

南开大学

Dec2hex

(汇编语言与逆向技术实验 2)



姓名: 申宗尚

学号: 2213924

专业: 信息安全

一. 实验目的

- 1、熟悉汇编语言的数据传送、寻址和算术运算；
- 2、熟悉汇编语言过程的定义和使用；
- 3、熟悉十进制和十六进制的数制转换

二. 实验环境

MASM32 编译环境

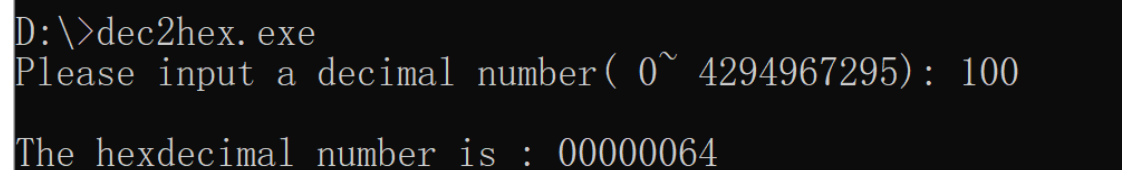
Windows 命令行窗口

三. 实验内容

编写汇编程序 `dec2hex.asm`，编译成 `dec2hex.exe`。`dec2hex.exe` 的功能是将 Windows 命令行输入的十进制无符号整数，转换成对应的十六进制整数，输出在 Windows 命令行中，如图 1 所示。

输入的十进制无符号整数的范围是 **0 到 4294967295** ($2^{32}-1$)。

输出对应的十六进制整数，对应的范围是 **00000000h 到 FFFFFFFFh**。



```
D:\>dec2hex.exe
Please input a decimal number( 0~ 4294967295): 100
The hexadecimal number is : 00000064
```

图 1. `dec2hex.exe` 将十进制 100 转换成十六进制 00000064

使用 `StdIn` 函数获得用户输入的十进制整数。`StdIn` 函数的定义在 `\masm32\include\masm32.inc`，库文件是 `\masm32\lib\masm32.lib`。`StdIn` 函数的定义 “`StdIn PROTO :DWORD,:DWORD`”，有两个参数，第一个是内存存储空间的起始地址，第二个是内存存储空间的大小。函数的例子：

```
.data
buf BYTE 20 DUP(0)
.code
invoke StdIn, addr buf, 20
invoke StdOut, addr buf
```

3.2 用户输入的十进制数对应的 ASCII 编码字符串存储在内存中，编写过程 `dec2dw`，将 ASCII 字符串转换成 `DWORD` 数据。例如，将字符串 “100” 转换成 `DWORD` 数据 00000064h。

3.3 编写过程 `dw2hex`，将 `DWORD` 数据转换成十六进制数的 ASCII 字符串。

例如，将 DWORD 数据 00000064h 转换成 ASCII 字符串“00000064”

使用 StdOut 函数在 Windows 命令函中输出十六进制整数的 ASCII 字符串。

StdOut 函数的定义在\masm32\include\masm32.inc，库文件是

\masm32\lib\masm32.lib。StdOut 函数的定义“StdOut PROTO :DWORD”，只有一个参数，是内存存储空间的起始地址。函数使用的例子同 StdIn 函数的例子。

使用 ml 将 dec2hex.asm 文件汇编到 dec2hex.obj 目标文件，编译命令：

“\masm32\bin\ml /c /coff dec2hex.asm”

使用 link 将目标文件 dec2hex.obj 链接成 dec2hex.exe 可执行文件，链接命令：

“\masm32\bin\link /SUBSYSTEM: CONSOLE dec2hex.obj”

四．实验代码及分段解析

.386

.model flat, stdcall

option casemap :none

include \masm32\include\windows.inc

include \masm32\include\kernel32.inc

include \masm32\include\user32.inc

include \masm32\include\masm32.inc

includelib \masm32\lib\kernel32.lib

includelib \masm32\lib\masm32.lib

.data

tmp DWORD 0

inp BYTE 20 DUP(0)

value BYTE 0

inputstr BYTE “请输入数字”, 0Dh, 0Ah, 0

strsize =(\$-inputstr)

