**《编译技术》课程设计**

**测试报告**

## 测试程序一

### 测试目标

函数调用

### 1.2测试程序

int a, b, c;

int max(int x, int y) {

if (x >= y) {

return (x);

}

else {

return (y);

}

}

int min(int x, int y) {

if (x >= y) {

return (y);

}

else {

return (x);

}

}

int middle(int x, int y, int z) {

if (x <= max(y, z)) {

if(x >= min(y, z)) {

return (x);

}

else {

return (min(y, z));

}

}

else {

return (max(y, z));

}

}

void main() {

int res1, res2, res3;

printf("please input 3 numbers:");

scanf(a, b, c);

res1 = max(a, max(b, c));

res2 = middle(a, b, c);

res3 = min(a, min(b, c));

printf("max:", res1);

printf("middle:", res2);

printf("min:", res3);

}

### X86汇编指令

.386

.model flat

option casemap: none

includelib lib\msvcrt.lib

printf proto c

scanf proto c

.data

$a dword 0

$b dword 0

$c dword 0

@str byte "please input 3 numbers:", 0

@str1 byte "max:", 0

@str2 byte "middle:", 0

@str3 byte "min:", 0

@figure dword ?

@figure\_format\_d byte "%d", 0

@figure\_format\_c byte "%c", 0

@figure\_format\_f byte "%f", 0

@space byte 20h, 0

@new\_line byte 0dh, 0ah, 0

.code

\_main proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

sub esp, 32

lea eax, @str

push eax

call printf

add esp, 4

lea eax, @new\_line

push eax

call printf

add esp, 4

lea eax, @figure

push eax

lea eax, @figure\_format\_d

push eax

call scanf

add esp, 8

mov eax, @figure

mov $a, eax

lea eax, @figure

push eax

lea eax, @figure\_format\_d

push eax

call scanf

add esp, 8

mov eax, @figure

mov $b, eax

lea eax, @figure

push eax

lea eax, @figure\_format\_d

push eax

call scanf

add esp, 8

mov eax, @figure

mov $c, eax

push $c

push $b

call \_max

add esp, 8

push eax

push $a

call \_max

add esp, 8

mov edi, eax

push $c

push $b

push $a

call \_middle

add esp, 12

mov esi, eax

push $c

push $b

call \_min

add esp, 8

push eax

push $a

call \_min

add esp, 8

mov ebx, eax

lea eax, @str1

push eax

call printf

add esp, 4

push edi

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

lea eax, @str2

push eax

call printf

add esp, 4

push esi

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

lea eax, @str3

push eax

call printf

add esp, 4

push ebx

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

L8:

@main\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_main endp

\_max proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

mov eax, dword ptr [ebp + 20]

cmp eax, dword ptr [ebp + 24]

jl L1

mov dword ptr [ebp + 20], eax

mov eax, eax

jmp @max\_end

mov dword ptr [ebp + 20], eax

jmp L2

L1:

mov eax, dword ptr [ebp + 24]

jmp @max\_end

L2:

@max\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_max endp

\_middle proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

sub esp, 16

push dword ptr [ebp + 28]

push dword ptr [ebp + 24]

call \_max

add esp, 8

mov ecx, dword ptr [ebp + 20]

cmp ecx, eax

jg L6

push dword ptr [ebp + 28]

push dword ptr [ebp + 24]

call \_min

add esp, 8

mov ecx, dword ptr [ebp + 20]

cmp ecx, eax

jl L5

mov eax, dword ptr [ebp + 20]

jmp @middle\_end

jmp L7

L5:

push dword ptr [ebp + 28]

push dword ptr [ebp + 24]

call \_min

add esp, 8

mov dword ptr [ebp - 4], eax

mov eax, eax

jmp @middle\_end

jmp L7

L6:

push dword ptr [ebp + 28]

push dword ptr [ebp + 24]

call \_max

add esp, 8

mov dword ptr [ebp - 8], eax

mov eax, eax

jmp @middle\_end

L7:

@middle\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_middle endp

\_min proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

mov eax, dword ptr [ebp + 20]

cmp eax, dword ptr [ebp + 24]

jl L3

mov dword ptr [ebp + 20], eax

mov eax, dword ptr [ebp + 24]

jmp @min\_end

mov dword ptr [ebp + 20], eax

jmp L4

L3:

mov eax, dword ptr [ebp + 20]

jmp @min\_end

L4:

@min\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_min endp

\_start:

call \_main

ret

end \_start

### 1.4结果显示



## 测试程序二

### 2.1测试目标

循环语句

### 2.2 测试程序

void main()

{

int fahr,celsius;

int lower,upper,step;

lower=0;

upper=300;

step=20;

fahr=lower;

printf("while begin");

while(fahr<=upper)

{

printf(fahr);

fahr=fahr+step;

}

printf("while end");

printf("for begin");

for(fahr=300;fahr>=0;fahr=fahr - 20){

printf(fahr);

}

printf("for end");

}

### 2.3 X86汇编指令

.386

.model flat

option casemap: none

includelib lib\msvcrt.lib

printf proto c

scanf proto c

.data

@str byte "while statement", 0

@str1 byte "while statement", 0

@str2 byte "for statement", 0

@str3 byte "for statement", 0

@figure dword ?

@figure\_format\_d byte "%d", 0

@figure\_format\_c byte "%c", 0

@figure\_format\_f byte "%f", 0

@space byte 20h, 0

@new\_line byte 0dh, 0ah, 0

.code

\_main proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

sub esp, 28

mov edi, 0

mov esi, 300

mov ebx, 20

mov eax, edi

mov dword ptr [ebp - 8], eax

lea eax, @str

push eax

call printf

add esp, 4

lea eax, @new\_line

push eax

call printf

add esp, 4

L1:

mov eax, dword ptr [ebp - 8]

cmp eax, esi

jg L2

mov dword ptr [ebp - 8], eax

push dword ptr [ebp - 8]

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

mov eax, dword ptr [ebp - 8]

add eax, ebx

mov dword ptr [ebp - 8], eax

jmp L1

L2:

lea eax, @str1

push eax

call printf

add esp, 4

lea eax, @new\_line

push eax

call printf

add esp, 4

lea eax, @str2

push eax

call printf

add esp, 4

lea eax, @new\_line

push eax

call printf

add esp, 4

mov eax, 300

mov dword ptr [ebp - 8], eax

L3:

mov eax, dword ptr [ebp - 8]

cmp eax, 0

jl L4

mov dword ptr [ebp - 8], eax

push dword ptr [ebp - 8]

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

mov eax, dword ptr [ebp - 8]

sub eax, 20

mov dword ptr [ebp - 8], eax

jmp L3

L4:

lea eax, @str3

push eax

call printf

add esp, 4

lea eax, @new\_line

push eax

call printf

add esp, 4

L5:

@main\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_main endp

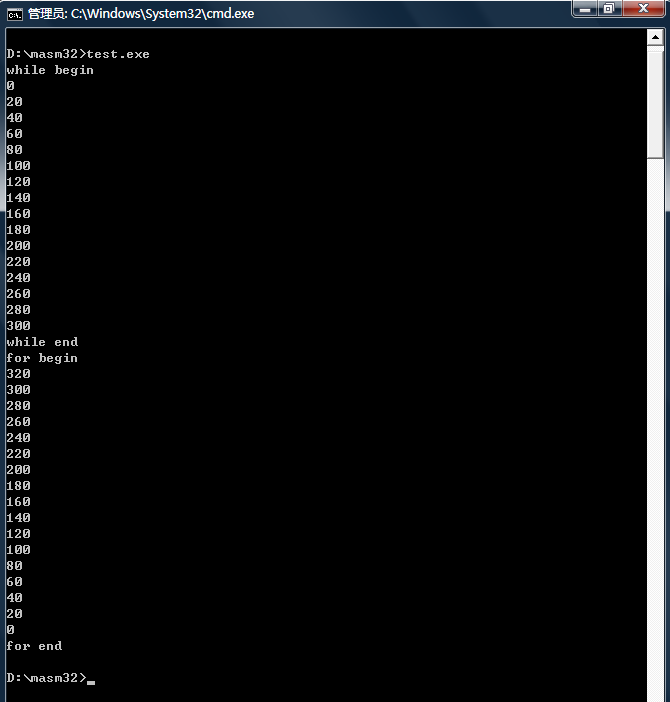
\_start:

call \_main

ret

end \_start

### 2.4结果显示



## 测试程序3

### 3.1测试目标

表达式计算与类型转换

### 3.2 测试程序

const int A = -8, B = 6;

int C;

char int\_to\_char(int n)

{

return (n + 48);

}

int char\_to\_int(char c)

{

return (c - 48);

}

void main()

{

const int pi = 3;

int m, n, x, y, p, q;

char c1, c2;

scanf(m, n);

x = 5;

y = -4;

c = 2;

c1 = int\_to\_char(m);

c2 = int\_to\_char(n);

printf(c1);

printf(c2);

printf(c1 + c2 - m - n);

printf((-3 + pi) \* y / c + (char\_to\_int(c2) - char\_to\_int(c1)));

printf((3 \* c2 + (c - a / y) - c1 \* pi) \* (b - x));

}

### 3.3 X86汇编指令

.386

.model flat

option casemap: none

includelib lib\msvcrt.lib

printf proto c

scanf proto c

.data

$a dword -8

$b dword 6

$c dword 0

@figure dword ?

@figure\_format\_d byte "%d", 0

@figure\_format\_c byte "%c", 0

@figure\_format\_f byte "%f", 0

@space byte 20h, 0

@new\_line byte 0dh, 0ah, 0

.code

\_char\_to\_int proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

sub esp, 4

mov eax, dword ptr [ebp + 20]

sub eax, 48

mov dword ptr [ebp - 4], eax

mov eax, eax

jmp @char\_to\_int\_end

L2:

@char\_to\_int\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_char\_to\_int endp

\_int\_to\_char proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

sub esp, 4

mov eax, dword ptr [ebp + 20]

add eax, 48

mov dword ptr [ebp - 4], eax

mov eax, eax

jmp @int\_to\_char\_end

L1:

@int\_to\_char\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_int\_to\_char endp

\_main proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

sub esp, 120

mov eax, 3

mov dword ptr [ebp - 36], eax

lea eax, @figure

push eax

lea eax, @figure\_format\_d

push eax

call scanf

add esp, 8

mov eax, @figure

mov dword ptr [ebp - 12], eax

lea eax, @figure

push eax

lea eax, @figure\_format\_d

push eax

call scanf

add esp, 8

mov eax, @figure

mov edi, eax

mov esi, 5

mov eax, 4

neg eax

mov ebx, eax

mov ecx, 2

push dword ptr [ebp - 12]

mov $c, ecx

call \_int\_to\_char

add esp, 4

push edi

mov dword ptr [ebp - 4], eax

call \_int\_to\_char

add esp, 4

mov dword ptr [ebp - 8], eax

push dword ptr [ebp - 4]

lea eax, @figure\_format\_c

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

push dword ptr [ebp - 8]

lea eax, @figure\_format\_c

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

mov eax, dword ptr [ebp - 4]

add eax, dword ptr [ebp - 8]

mov ecx, eax

sub ecx, dword ptr [ebp - 12]

mov eax, ecx

sub eax, edi

mov dword ptr [ebp - 40], eax

push eax

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

mov eax, 3

neg eax

mov ecx, eax

add ecx, 3

mov eax, ecx

imul eax, ebx

mov ecx, eax

xchg eax, ecx

mov dword ptr [ebp - 44], ecx

mov ecx, $c

cdq

idiv ecx

push dword ptr [ebp - 8]

mov dword ptr [ebp - 48], eax

call \_char\_to\_int

add esp, 4

push dword ptr [ebp - 4]

mov dword ptr [ebp - 52], eax

call \_char\_to\_int

add esp, 4

mov ecx, dword ptr [ebp - 52]

sub ecx, eax

mov eax, dword ptr [ebp - 48]

add eax, ecx

mov dword ptr [ebp - 56], eax

push eax

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

mov eax, 3

imul eax, dword ptr [ebp - 8]

mov ecx, $a

xchg eax, ecx

mov dword ptr [ebp - 60], ecx

mov ecx, ebx

cdq

idiv ecx

mov edx, $c

sub edx, eax

mov eax, dword ptr [ebp - 60]

add eax, edx

mov edx, dword ptr [ebp - 4]

imul edx, 3

mov dword ptr [ebp - 64], eax

sub eax, edx

mov edx, $b

sub edx, esi

mov dword ptr [ebp - 68], eax

imul eax, edx

mov dword ptr [ebp - 72], eax

push eax

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

L3:

@main\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_main endp

\_start:

call \_main

ret

end \_start

### 3.4结果显示



## 测试程序四

### 4.1测试目标

变量作用域测试

### 4.2 测试程序

int a, b;

int ab()

{

a = 2;

b = 4;

return (a + b);

}

void main()

{

int a, b;

a = 1;

b = 3;

printf(ab());

printf(a + b);

}

### 4.3 X86汇编指令

.386

.model flat

option casemap: none

includelib lib\msvcrt.lib

printf proto c

scanf proto c

.data

$a dword 0

$b dword 0

@figure dword ?

@figure\_format\_d byte "%d", 0

@figure\_format\_c byte "%c", 0

@figure\_format\_f byte "%f", 0

@space byte 20h, 0

@new\_line byte 0dh, 0ah, 0

.code

\_ab proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

sub esp, 4

mov eax, 2

mov ecx, 4

mov edx, eax

add edx, ecx

mov $a, eax

mov $b, ecx

mov dword ptr [ebp - 4], edx

mov eax, edx

jmp @ab\_end

mov $a, eax

mov $b, ecx

L1:

@ab\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_ab endp

\_main proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

sub esp, 16

mov esi, 1

mov ebx, 3

call \_ab

mov dword ptr [ebp - 12], eax

push eax

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

mov eax, esi

add eax, ebx

mov dword ptr [ebp - 16], eax

push eax

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

L2:

@main\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_main endp

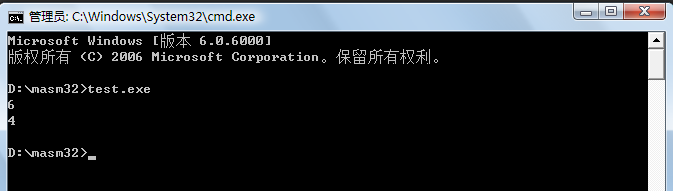
\_start:

call \_main

ret

end \_start

### 4.4结果显示



## 测试程序五

### 5.1测试目标

综合测试

### 5.2 测试程序

const int pi = 3;

void loop() {

int m;

printf("Loop number：");

scanf(m);

while(m > 0) {

printf(m);

m = m - 1;

}

printf(" ");

}

int prime(int n) {

if (n <= 1) {

return (1);

} else {

return (n \* prime(n - 1));

}

}

void cal\_prime() {

int n;

printf("Calculate n!, input n: ");

scanf(n);

printf(prime(n));

}

void main() {

loop();

cal\_prime();

}

### 5.3 X86汇编指令

.386

.model flat

option casemap: none

includelib lib\msvcrt.lib

printf proto c

scanf proto c

.data

$pi dword 3

@str byte "Calculate n!, input n: ", 0

@str1 byte "Loop number：", 0

@str2 byte " ", 0

@figure dword ?

@figure\_format\_d byte "%d", 0

@figure\_format\_c byte "%c", 0

@figure\_format\_f byte "%f", 0

@space byte 20h, 0

@new\_line byte 0dh, 0ah, 0

.code

\_cal\_prime proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

sub esp, 8

lea eax, @str

push eax

call printf

add esp, 4

lea eax, @new\_line

push eax

call printf

add esp, 4

lea eax, @figure

push eax

lea eax, @figure\_format\_d

push eax

call scanf

add esp, 8

mov eax, @figure

mov ebx, eax

push ebx

call \_prime

add esp, 4

mov dword ptr [ebp - 8], eax

push eax

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

L6:

@cal\_prime\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_cal\_prime endp

\_loop proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

sub esp, 8

lea eax, @str1

push eax

call printf

add esp, 4

lea eax, @new\_line

push eax

call printf

add esp, 4

lea eax, @figure

push eax

lea eax, @figure\_format\_d

push eax

call scanf

add esp, 8

mov eax, @figure

mov ebx, eax

L1:

cmp ebx, 0

jle L2

push ebx

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

mov eax, ebx

sub eax, 1

mov ebx, eax

jmp L1

L2:

lea eax, @str2

push eax

call printf

add esp, 4

lea eax, @new\_line

push eax

call printf

add esp, 4

L3:

@loop\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_loop endp

\_main proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

call \_loop

call \_cal\_prime

L7:

@main\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_main endp

\_prime proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

sub esp, 12

mov eax, dword ptr [ebp + 20]

cmp eax, 1

jg L4

mov dword ptr [ebp + 20], eax

mov eax, 1

jmp @prime\_end

mov dword ptr [ebp + 20], eax

jmp L5

L4:

mov eax, dword ptr [ebp + 20]

sub eax, 1

push eax

call \_prime

add esp, 4

mov ecx, dword ptr [ebp + 20]

imul ecx, eax

mov dword ptr [ebp - 4], ecx

mov eax, ecx

jmp @prime\_end

L5:

@prime\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_prime endp

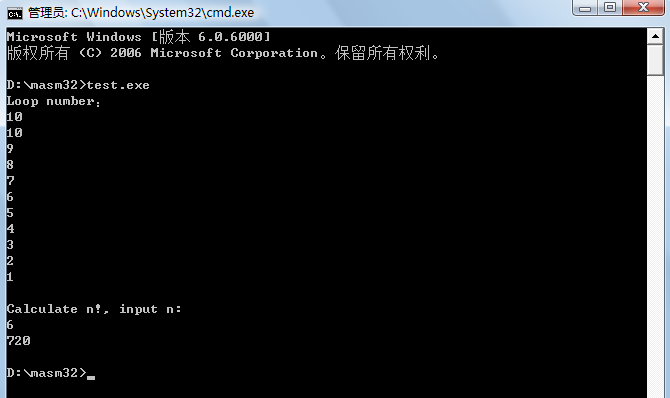
\_start:

call \_main

ret

end \_start

### 5.4结果显示



## 6．测试程序六

### 6.1测试目标

变量未定义和重复定义

### 6.2 测试程序

void main() {

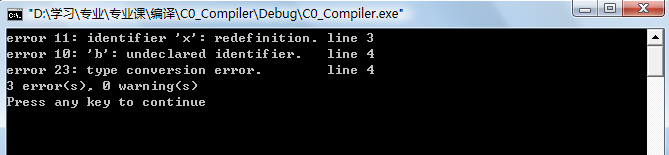
int a, x;

float x;

b = a;

}

### 6.3结果显示



## 7．测试程序七

### 7.1测试目标

函数未定义和重复定义

### 7.2 测试程序

int add(int a, int b) {

return (a + b);

}

int add(int c, int d) {

return (c + d);

}

void main() {

int m, n;

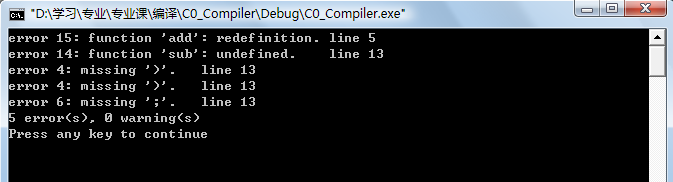
scanf(m, n);

printf(add(m, n));

printf(sub(m, n));

}

### 7.3结果显示



## 8．测试程序八

### 8.1测试目标

缺少字符

### 8.2 测试程序

void main()

int a, b;

char c, d

c = 'c';

d = 'd;

scanf(a b);

if a > b) {

printf(c;

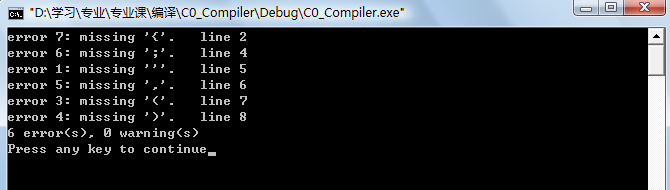
}

else

printf(d);

}

### 8.3 结果显示



## 9．测试程序九

### 9.1测试目标

函数返回值错误和函数调用错误

### 9.2 测试程序

int fun1(int a, float b, char c) {

return (a);

}

void fun2() {

int a;

a = 0;

return (a);

}

int fun3() {

float a;

return (a);

}

int fun4() {

}

void main() {

int a, b, c, d;

float f;

scanf(a, b, c, d);

printf(fun1(a, b));

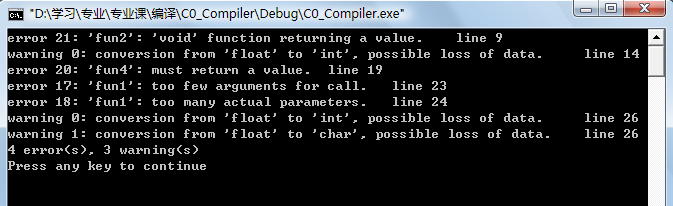
printf(fun1(a, b, c, d));

printf(fun1(a, b, c));

printf(fun1(f, f, f));

