МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический университет им. А.Н. Туполева-КАИ»
(КНИТУ-КАИ)
Чистопольский филиал «Восток»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

по дисциплине

ЭЛЕКТРОНИКА И МИКРОПРОЦЕССОРНАЯ ТЕХНИКА

Семестр 6

Индекс по учебному плану: **Б1.В.09**
Направление подготовки: **12.03.01 Приборостроение**
Квалификация: **Бакалавр**
Профиль подготовки: **Приборостроение**
Типы задач профессиональной деятельности: **проектно-конструкторский, производственно-технологический**

Рекомендовано УМК ЧФ КНИТУ-КАИ

Чистополь
2023 г.

Введение

Данные методические указания предназначены для проведения лабораторных работ по дисциплине «Электроника и микропроцессорная техника»

Цикл лабораторных работ включает задания различного уровня.

Лабораторные работы №1, №2, №3, №4 ориентированы на изучение структуры команд микропроцессора МП 580 (аналог МП I8080).

№ п/п	№ раздела	Наименование лабораторных работ	Трудоемкость (час.)
1	2	Технология подготовки и выполнения программ на Ассемблере	8
2	2	Программирование простейших арифметических выражений с целочисленными переменными»	8
3	2	Ввод-вывод и преобразование числовых данных	8
4	2	«Программирование разветвлений»	8

ЛАБОРАТОРНАЯ РАБОТА № 1 «Технология подготовки и выполнения программ на Ассемблере»

Цель работы: изучение последовательности работы при подготовке программ на макроассемблере, их отладке и выполнении в системе программирования TEDASM; изучение функций и последовательности работы с отладчиком программ AFD.

Основные сведения

Система программирования TEDASM предназначена для написания и выполнения программ на языке Ассемблера. Основными функциями системы являются:

подготовка исходных файлов на макроассемблере. При создании нового исходного файла, его имя создается по формату name.asm;

трансляция исходного файла с получением загрузочного модуля (name.com);

выполнение программы.

Вызов системы выполняется из директории TEDASM, расположенной на диске «C:» запуском на выполнение загрузочного модуля tedasm.com. После загрузки системы на экране появляется главное меню, содержащее перечень команд системы и приглашение (C).

Основными командами системы являются:

Assem Sourse – ассемблировать исходный файл;

Edit Sourse – редактировать исходный файл;

Get Sourse – считать исходный файл в буфер системы;

Write Sourse – записать исходный файл в текущую директорию;

Run Codefile – запустить программу в кодах на выполнение;

Hexdump File – вывести на экран программу в HEX- и в ASCII-кодах;

Kill File – удалить файл из текущей директории;

List File – вывести на экран листинг программы;

Symbol List – вывести на экран таблицу имен программы, их типов и значений;

Xrefer List – вывести на экран список количественных параметров программы;

Directory – вывести на экран содержание текущей директории;

Asm Options – установить опции ассемблирования;

New Drive-Directory – сменить устройство и директорию;

Value – перевести введенное числовое значение в десятичную, восьмеричную, двоичную, шестнадцатеричную системы счисления и ASCII-код;

Quit – выйти из системы.

Рекомендуется следующий порядок работы в системе TEDASM:

1. Смена диска.

После загрузки TEDASM на экране появляется приглашение (C). Как правило, файлы пользователя размещаются на гибком диске A или B, поэтому необходимо воспользоваться командой смены диска и директории:

(C) N

New drive - B <Enter>

New Directory - <Enter>

2. Загрузка или редактирование исходного файла.

Если на диске B уже есть исходный файл, то необходимо:

- загрузить исходный файл NAME.ASM:

(B) G

- вызвать редактор:

(B) E

- отредактировать файл;

- выйти из редактора клавишей ESC;

- записать скорректированный файл на диск B:

(B) W

Если же исходный файл необходимо создать, то выполняются следующие действия:

- вызов редактора:

(B) E

- ввод текста с клавиатуры;

- выход из редактора клавишей ESC;

- запись созданного файла на диск B:

(B) W

3. Установление опций ассемблирования.

