

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МОЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №7**  
**по дисциплине «Организация ЭВМ и систем»**  
**Тема: Использование арифметических операций над целыми числами и**  
**процедур в Ассемблере**

Студент гр. 9382

\_\_\_\_\_

Русинов Д.А.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2020

## **Цель работы.**

Научиться использовать арифметические операции над целыми числами и процедуры в Ассемблере.

## **Основные теоретические положения.**

Разработать на языке Ассемблер процессора IntelX86 две процедуры:

- Одна – выполняет прямое преобразование целого числа, заданного в регистре AX ( или в паре регистров DX:AX) в строку, представляющую его символьное изображение в заданной системе счисления (с учетом или без учета знака в зависимости от варианта задания);
- Другая - обратное преобразование строки, представляющей символьное изображение числа в заданной системе счисления в целое число, помещаемое в регистр AX ( или в пару регистров DX:AX)

Строка должна храниться в памяти, а также выводиться на экран для индикации.

Отрицательные числа при представлении с учетом знака должны в памяти храниться в дополнительном коде, а на экране изображаться в прямом коде с явным указанием знака или в символьном виде со знаком.

## **Задание.**

Вариант 1.1.1

16 бит, с учетом знака, двоичная СС

## **Выполнение работы.**

В регистр ax задается число, которое необходимо перевести в строку. Процедура DEC\_TO\_BIN отвечает за перевод регистра в строку. Сначала проверяет знак числа. Если число отрицательное, то оно переводится в прямой код, иначе оно не нуждается в переводе. Затем происходит поиск значащего бита в числе. Если он не найден, то число либо нулевое, либо это самое большое

отрицательное число. Если был найден значащий бит, то формируется строка. Биты рассматриваются с помощью битмаски.

Процедура BIN\_TO\_DEC переводит строку к регистру. Рассмотрены отдельно случаи 0 и самое большое отрицательное число. В других случаях проходим по строке и формируем число с помощью битмаски.

В основной процедуре вызывается сначала DEC\_TO\_BIN, и выводится представление числа в строке, затем вызывается процедура BIN\_TO\_DEC, так как вручную ее проверять довольно сложно, вызывается процедура DEC\_TO\_BIN еще раз, чтобы убедиться, что в регистре после процедуры BIN\_TO\_DEC хранится правильное значение. Если два строковых представления совпадают, значит программа отработала правильно.

### **Тестирование.**

| Номер | Входные данные | Выходные данные  |
|-------|----------------|--|
| 1     | ax=F100h       | From register to string: -<br>111100000000<br><br>From string to register<br>[then for visual from<br>register to string]: -<br>111100000000         |
| 2     | Ax=8000h       | From register to string: -<br>1000000000000000<br><br>From string to register<br>[then for visual from<br>register to string]: -<br>1000000000000000 |
| 3     | Ax=0h          | From register to string: 0   |

|   |            |  |
|---|------------|--|
|   |            | From string to register<br>[then for visual from<br>register to string]: 0                                 |
| 4 | Ax = FFFFh | From register to string: -1<br>From string to register<br>[then for visual from<br>register to string]: -1 |
| 5 | Ax = 1h    | From register to string: +1<br>From string to register<br>[then for visual from<br>register to string]: +1 |

## Альтернативное тестирование

### 1. Первый тест.

В ax было записано 1h, после вызова процедуры DEC\_TO\_BIN в сегмент данных была записана строка +1\$.