.code

start:

invoke StdOut, addr inputstr

invoke StdIn, addr inp, 20

mov eax, 0

mov ebx, 0

mov ecx, 0

dec2dw:

mov bl, [inp+ecx]

sub bl, 48

imul eax, 10

```

    add al,bl
    inc ecx
    mov bl,[inp+ecx]
    cmp bl,0
    jg dec2dw

    mov tmp,eax
    push 0
dw2hex:
    mov ebx,tmp
    and bx,15.
    add bl,48
    cmp bl,58
    jl isnum
    add bl,7
isnum:
    mov eax,tmp
    shr eax,4
    mov tmp,eax
    push bx
    cmp tmp,0
    jg dw2hex

cout:
    pop bx
    cmp bx,0
    je exit
    mov value,bl
    invoke StdOut, addr value
    cmp ebx,0
    jg cout
exit:
    invoke ExitProcess, 0

END start

```

以下是对以上代码的分段解析：

```

.386
.model flat, stdcall
option casemap :none

include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
include \masm32\include\user32.inc

```

```
include \masm32\include\masm32.inc
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\masm32.lib
```

这部分是程序的头部，选择了指令集和必要的库与头文件。
.386 声明使用 80386 或更高版本的指令集。
.model flat, stdcall 使用平坦模型和 stdcall 调用约定。
option casemap :none 大小写不敏感
include 和 includelib 几句 包含库、头文件

```
.data
tmp DWORD 0
inp BYTE 20 DUP(0)
value BYTE 0
inputstr BYTE "请输入数字", 0Dh, 0Ah, 0
strsize =($-inputstr)
```

数据段定义了程序中会用到的各种变量
tmp 是一个 DWORD 类型的 32 位整数，初始值为 0。
inp 是一个包含 20 个字节的数组，用于存储输入的字符串。
inputstr 是提示消息。
value 是一个 BYTE 类型变量，初始值为 0。
strsize 的值存储字符串 inputstr 的长度。

```
.code
start:
    invoke StdOut, addr inputstr
    invoke StdIn, addr inp, 20
    mov eax, 0
    mov ebx, 0
    mov ecx, 0
```

代码段是程序执行的开始
先用 invoke 调用 StdOut 函数输出提示字符串
再调用 StdIn 函数获取输入的字符串
然后清零 eax, ebx, ecx 寄存器，供后续使用

```
dec2dw:
    mov bl, [inp+ecx]
    sub bl, 48
    imul eax, 10
    add al, bl
```

```

inc ecx
mov bl, [inp+ecx]
cmp bl, 0
jg dec2dw

```

这段是 dec2dw 过程，将输入的字符串 inp 变成十进制整数储存在 al 寄存器中
 先将 inp 中的一个字符存在 b1 寄存器中
 再将其-48 将字符转为数字
 将 eax 中已有的数字乘 10，实现左移一个十进制位
 再将 b1 中的数字加入 eax 中
 递增 ecx 寄存器，实现按位读取字符串
 最后通过 cmp bl 0 判断字符串是否结束 未结束则 dec2dw 过程继续

```

dw2hex:
mov ebx, tmp
and bx, 15.
add bl, 48
cmp bl, 58
jl isnum
add bl, 7
isnum:
mov eax, tmp
shr eax, 4
mov tmp, eax
push bx
cmp tmp, 0
jg dw2hex