Вызов процедуры установки опций осуществляется с помощью команды

(B) O

После вызова на экране появляется меню для задания нужного набора опций:

F1	- Screen	(OFF)
F2	- Printer	(OFF)
F3	- Symbols	(OFF)
F4	- Xrefer	(OFF)
F5	- Memory	(ON)
F6	- ErrWait	(ON)
F7	- OBJ File	(OFF)
F8	- COM File	(OFF)
F9	- LST File	(OFF)
F10	- Unefd.	(OFF)

В исходном состоянии подключены опции памяти размещения исполняемого модуля программы (Memory) и контроля ошибок при трансляции (ErrWait). Нажатием функциональных клавиш можно изменить значение опций, в частности, клавишами F7, F8, F9 можно настроить систему на формирование и запись на диск объектного файла (OBJ), исполняемого файла (COM) и листинга программы (LST).

4. Ассемблирование.

Процесс ассемблирования запускается командой:

(C) A

При наличии ошибок процесс ассемблирования приостанавливается на ошибочной строке (если опция ErrWait находится в состоянии ON). Нажатием клавиши ENTER можно продолжить трансляцию программы. В случае успешной трансляции в буферной памяти системы формируется исполняемый файл name.com (опция MEMORY в состоянии ON), который может быть запущен на выполнение в системе TEDASM. Если опция MEMORY в состоянии OFF, то исполняемый файл записывается на диск C и не может быть запущен на выполнение из среды TEDASM.

5. Выполнение.

Программа запускается на выполнение командой:

(C) R

6. Выход из системы TEDASM.

Для выхода из системы необходимо ввести команду:

(C) Q

Система AFD предназначена для отладки программ, написанных на макроассемблере ASM-86. Для запуска системы необходимо войти в директорию TEDASM и, выбрав курсором исполняемый файл afd.com, нажать клавишу ENTER. После загрузки AFD на экране появится кадр, разделенный на несколько полей, имеющих свое функциональное назначение. Структура кадра представлена ниже:

<table>
<tr><td colspan="3">(1) Содержимое регистров микропроцессора и вершины стека</td></tr>
<tr><td>(2) Командная строка</td><td rowspan="2">(3) Первое окно для индикации памяти (HEX-форма)</td></tr>
<tr><td>(4) Окно дизассемблера</td></tr>
<tr><td>(5) Второе окно для индикации памяти (HEX-форма)</td><td>(6) ASCII-коды второго окна</td></tr>
</table>

Поле 1 – предназначено для индикации текущих состояний регистров микропроцессора и четырех верхних слов стека.

Поле 2 – служит для ввода команд загрузки отлаживаемой программы, просмотра содержимого памяти, запуска программы на выполнение, выхода из системы и т. д.

Поле 3 – предназначено для индикации памяти данных (адресация через сегментный регистр DS) в шестнадцатеричной форме.

Поле 4 – предназначено для просмотра памяти программ (адресация через сегментный регистр CS). Рассматриваемое поле содержит фрагмент программы из восьми команд в символьном виде и в кодах. Просмотр может осуществляться одним их двух способов:

- просмотр без выполнения программы. В этом режиме пользователь может просматривать текст и коды программы, без ее выполнения. Перемещение по тексту программы осуществляется при помощи клавиш перемещения курсора;

- просмотр с выполнением программы. В этом режиме осуществляется запуск программы и ее пошаговое выполнение. Покомандное выполнение программы осуществляется с помощью функциональной клавиши F1, а в том случае, когда необходимо выполнить программу до конца текущей подпрограммы, используется функциональная клавиша F2.

Поле 5 – служит для индикации памяти данных (адресация осуществляется через сегментный регистр DS) в шестнадцатеричной форме.

Поле 6 – представляет данные, индицируемые в поле 5 в ASCII кодах.

