第五章作业

5.1　简述 VC2010 的编译选项“大小最小化”和“速度最大化”分别追求的目标。并简要说明编译器所采取的若干做法。

大小最小化：使用尽可能短的指令以及尽可能简短的算法来执行编译。
速度最大化：使用尽可能快的算法来执行编译。
例如，寄存器数据清零，

```
MOV EAX, 0; MOV 指令更短
XOR EAX, EAX; 逻辑运算指令执行更快
```

5.2　基于 VC2010 的集成开发环境，请说明“结构体类型变量”的尺寸往往大于其“各成员”尺寸之和的原因。

数据对齐，例如，在 4 字节对齐的系统中，结构体类型变量会被对齐到 4 字节的倍数。其中如有 1 字节的 char 型数据类型就会引起数据对齐。

5.3　基于 VC2010 的集成开发环境，请分别举例说明指针变量和引用的本质。

指针变量就是地址变量，32 位程序中，指针长度 4 字节，即 32 位，存储虚拟地址。
对指针的改变意味着对本身存储的地址进行改变，将指针指向别处。
引用对象的别名，一旦初始化绑定到具体的实例对象就不再指向别处。
对引用的修改就是对实例对象的修改。

5.4　基于 VC2010 集成开发环境，编写一个实现如下功能的程序：由用户从键盘输入一个十进制正整数，然后输出其各个因子。其他要求：输入和输出只能采用字符串形式（只能使用格式符“%s”）；应该设计合适的函数。请在编译选项“大小最小化”和“速度最大化”的情况下，生成汇编格式的目标代码，并对照源程序分析目标代码。

C 程序代码：

```
#include <stdio.h>
#include <stdlib.h>

#define MAX_LENGTH 20

int main()
{
    char str_in[MAX_LENGTH], str_out[MAX_LENGTH][10];
    unsigned int num = 0, factors_count = 0;
    puts("ex51_c\n 请输入一个十进制数：");
    gets_s(str_in);
```

```
    num = atoi(str_in);             // 字符串->数值
    for (int i=num; i > 0; i--)
    {
        if (num%i == 0)
        {
            itoa(i, str_out[factors_count++], 10);       // 数值->字符串
        }
    }

    puts("该十进制数值的因子为：");
    for (unsigned int i = 0; i < factors_count; i++)
    {
        printf("%s ", str_out[i]);
    }
    printf("\n");

    return 0;
}
```

编译优化：大小最小化（节选核心部分）——

```
; 29    :  num = atoi(str_in);

    lea  eax, DWORD PTR _str_in$[ebp]     // 准备入口参数：取得变量 str_in 数组的有效地址
    push     eax
    call  DWORD PTR __imp__atoi           // 调用 EXTERN 的库函数 atoi
    add esp, 16                           ; 00000010H
    movDWORD PTR _num$1$[ebp], eax   // 出口参数送入 num

; 30    :  for (int i=num; i > 0; i--)

    movedi, eax       // edi 即 int i, eax 即 num
    test eax, eax       // 先判断 num 是否为 0，若是，直接不执行循环
    jle   SHORT $LN3@main  // 负责跳出循环，开始显示结果

; 24    :  char str_in[MAX_LENGTH], str_out[MAX_LENGTH][10];
; 25    :  unsigned int num = 0, factors_count = 0;

    push     ebx
    lea  ebx, DWORD PTR _str_out$[ebp]
$LL4@main:

; 31    :  {
```

```
; 32    :        if (num%i == 0)

      xor  edx, edx       // edx:eax÷edi，此处高 32 位 edx 清零
      div  edi
      test edx, edx       // 判断余数 num%i 是否为 0
      jne  SHORT $LN2@main  // 不为 0，即非因子，则准备下一次循环

; 33    :        {
; 34    :              itoa(i, str_out[factors_count++], 10);

      push      10                      ; 0000000aH
      push      ebx
      push      edi
      call  DWORD PTR __imp__itoa       // 为 0，即因子，则调用 EXTERN 库函数 itoa
      add esp, 12                       ; 0000000cH
      inc   esi
      add ebx, 10                       ; 0000000aH
$LN2@main:

; 30    : for (int i=num; i > 0; i--)

      mov eax, DWORD PTR _num$1$[ebp]   // 新一轮循环的准备(赋初值)
      dec edi         // 新一轮循环的准备（i--）
      test edi, edi
      jg    SHORT $LL4@main
      pop ebx
```

编译优化：速度最大化（节选核心部分）——

```
; 29    : num = atoi(str_in);

      lea   eax, DWORD PTR _str_in$[ebp]     // 准备入口参数：取得变量 str_in 数组的有效地址
      push      eax
      call  DWORD PTR __imp__atoi           // 调用 EXTERN 的库函数 atoi
      add esp, 16                       ; 00000010H
      mov DWORD PTR _num$1$[ebp], eax   // 出口参数送入 num

; 30    : for (int i=num; i > 0; i--)

      mov edi, eax        // edi 即 int i, eax 即 num
      test eax, eax       // 先判断 num 是否为 0，若是，直接不执行循环
      jle   SHORT $LN3@main  // 负责跳出循环，开始显示结果
```

```
; 24    : char str_in[MAX_LENGTH], str_out[MAX_LENGTH][10];
; 25    : unsigned int num = 0, factors_count = 0;

    push    ebx
    lea ebx, DWORD PTR _str_out$[ebp]
$LL4@main:

; 31    : {
; 32    :     if (num%i == 0)

    xor edx, edx        // edx:eax÷edi，此处高 32 位 edx 清零
    div edi
    test edx, edx       // 判断余数 num%i 是否为 0
    jne SHORT $LN2@main  // 不为 0，即非因子，则准备下一次循环

; 33    :     {
; 34    :         itoa(i, str_out[factors_count++], 10);

    push    10                      ; 0000000aH
    push    ebx
    push    edi
    call DWORD PTR __imp__itoa      // 为 0，即因子，则调用 EXTERN 库函数 itoa
    add esp, 12                     ; 0000000cH
    inc esi
    add ebx, 10                     ; 0000000aH
$LN2@main:

; 30    : for (int i=num; i > 0; i--)

    mov eax, DWORD PTR _num$1$[ebp]   // 新一轮循环的准备(赋初值)
    dec edi         // 新一轮循环的准备（i--）
    test edi, edi
    jg  SHORT $LL4@main
    pop ebx
```