计算机组成原理第七次理论作业

3.1

5730

3.2

5730

3.13

Step	Action	Multipicand	Product/Multiplier
0	Initial Vals	0110 0010	0000 0000 0001 0010
1	nop Rshif Mplier	0110 0010	0000 0000 0001 0010 0000 0000 0000 1001
2	Prod = Prod + Mcand Rshift Mplier	0110 0010	0110 0010 0000 1001 0011 0001 0000 0100
3	nop Rshif Mplier	0110 0010	0011 0001 0000 0100 0001 1000 1000 0010
4	nop Rshif Mplier	0110 0010	0001 1000 1000 0010 0000 1100 0100 0001
5	Prod = Prod + Mcand Rshift Mplier	0110 0010	0110 1110 0100 0001 0011 0111 0010 0000
6	nop Rshif Mplier	0110 0010	0011 0111 0010 0000 0001 1011 1001 0000
7	nop Rshif Mplier	0110 0010	0001 1011 1001 0000 0000 1101 1100 1000

Step	Action	Multipicand	Product/Multiplier
8	nop Rshif Mplier	0110 0010	0000 1101 1100 1000 0000 0110 1110 0100

即: $62H \times 12H = 6E4H$

3.18

3.18: 74/21 = 3 remainder 9

Step	Action	Quotient	Divisor	Remainder
			2 1	7 4
0	Init Valc	000 000	010 001 000 000	000 000 111 100
1	Rem=Rem-Div	000 000	010 001 000 000	101 111 111 100
	Rem<0, R+D, Q<<	000 000	010 001 000 000	000 000 111 100
	Rchift Div	000 000	001 000 100 000	000 000 111 100
2.	Rem=Rem-Div	000 000	001 000 100 000	111 000 011 100
	Rem<0, R+D, Q<<	000 000	001 000 100 000	000 000 111 100
	Rchift Div	000 000	000 100 010 000	000 000 111 100
	Rem=Rem-Div	000 000	000 100 010 000	111 100 101 100
3.	Rem<0, R+D, Q<<	000 000	000 100 010 000	000 000 111 100
	Rchift Div	000 000	000 010 001 000	000 000 111 100
	Rem=Rem-Div	000 000	~~000 010 001 000~~	111 110 110 100
4	Rem<0, R+D, Q<<	000 000		~~000 000 111 100~~
	Rshift Div	000 000	000 001 000 100	
	Rem=Rem-Div			111 111 111 000
5	Rem<0, R+D, Q<<	000 000	000 001 000 100	~~000 000 111 100~~
	Rchift Div		000 000 100 010	
	Rem = Rem-Div	000 000	~~000 000 100 010~~	
6.	Rem>0, Q<<1	~~000 001~~		000 000 011 0
	Rchift Div		000 000 010 001	
7.	Rem=Rem-Div	000 001	000 000 010 00	
	Rem>0, Q<<1	000 011		000 000 001
	Rshift Div		000 000 001 000	

