Московский государственный
университет им. М.В. Ломоносова
Факультет вычислительной математики и кибернетики

Пильщиков В.Н.

# Упражнения по языку ассемблера MASM

(Учебное пособие)

1997

Пособие содержит упражнения по языку ассемблера MASM и дополняет книгу автора «Программирование на языке ассемблера IBM PC» (М.: ДИАЛОГ-МИФИ, 1994). Пособие может быть использовано на семинарских занятиях по курсу «Архитектура ЭВМ и язык ассемблера». Порядок следования тем в пособии согласован с изложением материала в указанной книге.

Рецензенты:
доц. Баула В.Г.
доц. Корухова Л.С.

**Пильщиков В.Н.**
«Упражнения по языку ассемблера MASM (Учебное пособие)» — М.: Изд-во факультета ВМиК МГУ (лицензия ЛР №040777 от 23.07.1996), 1997. — 40 с.

Печатается по решению Ученого Совета факультета вычислительной математики и кибернетики МГУ им. М.В. Ломоносова.

**ISBN 5-89407-011-2**

СОДЕРЖАНИЕ

# 1. ОПИСАНИЕ ДАННЫХ. ПЕРЕСЫЛКИ.

1.1. Для каждой из указанных директив выписать эквивалентную ей директиву, где начальное значение переменной записано в 16-ричном виде.

| | | | |
|---|---|---|---|
| а) `A DB 17` | б) `B DB 15` | в) `V DB 255` | г) `G DB 150` |
| д) `D DB –17` | е) `E DB –1` | ж) `J DB –106` | з) `Z DB –128` |
| и) `I DW 17` | к) `K DW –17` | л) `L DW –1` | м) `M DW 256` |

1.2. Для каждой из указанных директив выписать две эквивалентные ей директивы, в первой из которых начальное значение переменной записано в виде десятичного числа со знаком, а во второй — без знака.

| | | | |
|---|---|---|---|
| а) `A DB 0Ah` | б) `B DB 0A5h` | в) `V DB 7Fh` | г) `G DB 80h` |
| д) `D DB 101b` | е) `E DW 0FFFEh` | ж) `J DW 7Fh` | з) `Z DW 80h` |

1.3. Дано описание:

```
A DW 1020h
B DD 10203040h
```

Указать в 16-ричном виде значения байтов с адресами: *A* и *A*+1; *B*, *B*+1, *B*+2 и *B*+3.

1.4. Описать переменную *D* размером в двойное слово с начальным значением $2^{16}$.

1.5. Записать более простым способом директиву

```
C DB '5'+1
```

1.6. Описать переменную-слово *X*, начальным значением которой является:

а) адрес этой же переменной;
б) адрес следующего слова памяти;
в) адрес предыдущего слова памяти.

1.7. Дано описание:

```
A DB 0,1,2
B DB 3,4,5,6
```

Указать значения байтов с адресами: *A*+1, *B*+2, *A*+4 и *B*−1.

1.8. Описать байтовый массив *PRIM* из 7 элементов, начальными значениями которых являются первые семь простых чисел (2, 3, 5 и т.д.).

1.9. Выписать все возможные варианты (кроме тех, где указываются коды букв) описания символьного массива *S*, начальными значениями элементов которого являются первые три большие буквы русского алфавита.

1.10. Описать массив *X* из 85 элементов-слов со следующими начальными значениями:

а) первые 40 элементов имеют значение 10, следующие 20 элементов — значение '*', остальные — без начального значения;
б) средний элемент имеет значение 1, а все остальные — значение 0.

1.11. Описать байтовую матрицу *M* размера 30×50, в каждой строке которой первые 47 элементов имеют значение −1, а последние три — значение 'q'.

1.12. Дано описание:

```
S DB 'ABCD'
W DW 10 DUP(?)
```

Указать значения выражений *TYPE S* и *TYPE W*.

1.13. Дано описание:

```
K EQU 90
X DB K+10 DUP((K+9)/3 DUP(?))
```

Сколько всего байтов занимает массив *X*?

1.14. Дано описание:

```
N EQU 30
```

Описать байтовую единичную матрицу *E* размера *N*×*N*.

1.15. Воспользоваться подходящей директивой *EQU* и предложить более короткий вариант записи директивы

```
S DB "abcdefhg=abcdef+k"
```

1.16. Указать начальные значения элементов массивов *X* и *Y*:

```
A=10
B=A
C EQU A
D EQU +A
X DB A, B, C, D
A=2*A
Y DB A, B, C, D
```

1.17. Дано описание:

```
N EQU 10
X DB ?
Y DW ?
```

Разделить следующие конструкции на три группы: 1) на константные выражения, 2) на адресные выражения и 3) на неправильные выражения.

а) `Y+Y-X` б) `Y+(Y-X)` в) `(X-Y)/2` г) `X-Y/2`

д) `Y-N` е) `N-Y` ж) `2*N+1` з) `2*X+1`

*Замечание. В упражнениях 1.18–1.22 требуется выписать команды, решающие указанные задачи.*

1.18. Дано описание:

```
A DB 3 DUP(?)
```

Рассматривая эту переменную как массив *A*[1..3]:

а) присвоить *A*[1] := 1, *A*[2] := 2, *A*[3] := 3;

б) циклически сдвинуть на 1 позицию влево элементы массива *A*.

1.19. Дано описание:

```
A DW ?,?
B DW ?,?
X DD ?
Y DD ?
```

Переменной *A* присвоить значение (два слова) переменной *B*, а переменной *X* — значение переменной *Y*.

1.20. Дано описание:

```
Q DD ?
```

Переменной *Q* присвоить значение 75535. (Вспомогательные переменные не использовать.)

1.21. Дано описание:

```
X DB ?
Y DB ?
```

Поменять местами значения переменных *X* и *Y*.

1.22. Дано описание:

```
Z DW ?
```

Поменять местами байты слова *Z*.

1.23. Дано описание:

```
A DW -73
B DW ?
```

Указать в 16-ричном виде значение переменной B после выполнения следующих команд:

```
MOV AX, A
MOV BYTE PTR B, AH
MOV BYTE PTR B+1, AL
```

1.24. Дано описание:

```
B DB ?
W DW ?
```

Среди перечисленных команд указать те, что записаны с ошибкой.

| | | | |
|---|---|---|---|
| а) `MOV BP,SP` | б) `MOV AX,BL` | в) `MOV DX,'*'` | г) `MOV CH,500` |
| д) `MOV BX,B` | е) `MOV SI,W+1` | ж) `MOV W,SI+1` | з) `MOV B,BYTE PTR CX` |
| и) `MOV B,80h` | к) `MOV B,W-B` | л) `MOV B,-130` | м) `MOV W,W+2` |
| н) `XCHG AH,AL` | о) `XCHG W,CX` | п) `XCHG B,B+1` | р) `XCHG W,15` |

# 2. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ.

2.1. Указать значения регистра *AL* (в виде десятичного числа без знака) и флагов *CF* и *ZF* после выполнения следующей пары команд:

| | | | |
|---|---|---|---|
| а) `MOV AL,100`<br>`ADD AL,100` | б) `MOV AL,100`<br>`ADD AL,156` | в) `MOV AL,100`<br>`SUB AL,90` | г) `MOV AL,100`<br>`SUB AL,190` |

2.2. Указать значения регистров *AH* и *AL* (в виде десятичных чисел без знака) и флагов *CF* и *ZF* после выполнения следующих команд:

| | | | |
|---|---|---|---|
| а) `MOV AH,0`<br>`MOV AL,160`<br>`ADD AL,60`<br>`ADC AH,3` | б) `MOV AH,0`<br>`MOV AL,160`<br>`ADD AL,160`<br>`ADC AH,3` | в) `MOV AH,255`<br>`MOV AL,255`<br>`ADD AL,1`<br>`ADC AH,0` | г) `MOV AH,20`<br>`MOV AL,10`<br>`SUB AL,16`<br>`SBB AH,0` |

2.3. Указать значения регистра *BH* (в виде десятичного числа со знаком) и флагов *OF* и *SF* после выполнения следующей пары команд:

| | | | |
|---|---|---|---|
| а) `MOV BH,80`<br>`ADD BH,40` | б) `MOV BH,80`<br>`ADD BH,50` | в) `MOV BH,-80`<br>`ADD BH,-40` | г) `MOV BH,-80`<br>`ADD BH,-50` |
| д) `MOV BH,80`<br>`ADD BH,-40` | е) `MOV BH,-80`<br>`ADD BH,40` | ж) `MOV BH,80`<br>`SUB BH,100` | з) `MOV BH,-80`<br>`SUB BH,50` |

2.4. Указать значения регистра *CL* (в виде как знакового, так и беззнакового десятичных чисел) и флагов *CF*, *OF*, *SF* и *ZF* после выполнения следующей пары команд:

| | | | |
|---|---|---|---|
| а) `MOV CL,128`<br>`ADD CL,128` | б) `MOV CL,-10`<br>`ADD CL,-40` | в) `MOV CL,246`<br>`ADD CL,216` | г) `MOV CL,40`<br>`SUB CL,100` |

2.5. Указать исходное значение регистра *AL* (любое из возможных), при котором после выполнения команды *ADD AL,2* флаги имели бы следующие значения:

а) $CF = 1$, $OF = 0$, $SF = 0$ б) $CF = 0$, $OF = 1$, $SF = 1$

2.6. Доказать следующие утверждения:

а) Если операнды команды *ADD* интерпретировать как знаковые числа, то флаг *OF* получает в этой команде значение 1 тогда и только тогда, когда эти числа имеют один и тот же знак, а у результата команды — иной знак.

б) Если операнды команды *SUB x, y* интерпретируются как знаковые числа и $x < y$, то после выполнения этой команды флаги *OF* и *SF* обязательно будут иметь разные значения ($OF \neq SF$).

в) После выполнения команды *ADD* флаги никогда не будут иметь следующие значения:
1) $CF = OF = SF = 1$; 2) $CF = 0$, $OF = 1$, $SF = 0$;

г) После выполнения команды *SUB* возможны следующие значения флагов: $CF = OF = SF = 1$.

2.7. Дано описание:

```
X DB ?
```

Требуется записать в регистр *CL* значение переменной *X*, увеличенное на 2. Определить, какой из следующих двух фрагментов правильно решает эту задачу:

а)
```
MOV CL,X
ADD CL,2
```

б)
```
MOV CL,X+2
```

2.8. Пусть в регистре *BH* находится код какой-то большой латинской буквы и требуется записать в этот регистр код одноименной малой латинской буквы. Определить, какой из следующих фрагментов правильно решает эту задачу:

а)
```
SUB BH,'A'
ADD BH,'a'
```

б)
```
ADD BH,'a'-'A'
```

в)
```
MOV BH,BH-'A'+'a'
```

2.9. Дано описание:

```
A EQU 7
X DB ?      ; 0≤X≤80
```

Реализовать наименьшим числом команд следующее присваивание:

а) $X := 3X$ б) $X := 2(A + X) + 6$

2.10. Указать (в 16-ричном виде) значение регистра *AX* после выполнения следующих команд:

а)
```
MOV AL,80h
MOV BL,2
MUL BL
```

б)
```
MOV AL,80h
MOV BL,2
IMUL BL
```

в)
```
MOV AL,7Fh
CBW
```

г)
```
MOV AL,80h
CBW
```

*Замечание. В упражнениях 2.11–2.20 требуется выписать команды, решающие указанную задачу.*

2.11. Дано описание:

```
A DB ?
B DW ?
```

Записать в *B* число, равное по величине числу из *A*, при условии, что:

а) *A* — число без знака;

б) *A* — число со знаком.

2.12. Дано описание:

```
N DB ?   ; число без знака
K DW ?
```

Вычислить: $K := N(N+1)/2$

2.13. Дано описание:

```
A DW ?   ; число со знаком
B DW ?
```

Вычислить: $B := (A \text{ div } 1000)(A \text{ mod } 1000)$

2.14. Пусть *A*, *B*, *C* и *X* — знаковые байтовые переменные, а *Y* — переменная типа *DWORD*. Вычислить: $Y := AX^2 + BX + C$

2.15. Дано описание:

```
N  DW ?   ; 1≤N≤365
WD DB ?
```

Записать в *WD* номер дня недели (1 — понедельник, ..., 7 — воскресенье), на который приходится *N*-ый день года, считая, что 1 января этого года — понедельник.