DOSBox 0.74-3-2, Cpu speed: 3000 cycles, Frameskip 0, Program: AFDPRO

AX 0001 SI 0000 CS 1A4E IP 008D Stack +0 0000 Flags 7244  
 BX 0000 DI 0002 DS 1A45 +2 7246  
 CX 0001 BP 0000 ES 19F5 HS 19F5 +4 6D6F OF DF IF SF ZF AF PF CF  
 DX 0060 SP 0400 SS 1A05 FS 19F5 +6 7220 0 0 1 0 1 0 1 0

CMD >

|                   |     |              |         |                         |
|-------------------|-----|--------------|---------|-------------------------|
| 0089 EB02         | JMP | 008D         | DS:0000 | 00 00 46 72 6F 6D 20 72 |
| 008D 2BC0         | SUB | AX,AX        | DS:0008 | 65 67 69 73 74 65 72 20 |
| 008F 2BF6         | SUB | SI,SI        | DS:0010 | 74 6F 20 73 74 72 69 6E |
| 0091 80BC60002D   | CMP | [0060+SI],2D | DS:0018 | 67 3A 20 24 46 72 6F 6D |
| 0096 7506         | JNZ | 009E         | DS:0020 | 20 73 74 72 69 6E 67 20 |
| 0098 2EC6068C0001 | MOV | CS:[008C],01 | DS:0028 | 74 6F 20 72 65 67 69 73 |
| 009E 46           | INC | SI           | DS:0030 | 74 65 72 20 5B 74 68 65 |
| 009F 80BC600024   | CMP | [0060+SI],24 | DS:0038 | 6E 20 66 6F 72 20 76 69 |
| 00A4 75F8         | JNZ | 009E         | DS:0040 | 73 75 61 6C 20 66 72 6F |
|                   |     |              | DS:0048 | 6D 20 72 65 67 69 73 74 |

2

|         |                         |                         |                    |
|---------|-------------------------|-------------------------|--------------------|
| DS:0050 | 65 72 20 74 6F 20 73 74 | 72 69 6E 67 5D 3A 20 24 | er to st ringl: \$ |
| DS:0060 | 2B 31 24 30 30 30 30 30 | 30 30 30 30 30 30 30 30 | +1\$00000 00000000 |
| DS:0070 | 30 30 30 30 30 30 30 30 | 30 30 30 30 30 30 30 30 | 00000000 00000000  |
| DS:0080 | 30 30 30 00 00 00 00 00 | 00 00 00 00 00 00 00 00 | 000.....           |
| DS:0090 | EB 01 90 2B FF BB 00 80 | 3D 00 00 7D 0C C6 85 60 | δ.É+ 1.Ç =...}.fà  |

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 ↑ 8 ↓ 9 ← 10 →

Была вызвана процедура BIN\_TO\_DEC. Регистр ax обнулился.

DOSBox 0.74-3-2, Cpu speed: 3000 cycles, Frameskip 0, Program: AFDPRO

|         |         |         |         |               |                         |
|---------|---------|---------|---------|---------------|-------------------------|
| AX 0000 | SI 0002 | CS 1A4E | IP 00AB | Stack +0 0000 | Flags 7281              |
| BX 0000 | DI 0002 | DS 1A45 |         | +2 7246       |                         |
| CX 0001 | BP 0000 | ES 19F5 | HS 19F5 | +4 6D6F       | OF DF IF SF ZF AF PF CF |
| DX 0060 | SP 0400 | SS 1A05 | FS 19F5 | +6 7220       | 0 0 1 1 0 0 0 1         |

CMD > █

|                 |     |              |         |                         |
|-----------------|-----|--------------|---------|-------------------------|
| 00A9 7D1D       | JNL | 00CB         | DS:0000 | 00 00 46 72 6F 6D 20 72 |
| 00AB 83FE01     | CMP | SI,0001      | DS:0008 | 65 67 69 73 74 65 72 20 |
| 00AE 7429       | JZ  | 00D9         | DS:0010 | 74 6F 20 73 74 72 69 6E |
| 00B0 2BFF       | SUB | DI,DI        | DS:0018 | 67 3A 20 24 46 72 6F 6D |
| 00B2 BB0100     | MOV | BX,0001      | DS:0020 | 20 73 74 72 69 6E 67 20 |
| 00B5 4E         | DEC | SI           | DS:0028 | 74 6F 20 72 65 67 69 73 |
| 00B6 83FE00     | CMP | SI,0000      | DS:0030 | 74 65 72 20 5B 74 68 65 |
| 00B9 7413       | JZ  | 00CE         | DS:0038 | 6E 20 66 6F 72 20 76 69 |
| 00BB 80BC600031 | CMP | [0060+SI],31 | DS:0040 | 73 75 61 6C 20 66 72 6F |
|                 |     |              | DS:0048 | 6D 20 72 65 67 69 73 74 |