Поле 7 – задает значение используемых в системе программирования функциональных клавиш:

F1 (Step) – выполнение отлаживаемой программы по командам;

F2 (StepProc) – выполнение программы до конца очередной подпрограммы;

F4 (Help) – на экран монитора последовательно выдаются четыре страницы информации в помощь пользователю;

F7 (up) – перемещение курсора вверх;

F8 (dn) – перемещение курсора вниз;

F9 (le) – перемещение курсора влево;

F10 (ri) – перемещение курсора вправо.

Клавишами F7 – F10 можно переместить курсор в любое поле кадра и скорректировать содержимое любого регистра микропроцессора или ячейки памяти данных. После выполения корректировки необходимо вернуть курсор в начало командной строки для предотвращения случайного изменения содержимого регистров или ячеек памяти.

В том случае, когда курсор находится в командной строке, пользователь может ввести и выполнить одну из нижеперечисленных команд.

Загрузка отлаживаемой программы:

L <имя файла>, <адрес 0100> <Enter>

Просмотр текста программы в окне дизассемблирования:

D <начальный адрес фрагмента программы> <Enter>

Просмотр содержимого памяти данных:

M <номер окна> <начальный адрес> <Enter>

или

M <номер окна> <регистр, содержащий нач. адрес> <Enter>

Запуск программы на выполнение:

G <адрес старта> [, <адрес пром. останова>] <Enter>

Необходимо отметить, что при использовании команды G, программа выполняется в автоматическом режиме или до конца программы, или до точки промежуточного останова, заданного пользователем. В том случае, когда требуется покомандное выполнение программы, необходимо воспользоваться функциональными клавишами F1 и F2. В процессе выполнения программы, в командной строке выдается сообщение:

*** E X E C U T I N G ***

Прервать выполнение программы можно клавишами Ctrl+Esc.

Вывод на печать листинга программы и данных производится при помощи следующих команд.

Получение листинга:

PD <нач. адрес>,< длина в HEX-коде > [,<специф. файла>] <Enter>

Вывод на печать данных:

PH <нач. адрес>,< длина в HEX-коде > [,<специф. файла>] <Enter>

Выход из отладчика в DOS осуществляется следующей командой:

QUIT <Enter>

При работе с отладчиком AFD рекомендуется следующий порядок действий:

1) загрузить отлаживаемый файл, например:

L name.com, 0100 <Enter>

2) просмотром программы в окне дизассемблера найти адреса, соответствующие начальным байтам буферов и точкам промежуточных остановов, например:

buffer1 -----> 11c5,

buffer2 -----> 11f3,

останов1 -----> 1267,

останов2 -----> 12ad.

3) установить начальные адреса окон для индикации данных, например:

M 1 11c5 <Enter>

M 2 11f3 <Enter>

4) запустить программу на выполнение в автоматическом режиме до первого останова, например:

G 0100,1267 <Enter>

5) выполнить участок программы с ошибкой в покомандном режиме с помощью клавиш F1 или F2 с целью установления несоответствия ожидаемой и реальной информации в регистрах микропроцессора и в ячейках памяти. Определить ошибку в программе и выполнить корректировку регистров и буферов вручную.

6) запустить программу на выполнение с текущего состояния программного счетчика IP до следующей точки останова, например:

G ,12 ad <Enter>

7) продолжить выполнение в покомандном режиме аналогично п. 5.

8) выполнить оставшийся участок программы до конца, например:

G <Enter>

9) выйти из отладчика и с помощью любого редактора исправить ошибки в исходном файле программы.

Задание

1. Запустить на выполнение систему программирования.

2. Ввести с помощью редактора программу, установить опции ассемблирования, произвести трансляцию и выполнение программы.

3. Вторично оттранслировать программу с формированием исполняемого файла (name.com) на диске C.

4. Вызвать отладчик.

5. Загрузить отлаживаемую программу и выполнить ее в покомандном режиме и с промежуточными остановами. Для индикации памяти данных использовать соответствующие окна отладчика.