2.16. Пусть *H*, *M* и *S* — байтовые переменные, а *T* — переменная типа *DWORD*. Считая, что от начала суток прошло *H* часов ($0 \le H < 24$), *M* минут и *S* секунд ($0 \le M, S < 60$), определить, сколько всего секунд прошло от начала суток к этому моменту времени. Ответ записать в *T*.

2.17. Пусть *T* — переменная размером в двойное слово, а *H*, *M* и *S* — байтовые переменные. Считая, что прошло *T* секунд ($0 \le T < 86400$) от начала суток, определить, сколько полных часов (*H*), минут (*M*) и секунд (*S*) прошло к этому моменту времени.

2.18. Пусть *D1*, *D2* и *N* — байтовые переменные. Считая, что значения *D1* и *D2* — это символы-цифры (от «0» до «9»), записать в *N* число, десятичная запись которого составлена из этих цифр (*D1* — левая цифра).

2.19. Дано описание:

```
N DW ?   ; 100≤N≤999
```

Записать в *N* число, полученное выписыванием в обратном порядке десятичных цифр исходного числа из *N* (например, $125 \to 521$).

2.20. Дано описание:

```
X DD ?
Y DD ?
Z DD ?
W DD ?,?
```

Реализовать следующие операции над «длинными» числами:

а) $Z := X + Y$ (считать, что сумма укладывается в двойное слово);

б) $Z := X - Y$ (считать $X \ge Y$);

в) $W := X\,Y$ (считать *X* и *Y* беззнаковыми числами).

2.21. Дано описание:

```
A DB ?
B DW ?
```

Среди перечисленных команд указать те, что записаны с ошибкой:

а) `ADD BX,'*'` б) `SUB AL,400` в) `ADC BH,BL` г) `SBB AX,BL`

```
д) ADD BX,B     е) SUB CX,A     ж) ADC B,'B'    з) SUB A,BYTE PTR B
и) CBW AL       к) MUL AL       л) DIV 100      м) DEC A+1
```

# 3. ПЕРЕХОДЫ. ЦИКЛЫ. ВВОД-ВЫВОД.

3.1. Дано описание:

```
X DB 206   ; (–50)
```

Определить, будет ли сделан переход на метку MET при выполнении следующих команд:

```
а)  CMP X,210    б) CMP X,210    в) CMP X,-40   г) CMP X,–40   д) CMP X,216
    JA MET          JE LAB          JG MET         JL MET         JL MET
                    JB MET
```

3.2. Дано описание:

```
X DW ?  ; число со знаком
```

Определить, какой из следующих фрагментов правильно реализует оператор

```
if X>80 then X:=X-1 else X:=X+1
```

```
  а) CMP X,80      б) CMP X,80      в) CMP X,80      г) CMP X,80
     JLE M            JG M             JLE M            JNG M
     DEC X            DEC X            DEC X            DEC X
M:   INC X            JMP L            JMP L            JMP L
                  M:  INC X        M:  INC X        M:  INC X
                  L:               L:               L:
```

3.3. Упростить (записать меньшим числом команд) следующий фрагмент:

```
  а) CMP AH,0      б) CMP AH,0      в) L: ADD AX,AX   г) L:ADD AX,AX
     JE M             JL  M1             DEC BX            DEC CX
     JG M             JMP M              CMP BX,0          CMP CX,0
     NEG AH        M1:NEG AH             JE M              JNE L
M:                    M:                 JMP L
                                      M:
```

3.4. Пусть *X* и *Y* — знаковые байтовые переменные, а *L* — метка. Реализовать следующие условные переходы:

```
а) if (X>2) or  (Y<10) then goto L;
б) if (X>2) and (Y<10) then goto L;
```

3.5. Пусть *X*, *Y*, *Z* и *W* — знаковые переменные-слова. Реализовать следующее присваивание:

а) $Y := \mathrm{abs}(X)$ б) $Z := \min(X, Y)$ в) $W := \max(X + 10, 2Y, 8 - Z)$

3.6. В регистре *BL* находится число от 0 до 15. Записать в *BL* код соответствующей шестнадцатеричной цифры как символа (в качестве «буквенных» цифр использовать большие латинские буквы от 'A' до 'F').

3.7. Дано описание:

```
X DD ?
Y DD ?
```

Рассматривая значения *X* и *Y* как беззнаковые числа, реализовать условный переход на метку *LESS*, если $X < Y$.

3.8. Дано описание:

```
N DW ?   ; 0≤N≤999
```

Определить, содержит ли десятичная запись числа $N$ цифру 5. Ответ — 1 (содержит) или 0 — записать в регистр *DL*.

3.9. Пусть $Y$ — переменная-слово со значением от 1 до 2100. Определить, является ли год $Y$ високосным, и в регистр *CL* записать 1, если является, и 0 в противном случае. (По «новому стилю» високосными считаются года, кратные 4, однако из всех годов, кратных 100, високосными являются лишь кратные 400: 1700, 1800 и 1900 — невисокосный, 2000 — високосный).

3.10. Пусть $H$, $M$ и $S$ — байтовые переменные, причем $0 \le H < 24$, $0 \le M, S < 60$. Рассматривая $H$, $M$ и $S$ соответственно как число полных часов, минут и секунд, прошедших к некоторому моменту суток, присвоить этим переменным значения, соответствующие моменту, на 1 секунду большему. (Учесть смену суток.)

3.11. Пусть $A$, $B$ и $C$ — знаковые переменные-слова. Записать в регистр *DL* значение 3, 2, 1 или 0 в зависимости от того, может ли существовать треугольник с такими длинами сторон (и если да, то какого он вида — равносторонний (3), равнобедренный (2) или какой-то иной (1)) или не может (0).

3.12. Пусть $N$ — переменная-слово ($N \ge 1$), а $K$ — переменная-байт. Определить, является ли $N$ степенью числа 3 (1, 3, 9, 27, ...). Если является, то в $K$ записать показатель степени ($N = 3^K$), не является — записать −1.

3.13. Дано описание:

```
N DB ?    ; N≥1
F DW ?
```

Записать в $F$:

а) $N$-ое число Фибоначчи ($F_N$);

б) первое из чисел Фибоначчи, превосходящих 10000.

(Определение чисел Фибоначчи $F_k$: $F_0 = F_1 = 1$, $F_k = F_{k-1} + F_{k-2}$.)

3.14. Пусть $N$ и $K$ — переменные-слова и $2 < K < N$. Записать в регистр *BX* наибольший из остатков от деления $N$ на числа 2, 3, ..., $K$.

3.15. Дано описание:

```
N DW ?    ; N≥0
```

Записать в регистр *CL*:

а) наибольшую цифру из десятичной записи числа $N$;

б) количество значащих цифр из десятичной записи числа $N$;

в) сумму цифр из десятичной записи числа $N$.

3.16. Дано описание:

```
X DD ?    ; X≥0
```

Записать в регистр *CL* старшую (левую) значащую цифру из десятичной записи числа $X$.

3.17. Дано описание:

```
N DW ?  ; N>1
```

Определить, является ли $N$ простым числом. Ответ — 1 (да) или 0 — записать в регистр *BL*.

3.18. Дано описание:

```
N DW ?  ; N>1
```

Записать в регистр *BL* количество различных простых делителей числа *N*.

3.19. Дано описание:

```
Z DB ?
```

Записать в *Z* максимальное значение выражения $(X^2 + Y^2) \bmod 80$ в целочисленных точках квадрата $0 \le X \le 99$, $0 \le Y \le 99$.

3.20. Дано описание:

```
R DB ?   ; 0<R<150
```

Записать в регистр *DX* количество целочисленных точек на плоскости, попадающих в круг радиуса *R* с центром в начале координат.

**Замечание**. *В упражнениях 3.21–3.32 решение задачи описать в виде фрагмента программы с вводом-выводом, используя вспомогательные операции INCH, ININT, FLUSH, OUTCH, OUTINT, OUTWORD, OUTSTR и NEWLINE. Считать, что все вводимые числа «укладываются» в размер слова.*

3.21. Дана последовательность из 130 попарно различных знаковых чисел. Определить наибольшее из них и его порядковый номер в последовательности.

3.22. Дана последовательность символов (отличных от точки), за которой следует точка. Определить, сбалансирована ли эта последовательность по круглым скобкам. Ответ: ДА или НЕТ.

3.23. Напечатать таблицу умножения (в десятичной системе счисления).

3.24. Дан текст следующего вида:

$d_1 \pm d_2 \pm \ldots \pm d_k$.

где $d_i$ — цифра от 0 до 9, $k \ge 1$. Найти значение этой алгебраической суммы.

3.25. Используя только операцию *OUTCH*, вывести содержимое регистра *AX* в виде:

а) знакового десятичного числа,
б) беззнакового 16-ричного числа («буквенные» цифры — от A до F).

3.26. Используя только операцию *INCH*, ввести:

а) знаковое десятичное число,
б) беззнаковое 16-ричное число («буквенные» цифры — от A до F) и записать его в регистр *AX*.

Считать, что число записано без ошибок, оканчивается пробелом и «укладывается» в размер слова.

3.27. Дано 50 чисел, среди которых есть по крайней мере одно отрицательное. Найти наибольшее среди отрицательных чисел.

3.28. Дана последовательность из 40 чисел. Определить, у скольких чисел этой последовательности равные «соседи» (т.е. равны предыдущее и последующее числа).

3.29. Дана непустая последовательность символов (отличных от точки), за которой следует точка. Напечатать эту же последовательность:

а) заменив все 'PH' на 'F';
б) удалив все лишние пробелы (т.е. из нескольких подряд идущих пробелов оставить только один).

3.30. Дана непустая последовательность непустых слов из больших латинских букв; между соседними словами — запятая, за последним словом — точка. Подсчитать количество слов, которые:

а) начинаются с буквы 'S';
б) оканчиваются буквой 'Z';
в) начинаются и оканчиваются одной и той же буквой.

3.31. Дано описание:

```
P DW ?; P≥0
Q DW ?; Q>0
```

Напечатать дробь $P/Q$ в виде вещественного числа с 5 цифрами в дробной части.

3.32. Дано описание:

```
X DD ?
```

Напечатать значение $X$ в виде беззнакового десятичного числа. (Обратить особое внимание на возможность переполнения в команде *DIV*.)

# 4. ИНДЕКСИРОВАНИЕ. МАССИВЫ. СТРУКТУРЫ.

4.1. Дано описание:

```
X DW 500h, 600h, 700h
```

Пусть $BX = 8000h$ и $SI = 102h$. Указать (в форме $X + число$) исполнительный адрес следующей команды:

а) `DEC X+2` б) `DEC X[2]` в) `DEC [X+2]` г) `DEC [X]+2`
д) `INC X[SI]` е) `INC X[SI+2]` ж) `INC X[SI]-2` з) `INC X-2[SI]`
и) `NEG X[BX][SI]` к) `NEG X[BX+SI]` л) `NEG X[BX+SI+8000h]`

4.2. Пусть $X$ — имя некоторого байта памяти и $BX = i$ и пусть требуется обнулить байт памяти с адресом $X + 2i$. Определить, какой из следующих фрагментов правильно решает эту задачу:

а)
```
MOV X[BX+BX],0
```
б)
```
MOV DI,BX
MOV X[BX+DI],0
```
в)
```
ADD BX,BX
MOV X[BX],0
```

4.3. Пусть $X$ — имя некоторого байта памяти и $BX = i$ и пусть требуется обнулить байт памяти с адресом $X - i$. Определить, какой из следующих фрагментов правильно решает эту задачу:

а)
```
MOV X[-BX],0
```
б)
```
MOV [X-BX],0
```
в)
```
NEG BX
MOV X[BX],0
```

4.4. Дано описание:

```
X DW 0,1,2
```

Пусть $X$ обозначает ячейку памяти с адресом 100. Указать числовые значения регистров $AX$ и $BX$ после выполнения следующих команд:

а)
```
MOV AX,X
LEA BX,X
```
б)
```
MOV AX,X+2
LEA BX,X+2
```
в)
```
MOV SI,2
MOV AX,X[SI]
LEA BX,X[SI]
```
г)
```
LEA SI,X
MOV AX,2[SI]
LEA BX,2[SI]
```

4.5. С помощью одной команды *LEA* реализовать следующее действие:

а) увеличить значение регистра $BX$ на 10;
б) уменьшить значение регистра $BX$ на 10;
в) записать в регистр $AX$ значение регистра $DI$;
г) записать в регистр $SI$ значение регистра $BX$, увеличенное на 1.

