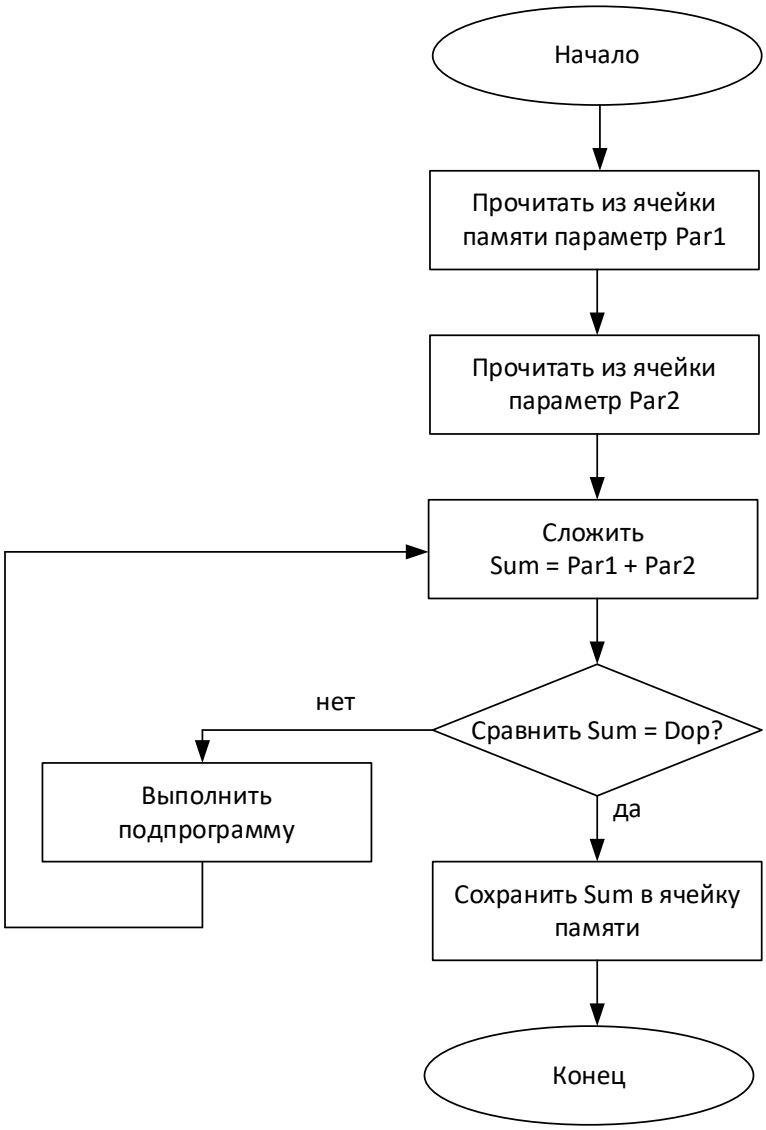
Вариант 4



Подпрограмма.

Прочитать par2 и увеличить его на 8, затем умножить логически на par1 и результат вывести в порт. Выйти из подпрограммы