|         |                         |                         |                     |
|---------|-------------------------|-------------------------|---------------------|
| DS:0050 | 65 72 20 74 6F 20 73 74 | 72 69 6E 67 5D 3A 20 24 | er to st ringl: \$  |
| DS:0060 | 2B 31 24 30 30 30 30 30 | 30 30 30 30 30 30 30 30 | +1\$00000 00000000  |
| DS:0070 | 30 30 30 30 30 30 30 30 | 30 30 30 30 30 30 30 30 | 00000000 00000000   |
| DS:0080 | 30 30 30 00 00 00 00 00 | 00 00 00 00 00 00 00 00 | 000.....            |
| DS:0090 | EB 01 90 2B FF BB 00 80 | 3D 00 00 7D 0C C6 85 60 | δ.É+ 7.Ç =...f.  à` |

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 ↑ 8 ↓ 9 ◀ 10 ▶

На момент выхода из процедуры BIN\_TO\_DEC значение ax теперь то же самое, что и в начале программы, то есть 1h.

```

DOSBox 0.74-3-2, Cpu speed: 3000 cycles, Frameskip 0, Program: AFDPRO
AX 0001 SI 0000 CS 1A4E IP 00D9 Stack +0 0000 Flags 7244
BX 0002 DI 0000 DS 1A45 +2 7246
CX 0001 BP 0000 ES 19F5 HS 19F5 +4 6D6F OF DF IF SF ZF AF PF CF
DX 0060 SP 0400 SS 1A05 FS 19F5 +6 7220 0 0 1 0 1 0 1 0

CMD >

00D4 7403 JZ 00D9
00D9 CB RET Far
00DA B8451A MOV AX,1A45
00DD 8ED8 MOV DS,AX
00DF BA0200 MOV DX,0002
00E2 B409 MOV AH,09
00E4 CD21 INT 21
00E6 B80100 MOV AX,0001
00E9 E814FF CALL 0000

DS:0000 00 00 46 72 6F 6D 20 72
DS:0008 65 67 69 73 74 65 72 20
DS:0010 74 6F 20 73 74 72 69 6E
DS:0018 67 3A 20 24 46 72 6F 6D
DS:0020 20 73 74 72 69 6E 67 20
DS:0028 74 6F 20 72 65 67 69 73
DS:0030 74 65 72 20 5B 74 68 65
DS:0038 6E 20 66 6F 72 20 76 69
DS:0040 73 75 61 6C 20 66 72 6F
DS:0048 6D 20 72 65 67 69 73 74

2 0 1 2 3 4 5 6 7 8 9 A B C D E F
DS:0050 65 72 20 74 6F 20 73 74 72 69 6E 67 5D 3A 20 24 er to st ringl: $
DS:0060 2B 31 24 30 30 30 30 30 30 30 30 30 30 30 30 +1$000000 00000000
DS:0070 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 00000000 00000000
DS:0080 30 30 30 00 00 00 00 00 00 00 00 00 00 00 00 000..... .....
DS:0090 EB 01 90 2B FF BB 00 80 3D 00 00 7D 0C C6 85 60 δ.É+ ȳ.Ç =...}.|à'

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 ↑ 8 ↓ 9 ←= 10 ⇒

```

## 2. Второй тест.

Значение  $ax = 8000h$ . Сначала переводим  $ax$  в строку, в данный момент строка пустая.

```

DOSBox 0.74-3-2, Cpu speed: 3000 cycles, Frameskip 0, Program: AFDPRO
AX 8000 SI 0000 CS 1A4E IP 00E9 Stack +0 0000 Flags 7200
BX 0000 DI 0000 DS 1A45 +2 7246
CX 05A3 BP 0000 ES 19F5 HS 19F5 +4 6D6F OF DF IF SF ZF AF PF CF
DX 0002 SP 0400 SS 1A05 FS 19F5 +6 7220 0 0 1 0 0 0 0 0

CMD >

00E9 E814FF CALL 0000
00EC B409 MOV AH,09
00EE CD21 INT 21
00F0 B20A MOV DL,0A
00F2 B402 MOV AH,02
00F4 CD21 INT 21
00F6 B20D MOV DL,0D
00F8 B402 MOV AH,02

DS:0000 00 00 46 72 6F 6D 20 72
DS:0008 65 67 69 73 74 65 72 20
DS:0010 74 6F 20 73 74 72 69 6E
DS:0018 67 3A 20 24 46 72 6F 6D
DS:0020 20 73 74 72 69 6E 67 20
DS:0028 74 6F 20 72 65 67 69 73
DS:0030 74 65 72 20 5B 74 68 65
DS:0038 6E 20 66 6F 72 20 76 69
DS:0040 73 75 61 6C 20 66 72 6F
DS:0048 6D 20 72 65 67 69 73 74

2 0 1 2 3 4 5 6 7 8 9 A B C D E F
DS:0050 65 72 20 74 6F 20 73 74 72 69 6E 67 5D 3A 20 24 er to st ringl: $
DS:0060 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 00000000 00000000
DS:0070 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 00000000 00000000
DS:0080 30 30 30 00 00 00 00 00 00 00 00 00 00 00 00 00 000.....
DS:0090 EB 01 90 2B FF BB 00 80 3D 00 00 7D 0C C6 85 60 δ.É+ η.ç =...}.|à'

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 ↑ 8 ↓ 9 ◀ 10 ▶

```

После выхода из процедуры значение строки находится в сегменте данных.

```

DOSBox 0.74-3-2, Cpu speed: 3000 cycles, Frameskip 0, Program: AFDPRO
AX 8000 SI 0011 CS 1A4E IP 00EC Stack +0 0000 Flags 7204
BX 0000 DI 0001 DS 1A45 +2 7246
CX 0000 BP 0000 ES 19F5 HS 19F5 +4 6D6F OF DF IF SF ZF AF PF CF
DX 0060 SP 0400 SS 1A05 FS 19F5 +6 7220 0 0 1 0 0 0 1 0

CMD >

0088 C3 RET
00EC B409 MOV AH,09
00EE CD21 INT 21
00F0 B20A MOV DL,0A
00F2 B402 MOV AH,02
00F4 CD21 INT 21
00F6 B20D MOV DL,0D
00F8 B402 MOV AH,02
00FA CD21 INT 21

DS:0000 00 00 46 72 6F 6D 20 72
DS:0008 65 67 69 73 74 65 72 20
DS:0010 74 6F 20 73 74 72 69 6E
DS:0018 67 3A 20 24 46 72 6F 6D
DS:0020 20 73 74 72 69 6E 67 20
DS:0028 74 6F 20 72 65 67 69 73
DS:0030 74 65 72 20 5B 74 68 65
DS:0038 6E 20 66 6F 72 20 76 69
DS:0040 73 75 61 6C 20 66 72 6F
DS:0048 6D 20 72 65 67 69 73 74

2 0 1 2 3 4 5 6 7 8 9 A B C D E F
DS:0050 65 72 20 74 6F 20 73 74 72 69 6E 67 5D 3A 20 24 er to st ringl: $
DS:0060 2D 31 30 30 30 30 30 30 30 30 30 30 30 30 30 30 -1000000 00000000
DS:0070 30 24 30 30 30 30 30 30 30 30 30 30 30 30 30 30 0$000000 00000000
DS:0080 30 30 30 00 00 00 00 00 00 00 00 00 00 00 00 00 000.....
DS:0090 EB 01 90 2B FF BB 00 80 3D 00 00 7D 0C C6 85 60 δ.É+ η.ç =...}.|à'

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 ↑ 8 ↓ 9 ◀ 10 ▶

```

Был совершен вызов процедуры BIN\_TO\_DEC. Значение в ах обнулилось.

```

DOSBox 0.74-3-2, Cpu speed: 3000 cycles, Frameskip 0, Program: AFDPRO
AX 0000 SI 0011 CS 1A4E IP 008F Stack +0 0108 Flags 7244
BX 0000 DI 0001 DS 1A45 +2 1A4E
CX 0000 BP 0000 ES 19F5 HS 19F5 +4 0000 OF DF IF SF ZF AF PF CF
DX 001C SP 03FC SS 1A05 FS 19F5 +6 7246 0 0 1 0 1 0 1 0

CMD >

008D 2BC0 SUB AX,AX
008F 2BF6 SUB SI,SI
0091 80BC60002D CMP [0060+SI],2D
0096 7506 JNZ 009E
0098 2EC6068C0001 MOV CS:[008C],01
009E 46 INC SI
009F 80BC600024 CMP [0060+SI],24
00A4 75F8 JNZ 009E
00A6 83FE11 CMP SI,0011

DS:0000 00 00 46 72 6F 6D 20 72
DS:0008 65 67 69 73 74 65 72 20
DS:0010 74 6F 20 73 74 72 69 6E
DS:0018 67 3A 20 24 46 72 6F 6D
DS:0020 20 73 74 72 69 6E 67 20
DS:0028 74 6F 20 72 65 67 69 73
DS:0030 74 65 72 20 5B 74 68 65
DS:0038 6E 20 66 6F 72 20 76 69
DS:0040 73 75 61 6C 20 66 72 6F
DS:0048 6D 20 72 65 67 69 73 74

2 0 1 2 3 4 5 6 7 8 9 A B C D E F
DS:0050 65 72 20 74 6F 20 73 74 72 69 6E 67 5D 3A 20 24 er to st ringl: $
DS:0060 2D 31 30 30 30 30 30 30 30 30 30 30 30 30 30 -1000000 00000000
DS:0070 30 24 30 30 30 30 30 30 30 30 30 30 30 30 30 0$000000 00000000
DS:0080 30 30 30 00 00 00 00 00 00 00 00 00 00 00 00 000.....
DS:0090 EB 01 90 2B FF BB 00 80 3D 00 00 7D 0C C6 85 60 δ.É+ ȳ.Ş =...}.|à

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 ↑ 8 ↓ 9 ◀ 10 ▶

```

После выхода из процедуры имеем начальное значение ax.

```

DOSBox 0.74-3-2, Cpu speed: 3000 cycles, Frameskip 0, Program: AFDPRO
AX 8000 SI 0011 CS 1A4E IP 0108 Stack +0 0000 Flags 7244
BX 0000 DI 0001 DS 1A45 +2 7246
CX 0000 BP 0000 ES 19F5 HS 19F5 +4 6D6F OF DF IF SF ZF AF PF CF
DX 001C SP 0400 SS 1A05 FS 19F5 +6 7220 0 0 1 0 1 0 1 0

CMD >

00D9 CB RET Far
0108 E8F5FE CALL 0000
010B B409 MOV AH,09
010D CD21 INT 21
010F B44C MOV AH,4C
0111 CD21 INT 21
0113 0000 ADD [BX+SI],AL
0115 0000 ADD [BX+SI],AL
0117 0000 ADD [BX+SI],AL

DS:0000 00 00 46 72 6F 6D 20 72
DS:0008 65 67 69 73 74 65 72 20
DS:0010 74 6F 20 73 74 72 69 6E
DS:0018 67 3A 20 24 46 72 6F 6D
DS:0020 20 73 74 72 69 6E 67 20
DS:0028 74 6F 20 72 65 67 69 73
DS:0030 74 65 72 20 5B 74 68 65
DS:0038 6E 20 66 6F 72 20 76 69
DS:0040 73 75 61 6C 20 66 72 6F
DS:0048 6D 20 72 65 67 69 73 74

2 0 1 2 3 4 5 6 7 8 9 A B C D E F
DS:0050 65 72 20 74 6F 20 73 74 72 69 6E 67 5D 3A 20 24 er to st ringl: $
DS:0060 2D 31 30 30 30 30 30 30 30 30 30 30 30 30 30 -1000000 00000000
DS:0070 30 24 30 30 30 30 30 30 30 30 30 30 30 30 30 0$000000 00000000
DS:0080 30 30 30 00 00 00 00 00 00 00 00 00 00 00 00 000.....
DS:0090 EB 01 90 2B FF BB 00 80 3D 00 00 7D 0C C6 85 60 δ.É+ ȳ.Ş =...}.|à

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 ↑ 8 ↓ 9 ◀ 10 ▶