4.6. Дано описание:

```
X DB 400 DUP(?)
Y DW 400 DUP(?)
```

Определить, какая величина должна находиться в регистре *BX*, чтобы выражение *X*[*BX*] определяло адрес 20-го от начала элемента массива *X*, если нумерация элементов этого массива начинается с:

а) 0 б) 1 в) 10 г) −10

Ответить на аналогичные вопросы для массива *Y* и выражения *Y*[*BX*].

4.7. Дано описание:

```
Z DB 61 DUP(?)    ; Z[0..60]
```

Пусть в регистре *BX* находится величина *i*, а в регистре *DI* — величина *j* ($10 \le i, j \le 30$). Указать адресные выражения, с помощью которых можно сослаться на элемент массива *Z* с индексом:

а) *i* б) *i* − 10 в) *j* + 10 г) *j* + *i* − 1 д) 42 е) 0

4.8. Дано описание:

```
X DB ?
Y DW ?
```

Указать, какие из следующих конструкций являются допустимыми формами записи операндов команд, а какие — нет и почему.

| | | | |
|---|---|---|---|
| а) `X[SI-1]` | б) `X[1-BX]` | в) `Y[DI]-2` | г) `2-Y[DI]` |
| д) `BX[AX]` | е) `AX[BX]` | ж) `SI+2` | з) `X[BH]` |
| и) `[X]` | к) `[5]` | л) `X[5]` | м) `5[X]` |
| н) `[Y-X]` | о) `Y-X[DI]` | п) `X-Y[BX]` | р) `X+Y[SI]` |
| с) `X[BX]+2[SI]` | т) `X[SI]+2[DI]` | у) `[DI]X` | ф) `Y[DI]` |

4.9. Дано описание:

```
X DB 50 DUP(?)  ; X[0..49]
```

Пусть в регистре *BX* находится адрес некоторого элемента массива *X* (т.е. *X* + *i*, где *i* — число от 0 до 49). Требуется в этом же регистре получить индекс (*i*) этого элемента. Определить, какая из следующих групп команд правильно решает эту задачу:

а)
```
SUB BX,X
```
б)
```
SUB BX,[X]
```
в)
```
LEA AX,X
SUB BX,AX
```
г)
```
LEA BX,-X[BX]
```

4.10. Описать подходящий массив и, используя команду *XLAT*, решить следующую задачу: в регистре *DH* находится число от 2 до 19; требуется записать 1 в регистр *AL*, если число *DH* простое, и записать 0 иначе.

4.11. Дано описание:

```
Q DB 256 DUP(?)    ; Q[0..255] of 0..255
```

Считая, что в регистре *AL* находится число *i* от 0 до 255, реализовать присваивание *Q*[*Q*[*i*]] := *i*.

4.12. Дано описание:

```
X DW 100 DUP(?)   ; числа со знаком
```

Решить следующую задачу:

а) записать в регистр *AX* число нулевых элементов массива *X*;

б) записать в регистр *AX* наименьший элемент массива *X*;

в) обнулить все отрицательные элементы массива *X*;

г) заменить на 1 первый нулевой элемент массива *X*;

д) изменить знак у первого элемента массива *X*, отличного от 1.

4.13. Дано описание:

```
N EQU 100
Y DB N DUP(?)   ; Y[0..N-1]
```

Считая, что все элементы массива *Y* неотрицательны и попарно различны, поменять местами наибольший и 20-ый элементы этого массива.

4.14. Дано описание:

```
Z DW 30 DUP(?)   ; Z[0..29], числа со знаком
```

Решить следующую задачу:

а) обнулить последний положительный элемент массива *Z*;
б) если во второй половине массива *Z* есть элементы, равные 1, то первый из них заменить на 45;
в) поменять знак у всех элементов массива *Z* с четными индексами.

4.15. Дано описание:

```
X DW 100 DUP(?)
```

Решить следующую задачу:

а) определить, у скольких элементов массива *X* равные соседи (предыдущий и последующий элементы), и записать ответ в регистр *AL*;
б) элементы массива *X* циклически сдвинуть на 2 позиции вправо;
в) если левая и правая половины массива *X* совпадают, то обнулить последний элемент этого массива.

4.16. Дано описание:

```
S DB 200 DUP(?)
T DB 200 DUP(?)
```

Рассматривая *S* и *T* как символьные массивы (строки), решить следующую задачу:

а) все цифры строки *S* записать в начало строки *T*, а остальные символы — в конец (в любом порядке);
б) проверить на равенство строки *S* и *T* при условии, что пробелы не учитываются, и записать ответ 1 (равны) или 0 в регистр *AL*;
в) перенести в конец строки *S* все её пробелы, сохранив взаимный порядок следования остальных символов;
г) определить, есть ли в строке *S* хотя бы два одинаковых символа, и записать ответ 1 (есть) или 0 в регистр *AL*.

4.17. Дано описание:

```
LW DB 150 DUP(12 DUP(?))
```

Рассматривая *LW* как массив из 150 слов по 12 символов в каждом, решить следующие задачи:

а) отсчитать количество симметричных слов в этом массиве и записать ответ в регистр *AL*;
б) определить, упорядочены ли слова этого массива по алфавиту (по неубыванию), и записать ответ 1 (да) или 0 в регистр *AL*;
в) определить, есть ли в массиве *LW* хотя бы два одинаковых слова, и записать ответ 1 (есть) или 0 в регистр *AL*.

4.18. Дано описание:

```
N DW ?            ; 0≤N≤9999
S DB 4 DUP(?)     ; символьная строка из цифр
```

Требуется:

а) записать в *S* десятичные цифры *N* (например: $N = 304 \rightarrow S =$ '0304');
б) решить обратную задачу.

4.19. Дано описание:

```
A DD 40 DUP(?)    ; числа без знака
```

Заменить начальный элемент массива *A* на максимальный элемент массива.

4.20. Дано описание:

```
X DB 100 DUP(?)    ; X[0..99]
Y DW 100 DUP(?)    ; Y[0..99]
```

Решить следующую задачу:

а) заполнить массивы *X* и *Y* по правилу: $X[i] := i$, $Y[i] := i$;

б) записать в регистр *DH* количество элементов массива *X*, для которых выполняется условие $X[i] > i$ (считать, *X*[*i*] числами без знака).

4.21. Для ввода задана непустая последовательность малых латинских букв, за которой следует точка. Используя подходящий вспомогательный массив, определить (ответ напечатать):

а) сколько различных букв входит в эту последовательность;

б) какая из букв чаще всего встречается в этой последовательности (если таких букв несколько, выбрать любую).

4.22. Дано описание:

```
N EQU 1000
X DB N DUP(?)
K DW ?        ; 0<K<N
```

Используя подходящий вспомогательный массив, циклически сдвинуть элементы массива *X* на *K* позиций влево.

4.23. Для ввода дана непустая последовательность символов (отличных от точки), за которой следует точка и в которой не более 1000 символов. Ввести эти символы и распечатать их в обратном порядке, удалив предварительно все большие гласные русские буквы (А, Е, И, О, У, Ы, Э, Ю, Я).

4.24. Пусть в регистрах *SI* и *DI* находятся начальные адреса двух (непересекающихся) областей памяти из 20 слов в каждой. Решить следующую задачу:

а) обнулить все слова первой из этих областей (*SI*);

б) записать в каждое слово первой области его адрес;

в) сравнить содержимое обеих областей и записать ответ 1 (равны) или 0 в регистр *AL*.

4.25. Дано описание:

```
N EQU 100
X DB N DUP(?)    ; числа со знаком
Y DB ?
```

Считая, что элементы массива *X* упорядочены по возрастанию, определить, есть ли в *X* элемент, равный *Y*, и записать ответ 1 (есть) или 0 а в регистр *AX*.

При решении задачи использовать метод бинарного поиска (деления пополам): сравнить *Y* со средним (или около него) элементом массива *X*; если они равны, то поиск закончить; если *Y* меньше (больше) среднего элемента, то далее рассматривать только левую (правую) половину массива, применив к ней этот же метод.

4.26. Дано описание:

```
N EQU 100
X DW N DUP(?)    ; числа со знаком
```

Упорядочить массив $X$ по неубыванию ($X_1 \le X_2 \le X_3 \le \ldots X_N$), используя следующий метод сортировки:

а) *сортировка выбором*: найти максимальный элемент массива и переставить его с последним элементом; затем этот же метод применить ко всем элементам, кроме последнего (он уже находится на своем окончательном месте); и т.д.

б) *сортировка обменом (метод пузырька)*: последовательно сравнивать пары соседних элементов массива ($X_1$ с $X_2$, $X_2$ с $X_3$ и т.д.) и, если первый элемент пары больше второго, то переставлять их; тем самым наибольший элемент окажется в конце массива; затем этот же метод применить ко всем элементам, кроме последнего; и т.д.

в) *сортировка вставками*: пусть первые $k$ элементов уже упорядочены по неубыванию; взять ($k$ + 1)-ый элемент и разместить его среди первых $k$ элементов так, чтобы упорядоченными оказались уже $k$ + 1 первый элемент; этот метод применять при $k$ от 1 до $N$ − 1.

4.27. Дано описание:

```
DATE  STRUC              ; дата
    Y DB 1997            ; год
    M DB ?               ; номер месяца
    D DB ?               ; число
   WD DB 'воскресенье'   ; день недели
DATE  ENDS
```

Описать переменные *D1*, *D2*, *D3* и *D4* типа *DATE* со следующими начальными значениями их полей (знак ? означает, что поле не должно иметь начального значения):

```
          Y      M     D          WD
D1:   1945       5    13       среда
D2:   1997      12     ?     четверг
D3:      ?       ?     ?           ?
D4:   1997       ?     ? воскресенье
```

Директивы, описывающие эти переменные, должны быть максимально короткими.

4.28. Дано описание:

```
TIME STRUC   ; время какого-то момента суток
      H     DB ?      ; час (от 0 до 23)
      M     DB ?      ; минута (от 0 до 59)
      S     DB ?      ; секунда (от 0 до 59)
TIME ENDS
   T TIME <>
  T1 TIME <>
```

Присвоить переменной *T1* время, на 1 секунду большее времени *T*. (Учесть смену суток.)

4.29. Описать структурный тип *PERSON* (человек) со следующими тремя полями: *FAM* (фамилия) — из 20 байтов, *NAME* (имя) — из 10 байтов и *AGE* (возраст) — 1 байт. Описать также (без начальных значений) переменную *P* типа *PERSON* и массив *GR* (группа людей) из 40 элементов того же типа. Считая, что этим переменным в программе уже присвоены какие-то значения, решить следующую задачу (ответ записать в регистр *AL*):

а) определить, является ли *P* человеком в возрасте 17 лет, вторая буква в фамилии которого — это «е» (ответ: 1 (является) или 0);

б) определить, сколько людей из *GR* имеют тот же возраст, что и *P*;

в) напечатать фамилию самого молодого человека из группы *GR* (любого, если таких несколько);

г) определить, сколько людей из *GR* имеют то же имя, что и *P*;
д) определить, есть ли в *GR* хотя бы одна пара однофамильцев (ответ: 1 (есть) или 0).