}

### 9.3结果显示



## 10．测试程序十

### 10.1测试目标

全局寄存器优化，即比较使用活跃变量分析-建立冲突图的方法来分配寄存器对代码的优化。

### 10.2 测试程序

void main() {

int x, y, i, a, b, z;

scanf(a, b);

x = a;

y = b;

i = 0;

while (i < 100) {

z = a \* 10;

x = x + y;

if (x < z) {

x = x - y;

}

y = y + 1;

i = i + 1;

}

printf(x);

printf(y);

printf(z);

}

### 10.3 X86汇编指令

优化关闭：

共产生14条内存访问指令

.386

.model flat

option casemap: none

includelib lib\msvcrt.lib

printf proto c

scanf proto c

.data

@figure dword ?

@figure\_format\_d byte "%d", 0

@figure\_format\_c byte "%c", 0

@figure\_format\_f byte "%f", 0

@space byte 20h, 0

@new\_line byte 0dh, 0ah, 0

.code

\_main proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

sub esp, 44

lea eax, @figure

push eax

lea eax, @figure\_format\_d

push eax

call scanf

add esp, 8

mov eax, @figure

mov ebx, eax

lea eax, @figure

push eax

lea eax, @figure\_format\_d

push eax

call scanf

add esp, 8

mov eax, @figure

mov esi, eax

mov eax, ebx

mov ecx, esi

mov edi, 0

mov dword ptr [ebp - 16], eax

mov dword ptr [ebp - 20], ecx

L2:

cmp edi, 100

jge L3

mov eax, ebx

imul eax, 10

mov ecx, dword ptr [ebp - 16]

add ecx, dword ptr [ebp - 20]

mov dword ptr [ebp - 24], eax

mov dword ptr [ebp - 16], ecx

cmp ecx, eax

jge L1

mov eax, dword ptr [ebp - 16]

sub eax, dword ptr [ebp - 20]

mov dword ptr [ebp - 16], eax

L1:

mov eax, dword ptr [ebp - 20]

add eax, 1

mov ecx, edi

add ecx, 1

mov edi, ecx

mov dword ptr [ebp - 20], eax

jmp L2

L3:

push dword ptr [ebp - 16]

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

push dword ptr [ebp - 20]

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

push dword ptr [ebp - 24]

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

L4:

@main\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_main endp

\_start:

call \_main

ret

end \_start

优化开启：

共产生10条内存访问指令

.386

.model flat

option casemap: none

includelib lib\msvcrt.lib

printf proto c

scanf proto c

.data

@figure dword ?

@figure\_format\_d byte "%d", 0

@figure\_format\_c byte "%c", 0

@figure\_format\_f byte "%f", 0

@space byte 20h, 0

@new\_line byte 0dh, 0ah, 0

.code

\_main proc

push ebx

push ebp

push esi

push edi

mov ebp, esp

sub esp, 44

lea eax, @figure

push eax

lea eax, @figure\_format\_d

push eax

call scanf

add esp, 8

mov eax, @figure

mov dword ptr [ebp - 4], eax

lea eax, @figure

push eax

lea eax, @figure\_format\_d

push eax

call scanf

add esp, 8

mov eax, @figure

mov dword ptr [ebp - 8], eax

mov edi, dword ptr [ebp - 4]

mov ebx, dword ptr [ebp - 8]

mov eax, 0

mov dword ptr [ebp - 12], eax

L2:

mov eax, dword ptr [ebp - 12]

cmp eax, 100

jge L3

mov dword ptr [ebp - 12], eax

mov eax, dword ptr [ebp - 4]

imul eax, 10

mov esi, eax

mov ecx, edi

add ecx, ebx

mov edi, ecx

cmp edi, esi

jge L1

mov eax, edi

sub eax, ebx

mov edi, eax

L1:

mov eax, ebx

add eax, 1

mov ebx, eax

mov ecx, dword ptr [ebp - 12]

add ecx, 1

mov dword ptr [ebp - 12], ecx

jmp L2

L3:

push edi

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

push ebx

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

push esi

lea eax, @figure\_format\_d

push eax

call printf

add esp, 8

lea eax, @new\_line

push eax

call printf

add esp, 4

L4:

@main\_end:

mov esp, ebp

pop edi

pop esi

pop ebp

pop ebx

ret

\_main endp

\_start:

call \_main

ret

end \_start

### 10.4结果显示