ЛАБОРАТОРНАЯ РАБОТА № 2 «Программирование простейших арифметических выражений с целочисленными переменными»

Цель работы: изучение команд пересылки данных, сложения и вычитания микропроцессора М580.

Основные сведения

Система команд микропроцессора М580 содержит 91 мнемокод и позволяет совершать операции над байтами, словами (2 байта), отдельными битами и цепочками байтов и слов. По функциональному признаку систему команд ВМ86 можно разбить на пять групп: команды пересылки данных, арифметические операции, логические операции и сдвиги, команды передачи управления и команды управления микропроцессором. В настоящей лабораторной работе рассматриваются первые две группы команд, мнемокоды которых приведены в табл. 1, 2.

Команды пересылки данных составляют четыре подгруппы: общие, стековые, ввода-вывода и пересылки цепочек. Все эти команды, за исключением POPF и SAHF, не влияют на флаги.

Команда MOV пересылает данные из регистра в регистр, из регистра в память и наоборот, но не обеспечивает пересылку типа память-память, что несколько ограничивает возможности микропроцессора ВМ86.

Команда XCHG осуществляет обмен данными между источником и приемнмком. Существует два формата этой команды. Общий формат позволяет обмениваться содержимым любой пары общих регистров, а также обмен между общим регистром и ячейкой памяти. Укороченный формат позволяет обмениваться данными между аккумулятором и общим регистром.

Команда XLAT предназначена для быстрого преобразования кодов и заменяет содержимое младшего байта аккумулятора AL на байт из 256-байтовой таблицы, базовый адрес которой содержится в регистре BX. При выполнении этой команды к содержимому BX прибавляется содержимое AL, а полученный таким образом результат используется как смещение относительно начала текущего

сегмента данных. Байт, расположенный по полученному адресу, помещается в AL, заменяя в нем прежнее содержимое.Таблица 1

Команды пересылки

Мнемо-ника	Опе-ранды	Размер	Описание команды
MOV	dst,src	b,w	src → dst
MOV	dst,data	b,w	data → dst
MOV	dst,sreg	w	sreg → dst
MOV	dst,reg	b,w	reg → dst
XCHG	dst,reg	b,w	dst ↔ reg
PUSH	src	w	src → стек
PUSH	sreg	w	src → стек
PUSHF		w	FLAGS → стек
POP	dst	w	стек → dst
POP	sreg	w	стек → sreg
POPF		w	стек → FLAGS
LAHF		b	FLAGS_L → AH
SAHF		b	AH → FLAGS_L
XLAT		b	MEM[BX+AL] → AL
LEA		w	Загрузка в reg эффективного адреса src
LDS		d	Загрузка в регистры DS и reg двойного слова
LES		d	Загрузка в регистры ES и reg двойного слова

Условные обозначения:

b – байт (8 бит);

w – слово (16 бит);

d – двойное слово (32 бита);

reg – регистр общего назначения или указ. регистр;

sreg – сегментный регистр;

mem – ячейка памяти;

src – источник операнда.

Таблица 2

Арифметические команды

Условные обозначения:

b – байт (8 бит);

w – слово (16 бит);

d – двойное слово (32 бита);

reg – регистр общего назначения или указ. регистр;

sreg – сегментный регистр;

mem – ячейка памяти;

src – источник операнда.

В отличие от остальных команд пересылок команды LEA, LDS и LES пересылают в регистр-приемник не сами данные, а их адрес. Основное назначение этих команд – инициализация регистров перед выполнением цепочечных команд или перед вызовом подпрограмм.

Стековые команды PUSH и POP предназначены для обмена данными со стеком, а команды PUSHF и POPF – для обмена со стеком регистра флагов микропроцессора.

Арифметические команды BM86 также весьма разнообразны и могут выполняться над целыми числами четырех типов: беззнаковыми и знаковыми двоичными, упакованными и неупакованными десятичными. Двоичные числа могут быть как байтовыми, так и двухбайтовыми. В состав группы арифметических команд входят команды десятичной коррекции, необходимые при работе с десятичными числами.

Полезными могут оказаться команды преобразования формата CBW и CWD. Первая из них расширяет байт из AL до двухбайтового числа и помещает его в AX. Вторая команда преобразует двухбайтовое слово в двойное слово (че-

тырехбайтовое) и размещает его в регистрах AX и DX. Расширение в обоих случаях происходит путем распространения знакового разряда источника на старшие байты. Подобные преобразования могут потребоваться для обеспечения корректной работы команды целочисленного деления IDIV.

Задание

Используя приведенные в прил. 2 подпрограммы PSN и PNS для перевода чисел из символьной формы в десятичную и обратно, написать и отладить программу вычисления заданного арифметического выражения. Для ввода и вывода значений переменных использовать буферы в виде символьных строк. Входные переменные полагать двухзначными целыми. Ввод переменных осуществляется оператором с клавиатуры, вывод результата производить на экран монитора.

Варианты заданий

1. (I + J - K) - (N - M)
2. (I + N) - (J - M) + K
3. (I – J - K) - (J + M)
4. (I + K) - (N - J) + (I - M)
5. (M – N - J) - (J + N)
6. (M + N + I) - (K + J - M)
7. (I + K) - (J + K) - (N + M)
8. (I + K - M + N) - (M + J)
9. (J - M + N) - (N - M + I)
10. (I + K) + (I + K) - N - M
11. (I + N - M - K) - (M - N)
12. (I + N - M) - (J - K)
13. (I + N) - (M + K)
14. (J + I) + (N + M) + (I - J)

ЛАБОРАТОРНАЯ РАБОТА №3 «Ввод-вывод и преобразование числовых данных»

Цель работы: изучение средств ввода и вывода данных в символьной форме, а также приемов преобразования числовых данных из символьной формы в двоичную и обратно.

Основные сведения

1. Средства MS-DOS для ввода и вывода символьных данных

Ввод и вывод данных при работе на ПЭВМ выполняется в символьной форме под управлением операционной системы MS-DOS. В составе операционной системы имеются специальные процедуры ввода и вывода, обращение к которым из программы пользователя выполняется с помощью команды программного прерывания INT 21H.

Выбор процедуры ввода/вывода задается с помощью номера функции прерывания, код которого перед выполнением команды прерывания заносится в регистр AH микропроцессора. При необходимости в регистр DX заносится параметр для процедуры ввода/вывода.

1.1. Ввод одного символа с клавиатуры (функция 01H)

```
MOV AH,01H      ; выбор функции
INT   21H       ; обращение к MS-DOS
```

После нажатия на клавишу вводимого символа его ASCII-код заносится в регистр AL.

1.2. Вывод одного символа на дисплей (функция 02H)

```
MOV       DL,<код выводимого символа>
MOV       AH,02H
INT   21H
```

1.3. Ввод символьной строки (функция 0AH)

Для ввода символьной строки пользователь должен организовать в своей программе буфер, содержащий n+3 байта памяти, где n - количество вводимых символов. Структура буфера содержит следующие составные части:

- дескриптор, расположенный в первых двух байтах буфера и предназначенный для записи константы n+1 и фактического количества введенных символов;

- n байтов для размещения вводимых символов;

байт, предназначенный для занесения признака конца ввода (0DH) по нажатию клавиши ВВОД.

Ввод константы в первый байт дескриптора выполняется программой пользователя, ввод значения и фактического количества введенных символов выполняется операционной системой в процессе обработки процедуры ввода символьной строки.

Фрагмент программы для ввода символьной строки имеет вид:

```
. . .
BUFIN       DB  <n+3>DUP(0)
. . .
LEA  DX,BUFIN ; адрес буфера для DOS и пользователя
MOV BX,DX
MOV AL,<n+1>   ; запись <n+1> в
MOV [BX],AL    ; 1-й байт дескриптора
MOV AH,0AH
INT   21H
. . .
```

Ввод символьной строки может быть досрочно прекращен нажатием клавиши ВВОД. При попытке ввода количества байтов, превышающего заданное, ПЭВМ подает звуковой сигнал. Функция 0AH позволяет в процессе ввода редактировать вводимую строку.