4.30. Имеются символьные переменные S и T:

```
S DB 256 DUP(?)    ; S[0..255]
T DB 80 DUP(?)     ; T[0..79]
```

подстроки которых будем представлять структурами типа SUBSTR:

```
SUBSTR STRUC
  ASTR DW ? ;начальный адрес строки, в которую входит подстрока
  INDX DB ? ;индекс элемента строки, с которого начинается
            ;подстрока
  LENG DB ? ;длина подстроки (число символов в ней)
SUBSTR ENDS
```

(Например, директива *X SUBSTR <T*, 60, 20> описывает подстроку из последних 20 символов строки *T*.)
Имеются две подстроки:

```
A SUBSTR <>
B SUBSTR <>
```

которые в процессе выполнения программы получили некоторые значения (считать, что эти значения корректно задают подстроки: подстрока не выходит за пределы строки и т.п.).
Решить следующую задачу:
а) если в подстроку *A* входит пробел, тогда сделать значением переменной *B* подстроку из 15 начальных символов строки *S*;
б) если подстроки *A* и *B* равны (состоят из равного числа попарно равных символов), тогда в регистр *AL* записать 1, а иначе — 0.

# 5. БИТОВЫЕ ОПЕРАЦИИ. УПАКОВАННЫЕ ДАННЫЕ.

5.1. Указать значения регистра *AL* и флага *ZF* после выполнения следующей пары команд:

```
а) MOV AL,1010b   б) MOV AL,0     в) MOV AL,1101b  г) MOV AL,100b
   NOT AL            NOT AL          AND AL,0111b     AND AL,011b
д) MOV AL,100b    е) MOV AL,0F0h  ж) MOV AL,0      з) MOV AL,101b
   TEST AL,011b      OR AL,0Fh       XOR AL,0FFh      XOR AL,AL
```

5.2. Дано описание:

```
A DB ?
B DB ?
X DW ?   ; число со знаком
```

Рассматривая *A* и *B* как логические переменные, принимающие лишь значения 0 («ложь») и 0FFh («истина»), реализовать следующее присваивание:
а) A := A and not *B* or not *A* and *B*
б) A := B or (*X* > 2) and *A*
в) A := A ≥ B (команды условного перехода не использовать)

5.3. Пусть под логические переменные *A* и *B* выделено по байту:

```
A DB ?
B DB ?
```

и пусть выбрано следующее представление для логических значений: «ложь» — нулевой байт, «истина» — любой ненулевой байт. Реализовать при этом представлении следующие операции:
а) *A* := not *A*  б) *A* := *A* and *B*  в) *A* := *A* or *B*

5.4. Предложить машинное представление (размером в байт) для логических значений, которое отлично от известного представления

*false* = 00h, *true* = 0FFh

и при котором команды *NOT*, *AND* и *OR* правильно реализуют логические операции отрицания, конъюнкции и дизъюнкции соответственно.

5.5. Указать значения регистра *AL* и флага *CF* после выполнения следующей группы команд:

а)
```
MOV AL,101b
SHL AL,1
```
б)
```
MOV AL,1100b
MOV CL,5
SHL AL,CL
```
в)
```
MOV AL,110b
SHR AL,1
```
г)
```
MOV AL,110b
MOV CL,3
SHR AL,CL
```
д)
```
MOV AL,0F0h
ROL AL,1
```
е)
```
MOV AL,101b
MOV CL,2
ROR AL,CL
```
ж)
```
MOV AX,1122h
SHR AH,1
RCL AL,1
```
з)
```
MOV AX,1122h
SHR AH,1
RCR AL,1
```

5.6. Пусть *X* и *Y* — беззнаковые переменные-слова. Не используя команды умножения и деления, реализовать следующее присваивание:

а) $Y := 4X - X \operatorname{div} 8 + X \bmod 16$

б) $Y := 35X$

5.7. Дано описание:

```
X DW ?    ; число без знака
```

Реализовать условный переход на метку *L*, если *X* — четное число, четырьмя способами — соответственно с помощью команд *DIV*, *AND*, *TEST* и *SHR*. Какой из этих способов лучше и почему?

5.8. Дано описание:

```
Y20 DW 30 DUP(?)
```

Рассматривая элементы массива *Y20* как порядковые номера некоторых годов 20 века (от 1901 до 2000), подсчитать количество високосных годов среди них и записать ответ в регистр *AL*.

5.9. Дано описание:

```
X DD ?    ; число со знаком
```

Не используя команды условного перехода, записать в регистр *AL* знаковый бит числа *X*, т.е. 1, если $X < 0$, и 0 иначе.

5.10. Реализовать следующую операцию:

а) сделать переход на метку *L*, если 4 средних бита регистра *AL* — это 1001b;
б) сделать переход на метку *L*, если 2 правых бита регистра *AL* — равны 2 правым битам регистра *BL*;
в) сделать переход на метку *L*, если равны 3 левых и 3 правых бита регистра *AX*;
г) записать в регистр *CL* байт, составленный из 4 левых битов регистра *AL* и 4 правых битов регистра *BL*;
д) в регистре *AX* заменить 5 левых битов на 5 правых.

5.11. Решить следующую задачу:

а) подсчитать число двоичных единиц в значении регистра *AX* и записать это число в регистр *DH*;
б) не используя команды деления, напечатать значение регистра *AX* в виде беззнакового двоичного числа без незначащих нулей;
в) перевернуть содержимое регистра *AX*;

г) не используя команды умножения, ввести беззнаковое двоичное число и записать его в регистр *AX* (считать, что число задано без ошибок, содержит от 1 до 16 цифр и заканчивается пробелом).

5.12. Пусть *A*, *B* — беззнаковые переменные-слова, а *C* — переменная типа *DWORD*. Без команды умножения реализовать операцию $C := 16A + B$.

5.13. Не используя команды умножения и деления и используя буквы A-F как «буквенные» 16-ричные цифры, решить следующую задачу:

а) вывести значение регистра *AX* в виде 4-значного шестнадцатеричного числа;
б) ввести шестнадцатеричное число и записать его в регистр *AX* (считать, что число записано без ошибок, содержит от 1 до 4 цифр, за числом следует пробел).

5.14. При так называемом двоично-десятичном представлении целых (неотрицательных) чисел каждая цифра в десятичной записи числа заменяется на 4-битовый двоичный код этой цифры ($0 \to 0000$, $1 \to 0001$, ..., $9 \to 1001$), причем соседние пары таких четверок упаковываются в один байт. Например, десятичное число 193 в этом представлении имеет следующий вид:

```
0000 0001   1001 0011
```

Решить следующую задачу:

а) в регистре *BX* хранится число от 0 до 9999, представленное в двоично-десятичном виде; перевести его в обычное двоичное представление и записать в регистр *AX*;
б) регистре *AX* хранится число от 0 до 9999 в обычном двоичном представлении; записать в регистр *BX* двоично-десятичное представление этого числа.

5.15. Дано описание:

```
T RECORD  A:1=1,  B:3=5,  C:2
X T < ,2>
```

Указать значение:

а) всего байта *X* и полей *A*, *B* и *C* в нем;
б) выражений *WIDTH X*, *WIDTH A*, *WIDTH B* и *WIDTH C*;
в) выражений *MASK X*, *MASK A*, *MASK B* и *MASK C*;
г) выражений *A*, *B* и *C*.

5.16. Дано описание:

```
TR RECORD A:3, B:2, C:3
 R TR <>
```

Реализовать следующую операцию (все поля — из переменной *R*):

а) полю *A* присвоить значение поля *C*;
б) поменять местами значения полей *A* и *C*;
в) перейти на метку *L*, если значения полей *A* и *B* равны.

5.17. Дано описание:

```
DATE1 STRUC ; дата в виде структуры
     D1     DB ?    ; день (от 1 до 31)
     M1     DB ?    ; номер месяца (от 1 до 12)
     Y1     DB ?    ; год (две последние цифры - от 0 до 99)
DATE1 ENDS
DATE2 RECORD  D2:5,  M2:4,  Y2:7   ; дата в виде записи
     DT1    DATE1 <>
     DT2    DATE2 <>
```

Решить следующую задачу:

а) переменной *DT1* присвоить дату, являющуюся значением *DT2* (распаковать дату);
б) переменной *DT2* присвоить дату, являющуюся значением *DT1* (упаковать дату);
в) проверить на равенство даты *DT1* и *DT2* и записать в регистр *L* значение 1, если даты равны, и значение 0 иначе.

5.18. Дано описание:

```
DATE  RECORD  D:5,  M:4,  Y:7 ; дата в формате
                             ; "день-месяц-год (две последние цифры)"
DATE1 RECORD Y1:7, M1:4, D1:5 ; дата в формате
                              ; "год-месяц-день"
      A     DATE <>
      A1    DATE1 <>
```

Решить следующую задачу:
а) вывести дату *A* в виде *dd.mm.yy* (например: 16.5.97);
б) присвоить переменной *A1* ту же дату, что записана в переменной *A*.

5.19. Имеются числовые константы *L* и *R* (*L* < *R*) и переменные

```
M DB (R-L)/8+1 DUP(?)
S DB (R-L)/8+1 DUP(?)
X DW ?    ; L≤X≤R
```

Рассматривая *M* и *S* как упакованные множества целых чисел из отрезка [*L*, *R*], решить следующую задачу:
а) из множества *M* удалить все элементы, входящие в множество *S*;
б) перейти на метку *SUBSET*, если множество *S* является подмножеством множества *M*;
в) в множество *M* добавить элемент *X*;
г) из множества *M* удалить элемент *X*;
д) определить, сколько элементов входит в множество *M*, и записать ответ в регистр *AX*;
е) напечатать все числа, входящие в множество *M*.

5.20. Дано описание на языке Паскаль:

```
var
  B : packed array[0..99] of boolean;
  i : 0.99;
```

Предложить для массива *B* упакованное машинное представление, описать этот массив и реализовать следующую операцию:
а) присвоить всем элементам массива *B* значение *true*;
б) поменять на противоположное значение всех элементов массива *B*;
в) *B*[28] := not *B*[28];
г) *B*[*i*] := *false*;
д) *B*[*i*] := *B*[*i*] and *B*[41] or *B*[80]

# 6. СЕГМЕНТИРОВАНИЕ. ПОЛНЫЕ ПРОГРАММЫ. СТЕК.

6.1. Пусть *DS* = 8208h, *ES* = 8400h, *BX* = 0FFF0h, *SI* = 12h, *offset X* = 28Ah. Вычислить (20-разрядный) абсолютный адрес второго операнда следующей команды:

а) `MOV AX,DS:210Ah`
б) `CMP BX,ES:X`
в) `ADD CX,DS:[BX]`
г) `OR DX,ES:X[BX+SI]`

6.2. Пусть *DS* = 1100h, *SS* = 0FF00h, *BX* = 8000h, *BP* = 8000h, *SI* = 9003h. Определить, какой сегментный регистр подразумевается по умолчанию в следующей команде, и вычислить абсолютный адрес ее первого операнда:

а) `SUB [SI],CL`
б) `MOV [BX+2],DX`
в) `MOV [BP+2],DX`
г) `AND [BP+SI-1],BH`

6.3. Дано описание:

```
X DW ?
```

Среди перечисленных команд указать ошибочные:

а) `MOV DS,1000h` б) `MOV DS,AX` в) `MOV CS,AX` г) `MOV AX,CS`
д) `MOV ES,DS` е) `MOV DS,X` ж) `MOV X,CS` з) `LEA ES,X`

6.4. Требуется записать в регистр *AL* содержимое байта памяти с абсолютным адресом 01234h. Определить, какие из следующих групп команд правильно решают эту задачу, а какие — нет и почему.

а)
```
MOV AL,1234h
```

б)
```
MOV BX,1234h
MOV AL,[BX]
```

в)
```
MOV DS,0
MOV AL,DS:1234h
```

г)
```
MOV AX,0
MOV ES,AX
MOV AL,ES:1234h
```

д)
```
MOV AX,1234h
MOV ES,AX
MOV AL,ES:0
```

е)
```
MOV AX,123h
MOV DS,AX
MOV AL,DS:4
```

6.5. Реализовать следующую операцию:

а) поменять местами значения байтов с абсолютными адресами 80000h и 80024h;
б) в слово памяти с абсолютным адресом 803FAh записать содержимое слова памяти с абсолютным адресом 90400h;
в) подсчитать количество нулевых байтов среди 20 байтов памяти, начинающихся с абсолютного адреса 12345h, и записать ответ в регистр *AL*.

6.6. Пусть в программе имеются следующие программные сегменты:

```
A     SEGMENT
      DW  100 DUP(?)
A     ENDS
B     SEGMENT
      DW  100 DUP(?)
B     ENDS
```

Реализовать перепись содержимого всего сегмента *B* в сегмент *A*.