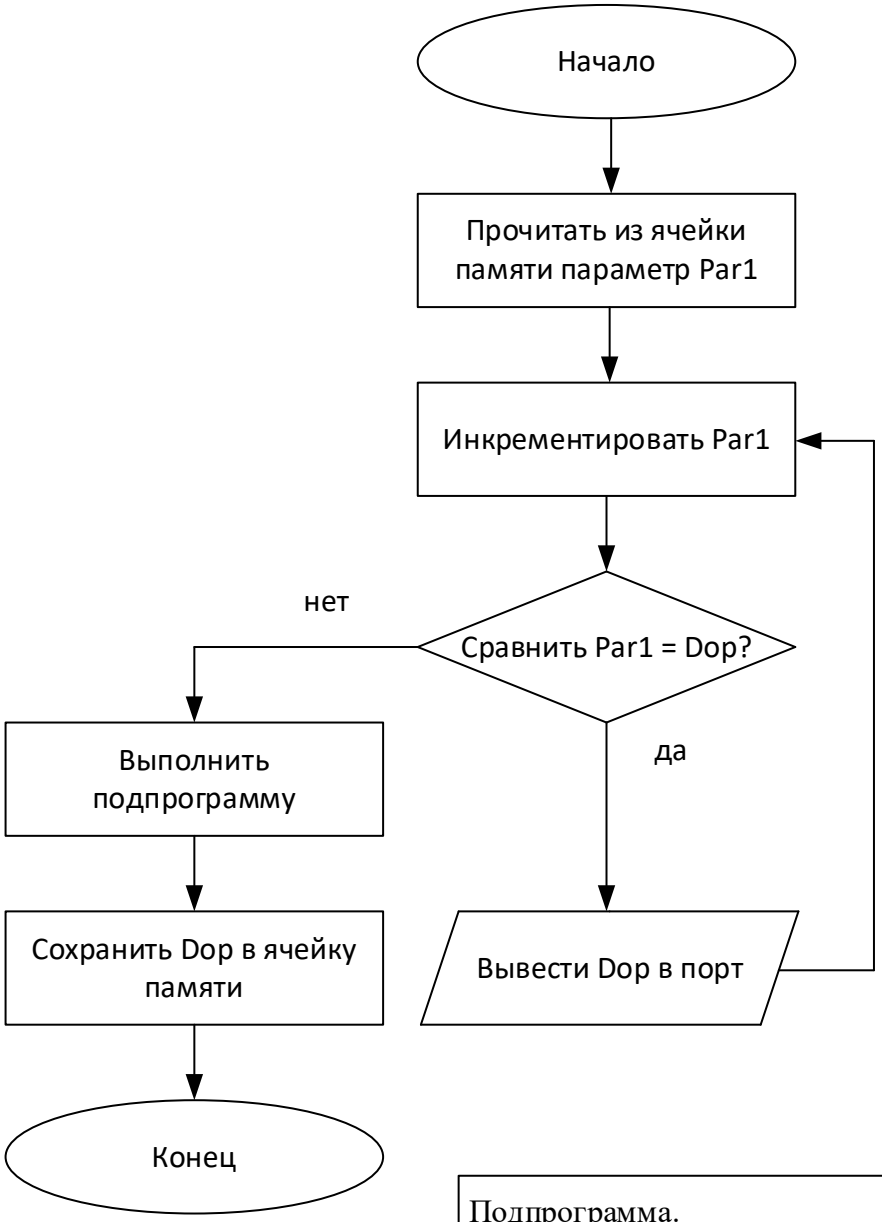
Вариант 5



Подпрограмма.

Увеличить par1 в 2 раза. Найти разность параметров par1 и par2, результат сохранить в ячейку памяти. Если разность par1 и par2 равна 0, то увеличить par1 на 4. Выйти из подпрограммы

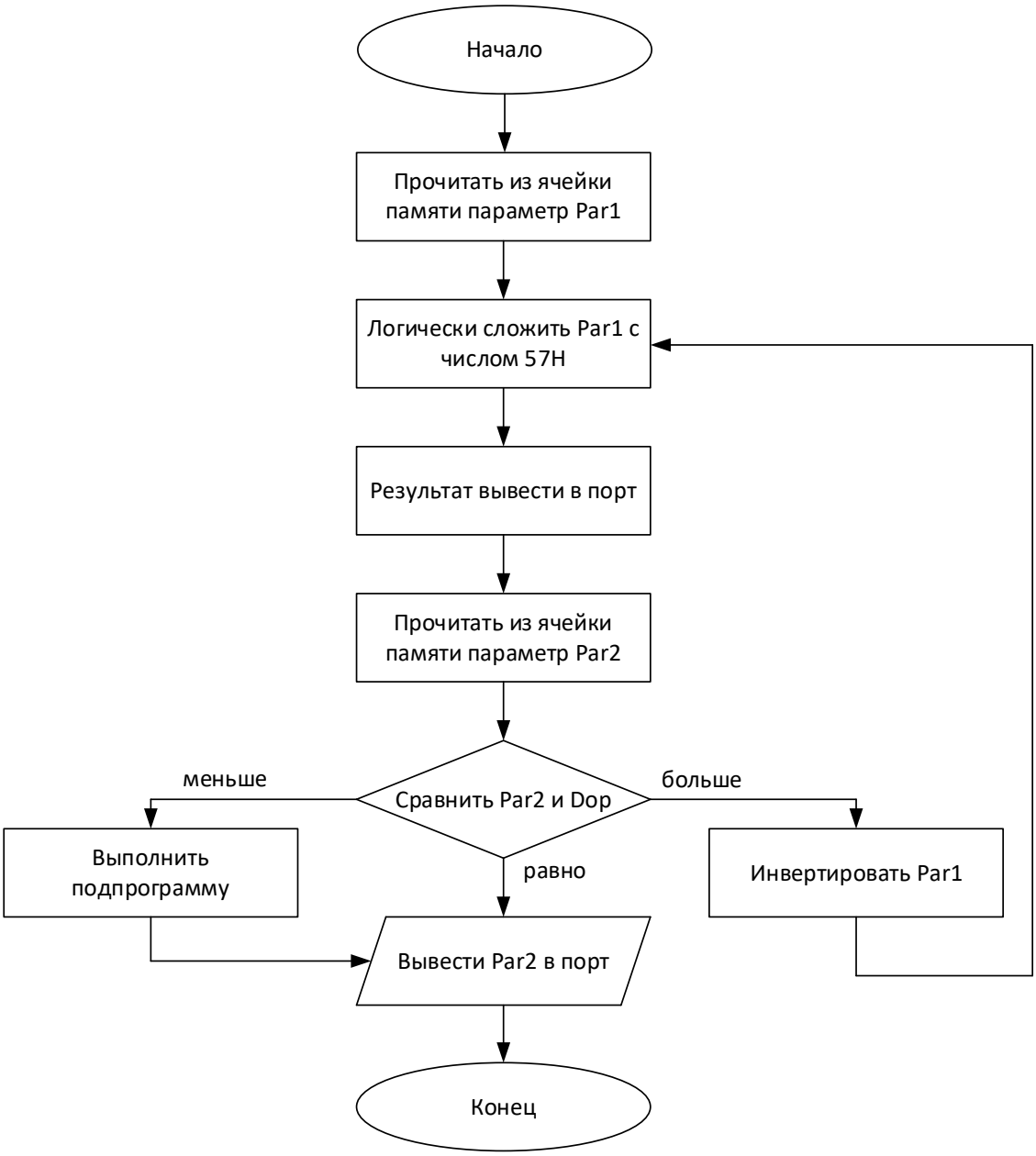
Вариант 6



Подпрограмма.

Сохранить par1 в ячейку памяти, а также сравнить par1 и dop. Если par1 окажется меньше, то увеличить dop в 2 раза, иначе уменьшить dop на 2. Выйти из подпрограммы

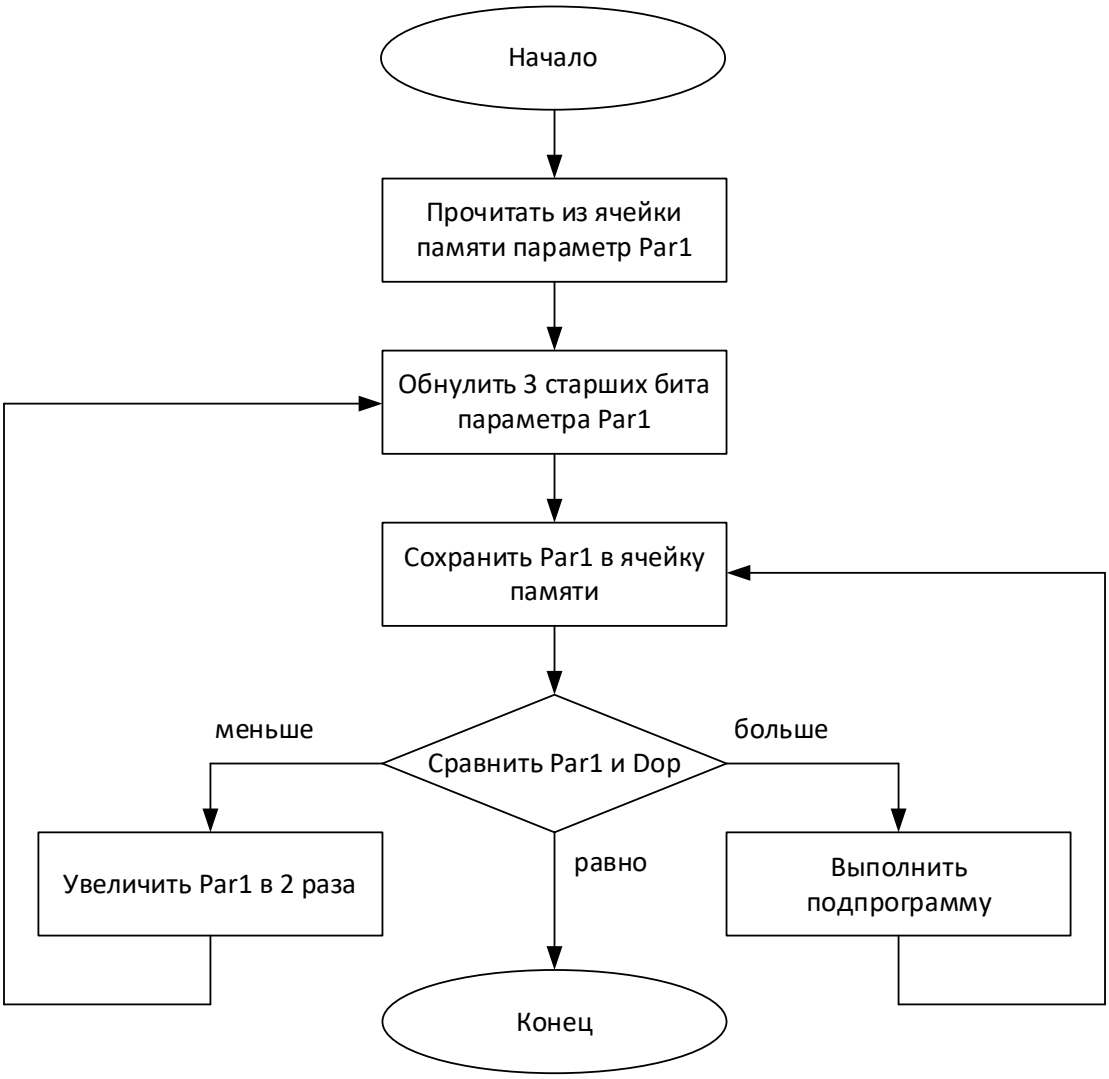
Вариант 7



Подпрограмма.

Увеличить par2 на 2 и сравнить с dop. Если par2 не равен dop, то увеличивать par2 на 2 до изменения результата сравнения. Выйти из подпрограммы

