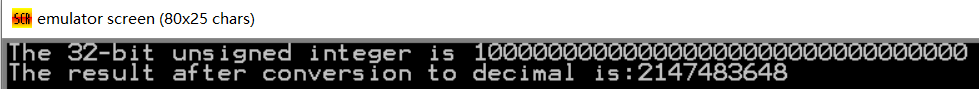
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓 名 | 胡聿鑫 | 学 号 | 2220192813 | 专业班级 | 计科3班 |
| 课程名称 | 微机原理与汇编语言 | | | 日 期 | 2021.11.6 |
| 作业序号 | **作业1** | | | 成 绩 |  |

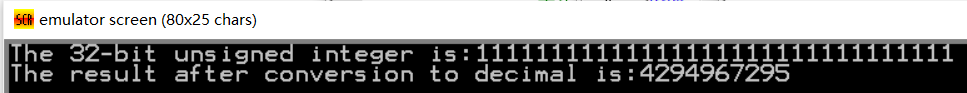
程序设计思路

首先给定任意一个32位无符号整数，将其存储到NUM中。将NUM的地址送入SI，NUM的内容送到AX与DX中。由于寄存器一次只能传送16位数，因此需要分两次将数的高、低16位分别送入。最后将二进制数转换成十进制数输出。

需要注意的是，要输出一个32位的无符号整数，需要进行32位除法的运算，要着重避免溢出问题的发生。具体做法是：首先让数的低16位入栈，除法运算的商和余数分别存储在AX和DX中，由于后续还需要用AX存储低16位的商，因此需先将高16位的商存储到BX中进行保护，然后在AX出栈时即可将低16位送入AX。结束上述步骤后，让高16位入栈，由于DX中存有除法运算的余数，但又需要它存储高16位的数，因此可以将DX存储的余数送CX，然后再出栈，即可将数的高16位送入DX。

程序运行结果





实验总结

32位无符号整数的十进制输出，不同于16位，涉及到了更多寄存器之间的传值，因此刚开始有些无从下手。32位的数必须要用AX和DX两个寄存器分别存储数的低、高16位，因此首要问题是如何将数送进这两个寄存器。由于定义时采用双字变量定义NUM，因此可以采用强制类型转换，将其拆分为两个字变量，低16位使用指令MOV AX, WORD PTR NUM[0]，高16位使用指令MOV DX, WORD PTR NUM[2]，从而完成NUM内容向寄存器的传递。

在进行32位除法运算时，由于被除数有32位，而除数为10，若直接进行除法则商必然会超过16位，发生溢出，因此需要分两次进行除法运算，即先运算高16位，再运算低16位。无论是对高位还是低位运算，都需要借助堆栈，先将数送入堆栈，在完成除法运算后在出栈时将数送到目标寄存器。为了简化子程序的内容，除法运算的子程序DIVDW只需完成一次除法运算即可，而循环部分可以在主程序中编写，即多次调用除法运算子程序DIVDW直到商为0.

此次实验使得我能够利用汇编程序输出任意的32位无符号整数，也加深了对于各个寄存器使用方法和时机的理解，对子程序的运用也更加娴熟。

附录（程序完整代码）

.model small

.stack

.data

NUM DD FFFFFFFFH

INF1 DB "The 32-bit unsigned integer is:"

DB "11111111111111111111111111111111",0AH,0DH,'$'

INF2 DB "The result after conversion to decimal is:$"

OBUF DB 11 DUP(0)

.code

DSTRING MACRO STRING

PUSH DX

PUSH AX

MOV DX, OFFSET STRING

MOV AH, 09H

INT 21H

POP AX

POP DX

ENDM

start:

MOV AX, data

MOV DS, AX

DSTRING INF1

MOV AX, WORD PTR NUM[0];NUM的低16位送入AX

MOV DX, WORD PTR NUM[2];NUM的高16位送入DX

MOV BX, OFFSET OBUF+10

MOV BYTE PTR [BX],'$'

AGAIN:

MOV CX,10;为除法运算(DX):(AX)/10 事先存入除数

CALL DIVDW

ADD CX,30H;将CX中的十进制数转换为ASCII码

DEC BX

MOV [BX],CL

CMP AX,0

JNZ AGAIN;商不为0则再次进行除法运算

DSTRING INF2

MOV DX,BX

MOV AH,09H

INT 21H

MOV AH,4CH

INT 21H

;进行32位除法运算（防止溢出）

DIVDW PROC

;先对高16位进行除法运算

PUSH BX;保护BX存储的内容

PUSH AX;低16位入栈

MOV AX,DX

MOV DX,0

DIV CX;商在AX,余数在DX

MOV BX,AX;BX保存高16位商

POP AX;低16位送入AX

;对低16位进行除法运算

PUSH BX;高16位入栈

DIV CX

MOV CX,DX;DX中保存的余数送CX

POP DX;此时DX存储数的高16位

POP BX

RET

ENDP DIVDW

END start