3.20

无论是有符号补码整数还是无符号整数其表示的都为 201326592

3.22

0x0C000000 = 0000 1100 0000 0000 0000 0000 0000 0000

若其表示单精度浮点数，那么：

S = 0
E = 0001 1000 = 24 - 127 = -103
F = 1 + 0000 0000 0000 0000 0000 00 = 1

即其表示的数为 $(-1)^0 \times 2^{-103} \times 1 = 2^{-103}$

3.23

63.35 = 111111.01 = 1.1111101 * 2^5

则：

S = 0
E = 5 + 127 = 132 = 1000 0100
F = 1111101

故IEEE标准下的位模式为：0 1000 0100 1111 1010 0000 0000 0000 000

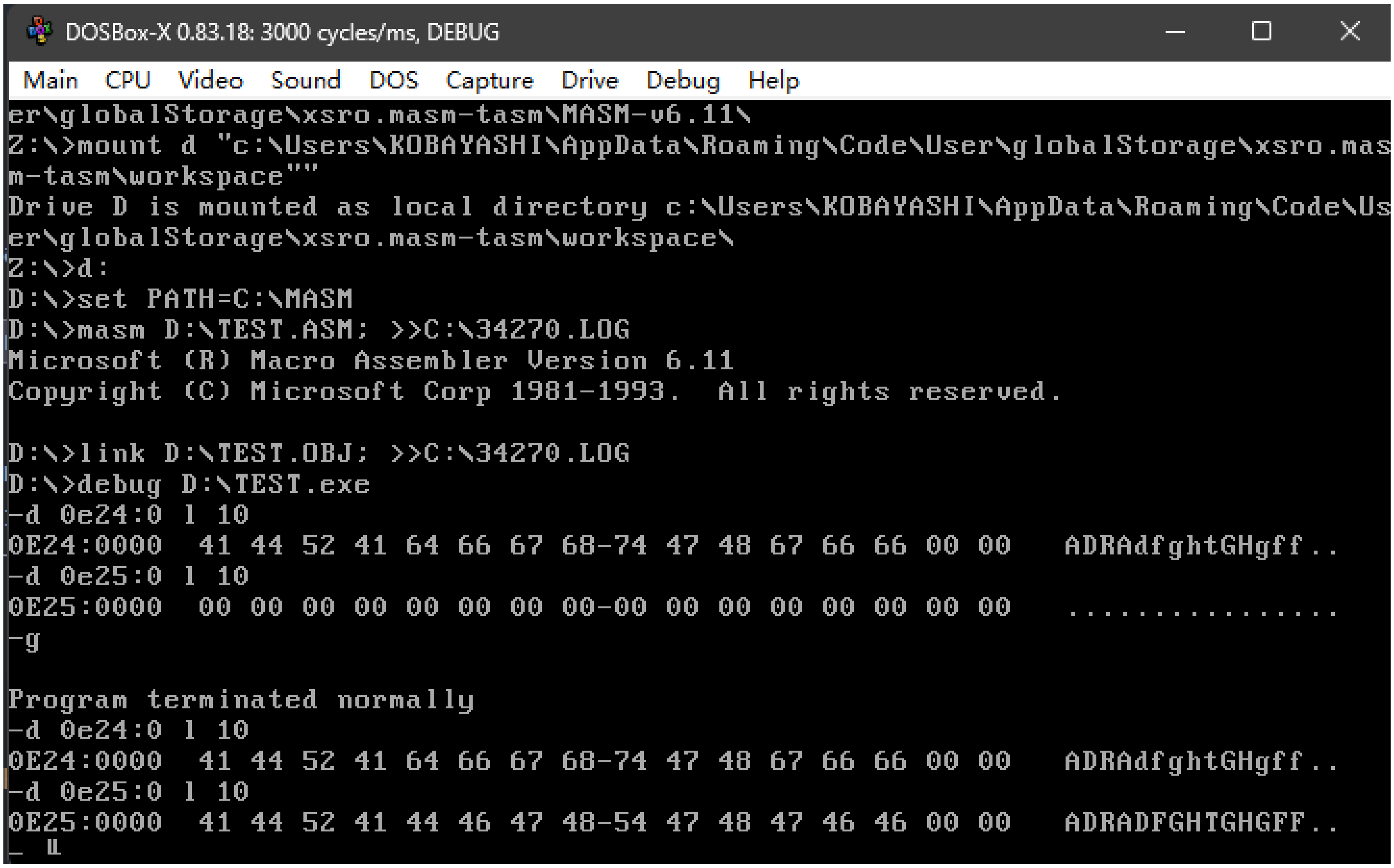
运行下列8086程序，分析该程序实现什么功能？截屏显示结果。

```
assume cs:code, ds:data, es:extra

DATA SEGMENT
string db 'ADRAdfghtGHgff'
count equ $-string
DATA ENDS

EXTRA SEGMENT
dest db count dup (?)
EXTRA ENDS

CODE SEGMENT
begin:

    mov ax, data
    mov ds, ax
    mov ax, extra
    mov es, ax

    mov cx, count
    lea si, string
    lea di, dest
    cld

again:
    lodsb
    and al, 0DFH
    stosb
    loop again

    mov ah, 4CH
    int 21H
CODE ENDS
end begin
```