```



### **Выводы.**

В ходе выполнения работы были использованы арифметические операции над целыми числами и процедуры в ассемблере, разработаны 2 процедуры по преобразованию числа в строку в двоичном виде и обратно.

### **ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ**

**Название файла: PROG.ASM**

```
STACKSG SEGMENT  PARA STACK 'Stack'  
        DW        512 DUP(?)  
STACKSG      ENDS
```

```
DATASG  SEGMENT  PARA 'Data'  
        KEEP_CS DW 0 ;  
        MESSAGE1 DB 'From register to string: $'
```

```

        MESSAGE2 DB 'From string to register [then for visual from register to
string]: $'
        STRING DB 35 DUP('0')
DATASG      ENDS

```

```

CODE SEGMENT; SEG CODE
ASSUME  DS:DataSG, CS:Code, SS:STACKSG

```

```

DEC_TO_BIN PROC NEAR
        JMP START

```

```

; ax - 16 бит, записано в доп. коде (первый бит - знак, далее число)
; di - индекс текущего символа строки
; bx - маска для числа, с ее помощью достаем цифру, первоначальное значение
- 8000h (16 бит - 1)
; Отрицательное число записано в доп. коде, в этом случае необходимо
инвертировать число и прибавить 1

```

```

START:
        SUB di, di                ; Обнуляем di
        MOV bx, 8000h            ; Кладем маску в bx
        CMP ax, 0                ; Сравниваем число с 0
        JGE POSITIVE_NUMBER      ; Если dx >= 0, то число положительное
                                   ; Если dx < 0, то число отрицательное

        MOV STRING[di], '-'
        INC di
        NOT ax
        INC ax                    ; Перевод доп. кода в прямой код
        JMP FIND_SIGNIFICANT

```

```

POSITIVE_NUMBER:
        MOV STRING[di], '+'
        INC di
        JMP FIND_SIGNIFICANT

```

```

END_REGISTER:

```

```

SUB si, si
CMP STRING[si], '-'
JNE NULL_REGISTER

MOV cx, 15
INC si
MOV STRING[si], '1'

OVERFLOW_REGISTER:
MOV si, cx
INC si
MOV STRING[si], '0'
LOOP OVERFLOW_REGISTER
MOV si, 16

NULL_REGISTER:
MOV STRING[si], '0'
INC si

MOV STRING[si], '$'
JMP END_CONVERT

FIND_SIGNIFICANT:
SHR bx, 1                                ; Сдвигаем маску вправо
CMP bx, 0                                ; Сравниваем маску с 0
JE END_REGISTER                          ; Если она 0, то мы полностью прошли
регистр                                  ; (В этом случае либо число 0, либо слишком
                                         большое)
MOV cx, bx                                ; Иначе выдергиваем бит
AND cx, ax
CMP cx, 0                                ; И сравниваем его с 0
JE FIND_SIGNIFICANT                      ; Если он все еще 0, то продолжаем искать
значений бит дальше

MATCHED_DIGIT:
CMP cx, 0                                ; Смотрим, что находится в бите
JE ZERO_DIGIT                            ; Если 0, то записываем 0 в строку

```

```

        MOV STRING[di], '1'           ; Иначе записывает в строку 1
        INC di
        JMP NEXT_DIGIT

ZERO_DIGIT:                               ; Если соответствующий бит равен нулю
        MOV STRING[di], '0'
        INC di

NEXT_DIGIT:
        SHR bx, 1                     ; Сдвигаем маску вправо
        CMP bx, 0                     ; Сравниваем маску с 0
        JE LOOP_END                   ; Если маска 0, то мы прошли полностью
регистр
        MOV cx, bx
        AND cx, ax                     ; Иначе выдергиваем бит
        JMP MATCHED_DIGIT

LOOP_END:                               ; Регистр пройден до конца
        MOV STRING[di], '$'           ; добавляем в конец строки символ конца
строки

END_CONVERT:
        MOV dx, offset STRING         ; Записываем в dx сдвиг строки
        ret
DEC_TO_BIN ENDP

BIN_TO_DEC PROC FAR
        JMP START_B2D
        SIGN_NUMBER DB 0; Отвечает за знак числа

; STRING - Строка, которую будем переводить в число
; ax - Результативный регистр

START_B2D:
        SUB ax, ax                     ; Обнуляем ax
        SUB si, si                     ; Обнуляем si (индекс текущего элемента
строки)

```

```

        CMP STRING[si], '-'           ; Сравниваем первый элемент строки с
минусом
        JNE LENGTH_LOOP              ; Если не равен минусу, то число
положительное

                                         ; Если равен, то число отрицательное
        MOV SIGN_NUMBER, 1

LENGTH_LOOP:                          ; Подсчет длины строки
        INC si
        CMP STRING[si], '$'          ; Сравниваем элемент строки с $
        JNE LENGTH_LOOP              ; Если не равен $ то возвращаемся в цикл

        CMP si, 11h                  ; Сравниваем длину строки с 17
                                         ; Если длина строки 17 (знак + 16), то
это последнее отрицательное число
        JGE MAX_NEGATIVE_NUMBER
        CMP si, 1                    ; Если длина строки 1, то это ноль
        JE BINARY_END

        SUB di, di                   ; Начинаем рассматривать регистр ax
        MOV bx, 1                    ; Битмаска

TRAVERSE_LOOP:
        DEC si
        CMP si, 0                    ; Сравниваем si с 0
        JE CHECK_NEGATIVE            ; Все символы пройдены, завершение

        CMP STRING[si], '1'          ; Сравниваем элемент в строке в индексе
si с единицей
        JNE FORM_BITMASK             ; Если не равно единице

        OR ax, bx                    ; В ax на месте в том, котором стоит
единица в маске BX, ставится 1

FORM_BITMASK:
        SHL bx, 1                    ; Сдвигаем битмаску влево
        JMP TRAVERSE_LOOP

```

```

MAX_NEGATIVE_NUMBER:
    MOV ax, 8000h
    JMP BINARY_END

CHECK_NEGATIVE:
    CMP SIGN_NUMBER, 0
    JE BINARY_END          ; Если число положительное, то завершение
                             программы
                             ; Если число отрицательное, записываем в
                             доп. коде
                             ; Инвертируем и добавляем единицу

    NOT ax;
    INC ax;

BINARY_END:
    ret

BIN_TO_DEC ENDP

Main PROC FAR
    MOV ax, DATASG
    MOV ds, ax

    MOV dx, offset MESSAGE1
    MOV ah, 09h
    INT 21h

    MOV ax, 0FFFFh
    CALL DEC_TO_BIN
    MOV ah, 09h;
    INT 21h;

    MOV dl, 10
    MOV ah, 02h
    INT 21h
    MOV dl, 13
    MOV ah, 02h

```

```

        INT 21h

MOV dx, offset MESSAGE2
MOV ah, 09h;
        INT 21h;

        CALL BIN_TO_DEC
        CALL DEC_TO_BIN

MOV ah, 09h
        INT 21h

MOV ah, 4Ch
        INT 21h
Main      ENDP
CODE      ENDS
END Main

```