6.7. Дано описание:

```
ABS DD  ?
PAIR DD ?
```

Значение переменной *ABS* трактуется как 20-разрядный абсолютный адрес некоторой ячейки памяти, а значение переменной *PAIR* — как адрес этой же ячейки, но заданный в виде адресной пары сегмент:смещение (*seg*:*ofs*), причем в силу «перевернутого» представления двойных слов в памяти часть *ofs* находится в первом слове этой переменной, а часть *seg* — во втором.

Не используя команды умножения и деления, решить следующие задачи:

а) преобразовать абсолютный адрес *ABS* в адресную пару *PAIR*;
б) преобразовать адресную пару *PAIR* в абсолютный адрес *ABS*;
в) нормализовать адресную пару *PAIR*, т.е. преобразовать ее к виду, где смещение *ofs* не превосходит 15 (0Fh); например, пару 4567h:1112h надо привести к виду 4678h:2h.

6.8. Пусть переменная-слово *D* описана в программном сегменте *D1*, переменная *E* — в сегменте *E1*, а переменная *S* — в сегменте *S1*. Пусть имеется директива (*C* — сегмент команд)

```
ASSUME  CS:C,  DS:D1,  ES:E1,  SS:S1
```

Для каждой из следующих команд определить, какой из префиксов (*DS:*, *ES:* или *SS:*) выберет ассемблер при трансляции ее второго операнда, и определить, сохранит ли ассемблер этот префикс при формировании соответствующей машинной команды.

а) `MOV AX,D`  б) `MOV AX,E`  в) `MOV AX,S`
г) `ADD AX,D[BX]`  д) `ADD AX,E[SI]`  е) `ADD AX,S[BX+DI]`
ж) `CMP AX,D[BP]`  з) `CMP AX,E[BP+SI+2]`  и) `CMP AX,S[BP+DI]`

6.9. Пусть переменная-слово *D* описана в сегменте *D1*, а переменная-слово *E* — в сегменте *S1* и пусть регистры *DS* и *ES* уже установлены на начало сегментов *D1* и *S1* соответственно. Реализовать присваивание $AX := D + E$ при условии, что имеется следующая директива *ASSUME* (*C* — имя сегмента команд):

а) `ASSUME  CS:C,  DS:D1,  ES:S1`
б) `ASSUME  CS:C,  DS:D1`
в) `ASSUME  CS:C,  ES:S1`
г) `ASSUME  CS:C`

6.10. Пусть в программе описаны следующие три сегмента:

```
A SEGMENT              B SEGMENT            C SEGMENT
  ...                    ...                  ...
X DW 1000 DUP(?)       Y DW 1000 DUP(?)     Z DW 1000 DUP(?)
  ...                    ...                  ...
A ENDS                 B ENDS               C ENDS
```

и пусть сегмент команд имеет имя *CODE*. Предполагая неопределенными значения регистров *DS* и *ES* и не используя регистр *SS*, реализовать (выписать подходящую директиву *ASSUME* и группу команд) следующее сложение массивов:
а) $Y := X + Y$  б) $Z := X + Y$

6.11. Дано описание:

```
C SEGMENT
     ASSUME CS:C
     X DW 1
BEG: MOV AX,Y
     ADD AX,X
     ...
     Y DW 2
C ENDS
```

При трансляции команды *MOV* ассемблер зафиксирует ошибку, тогда как трансляция команды *ADD* пройдет без ошибки. В чем разница между этими двумя случаями? Как исправить ошибку в команде *MOV*?

6.12. Дано описание:

```
A SEGMENT
    DB 400 DUP(?)
A ENDS
```

Выписать группу команд, копирующих самих себя в начало сегмента *A*.

6.13. Пусть в программе описан следующий сегмент данных:

```
S SEGMENT
    A DB 100 DUP(4)
    B DW 5,8,6
S ENDS
```

и пусть регистр *DS* уже установлен на начало этого сегмента. Определить, какое значение будет иметь регистр *AX* после выполнения каждой из следующих команд:

а) `MOV AX,B` б) `MOV AX,OFFSET B` в) `LEA AX,B`
г) `MOV AX,B+2` д) `MOV AX,OFFSET B + 2` е) `MOV AX,B – OFFSET B`

6.14. Дано описание:

```
X DB 200 DUP(?)   ; X[0..199]
```

Пусть в регистре *BX* находится адрес (смещение) одного из элементов массива *X*. Используя оператор *OFFSET*, реализовать переход на метку *L*, если:
а) в *BX* находится адрес элемента *X*[37];
б) *BX* указывает на вторую половину массива *X*.

6.15. Описать полную программу для решения следующей задачи.

а) Напечатать фразу «Hello World!».
б) Дан текст из 50 символов. Определить, симметричен ли он. Ответ: «симметричен» или «не симметричен».
в) Дан массив из 40000 чисел (размером в слово). Определить, симметричен ли этот массив. Ответ: «да» или «нет».
г) Дан текст из любых символов (кроме точки), за которым следует точка, и в котором не более 66000 символов. Напечатать этот текст в обратном порядке.
д) Дан непустой текст из любых символов (кроме точки), за которым следует точка. Напечатать этот текст, удалив из него лишние пробелы (из подряд идущих пробелов оставить только один).
е) Дан текст из больших латинских букв, за которым следует точка. Для каждой буквы указать, сколько раз она входит в этот текст.
ж) Дан непустой текст из любых символов (кроме точки), за которым следует точка. Определить, сколько различных больших русских гласных букв (А, Е, И, О, У, Ы, Э, Ю, Я) входит в этот текст.

6.16. Пусть под сегмент стека отведено 900 байтов. Реализовать переход на метку *L*, если в текущий момент:

а) стек полон;
б) стек пуст;
в) стек занимает ровно треть сегмента стека.

6.17. Используя любые регистры как вспомогательные, описать через другие команды действие команды:

а) `PUSH AX` б) `POP AX`

6.18. Определить, какие значения будут находиться в байте с адресом *SS*:*SP* и в байте с адресом *SS*:(*SP* + 1) после выполнения команд

```
MOV AX,0102h
PUSH AX
```

6.19. Выписать фрагмент программы, в котором вводится последовательность ненулевых чисел, заканчивающихся нулем, и эти числа выводятся в обратном порядке, но только если среди них нет отрицательных чисел (в противном случае ничего не выводить). Исходное значение регистра *SP* должно быть сохранено.

6.20. Пусть в стек записано 40 слов. Реализовать следующую операцию:

а) поменять местами два «верхних» слова стека, сохранив при этом значения всех регистров и не используя переменные;

б) определить, сколько среди этих слов нулевых, и записать ответ в регистр *AL*;
в) удалить из стека нулевые слова, «сжав» остальные (дополнительный массив не использовать);
г) определить, есть ли в стеке хотя бы два одинаковых слова, и записать ответ 1 (есть) или 0 в регистр *AL*;
д) рассматривая слова из стека как адреса (смещения) некоторых байтов из сегмента данных, обнулить все эти байты.

6.21. Пусть под сегмент стека отведено 200 слов и пусть в сегменте данных описан массив *X* из 200 слов. Выписать фрагмент программы, в котором в начало этого массива копируется текущее содержимое стека («верхнее» слово стека должно быть записано в начальный элемент массива) и заполняется нулями оставшаяся часть массива. Значение регистра *SP* должно быть сохранено.

6.22. Используя команды *PUSHF* и *POPF*, реализовать следующую операцию:

а) записать в регистр *AL* текущее значение флага *SF*, сохранив при этом значения всех флагов;
б) поменять на противоположное текущее значение флага *SF*, сохранив при этом значения всех остальных флагов.

(Замечание: флаг *SF* — это 7-ой справа (при нумерации с 0) бит регистра флагов.)

6.23. Выписать программу для решения следующей задачи.

а) Для ввода задана последовательность символов, представляющая собой (без ошибок) формулу следующего вида:

```
<формула> ::= <цифра> | (<формула> <знак> <формула>)
<знак>    ::= + | -
<цифра>   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Вычислить значение этой формулы. (Пример: $((5-2)+7) \rightarrow 10$)

б) Для ввода задана последовательность символов, представляющая собой (без ошибок) формулу следующего вида:

```
<формула> ::= <цифра> | M(<формула>,<формула>) |
              m(<формула>,<формула>)
<цифра>   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

где *M* трактуется как max (максимум), а *m* — как min (минимум).

Вычислить значение этой формулы. (Пример: $M(2, m(5, 7)) \rightarrow 5$)

# 7. ПРОЦЕДУРЫ.

7.1. Используя регистр *AX* как вспомогательный, описать через другие команды действие команды:

а) `CALL P` (*P* — имя близкой процедуры);
б) `RET n` (в случае близкого возврата).

7.2. Описать программу, которая вводит 4 положительных числа и определяет их наибольший общий делитель.

В программе описать и использовать процедуру нахождения наибольшего общего делителя двух чисел (эти числа-параметры передавать через регистры) при условии, что:
а) в программе должен быть только один сегмент команд;
б) процедура должна быть описана в отдельном сегменте команд.

7.3. Дано описание:

```
X DD ?
```

Описать близкую процедуру *OUTW16*, которая печатает в виде 4-значного беззнакового 16-ричного числа значение заданного слова. Используя эту процедуру, выписать фрагмент основной программы, печатающий значение переменной *X* в 16-ричном виде.

Выполнить это упражнение при условии, что параметр передается процедуре:
а) через регистр; б) через стек.

7.4. Дано описание:

```
X DB 100 DUP(?)     ; X[0..99]
```

Описать дальнюю процедуру *SETABS*, которой передается адрес (смещение в сегменте данных) некоторого знакового байта памяти и которая заменяет его значение на абсолютную величину. Используя эту процедуру, выписать фрагмент основной программы для решения следующей задачи: при $X[0] > X[27]$ заменить на абсолютную величину значение элемента $X[27]$, иначе — элемента $X[0]$.

Выполнить это упражнение при условии, что параметр передается процедуре:
а) через регистр; б) через стек.

7.5. Дано описание:

```
DATE   STRUC ; тип "дата"
        D     DB ?     ; день
        M     DB ?     ; месяц
        Y     DW ?     ; год
DATE   ENDS
```

Описать близкую процедуру *LESS*, которая сравнивает две даты типа *DATE* и возвращает через регистр *AL* свой ответ: 1, если первая дата предшествует второй, и 0 иначе. Процедуру описать при условии, что адреса дат (смещения в сегменте данных) передаются ей:

а) через регистры *SI* (первая дата) и *DI*;
б) через стек (сначала в стек записывается адрес первой даты).

7.6. Дано описание:

```
A DW ?   ; числа со знаком
B DW ?
C DW ?
```

Выписать фрагмент основной программы, в котором значения переменных *A*, *B* и *C* перераспределяются так, чтобы оказалось $A \ge B \ge C$. Для решения этой задачи предварительно описать близкую процедуру *MAXMIN*(*X*, *Y*), которая перераспределяет значения *X* и *Y* так, чтобы большее из них оказалось в *X*, а меньшее — в *Y*.

Выполнить это упражнение при условии, что параметры передаются процедуре:
а) через регистры; б) через стек.

7.7. Дано описание:

```
A DB  60 DUP(?)   ; числа со знаком
B DB 101 DUP(?)
```

Описать дальнюю процедуру *OUTARR8*, которой передается начальный адрес знакового байтового массива и число элементов в нем и которая печатает этот массив. Используя эту процедуру, выписать фрагмент основной программы для решения следующей задачи: если последний элемент массива *A* равен среднему элементу массива *B*, тогда напечатать массив *A*, иначе — массив *B*.

Выполнить это упражнение при условии, что параметры передаются процедуре:
а) через регистры; б) через стек.

7.8. Дано описание:

```
A DB 100 DUP(?)  ; числа со знаком
B DW ?
```

Описать близкую процедуру *SUM*(*X*, *N*, *S*), которая присваивает параметру-слову *S* сумму элементов массива *X* из *N* знаковых байтов, и выписать команды, соответствующие следующему обращению к процедуре: *SUM*(*A*, 100, *B*).
Выполнить это упражнение при условии, что параметры передаются процедуре:
а) через регистры; б) через стек.