分析：

该程序将存于DS段中的字符串string转为大写并复制到了ES段的dest中。

运行截图：

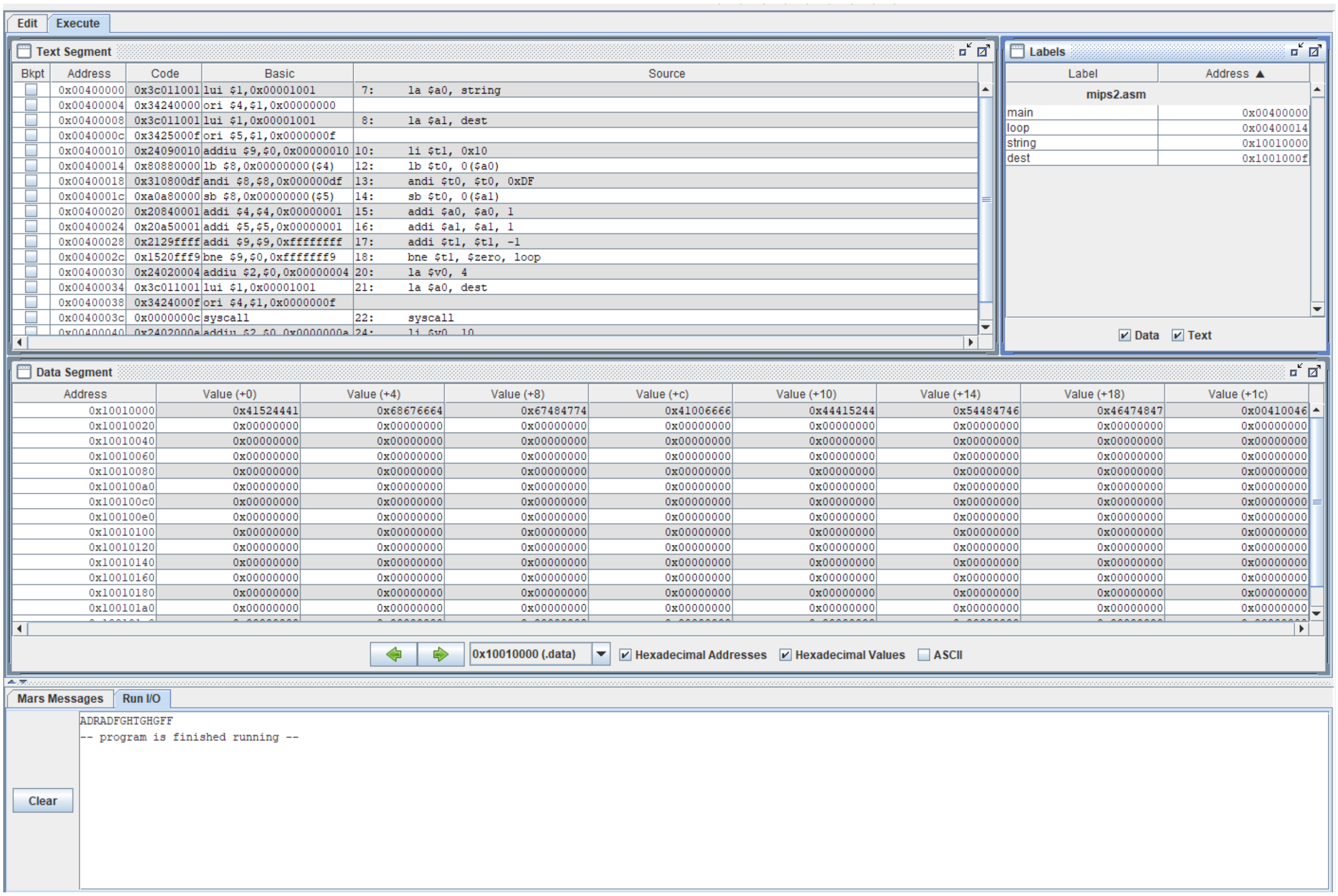


```
er\globalStorage\xsro.masm-tasm\MASM-v6.11\
Z:\>mount d "c:\Users\KOBAYASHI\AppData\Roaming\Code\User\globalStorage\xsro.mas
m-tasm\workspace""
Drive D is mounted as local directory c:\Users\KOBAYASHI\AppData\Roaming\Code\Us
er\globalStorage\xsro.masm-tasm\workspace\
Z:\>d:
D:\>set PATH=C:\MASM
D:\>masm D:\TEST.ASM; >>C:\34270.LOG
Microsoft (R) Macro Assembler Version 6.11
Copyright (C) Microsoft Corp 1981-1993.  All rights reserved.

D:\>link D:\TEST.OBJ; >>C:\34270.LOG
D:\>debug D:\TEST.exe
-d 0e24:0 l 10
0E24:0000  41 44 52 41 64 66 67 68-74 47 48 67 66 66 00 00   ADRAdfghtGHgff..
-d 0e25:0 l 10
0E25:0000  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-g

Program terminated normally
-d 0e24:0 l 10
0E24:0000  41 44 52 41 64 66 67 68-74 47 48 67 66 66 00 00   ADRAdfghtGHgff..
-d 0e25:0 l 10
0E25:0000  41 44 52 41 44 46 47 48-54 47 48 47 46 46 00 00   ADRADFGHTGHGFF..
_ ╙
```

