实验三 分支控制实验

1. 实验目的

- 了解分支控制流程
- 识别汇编码中的分支跳转条件，并了解如何修改分支条件
- 理解跳转表的原理

2.实验内容

2.1 do_loop

准备工作:

首先我阅读实验背景，编写do_loop.c文件。在main函数中使用scanf()函数读取输入流，再调用do_loop()函数。一开始我对于"从stdin中接收输入参数"有点不理解，后来通过搜索理解了stdin的含义。

实验过程和解答

1. x=2,y=4000,k=3,观察寄存器的值。
    - 执行cltd前%edx的值是4000
    - 刚执行完cltd后%edx的值是0
    - 执行指令idiv后%edx的值又变为1。这是因为在有符号除法中，32位运算商送回EAX，余数在EDX。在第一次循环中，用y%k，y=4000，k=3,计算得到的商是1333，余数为1，所以执行指令idiv后，%eax的值变为1333，而%edx的值变为1。
2. x=2,y=40000,k=3,观察寄存器的值。
    - 执行cltd前%edx的值是-25536
    - 刚执行完cltd后%edx的值是-1
    - 执行指令idiv后%edx的值又变为0。这是因为在有符号除法中，32位运算的商送回EAX,余数在EDX.通过观察寄存器可知实际上的被除数是-25536而不是40000，根据y%k,y=-25536,k=3,这是可以整除的，商为-8512,所以执行指令idiv后，%eax的值变为-8512,而%edx的值变为0，因为整除余数为0.
3. cltd指令的作用

使用gdb观察得到在第一题中，被除数为4000，执行指令cltd后，%edx的值变为0，机器数为0x0；在第二题中，被除数为-25536，执行指令cltd后，%edx的值变为-1，机器数为0xffffffff。由此推断，cltd指令的操作结果与%eax中被除数的符号有关，应该是将%eax中被除数的符号位扩展到32位并存放在%edx中。

为了检验这个推论，我查询了《深入理解计算机系统》（第三版），在书上找到了这样的解释：

指令	效果	描述
imulq S	R[%rdx]：R[%rax]←S×R[%rax]	有符号全乘法
mulq S	R[%rdx]：R[%rax]←S×R[%rax]	无符号全乘法
clto	R[%rdx]：R[%rax]←符号扩展(R[%rax])	转换为八字

o是八字的后缀，d是四子的后缀，由此看出指令cltd的作用也是符号扩展，转换为四字。

心得体会

1.通过这个实验我认识到，查看寄存器在gdb调试中的重要作用，它帮助我更好地理解栈的概念。

2.通过比较第一题和第二题，第二题中虽然一开始从输入流中读入y=40000，但是存储在寄存器中的y=-25536，这是由于short类型是2字节，有符号的范围是-2^(15)~2^(15)-1,即-32768~32767，因为40000超过了max值，所以造成了溢出，所以y的值为40000-2^(16)=-25536.这启示我了解计算机位级表示的重要性，在编写代码时对于数据的大小要选择声明合适的数据类型。

2.2 if_else

实验过程

见if_else_A.s和if_else_B.s

分析汇编代码可以得出返回条件是

$$\begin{cases} 0(x > 0\text{且}\, y <= 29) \\ 1(x > 0\text{且}\, y > 30) \\ 2(x <= 0\text{或}\, x > 0, 29 < y <= 30) \end{cases}$$

然后根据题目要求修改.s文件。

注：（B）中我的学号后4位为03 26

心得体会

在看汇编代码时，跳转条件很重要，要看清跳转条件。

2.3 switch

实验内容

1. 观察返回值

n	3	6	9	12	13	14
返回值	4	15	12	53	13	28

2. 填写完整switchCase的代码

```
int switchCase(int n){
    int result = 0;
    switch(n){
```

```
        case 3:
            result = (n>>1);
            break;
        case 6:
            result = 2*n-3;
            break;
        case 8:
        case 9:
            result = (n>>2)+1;
            break;
        case 10:
        case 12:
            result = 3*n+5;
            break;
        case 13:
            result = ((n>>31)+n)/2-6;
            break;
        default:
            result = n;
    }
    result = result+n;
    return result;
}
```

注：因为题目要求中写的是运算包括加减乘除和移位运算，因此在这里我保留了大部分移位运算，并没有将之改为整除。根据csapp我们可以知道，算术右移x>>k相当于[x/(2^k)]，因此可以转换为除法。

实验中遇到的困难和心得体会

在这个实验中，我一开始不是特别理解switch分支的跳转表格，许多汇编指令也是看过就忘。后来我仔细揣摩了书上和ppt上有关switch的部分，遇到不会的汇编指令也都通过搜索的方式搞清楚了。通过这个实验，我对于汇编语言的理解大大加深了，我深刻地意识到许多理论光看书是难以铭记的，只有在实验的过程中加以运用，才能更快的理解和掌握。我特别感谢在gdb调试中用si指令一步一步观察寄存器值的变化的过程，我努力使自己弄懂每一行汇编代码的含义，这一过程虽然痛苦，却也对我的理解产生了很大帮助。同时，我还了解了地址对于底层语言的重要性，它就像是一个唯一的标识，引导机器一步一步往下执行。

在完成switch语句的填写过程中，我尽力弄懂汇编代码的含义，最后为了确认结果的正确性，我自己写了一个.c文件，把完善后的switchCase代码放进去，验证得到了和观察寄存器一样的返回值。