7.9. Описать близкую процедуру *F*(*X*, *N*, *P*), определяющую, сколько элементов массива *X* из *N* байтов равно байту *P*, и возвращающую результат через регистр *AL*. Использовать эту процедуру для вычисления

$$K := F(A, 70, F(B, 30, K))$$

где *A* — массив из 70 байтов, *B* — массив из 30 байтов, а *K* — байтовая переменная.
Выполнить это упражнение при условии, что параметры передаются процедуре:
а) через регистры; б) через стек.

7.10. Описать подходящую процедуру и, используя её, выписать фрагмент основной программы для решения следующей задачи.

Имеются массивы *X*[0..59], *Y*[0..22] и *Z*[0..89] из знаковых чисел-слов. Требуется заменить максимальный элемент массива *X* на последний элемент массива *Y* и заменить все элементы массива *Z*, предшествующие его максимальному элементу, на максимальный элемент массива *Y*. (Считать, что в каждом массиве только один максимальный элемент.)

7.11. Описать подходящую процедуру и, используя ее, выписать фрагмент основной программы для решения следующей задачи.

Если в массиве *X* из 400 слов есть повторяющиеся элементы, а в массиве *Y* из 60 слов все элементы различны, тогда в регистр *AL* записать 1, а иначе записать 0.

7.12. Описать близкую процедуру *SHIFT23*, которой передается начальный адрес некоторого массива из 100 байтов и которая за один просмотр этого массива циклически сдвигает его элементы на 23 позиции вперед (влево). В своей работе процедура должна использовать вспомогательный массив, отведя ему место в стеке.

Выполнить это упражнение при условии, что параметр передается процедуре:
а) через регистр; б) через стек.

7.13. Описать дальнюю процедуру *MAXLET* для определения, какая из больших латинских букв чаще всего встречается в символьном массиве, начальный адрес которого передается через регистр *BX*, а число элементов в нем — через регистр *CX* (считать, что такая буква есть и она единственная). Свой ответ процедура должна вернуть через регистр *AL*. В своей работе процедура должна использовать вспомогательный массив, отведя ему место в стеке.

7.14. Используя из операций вывода только *OUTCH*, описать дальнюю процедуру *OUTSTRING*, действие которой эквивалентно операции вывода строки *OUTSTR*.

7.15. Описать близкую процедуру *OUTW*, печатающую значение регистра *AX* как беззнаковое число в системе счисления, основание которой (от 2 до 10) передается

через регистр *BL*. (Обратить внимание на возможность переполнения в команде *DIV*.)

7.16. Используя из операций ввода-вывода только *INCH* и *OUTSTR*, описать дальнюю процедуру *INDWORD*, которая вводит десятичное число от 0 до $2^{32}-1$ (концом числа считать первый символ, отличный от цифры) и записывает его в регистры *DX* (старшие цифры) и *AX* (младшие). Если число задано с ошибкой, то процедура должна выдать сообщение об ошибке («нет цифр», «слишком большое число») и попросить повторить ввод числа, после чего заново начать ввод числа. Все используемые данные должны быть размещены в самой процедуре, после команды *RET*. (Следует учесть, что операция *OUTSTR* печатает строку, чей абсолютный начальный адрес передается через регистры *DS*:*DX*.)

7.17. Описать близкую процедуру *SUM*, которой через регистр *BX* передается начальный адрес, а через регистр *CX* — число элементов некоторого массива, элементы которого (размером в слово) являются адресами каких-то знаковых байтов в сегменте данных. Процедура должна найти сумму значений всех этих байтов и вернуть ответ через регистр *AX*.

7.18. Описать дальнюю процедуру *ZERO* от 20 параметров, которые передаются через стек и каждый из которых (размером в слово) представляет собой адрес некоторой знаковой байтовой переменной из сегмента данных. Процедура должна обнулить те из этих переменных, значения которых положительны.

7.19. Описать дальнюю процедуру *CONCAT*($S_1$, $S_2$, ..., $S_n$, $S$, $n$), которая осуществляет конкатенакцию (сцепление) $n$ строк (символьных массивов из сегмента данных) $S_i$ и записывает получившуюся строку-результат (в ней сначала идут все символы из $S_1$, затем — из $S_2$, ..., в конце — из $S_n$) в строку $S$. Параметры процедуре передаются через стек в таком порядке: сначала в стек записываются данные (начальный адрес и длина-слово) о $S_1$, затем — о $S_2$, ..., в конце — о $S$ и, наконец, число-слово $n$ (> 0), указывающее количество сцепляемых строк $S_i$. Если длина строки-результат больше размера строки $S$, то лишние справа символы отбросить, если меньше — в конец $S$ дописать пробелы.

7.20. Дано описание:

```
A DB ?
B DB ?
```

Выписать фрагмент основной программы, в котором для каждой из переменных *A* и *B* выводится строка вида

```
<адрес>:<содержимое>
```

где *<адрес>* — это адрес (смещение) переменной в виде четырехзначного 16-ричного числа, а *<содержимое>* — значение переменной в виде двузначного 16-ричного числа (например: 01A8:F5).

При решении этой задачи описать и использовать следующие процедуры:

- — вывод числа от 0 до 15 в 16-ричном виде;
- — вывод байта в виде двух 16-ричных цифр;
- — вывод строки указанного вида для одной переменной.

7.21. Описать программу, которая в цикле вводит приказы указанных ниже типов и тут же выполняет их. (Считать, что приказы задаются без ошибок.)

Возможные приказы:

S *seg:ofs*. напечатать в 16-ричном виде (4 цифры) слово памяти, абсолютный адрес которого определяется парой *seg:ofs* (сегмент : смещение);

L *seg:ofs=w* в слово памяти, абсолютный адрес которого задан парой *seg:ofs*, записать новое значение *w*;

E. завершить работу программы.

Здесь *seg*, *ofs* и *w* — 16-ричные числа (от 1 до 4 цифр), в которых «буквенные» цифры записываются большими латинскими буквами от A до F.

7.22. Описать программу, которая вводит два массива из 100 знаковых чисел-слов в каждом и определяет, составлены ли эти массивы из одних и тех же чисел (без учета порядка их следования, но с учетом повторяющихся чисел). (Рекомендация: упорядочить оба массива, а затем сравнить их.)

7.23. Описать близкую рекурсивную процедуру *C*, вычисляющую биномиальный коэффициент $C(m, n)$, где $0 \le m \le n$, по следующей формуле:

$$C(m,n)=\begin{cases} 1, & m=0 \mid m=n \\ C(m,n-1)+C(m-1,n-1), & 0<m<n \end{cases}$$

Параметры *n* и *m* передаются процедуре через регистры *AH* и *AL* соответственно, а свой ответ она возвращает через регистр *BX*.

7.24. Для ввода задана последовательность символов (отличных от точки), за которой следует точка. Описать дальнюю рекурсивную процедуру *REVERSE* без параметров, которая вводит эти символы и выводит их (без точки) в обратном порядке.

7.25. Для ввода задана последовательность ненулевых чисел, за которой следует 0. Описать близкую рекурсивную процедуру PR без параметров, которая вводит эти числа и выводит сначала все отрицательные числа, а затем — все положительные (в любом порядке).

7.26. Описать близкую рекурсивную процедуру *BITS1*, которая подсчитывает количество двоичных единиц в значении регистра *AX* и возвращает ответ через регистр *BL*.

7.27. Для ввода задана последовательность символов, представляющая собой правильную запись формулы следующего вида:

```
<формула>  ::=  <цифра>  |  M(<формула>,<формула>) |
                m(<формула>,<формула>)
<цифра>    ::=  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

(*M* трактуется как максимум (max), а *m* — как минимум (min)). Описать близкую рекурсивную процедуру *MM* без параметров, которая вводит эту формулу, вычисляет ее значение (как число) и присваивает его регистру *AL*.

7.28. Описать программу, которая вводит текст вида

```
<формула> = <формула>
```

где

```
<формула>  ::=  <цифра> | (<формула> <знак> <формула>)
<знак>     ::=  + | -
<цифра>    ::=  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

и которая определяет, равны ли значения двух указанных формул. Ответ: ДА или НЕТ.

В программе описать и использовать рекурсивную процедуру, которая вводит и вычисляет значение одной формулы.

7.29. Описать близкую рекурсивную процедуру *FORM* без параметров, которая вводит текст из любых символов (кроме точки), за которым (обязательно) следует точка, и проверяет, удовлетворяет ли этот текст следующему определению:

```
<текст>    ::= <пусто> | <элемент> <текст>
<пусто>    ::=
<элемент>  ::= <буква> | (<текст>) |  [<текст>]  |  {<текст>}
<буква>    ::= A | B
```

# 8. ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ.

8.1. Используя, если надо, регистр *AX* как вспомогательный и считая, что флаг направления *DF* равен 0, описать через другие (не строковые) команды действие команды:

а) `MOVSB` б) `CMPSB` в) `SCASB` г) `LODSB` д) `STOSB` е) `REP MOVSB`

8.2. Дано описание:

```
S DB 'abcde',0
T DB 'abxyz'
```

Считая, что строки *S* и *T* описаны в сегменте данных и что регистры *DS* и *ES* уже установлены на начало этого сегмента, определить значения регистров *CX* и *BL* и флага *ZF* после выполнения следующей группы команд:

а)
```
CLD
LEA SI,S
LEA DI,T
MOV CX,5
REPE CMPSB
MOV BL,[SI]
```

б)
```
STD
LEA SI,S+4
LEA DI,T+4
MOV CX,5
REPNE CMPSB
MOV BL,[SI]
```

в)
```
CLD
LEA DI,S
MOV CX,5
MOV AL,'e'
REPNE SCASB
MOV BL,ES:[DI]
```

г)
```
CLD
LEA DI,S
MOV CX,5
MOV AL,'f'
REPNE SCASB
MOV BL,ES:[DI]
```

8.3. Дано описание:

```
S DB 100 DUP(?)    ; S[0..99] of char
T DB 100 DUP(?)    ; T[0..99] of char
```

Рассматривая *S* и *T* как символьные строки фиксированной длины, описанные в сегменте данных, на начало которого уже установлены регистры *DS* и *ES*, и используя строковые команды, выписать фрагмент программы для решения следующей задачи.

а) Определить, со скольких пробелов начинается строка *S*, и записать ответ в регистр *CL*.
б) Определить, сколькими пробелами заканчивается строка *S*, и записать ответ в регистр *CL*.
в) Найти индекс (от 0 до 99) первого вхождения буквы 'Q' в строку *S* и записать ответ в регистр *DI*; если эта буква не входит в *S*, то в *DI* записать −1.
г) Заменить в строке *S* последнее вхождение буквы 'n' на букву 'N'.
д) Записать в регистр *BH* число вхождений символа '*' в строку *S*.
е) Определить, равны ли левая и правая половины строки *S*, и записать ответ 1 (равны) или 0 в регистр *AL*.
ж) Определить, входят ли первые 5 символов строки *T* в строку *S* как подстрока, и записать ответ 1 (входят) или 0 в регистр *AL*.
з) Заменить последние 10 символов строки *S* на 10 первых символов строки *T*.
и) Циклически сдвинуть элементы строки *S* на две позиции влево.
к) Если первая и вторая четверти строки *S* не равны, то заменить третью четверть на последнюю.
л) Переписать содержимое строки *T* в строку *S* с заменой всех пробелов на символ '*'.

м) В строке *S* заменить все пробелы на символ '*'.
н) В строке *S* заменить все большие латинские буквы на соответствующие малые.
о) Удалить из строки *S* все цифры, сдвинув к ее началу все остальные символы и дописав пробелы в ее конец.

8.4. Считая, что в программе на сегмент стека отведено 1000 байтов, и используя строковые команды, выписать фрагмент программы для решения следующей задачи: если в стеке нет ни одного нулевого байта, тогда переписать содержимое стека в область памяти, начинающейся с абсолютного адреса 12345h.

8.5. Не делая никаких предположений о кодировке букв русского алфавита и используя команду *SCASB*, описать близкую процедуру *LOWRUS*, которой через регистр *AL* передается некоторый символ и которая, если это большая русская буква, заменяет ее (в *AL*) на соответствующую малую русскую букву (иные символы не менять). Все данные, необходимые процедуре, описать в самой процедуре.