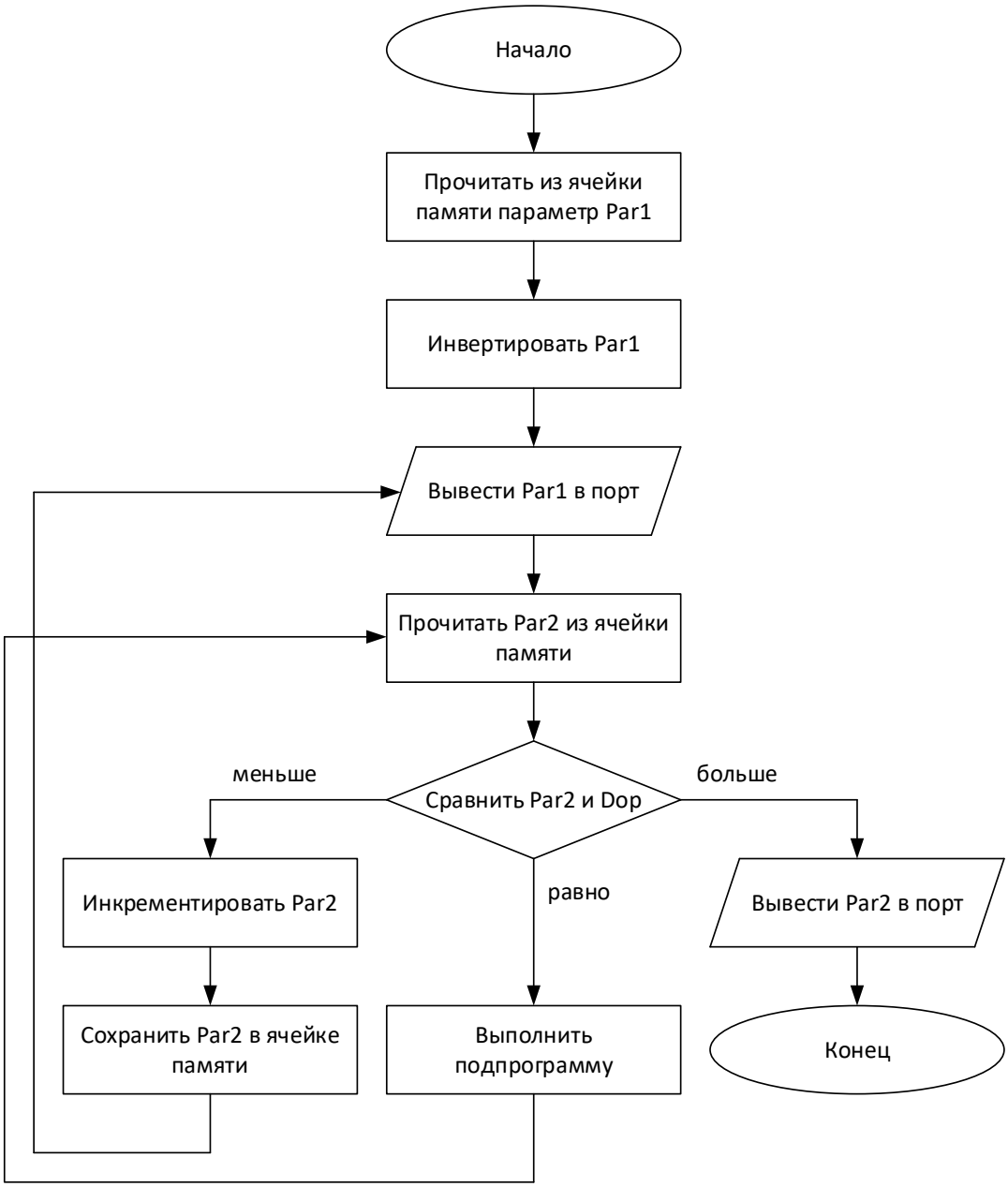
Вариант 8



Подпрограмма.

Прочитать par2 из ячейки памяти и сложить с par1. Если сумма окажется меньше 0, то увеличить ее на 2, иначе инвертировать сумму и выйти из подпрограммы

Вариант 9



Подпрограмма.

Записать в регистр C число 19H. увеличить par2 на содержимое регистра C, результат сохранить в ячейку памяти. Выйти из подпрограммы

Приложение 4

Выбор исходных данных

Нумерация портов согласно выбранному варианту задания

Предпоследняя цифра номера зачетной книжки									
0	1	2	3	4	5	6	7	8	9
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9
Присваиваемый номер порта ввода/вывода									