**Computer Architecture II**

**Tutorial 1**

Ryan

Barron

16329561

1)Using the code generation method from the slides, this is what we get as a basic solution:

Caller code for min:

min(1, 2, 2);

00A927BE push 2

00A927C0 push 2

00A927C2 push 1

00A927C4 call min (0A91005h)

00A927C9 add esp,0Ch

Callee code for min:

int min(int a, int b, int c) {

00A926D0 push ebp

00A926D1 mov ebp,esp

00A926D3 sub esp,0CCh

00A926D9 push ebx

00A926DA push esi

00A926DB push edi

00A926DC lea edi,[ebp-0CCh]

00A926E2 mov ecx,33h

00A926E7 mov eax,0CCCCCCCCh

00A926EC rep stos dword ptr es:[edi]

int v = a;

00A926EE mov eax,dword ptr [a]

00A926F1 mov dword ptr [v],eax

if (b < v)

00A926F4 mov eax,dword ptr [b]

00A926F7 cmp eax,dword ptr [v]

00A926FA jge min+32h (0A92702h)

v = b;

00A926FC mov eax,dword ptr [b]

00A926FF mov dword ptr [v],eax

if (c < v)

00A92702 mov eax,dword ptr [c]

00A92705 cmp eax,dword ptr [v]

00A92708 jge min+40h (0A92710h)

v = c;

00A9270A mov eax,dword ptr [c]

00A9270D mov dword ptr [v],eax

return v;

00A92710 mov eax,dword ptr [v] }

00A92713 pop edi

00A92714 pop esi

00A92715 pop ebx

00A92716 mov esp,ebp

00A92718 pop ebp }

00A92719 ret

Caller code for p:

00A928EF push 0

00A928F1 push 3

00A928F3 push 2

00A928F5 push 1

00A928F7 push 0

00A928F9 call p (0A910D7h)

00A928FE add esp,10h

Callee code for p:

int p(int i, int j, int k, int l) {

00A92730 push ebp

00A92731 mov ebp,esp

00A92733 sub esp,0C0h

00A92739 push ebx

00A9273A push esi

00A9273B push edi

00A9273C lea edi,[ebp-0C0h]

00A92742 mov ecx,30h

00A92747 mov eax,0CCCCCCCCh

00A9274C rep stos dword ptr es:[edi]

return min(min(g, i, j), k, l);

00A9274E mov eax,dword ptr [l]

00A92751 push eax

00A92752 mov ecx,dword ptr [k]

00A92755 push ecx

00A92756 mov edx,dword ptr [j]

00A92759 push edx

00A9275A mov eax,dword ptr [i]

00A9275D push eax

00A9275E mov ecx,dword ptr [g (0A9C000h)]

00A92764 push ecx

00A92765 call min (0A91005h)

00A9276A add esp,0Ch

00A9276D push eax

00A9276E call min (0A91005h)

00A92773 add esp,0Ch

}

00A92776 pop edi

00A92777 pop esi

00A92778 pop ebx

}

00A92779 add esp,0C0h

00A9277F cmp ebp,esp

00A92781 call \_\_RTC\_CheckEsp (0A91172h)

00A92786 mov esp,ebp

00A92788 pop ebp

00A92789 ret

Caller code for gcd:

00A9296F push 7

00A92971 push 15h

00A92973 push 0Eh

00A92975 call gcd (0A91032h)

00A9297A add esp,8

Callee code for gcd:

int gcd(int a, int b) {

00A92610 push ebp

00A92611 mov ebp,esp

00A92613 sub esp,0C0h

00A92619 push ebx

00A9261A push esi

00A9261B push edi

00A9261C lea edi,[ebp-0C0h]

00A92622 mov ecx,30h

00A92627 mov eax,0CCCCCCCCh

00A9262C rep stos dword ptr es:[edi]

if (b == 0) {

00A9262E cmp dword ptr [b],0

00A92632 jne gcd+2Bh (0A9263Bh)

return a;

00A92634 mov eax,dword ptr [a]

00A92637 jmp gcd+3Fh (0A9264Fh)