8.6. Дано описание:

```
S DB 256 DUP(?)   ; длина(S)≤255
T DB 101 DUP(?)   ; длина(T)≤100
```

Рассматривая *S* и *T* как символьные строки переменной длины (с текущей длиной в начальном байте), размещенные в одном сегменте памяти, на начало которого уже установлены регистры *DS* и *ES*, и используя строковые команды, выписать фрагмент программы для решения следующей задачи.
а) В строке *S* оставить только первые 10 символов (не более).
б) Сделать значением *S* строку из 50 пробелов.
в) В строке *S* заменить все большие латинские буквы на соответствующие малые.
г) Удалить все пробелы в конце строки *S*.
д) Удалить все пробелы в начале строки *S*.
е) Если в строке *S* от 10 до 40 символов, то продублировать 10-й символ.
ж) Удалить из строки *S* все пробелы.
з) В конец строки *S* дописать символы строки *T*, отбросив при этом лишние (256-й и последующие) символы, если такие окажутся.
и) Сравнить строки *S* и *T* и записать ответ 1 ($S > T$), 0 ($S = T$) или −1 ($S < T$) в регистр *AL*.
к) Определить, является ли строка *T* подстрокой строки *S*, и записать ответ 1 (является) или 0 в регистр *AL*.
л) Удалить из строки *S* все вхождения подстроки *T*.

8.7. Описать указанные ниже процедуры при следующих условиях:

— все используемые строки — это строки переменной длины (с текущей длиной в начальном байте), их максимальная длина — 255, нумерация их символов начинается с 1;
— все параметры передаются через стек, причем порядок их записи в стек — слева направо;
— числовые параметры положительны и имеют размер слова;
— параметр-строка задается своим абсолютным начальным адресом в виде пары «сегмент : смещение» («сегмент» записывается в стек первым).

Процедуры (*len*(*S*) означает текущую длину строки *S*):

а) *copy*(*R*, *S*, *i*, *n*) — в строку *R* переписываются *n* символов строки *S* начиная с *i*-го (старое значение *R* уничтожается); если *i* > *len*(*S*), то строка *R* становится пустой; при *i* + *n* −1 > *len*(*S*) копируются все символы до конца *S*;
б) *delete*(*S*, *i*, *n*) — из строки *S* удаляются *n* символов начиная с *i*-го; при *i* > *len*(*S*) строка *S* не меняется; при *i* + *n* − 1 > *len*(*S*) удаляются все символы до конца *S*;
в) *insert*(*SS*, *S*, *i*) — строка *SS* вставляется в строку *S* между (*i* − 1)-м и *i*-м символами; если *i* > *len*(*S*), то *SS* добавляется в конец *S*; если длина строки-результата больше 255, то все лишние (256-й и последующие) символы отбрасываются;
г) *pos*(*SS*, *S*) — это функция, которая определяет, входит ли строка *SS* в строку *S* как подстрока, и возвращает (через регистр *AL*) как свое значение номер той позиции строки *S*, с которой начинается первое вхождение *SS* в *S*; если *SS* не входит в *S*, тогда значение функции равно 0.

8.8. Дано описание:

```
L  DW ?
L1 DW ?
X  DW 100 DUP(?)
```

Считая, что уже описаны константа *NIL*, тип *NODE* и процедуры *NEW* и *DISPOSE*, что регистр *ES* уже установлен на начало сегмента кучи и что значения *L* и *L1* трактуются как ссылки на списки (возможно, пустые) из знаковых чисел-слов, выписать фрагмент программы для решения следующей задачи.

а) Определить число положительных элементов в списке *L* и записать это число в регистр *AX*.
б) Обнулить все отрицательные элементы списка *L*.
в) Если в списке *L* не менее двух звеньев, то изменить знак у элемента его предпоследнего звена. (Рекомендация: хранить в двух модификаторах, например *BX* и *SI*, адреса текущего и предыдущего звеньев списка.)
г) Сравнить массив *X* и список *L* (равны ли последовательности их элементов?) и записать ответ 1 (равны) или 0 в регистр *AL*.
д) Сравнить списки *L* и *L1* и записать ответ 1 (равны) или 0 в регистр *AL*.
е) Определить, есть ли в списке *L* звенья с равными элементами, и записать ответ 1 (есть) или 0 в регистр *AL*.
ж) По массиву *X* построить список *L* — из тех же элементов и в том же порядке. (Рекомендация: список строить от конца.)
з) Ввести непустую последовательность ненулевых чисел, за которой следует 0, и напечатать эти числа в обратном порядке.
и) Создать *L1* — копию списка *L*.
к) В список *L* вставить звено с нулевым элементом после первого отрицательного элемента, если такой есть.
л) В список *L* вставить нулевой элемент перед первым отрицательным элементом, если такой есть.
м) В списке *L* продублировать каждое звено, содержащее положительный элемент.
н) Вставить между первым и вторым звеньями непустого списка *L* копии всех звеньев списка *L1*.
о) Удалить из непустого списка *L* его последний элемент.
п) Удалить из списка *L* первый нулевой элемент, если такой есть.
р) Перенести первый элемент непустого списка *L* в конец списка.
с) Перенести последний звено непустого списка *L* в начало списка.
т) Удалить из списка *L* все звенья с отрицательными элементами.

у) Уничтожить список *L*, освободив место, занимаемое его звеньями, и присвоить переменной *L* значение *NIL*.
ф) Удалить из списка *L* все звенья, элементы которых входят в список *L1*.

8.9. Описать близкую рекурсивную процедуру, параметр (ссылка на список) для которой передается через регистр *BX* и которая возвращает ответ через регистр *AX*, для решения следующей задачи.

а) Найти последнее звено непустого списка.
б) Найти длину (число звеньев) заданного списка.
в) Определить, входит ли заданное (в регистре *AX*) число в заданный список (ответ: 1 (входит) или 0).
г) Найти число звеньев с отрицательными элементами в заданном списке.
д) Определить, есть ли в заданном списке хотя бы одно звено с отрицательным элементом (ответ: 1 (есть) или 0).
е) Найти звено с максимальным элементом непустого заданного списка.
ж) Построить копию заданного списка.
з) Удалить из заданного списка первый звено с нулевым элементом, если оно есть.
и) Удалить из заданного списка все звенья с нулевыми элементами.

8.10. Пусть для работы с двоичными (бинарными) деревьями используется куча, организованная по аналогии с кучей для (линейных) списков (есть константа *NIL*, на начало кучи постоянно указывает регистр *ES* и т.д.), но при этом вершины деревьев описываются следующим структурным типом:

```
NODE   STRUC ; вершина дерева:
 ELEM  DW ?   ; элемент (знаковое число)
 LEFT  DW ?   ; ссылка на левое  поддерево или NIL
 RIGHT DW ?   ; ссылка на правое поддерево или NIL
NODE   ENDS
```

и процедуры *NEW* и *DISPOSE* работают с тремя соседними словами кучи.
Описать (рекурсивно или с использованием стека) близкую процедуру, которой через регистр *BX* передается ссылка на дерево (возможно, пустое) и которая свой ответ возвращает через регистр *AX*, для решения следующей задачи.
а) Найти сумму всех элементов заданного дерева.
б) Поменять знак у всех отрицательных элементов заданного дерева.
в) Найти максимальное значение элементов заданного непустого дерева.
г) Подсчитать число листьев в заданном дереве.
д) Найти максимальную глубину заданного непустого дерева (число ветвей в наиболее длинном пути).
е) Определить число вершин на *N*-м уровне заданного дерева, если число $N$ ($\geq 0$) передается через регистр *CX* (считать, что корень находится на 0-м уровне).
ж) Скопировать заданное дерево.
з) Ввести непустую последовательность попарно различных положительных чисел, за которой следует 0, и построить из них дерево поиска (в нем слева от любой вершины находятся только меньшие числа, а справа — только большие).
и) Распечатать в порядке возрастания все элементы заданного дерева поиска (см. предыдущий пункт).
к) Из заданного непустого дерева удалить все листья.

# 9. МАКРОСРЕДСТВА.

9.1. Описать с помощью подходящего блока повторения решение следующей задачи:

а) записать в регистр *AH* сумму чисел из регистров *AL*, *BL*, *CL* и *DH*;
б) обнулить переменные *A*, *B* и *C* типа *DWORD*;
в) используя из операций вывода только операцию *OUTCH*, вывести (с кавычками) текст "A+B=B+A";
г) зарезервировать (с помощью директивы *DB*) место в памяти для 40 байтов, присвоив им в качестве начальных значений первые 40 нечетных чисел (1, 3, 5, ..., 79).

9.2. Выписать текст окончательной программы, который построит макрогенератор по следующему фрагменту исходной программы:

а)
```
IRP C,<INC A,JE L>
  C
ENDM
```

б)
```
IRP W,<SW,< A,2>>
   MOV&W
ENDM
```

в)
```
N EQU 5
  IRP OP,<N,%N,N%N>
    ADD AX,OP
ENDM
```

г)
```
IRP C,<K,LL,M>
C EQU C&C&CC
  DB 'C&C&CC'
ENDM
```

д)
```
IRP T,<AB,C>
  IRPC U,T
    DW U,T&U,T&&U
  ENDM
  DW U,T&U
  ENDM
```

е)
```
IRPC CH,<!  %">
   ADD AL,'&CH' ;код CH
ENDM
```

9.3. Описать в виде указанных (слева) макросов указанные операции над двойными словами (*X*, *Y* и *Z* — переменные типа *DWORD*, *L* — метка):

```
DMOV X,Y    :  X:=Y
DADD X,Y,Z  :  X:=Y+Z
DSUB X,Y,Z  :  X:=Y-Z
DJNE X,Y,L  :  при X≠Y перейти на L
```

9.4. Описать в виде макроса *DEF X*, *T*, *N*, *V* определение массива *X* из *N* величин *V*, тип которых задается параметром *T*: при *T* = *B* это тип *BYTE*, при *T* = *W* — тип *WORD*, при *T* = *D* — тип *DWORD*.

Выписать текст, который построит макрогенератор по следующему фрагменту исходной программы:

```
K EQU 4
DEF C,B,10,'*'
DEF W,W,K+1,<TYPE C>
DEF ,D,%K+1,%(TYPE W)
DEF A,B,1,<1,2,3>
```

9.5. Описать полную программу, которая вводит три числа *H*, *M* и *S* и проверяет, удовлетворяют ли они следующим условиям: $0 \le H \le 23$, $0 \le M, S \le 59$. Если нет, программа должна выдать сообщение об ошибке, а иначе, трактуя эти числа как час (*H*), минута (*M*) и секунда (*S*) некоторого момента суток, должна напечатать время суток, на 1 секунду большее (с учетом смены суток).

Определить и использовать в этой программе два макроса, один из которых проверяет условие $a \le X \le b$, а другой — увеличивает *X* на 1 и, если $X > b$, обнуляет *X*.

9.6. Описать в виде указанного (слева) макроса указанное действие над знаковыми числами размером в байт:

а) `ABS R,X` : $R :=$ abs($X$), где *R* — регистр, *X* — переменная
б) `SUM X,N` : $AX :=$ сумма элементов массива *X* из *N* байтов ($N > 0$)

в) `MAX X,N` : $AL :=$ максимум элементов массива $X$ из $N$ байтов ($N > 0$)

9.7. Описать в виде макроса указанную команду, предполагая, что ее нет в ПК (в качестве вспомогательных можно использовать регистры *AX* и *BX*):

а) `PUSH X` ($X$ — переменная размером в слово);
б) `POP X` ($X$ — переменная размером в слово);
в) `CALL P` ($P$ — имя близкой процедуры);
г) `RET N` (близкий возврат);
д) `LOOP L` ($L$ — метка);

9.8. Дано описание:

```
X       DW 1234h
A       MACRO R,U
          MOV R,U
        ENDM
B       MACRO R,V
          A R,<V>
        ENDM
C       MACRO R,V
          A R,V
        ENDM
```

Определить значения регистров *AH* и *AL* после выполнения следующего фрагмента программы:

```
B AH,<BYTE PTR X>
C AL,<BYTE PTR X>
```

9.9. Описать в виде макроса *MAX2 M*, *X*, *Y* вычисление $M = \max(X, Y)$ и на его основе описать макрос *MAX3 M*, *X*, *Y*, *Z* для вычисления $M = \max(X, Y, Z)$, где *M*, *X*, *Y* и *Z* — знаковые байтовые переменные.