```

这段代码通过循环和位运算，将存储的十进制数字转化为十六进制字符串
 将 ebx 与 15 按位与操作，从而获得最低四位数字（也就是二进制转化为十六进制的一个转换单位）
 将此数字与 48 相加，与 58 相比，如果大于等于 58，说明该数字是字符，直接将其加 7 转化为对应字符存入 b1
 如果在 48 与 58 中间，则跳转到 isnum 过程
 先恢复 tmp 中的值，再右移 eax，以处理下一个 4 位，同时更新 tmp 的值
 再将当前的四位数压入 b1 存储
 检查 tmp 是否已经为 0，如果仍有未处理完，则继续进行 dw2hex 过程，直到所有数字都被转换后压入 b1 中存储。

```

cout:
pop bx
cmp bx, 0
je exit

```

```

mov value,bl
invoke StdOut, addr value
cmp ebx,0
jg cout

```

这段是十六进制字符串的输出过程，在输出循环中，先弹出 bx 栈中的四位十六进制值，再检查是否已经全部处理完，如果已经处理完，直接进入 exit 过程再将当前的四位值存入 value 中，使用 StdOut 函数输出字符最后检查是否 ebx 已经为空，如果还有未处理的，则继续进行 cout 过程

```

exit:
    invoke ExitProcess, 0

```

```

END start

```

这是程序的结束部分，调用了 ExitProcess 函数，终止程序的进行。

以下是对以上代码的编译、链接过程的代码和解析：

首先，使用 ml 将 dec2hex.asm 文件汇编到 dec2hex.obj 目标文件，编译命令：

“\masm32\bin\ml /c /coff dec2hex.asm”

```

C:\Users\KKkai>\masm32\bin\ml /c /coff C:\Users\KKkai\Desktop\dec2hex.asm
Microsoft (R) Macro Assembler Version 6.14.8444
Copyright (C) Microsoft Corp 1981-1997. All rights reserved.

Assembling: C:\Users\KKkai\Desktop\dec2hex.asm

*****
ASCII build
*****

```

“\masm32\bin\ml”代表打开盘中 masm32 中 bin 文件夹的 ml 应用程序。

/c 代表仅进行编译，不自动进行链接。

/coff 代表产生的 obj.文件格式为 COFF 格式。

C:\Users\KKkai\Desktop\dec2hex.asm 是.asm 文件的地址

然后，使用 link 将目标文件 dec2hex.obj 链接成 dec2hex.exe 可执行文件，链接命令：“\masm32\bin\link /SUBSYSTEM: CONSOLE dec2hex.obj”

```
C:\Users\KKkai>\masm32\bin\link /SUBSYSTEM:CONSOLE C:\Users\KKkai\dec2hex.obj
Microsoft (R) Incremental Linker Version 5.12.8078
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.


C:\Users\KKkai>
```

\masm32\bin\link 代表打开盘中 masm32 中 bin 文件夹的 link 应用程序

SUBSYSTEM:CONSOLE:具体设置哪个程序入口点由连接器的“/subsystem:”选项参数确定，它告诉操作系统如何运行编译生成的.exe 文件。可以指定四种方式：“CONSOLE|WINDOWS|NATIVE|POSIX”

C:\Users\KKkai\dec2hex.obj 操作对象文件名

五. 实验程序测试

 命令提示符

```
Microsoft Windows [版本 10.0.19045.3448]
(c) Microsoft Corporation。保留所有权利。

C:\Users\KKkai>C:\Users\KKkai\dec2hex.exe
请输入数字
64
40
C:\Users\KKkai>C:\Users\KKkai\dec2hex.exe
请输入数字
100
64
C:\Users\KKkai>C:\Users\KKkai\dec2hex.exe
请输入数字
11111
2B67
C:\Users\KKkai>C:\Users\KKkai\dec2hex.exe
请输入数字
0
0
C:\Users\KKkai>C:\Users\KKkai\dec2hex.exe
请输入数字
4294967295
FFFFFFFF
C:\Users\KKkai>_
```

如图，测试了输入十进制无符号整数，范围是 0 到 4294967295 ($2^{32}-1$)。

输出对应的十六进制整数，对应的范围是 00000000h 到 FFFFFFFFh。

经过测试，该程序可以实现十进制转换为十六进制的功能。