编写一个MIPS汇编程序实现上述功能，运行，并截屏显示结果

```
.data
string: .asciiz "ADRAdfghtGHgff"
dest: .space 16

.text
main:
    la $a0, string
    la $a1, dest

    li $t1, 0x10
loop:
    lb $t0, 0($a0)
    andi $t0, $t0, 0xDF
    sb $t0, 0($a1)
    addi $a0, $a0, 1
    addi $a1, $a1, 1
    addi $t1, $t1, -1
    bne $t1, $zero, loop

    la $v0, 4
    la $a0, dest
    syscall

    li $v0, 10
    syscall
```

运行截图：



Bkpt	Address	Code	Basic	Source
□	0x00400000	0x3c011001	lui $1,0x00001001	7: la $a0, string
□	0x00400004	0x34240000	ori $4,$1,0x00000000	
□	0x00400008	0x3c011001	lui $1,0x00001001	8: la $a1, dest
□	0x0040000c	0x3425000f	ori $5,$1,0x0000000f	
□	0x00400010	0x24090010	addiu $9,$0,0x00000010	10: li $t1, 0x10
□	0x00400014	0x80880000	lb $8,0x00000000($4)	12: lb $t0, 0($a0)
□	0x00400018	0x310800df	andi $8,$8,0x000000df	13: andi $t0, $t0, 0xDF
□	0x0040001c	0xa0a80000	sb $8,0x00000000($5)	14: sb $t0, 0($a1)
□	0x00400020	0x20840001	addi $4,$4,0x00000001	15: addi $a0, $a0, 1
□	0x00400024	0x20a50001	addi $5,$5,0x00000001	16: addi $a1, $a1, 1
□	0x00400028	0x2129ffff	addi $9,$9,0xffffffff	17: addi $t1, $t1, -1
□	0x0040002c	0x1520fff9	bne $9,$0,0xfffffff9	18: bne $t1, $zero, loop
□	0x00400030	0x24020004	addiu $2,$0,0x00000004	20: la $v0, 4
□	0x00400034	0x3c011001	lui $1,0x00001001	21: la $a0, dest
□	0x00400038	0x3424000f	ori $4,$1,0x0000000f	
□	0x0040003c	0x0000000c	syscall	22: syscall
□	0x00400040	0x2402000a	addiu $2,$0,0x0000000a	24: li $v0, 10

Label	Address ▲
mips2.asm	
main	0x00400000
loop	0x00400014
string	0x10010000
dest	0x1001000f

☑ Data ☑ Text

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x41524441	0x68676664	0x67484774	0x41006666	0x44415244	0x54484746	0x46474847	0x00410046
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

```
ADRADFGHTGHGFF
-- program is finished running --
```