Homework 5 & Homework 6

郭天魁
信息科学技术学院
1300012790

October 19, 2014

1 Homework 5

1.1 3.61

```
1  int fix_var_prod_ele(int n, int A[n][n], int B[n][n], int i, int k) {
2          register int *Aptr = A[i];
3          register int *Bptr = &B[n - 1][k];
4          register int j = n;
5          register int result = 0;
6          register int N = n;
7          while(--j != -1) {
8                  result += Aptr[j] * (*Bptr);
9                  Bptr -= N;
10         }
11         return result;
12 }
```

编译后，循环体中指令如下：

```
1  .L3:
2          movl    (%esi,%edx,4), %ebx
3          imull   (%ecx), %ebx
4          addl    %ebx, %eax
5          addl    %edi, %ecx
6          subl    $1, %edx
7          cmpl    $-1, %edx
8          jne     .L3
```

将循环变量j由从0到$n-1$循环替换为从$n-1$到0循环，可以避免在循环体中使用值n。

需要注意的是必须使用gcc的-O1编译选项。如果使用-O0，则imull指令中会用到%edx和%eax两个临时寄存器变量，导致使用的五个值至少有一个需要被保存在内存中。

1.2 3.62

A. $M = 13$.

B. i: %edi, j: %ecx.

C.

```
1  void transpose(int M, int A[M][M]) {
2          int i, j;
3          int M4 = M << 2;
4          int eax, *ebx, *esi, edx;
5          for (i = 0; i < M; i++) {
6                  ebx = &A[0][i];
7                  esi = A[i];
8                  for (j = 0; j < i; j++) {
9                          eax = *ebx;
10                         edx = esi[j];
11                         esi[j] = eax;
12                         *ebx = edx;
13                         ebx = (void*)ebx + M4;
14                 }
15         }
16 }
```

1.3 3.64

A. 8(%ebp)是result的地址，12(%ebp)是s1.a，16(%ebp)是s1.p.

B. 从高到低依次为s2.diff, s2.sum, s1.p, s1.a, &s2.

C. 将参数结构体内的变量由低到高存入栈中。

D. 将返回值的结构体地址压栈，在函数中直接操作结构体。

2 Homework 6

2.1 3.67

A. e1.p: 0, e1.y: 4, e2.x: 0, e2.next: 4.

B. 8.

C.

```
1  up->e2.next->e1.y=*(up->e2.next->e1.p)-up->e2.x;
```

2.2 3.68

```
#define L 10

void good_echo(){
        char buf[L];
        char *p;
        while(fgets(buf, L, stdin)) {
                for(p = buf; *p; p++) {
                        putchar(*p);
                        if(*p == '\n') return;
                }
        }
}
```

2.3 3.70

A.

```
long traverse(tree_ptr tp) {
        long rax = 1LL << 63;
        long rbx, r12;
        if(tp) {
                rbx = tp->val;
                r12 = traverse(tp->left);
                rax = traverse(tp->right);
                rax = r12 >= rax ? r12 : rax;
                rax = rax < rbx ? rbx : rax;
        }
        return rax;
}
```

B. Find the maximum value in the binary tree whose root is tp.