Выписать макрорасширение для макрокоманды

```
MAX3 A,<BYTE PTR B>,C+1,D
```

9.10. Пусть $K$ — числовая константа с положительным значением, а $\alpha$, $\beta$ и $\gamma$ — некоторые группы предложений. Используя средства условного ассемблирования, выписать фрагмент исходной программы, в котором $\alpha$, $\beta$ и $\gamma$ указаны лишь по разу и по которому в зависимости от значения $K$ формируются следующие варианты окончательной программы:

| при $K = 1$: | при $K = 2$: | при $K > 2$: | |
|---|---|---|---|
| $\alpha$ | $\alpha$ | $\beta$ | |
| $\beta$ | $\gamma$ | $\gamma$ | |
| | $\gamma$ | ... | ($K$ раз) |
| | | $\gamma$ | |

9.11. Пусть *RUN* — константа, которая своим значением 1 или 0 указывает режим текущего прогона программы: счет или отладка. Используя средства условного ассемблирования, выписать фрагмент программы, в котором находится наибольший общий делитель двух положительных чисел с записью его в регистр *AX*, при условии, что в режиме счета эти два числа должны вводиться, а в режиме отладки эти числа равны 45 и 30.

9.12. Имеются следующие описания макроса *M X*, который должен уменьшить значение *X* на 5, если *X* — это переменная-байт, или на 12, если *X* — это переменная-слово:

а)
```
M  MACRO X
   LOCAL L2,L
     CMP TYPE X,BYTE
     JNE L2
```

б)
```
M  MACRO X
   LOCAL L2,L
     MOV AL,TYPE X
     CMP AL,BYTE
```

в)
```
M  MACRO X
     IF TYPE X EQ BYTE
       SUB X,5
     ELSE
```

```
        SUB X,5              JNE L2               SUB X,12
        JMP L                SUB X,5            ENDIF
 L2:    SUB X,12             JMP L            ENDM
  L:                       L2: SUB X,12
ENDM                         L:
                            ENDM
```

Какое из этих описаний неправильно (и почему) и какое из двух правильных описаний лучше другого (и почему)?

9.13. Описать в виде макроса *SHIFT X*, *K* (*X* — имя байтовой переменной, *K* — явно заданное число) сдвиг значения *X* на | *K* | разрядов вправо (при *K* > 0) или влево (при *K* < 0).

Выписать макрорасширения для макрокоманд *SHIFT A*, −1 и *SHIFT B*, 5.

9.14. Описать в виде макроса *IF0 X*, *L* (*X* — переменная размером в байт, слово или двойное слово, *L* — метка) переход на метку *L* в том случае, когда значение переменной *X* равно 0.

Выписать макрорасширения для макрокоманд *IF0 B*, *L1* и *IF0 D*, *L2* при условии, что *B* — переменная типа *BYTE*, а *D* — типа *DWORD*.

9.15. Описать в виде макроса *SIGN X* (*X* — знаковая переменная размером в байт, слово или двойное слово) операцию засылки в регистр *AL* числа 1 при *X* > 0, числа 0 при *X* = 0 и числа −1 при *X* < 0.

Выписать макрорасширения для макрокоманд *SIGN W* и *SIGN D* при условии, что *W* — переменная размером в слово, а *D* — размером в двойное слово.

9.16. Описать в виде макроса *NULL X*, *N*, *T* (*X* — имя массива из *N* байтов, *N* — положительное целое число, *T* — это *FIRST* или *LAST*) обнуление либо начального (при *T* = *FIRST*), либо последнего (при *T* = *LAST*) элемента массива *X*.

Выписать макрорасширение для макрокоманды *NULL A*, 100, *LAST*.

9.17. Описать в виде макрос *VAR X*, *EQ*, *V* определение байтовой переменной с именем *X* и начальным значением *V*, а если последний параметр не задан — без начального значения (параметр *EQ* фиктивный).

Выписать макрорасширения для макрокоманд:
а) `VAR A = '*'` б) `VAR B : <10 DUP(?)>` в) `VAR C` г) `VAR`

9.18. Описать в виде макроса *NULL RS* (*RS* — это <$R_1$, $R_2$, ..., $R_k$>, где $R_i$ — имена регистров общего назначения, $k \ge 0$) обнуление регистров $R_i$.

Выписать макрорасширения для макрокоманд *NULL <AL*, *BX*, *SI>* и *NULL <>*.

9.19. Описать макрос *SUM R*, где *R* — имя (большими буквами) одного из 16-разрядных регистров общего назначения, для записи в *R* суммы значений всех остальных таких регистров.

9.20. Описать в виде макроса *OUTF* без параметров вывод текущих значений флагов *CF*, *OF*, *SF* и *ZF* в виде четверки из 0 и 1, причем значения всех флагов и используемых макросом регистров должны быть сохранены.

Воспользоваться командами *PUSHF* и *POPF*. В регистре флагов *Flags* указанным флагам соответствуют биты со следующими номерами (нумерация битов справа налево от 0): *CF* — 0, *OF* — 11, *SF* — 7, *ZF* — 6.

9.21. Описать в виде макроса *SL RS*, *OP* (*RS* — это <$R_1$, $R_2$, ..., $R_k$>, где $R_i$ — 16-разрядные регистры общего назначения, $k > 0$; *OP* — это *SAVE* или *LOAD*) за-

пись в стек (при *OP = SAVE*) или восстановление из стека (при *OP = LOAD*) регистров $R_i$.

Выписать макрорасширение для макрокоманды *SL <AX, CX, DI>, LOAD*.

9.22. Описать в виде макроса *SUM X* (*X* — это $<X_1, X_2, ..., X_k>$, где $X_i$ — имена знаковых байтовых переменных, $k > 0$) вычисление суммы значений $X_i$ и записи ее в регистр *BX*.

Выписать макрорасширение для макрокоманды *SUM <A, B, C>*.

9.23. Описать в виде макроса *MAX X* (*X* — это $<X_1, X_2, ..., X_k>$, где $X_i$ — имена знаковых байтовых переменных, $k > 0$) вычисление максимума $X_i$ и записи его в регистр *AL*. (Обратить особое внимание на метки, которые будут появляться в макрорасширениях.)

Выписать макрорасширение для макрокоманды *MAX <A, B, A>*.

9.24. Предположим, что имеется процедура *P* от двух параметров, которые по значению передаются через регистры *AX* и *BX*. Описать в виде макроса *CALL_P X,Y* команды обращения к этой процедуре, которые должны сохранять регистры *AX* и *BX* и которые должны корректно работать, когда в качестве *X* и *Y* указаны *AX* и/или *BX*. (Считать, что названия регистров записываются только большими буквами.)

Выписать макрорасширения для макрокоманд:
а) `CALL_P Q,2` б) `CALL_P AX,BX` в) `CALL_P BX,5` г) `CALL_P BX,AX`

9.25. Выписать при указанном макроопределении макрорасширение для указанной макрокоманды:

а)
```
MA  MACRO K
      JMP M1
      MOV AX,K
M&K:  ADD AX,K
    ENDM
    ...
    MA 1
```

б)
```
MB  MACRO K
      MOV AH,K
    IFIDN <AH>,<2>
      ADD AH,K
    ENDIF
    ENDM
    ...
    MB 2
```

в)
```
MV  MACRO K
  A EQU K
    IFDIF <A>,<K>
      ADD DX,A&K
    ENDIF
    ENDM
    ...
    MV 3
```

г)
```
MG  MACRO K
    IFDIF <K+1>,<5>
      MOV AX,K
    ENDIF
      MOV BX,K&K
    ENDM
  ...
  MG 4
```

д)
```
MD  MACRO K
    IF K+1 EQ 6
      X DB '&K+1'
    ENDIF
      Y&K DW K+1
    ENDM
    ...
    MD 5
```

е)
```
ME  MACRO K
      MOV AH,0
    IRPC C,K-1
      ADD AH,'&C'
    ENDM
    ENDM
    ...
    ME 6
```

# 10. МНОГОМОДУЛЬНЫЕ ПРОГРАММЫ.

10.1. Ответить на следующие вопросы относительно многомодульной программы:

а) Обязательно ли в каждом модуле программы описывать сегмент стека?
б) Если в программе используются вспомогательные процедуры ввода-вывода из файла *IO.ASM*, то обязательно ли в каждом модуле указывать директиву *INCLUDE IO.ASM*?
в) Как ассемблер, транслируя некоторый модуль программы, узнает, что какое-то имя в этом модуле является внешним? А общим?

г) Обязательно ли имя, объявленное в модуле общим, должно быть описано в этом модуле? Можно ли использовать данное имя в этом модуле?

д) Можно ли одно и то же имя объявить общим в нескольких модулях программы? А внешним?

10.2. Пусть в некотором модуле программы описана процедура *P*, которой через регистр *AX* передается по значению единственный параметр. Что надо сделать в другом модуле программы, чтобы обратиться к этой процедуре с нулевым фактическим параметром?

10.3. Пусть в первой строке текста некоторого модуля программы указана директива *EXTRN X*:*WORD* и пусть в этом модуле требуется записать значение переменной *X* в регистр *AX*. Определить, какой из следующих фрагментов правильно решает эту задачу.

а)
```
MOV AX,X
```

б)
```
MOV AX,ES:X
```

в)
```
MOV AX,SEG X
MOV ES,AX
MOV AX,ES:X
```

г)
```
MOV AX,SEG X
MOV DS,AX
MOV AX,X
```

10.4. Выписать (не основной) модуль программы, где описаны доступные другим модулям байтовая переменная *STEP* с начальным значением 1 и дальняя процедура *NEXT* без параметров, которая увеличивает на *STEP* значение байтовой переменной *TIME* из какого-то другого модуля.

10.5. Выписать вспомогательный модуль программы, содержащий описание доступной другим модулям дальней процедуры *NOD*, которая находит и записывает в регистр *AX* наибольший общий делитель двух натуральных чисел, переданных через регистры *AX* и *BX*.

Выписать также основной модуль программы, который вводит 4 натуральных числа и, используя процедуру *NOD*, определяет их наибольший общий делитель.

10.6. Выписать вспомогательный модуль программы, содержащий описание доступных другим модулям массива *PRIM* из 50 слов и дальней процедуры *INIT*, которая инициирует этот массив, записывая в него первые 50 простых чисел (2, 3, 5, ...).

Выписать основной модуль программы, который вводит последовательность натуральных чисел (за ней следует 0) и, используя процедуру *INIT* и массив *PRIM*, подсчитывает, сколько среди этих чисел равных первым 50 простым числам.

10.7. Используя для очереди («первый пришел — первым ушел») векторное представление, выписать вспомогательный модуль программы, в котором резервируется место (5 массивов по 1000 слов) для 5 очередей и описываются в виде процедур следующие операции над очередями:

— образовать новую (пустую) очередь (очереди можно идентифицировать по номерам 1, 2, ..., 5);
— проверить, пуста ли указанная (по номеру) очередь;
— записать элемент в указанную очередь;
— считать элемент из указанной очереди.

(Способ хранения элементов очереди в массиве, название процедур и т.п. - продумать самим.)

Выписать также основной модуль программы, который, используя процедуры из вспомогательного модуля, вводит последовательность ненулевых чисел (за ней следует 0) и печатает их в следующем порядке: сначала — все отрицательные

числа, затем — все числа, большие 500, и в конце — все остальные числа. При этом внутри каждой из этих трех групп чисел должно быть сохранено их исходное взаимное расположение.

10.8. Считая, что имеется вспомогательный модуль из предыдущего упражнения, выписать основной модуль программы, который вводит последовательность ненулевых чисел (за ней следует 0), в которой равное число положительных и отрицательных чисел, и печатает поочередно положительные и отрицательные числа (первым — положительное число), сохраняя при этом исходный взаимный порядок как среди положительных, так и среди отрицательных чисел. (Каждое число печатать, как только это становится возможным.)