1.4. Ввод символьной строки (функция 09H)

Под символьной строкой понимается последовательность, заключенная в апострофы:

'< последовательность символов> $ '

Как правило, символьная строка должна заканчиваться денежным знаком "$", который является признаком конца вывода для операционной системы.

Ниже преведен пример вывода символьной строки:

```
. . .
STR         DB  13,10,'Ввести исходные данные $'
. . .
            LEA  DX,STR ;адрес выводимой строки
            MOV AH,09H
            INT   21H
. . .
```

2. Преобразование данных

Для преобразования двухразрядных десятичных чисел со знаком в диапазоне -99 - +99 может использоваться подпрограмма PSN, обращение к которой оформляется следующим образом:

LEA SI,<адрес десятичной символьной формы>

LEA DI,<адрес двоичной численной формы>

CALL PSN

Для обратного преобразования двоичного числа длиной в один байт в двухразрядное десятичное число из того же диапазона предназначена подпрограмма PNS, обращение к которой оформляется так:

LEA SI,<адрес десятичной символьной формы>

LEA DI,<адрес двоичной численной формы>

CALL PNS

Двоичные числа в подпрограммах представляются в дополнительном коде.

Задание

Ввести исходные данные с клавиатуры в символьном виде, произвести вычисления согласно варианту задания и вывести результат на экран.

Варианты заданий

(I*J + K*N - I3)/M

(I + J)*K - I5/16

(I - J)/K - (N + M)*L

((N + M)*I - J)/(2*L)

(N5 + N6)*2 + (K + L)*6/(N + M)

((I + J)*5 + (N - M)*3)/K

((I - J)*K + (I + J)*5)/(N - M)

(K*(I + J - N)/(I1 - I2))/4

I5*I4/(I + J)/(I5 + I4)

(N*M - I*J)/(I5 - I4)

((I1*I2/I3) - (I1 + I2))/L

((J*N + K)/(L + I) - I

(J5*J6*L1 - I)/(J + 4)

(N - M)/(N + K) - I1/I5

(I8 + I6)*2 - I4/I1 + J

ЛАБОРАТОРНАЯ РАБОТА №4 «Программирование разветвлений»

Цель работы: изучение операторов передачи управления и приемов программирования арифметических выражений, содержащих разветвления.

Основные сведения

Команды передачи управления условно подразделяются на две группы:

команды безусловных переходов;

команды условных переходов

В табл. 3 перечислены команды передачи управления обоих типов. Команды безусловных переходов позволяют передавать управление как внутри текущего сегмента, так и между сегментами. Общий вид команды безусловного перехода:

JMP label

В результате выполнения этой команды в счетчик команд IP засылается адрес, соответствующий метке label. При переходе внутри сегмента изменяется только регистр IP, если же осуществляется межсегментный переход, то изменяются два регистра CS и IP.

Команды условных переходов позволяют передавать управление только внутри текущего сегмента. Общий вид команды условного перехода:

Jcc label,

где cc – мнемонический код условия перехода.

В результате выполнения команды условного перехода в счетчик команд заносится либо адрес, соответствующий метке label, если условие выполняется, либо адрес следующей команды при ложном значении условия.

В процессе ассемблирования программы команда передачи управления преобразуется к виду:

COP diff,

где COP – байт кода операции, a diff – смещение, т. е. восьми- или шестнадцати-разрядный код разницы между адресом перехода и содержимым счетчика команд IP.

Приведенная схема соответствует относительной адресации, которая используется только в командах передачи управления. При использовании относительной адресации адрес перехода вычисляется путем суммирования текущего состояния IP и смещения diff, заданного в команде. Если смещение diff отрицательно (в дополнительном коде), то переход выполняется в сторону меньших адресов. Если же смещение diff положительно, то переход выполняется в сторону больших адресов.

Команды условных переходов позволяют передавать управление в пределах текущего сегмента со смещением: -128 … +127.

В отличие от команд условного перехода микропроцессора К580ВМ80 аналогичные команды микропроцессора К1810ВМ86 позволяют анализировать одновременно состояние нескольких флагов условий.

Таблица 3

Команды передачи управления

Мнемоника	Операнд	Описание команды	Условие
а. Безусловные переходы			
JMP	Label	Адрес для label → IP	
JMP	Src	Reg → IP или Mem(reg) → IP	
б. Условные переходы			
JC	Lable	Есть перенос	CF=1
JNC	Lable	Нет переноса	CF=0

JZ/JE	Lable	Равно нулю	ZF=1
JNZ/JNE	Lable	Не равно нулю	ZF=0
JP	Lable	Есть паритет	PF=1
JPO	Lable	Нет паритета	PF=0
JO	Lable	Переполнение	OF=1
JNO	Lable	Нет переполнения	OF=0
JS	Lable	Отрицательное	SF=1
JNS	Lable	Положительное	SF=0
JCXZ	Lable	Регистр CX=0	
JA	Lable	Выше	(CF & ZF)=0
JAE	Lable	Выше/равно	CF=0
JB	Lable	Ниже	CF=1
JBE	Lable	Ниже/равно	(CF & ZF)=1
JG	Lable	Больше	(CF+OF)&ZF=1
JGE	Lable	Больше/равно	SF+OF=0
JL	Lable	Меньше	SF+OF=1
JLE	Lable	Меньше/равно	(CF+OF)&ZF=0

Задание

Составить программу арифметических и логических действий над целыми переменными и константами. Для ввода исходных данных и вывода результата использовать буферы в виде символьных строк, в которых каждое значение переменной размещается в трех байтах:

<знак числа> <старшая цифра> <младшая цифра>

Перевод чисел из символьной формы в числовую и обратно выполнить с помощью подпрограмм PSN и PNS соответственно.

Варианты заданий

1. $$M = \begin{cases} (K+1)*4, & N < 2 \\ N - 3*K + I/2, & 2 \le N < 4 \\ 2*N+1, & N \ge 4 \end{cases}$$

2. $$M = \begin{cases} 4*I - 7, & I > 3 \\ I*I + 4*I - 7, & I < 1 \\ (I*I*I)/(I*I+2), & 1 \le I \le 3 \end{cases}$$

3. $$M = \begin{cases} I + 4*J, & I = J \\ 2*I - (J + J/2), & I > J \\ I*4 + J, & I < J \end{cases}$$

4. $$M = \begin{cases} K*4, & N < 7 \\ (N - K)/2, & N > 20 \\ N+1, & 7 \le N \le 20 \end{cases}$$

5. $$M = \begin{cases} (J+I)*3, & I < -2 \\ I*2 + J*J/4, & I > 1 \\ 2*N+1, & -2 \le I \le 1 \end{cases}$$

6. $$M = \begin{cases} I*K + K/2, & I \ge 2 \\ 2*I*I + K, & I < 0 \\ 3*I*I*I - K, & 0 \le I \le 2 \end{cases}$$

7. $$M = \begin{cases} 2*I*I + I, & I*I < 5 \\ 5*I*I - 1, & 5 \le I*I \le 16 \\ 3*I/5 - 2, & I*I \ge 16 \end{cases}$$

8. $$M = \begin{cases} J + I*I/J, & I > J \\ J*(I+1), & I + J \\ 2*J - I, & I < J \end{cases}$$

9. $$M = \begin{cases} 2*J*J + 3, & J \ge 5 \\ 7*J + 8, & 2 \le J < 5 \\ -2*J*J + 2, & J < 2 \end{cases}$$

10. $$M = \begin{cases} 3*I + J/2, & I \ge 7 \\ 4*I - 6, & 1 \le I < 7 \\ 2*I*I*J, & I < 1 \end{cases}$$