}

else {

00A92639 jmp gcd+3Fh (0A9264Fh)

return gcd(b, a % b);

00A9263B mov eax,dword ptr [a]

00A9263E cdq

00A9263F idiv eax,dword ptr [b]

00A92642 push edx

00A92643 mov eax,dword ptr [b]

00A92646 push eax

00A92647 call gcd (0A91032h)

00A9264C add esp,8

}

}

00A9264F pop edi

00A92650 pop esi

00A92651 pop ebx

00A92652 add esp,0C0h

00A92658 cmp ebp,esp

00A9265A call \_\_RTC\_CheckEsp (0A91172h)

00A9265F mov esp,ebp

00A92661 pop ebp

00A92662 ret

2)Stack

The maximum number of stack frames for gcd(14,21) is 4.

|  |  |
| --- | --- |
| First gcd call | b:21 |
| a:14 |
| Return address to main |
| Saved ebp of main |
| Second gcd call | b:14 |
| a:21 |
| Return address to first gcd call |
| Saved ebp of first gcd call |
| Third gcd call | b:7 |
| a:14 |
| Return address to second gcd call |
| Saved ebp of second gcd call |
| Fourth gcd call | b:0 |
| a:7 |
| Return address to third gcd call |
| Saved ebp of third gcd call |

3) The code:

My code works, proof:

t1.h:

extern "C" int g;

extern "C" int \_cdecl min(int, int, int); // \_cdecl calling convention

extern "C" int \_cdecl p(int, int, int, int); // \_cdecl calling convention

extern "C" int \_cdecl gcd(int, int);

t1.asm:

.686 ; create 32 bit code

.model flat, C ; 32 bit memory model

option casemap:none ; case sensitive

.data

public g

g DWORD 4

.code

public min

min: push ebp ; save old frame pointer

mov ebp, esp ; set frame pointer to current stack pointer, which is where the old frame pointer is

sub esp, 4 ; allocate memory for one 32bit variable (v)

mov eax, [ebp+8] ; load first parameter in eax (a)

mov ecx, [ebp+12] ; load second parameter in ecx (b)

mov edx, [ebp+16] ; load third parameter in edx (c)

mov [ebp-4], eax ; int v = a

cmp [ebp-4], ecx ; if (b < v)

jl min0 ;

mov [ebp-4], ecx ; v = b

min0: cmp [ebp-4], edx ; if (c < v)

jl min1 ;

mov [ebp-4], edx ; v = c

min1: mov eax, [ebp-4] ; a = v, pass result in eax

mov esp, ebp ; deallocate memory, restore stack pointer

pop ebp ; restore old frame pointer

ret 0 ; return from function, result in eax

public p

p: push ebp

mov ebp, esp

sub esp, 4

mov eax, [g]

mov ecx, [ebp+8]

mov edx, [ebp+12]

push edx

push ecx

push eax

call min

add esp, 12

mov ecx, [ebp+16]

mov edx, [ebp+20]

push edx

push ecx

push eax

call min

add esp, 12

mov esp, ebp

pop ebp

ret 0

public gcd

gcd: push ebp

mov ebp, esp

mov ecx, [ebp+12]

mov eax, [ebp+8]

cmp ecx, 0

jne gcd0

mov esp, ebp

pop ebp

ret 0

gcd0: cdq

idiv ecx

push edx

push ecx

call gcd

add esp, 8

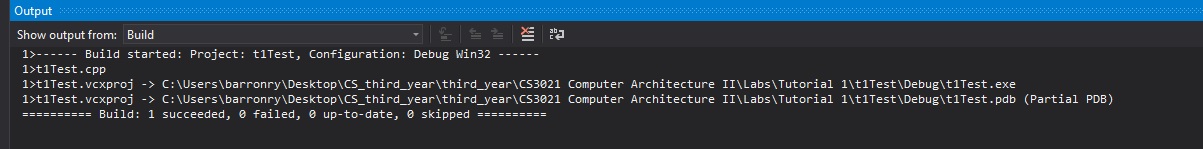
mov esp, ebp

pop ebp

ret 0

end

Successful build:



Console output: