// FactorialRM.c

#include <stdio.h>

int factorial(int n);

void DumpS(int n);

int main() {

int n, f;

printf("Enter an integer:");

scanf("%d",&n);

DumpS(64);

f = factorial(n);

DumpS(64);

printf("%d! is %d\n",n,f);

}

.file "FactorialRM.c"

.text

.globl main

.align 16, 0x90

.type main,@function

main: # @main

# BB#0:

pushl %ebp

movl %esp, %ebp

subl $40, %esp

leal .L.str, %eax

movl %eax, (%esp)

calll printf

leal .L.str1, %ecx

leal -4(%ebp), %edx

movl %ecx, (%esp)

movl %edx, 4(%esp)

movl %eax, -12(%ebp) # 4-byte Spill

calll scanf

movl $64, %ecx

movl $64, (%esp)

movl %eax, -16(%ebp) # 4-byte Spill

movl %ecx, -20(%ebp) # 4-byte Spill

calll DumpS

movl -4(%ebp), %eax

movl %eax, (%esp)

calll factorial

movl $64, %ecx

movl %eax, -8(%ebp)

movl $64, (%esp)

movl %ecx, -24(%ebp) # 4-byte Spill

calll DumpS

leal .L.str2, %eax

movl -4(%ebp), %ecx

movl -8(%ebp), %edx

movl %eax, (%esp)

movl %ecx, 4(%esp)

movl %edx, 8(%esp)

calll printf

movl $0, %ecx

movl %eax, -28(%ebp) # 4-byte Spill

movl %ecx, %eax

addl $40, %esp

popl %ebp

ret

.Ltmp0:

.size main, .Ltmp0-main

.type .L.str,@object # @.str

.section .rodata.str1.1,"aMS",@progbits,1

.L.str:

.asciz "Enter an integer:"

.size .L.str, 18

.type .L.str1,@object # @.str1

.L.str1:

.asciz "%d"

.size .L.str1, 3

.type .L.str2,@object # @.str2

.L.str2:

.asciz "%d! is %d\n"

.size .L.str2, 11

// FactorialRF.c

void DumpS(int n);

int factorial(int n) {

int result;

DumpS(64);

if(n <= 1) {

result = 1;

} else {

result = n \* factorial(n-1);

}

DumpS(64);

return result;

}

.file "FactorialRF.c"

.text

.globl factorial

.align 16, 0x90

.type factorial,@function

factorial: # @factorial

# BB#0:

pushl %ebp

movl %esp, %ebp

subl $24, %esp

movl 8(%ebp), %eax

movl $64, %ecx

movl %eax, -4(%ebp)

movl $64, (%esp)

movl %ecx, -12(%ebp) # 4-byte Spill

calll DumpS

cmpl $1, -4(%ebp)

jg .LBB0\_2

# BB#1:

movl $1, -8(%ebp)

jmp .LBB0\_3

.LBB0\_2:

movl -4(%ebp), %eax

movl -4(%ebp), %ecx

subl $1, %ecx

movl %ecx, (%esp)

movl %eax, -16(%ebp) # 4-byte Spill

calll factorial

movl -16(%ebp), %ecx # 4-byte Reload

imull %eax, %ecx

movl %ecx, -8(%ebp)

.LBB0\_3:

movl $64, %eax

movl $64, (%esp)

movl %eax, -20(%ebp) # 4-byte Spill

calll DumpS

movl -8(%ebp), %eax

addl $24, %esp

popl %ebp

ret

.Ltmp0:

.size factorial, .Ltmp0-factorial

.ident "FreeBSD clang version 3.4.1 (tags/RELEASE\_34/dot1-final 208032) 20140512"

.section ".note.GNU-stack","",@progbits

.data

Ra: .long 0

.long 0

Rb: .long 0

.long 0

Rc: .long 0

.long 0

Rd: .long 0

.long 0

FP: .long 0

.long 0

Curr: .long 0

.long 0

Last: .long 0

.long 0

Line: .ascii "------------------------------- FP=%x, %d bytes\n"

.asciz "(EAX=%11d),(EBX=%11d),(ECX=%11d),(EDX=%11d)\n"

Format: .asciz "%8x: %8x(%11d)\n"

AddrFt: .asciz "%8x: "

ValuFt: .asciz "%8x(%11d)"

ComaFt: .asciz ","

NLFt: .asciz "\n"

.text

.p2align 2

.globl DumpS

.type DumpS,@function

DumpS:

push $0

pop %ecx

movl %ebp,FP

movl %esp,Curr

movl %eax,Ra /\* Save register values \*/

movl %ebx,Rb

movl %ecx,Rc

movl %edx,Rd

push %ebp

movl %esp,%ebp

movl 8(%ebp),%ebx /\* The number of bytes to display \*/

movl %ebx,%ecx

movl Curr,%eax

addl %eax,%ebx /\* Compute the ending address \*/

movl %ebx,Last

movl FP,%eax /\* Print a header line \*/

addl $-4,%esp

movl Rd,%edx

push %edx

movl Rc,%edx

push %edx

movl Rb,%edx

push %edx

movl Ra,%edx

push %edx

push %ecx

push %eax

push $Line

call printf

addl $16,%esp

Loop: movl Curr,%eax /\* Iterate while Curr < Last \*/

movl Last,%ebx

cmpl %eax,%ebx

jle Done

movl 0(%eax),%ecx /\* Access memory for stact contents \*/

push %ecx /\* Print the stack element \*/

push %ecx

push %eax

push $Format /\* Address and value \*/

call printf

addl $16,%esp

movl Curr,%eax /\* Update Curr \*/

addl $4,%eax

movl %eax,Curr

jmp Loop

Done:

movl Rd,%edx /\* Restore register values \*/

movl Rc,%ecx

movl Rb,%ebx

movl Ra,%eax

leave

ret

Legend:

FP Address

Return Address

“Pushed Arguments”

Recursive Local Data

Global (mains)data

**Previous FP Addresses**

server1# cc -m32 FactorialRM.c FactorialRF.c DumpS32.s -o Factorial2016

server1# Factorial2016

Enter an integer:4

------------------------------- FP=ffffdac4, 64 bytes

(EAX= 1),(EBX= 1),(ECX= 0),(EDX= -9564)

ffffda98: 8048613( 134514195)

ffffda9c: 40( 64)

ffffdaa0: ffffdac0( -9536)

ffffdaa4: 2817a99f( 672639391)

ffffdaa8: 1( 1)

ffffdaac: 2804c560( 671401312)

ffffdab0: 40( 64)

ffffdab4: 1( 1)

ffffdab8: 11( 17)

ffffdabc: 206( 518)

ffffdac0: 4( 4)

ffffdac4: ffffdae8( -9496)

ffffdac8: 804851a( 134513946)

ffffdacc: 1( 1)

ffffdad0: ffffdb0c( -9460)

ffffdad4: ffffdb14( -9452)

------------------------------- FP=ffffda94, 64 bytes

(EAX= 4),(EBX= 1),(ECX= 0),(EDX= -9564)

ffffda78: 8048680( 134514304)

ffffda7c: 40( 64)

ffffda80: ffffdb14( -9452)

ffffda84: 1( 1)

ffffda88: 40( 64)

ffffda8c: ffffdaa4( -9564)

ffffda90: 4( 4)

ffffda94: **ffffdac4**( -9532)

ffffda98: 804861e( 134514206)

ffffda9c: 4( 4)

ffffdaa0: ffffdac0( -9536)

ffffdaa4: 2817a99f( 672639391)

ffffdaa8: 1( 1)

ffffdaac: 2804c560( 671401312)

ffffdab0: 40( 64)

ffffdab4: 1( 1)

------------------------------- FP=ffffda74, 64 bytes

(EAX= 3),(EBX= 1),(ECX= 0),(EDX= -9564)

ffffda58: 8048680( 134514304)

ffffda5c: 40( 64)

ffffda60: 1( 1)

ffffda64: 1( 1)

ffffda68: 40( 64)

ffffda6c: ffffdaa4( -9564)

ffffda70: 3( 3)

ffffda74: **ffffda94**( -9580)

ffffda78: 80486b0( 134514352)

ffffda7c: 3( 3)

ffffda80: ffffdb14( -9452)

ffffda84: 4( 4)

ffffda88: 40( 64)

ffffda8c: ffffdaa4( -9564)

ffffda90: 4( 4)

ffffda94: ffffdac4( -9532)

------------------------------- FP=ffffda54, 64 bytes

(EAX= 2),(EBX= 1),(ECX= 0),(EDX= -9564)

ffffda38: 8048680( 134514304)

ffffda3c: 40( 64)

ffffda40: ffffdac4( -9532)

ffffda44: 1( 1)

ffffda48: 40( 64)

ffffda4c: ffffdaa4( -9564)

ffffda50: 2( 2)

ffffda54: **ffffda74**( -9612)

ffffda58: 80486b0( 134514352)

ffffda5c: 2( 2)

ffffda60: 1( 1)

ffffda64: 3( 3)

ffffda68: 40( 64)

ffffda6c: ffffdaa4( -9564)

ffffda70: 3( 3)

ffffda74: ffffda94( -9580)

------------------------------- FP=ffffda34, 64 bytes

(EAX= 1),(EBX= 1),(ECX= 0),(EDX= -9564)

ffffda18: 8048680( 134514304)

ffffda1c: 40( 64)

ffffda20: ffffda94( -9580)

ffffda24: 1( 1)

ffffda28: 40( 64)

ffffda2c: ffffdaa4( -9564)

ffffda30: 1( 1)

ffffda34: **ffffda54**( -9644)

ffffda38: 80486b0( 134514352)

ffffda3c: 1( 1)

ffffda40: ffffdac4( -9532)

ffffda44: 2( 2)

ffffda48: 40( 64)

ffffda4c: ffffdaa4( -9564)

ffffda50: 2( 2)

ffffda54: ffffda74( -9612)

------------------------------- FP=ffffda34, 64 bytes

(EAX= 64),(EBX= 1),(ECX= 0),(EDX= -9564)

ffffda18: 80486cd( 134514381)

ffffda1c: 40( 64)

ffffda20: 40( 64)

ffffda24: 1( 1)

ffffda28: 40( 64)

ffffda2c: 1( 1)

ffffda30: 1( 1)

ffffda34: **ffffda54**( -9644)

ffffda38: 80486b0( 134514352)

ffffda3c: 1( 1)

ffffda40: ffffdac4( -9532)

ffffda44: 2( 2)

ffffda48: 40( 64)

ffffda4c: ffffdaa4( -9564)

ffffda50: 2( 2)

ffffda54: ffffda74( -9612)

------------------------------- FP=ffffda54, 64 bytes

(EAX= 64),(EBX= 1),(ECX= 0),(EDX= -9564)

ffffda38: 80486cd( 134514381)

ffffda3c: 40( 64)

ffffda40: 40( 64)

ffffda44: 2( 2)

ffffda48: 40( 64)

ffffda4c: 2( 2)

ffffda50: 2( 2)

ffffda54: **ffffda74**( -9612)

ffffda58: 80486b0( 134514352)

ffffda5c: 2( 2)

ffffda60: 1( 1)

ffffda64: 3( 3)

ffffda68: 40( 64)

ffffda6c: ffffdaa4( -9564)

ffffda70: 3( 3)

ffffda74: ffffda94( -9580)

------------------------------- FP=ffffda74, 64 bytes

(EAX= 64),(EBX= 1),(ECX= 0),(EDX= -9564)

ffffda58: 80486cd( 134514381)

ffffda5c: 40( 64)

ffffda60: 40( 64)

ffffda64: 3( 3)

ffffda68: 40( 64)

ffffda6c: 6( 6)

ffffda70: 3( 3)

ffffda74: **ffffda94**( -9580)

ffffda78: 80486b0( 134514352)

ffffda7c: 3( 3)

ffffda80: ffffdb14( -9452)

ffffda84: 4( 4)

ffffda88: 40( 64)

ffffda8c: ffffdaa4( -9564)

ffffda90: 4( 4)

ffffda94: ffffdac4( -9532)

------------------------------- FP=ffffda94, 64 bytes

(EAX= 64),(EBX= 1),(ECX= 0),(EDX= -9564)

ffffda78: 80486cd( 134514381)

ffffda7c: 40( 64)

ffffda80: 40( 64)

ffffda84: 4( 4)

ffffda88: 40( 64)

ffffda8c: 18( 24)

ffffda90: 4( 4)

ffffda94: **ffffdac4**( -9532)

ffffda98: 804861e( 134514206)

ffffda9c: 4( 4)

ffffdaa0: ffffdac0( -9536)

ffffdaa4: 2817a99f( 672639391)

ffffdaa8: 1( 1)

ffffdaac: 2804c560( 671401312)

ffffdab0: 40( 64)

ffffdab4: 1( 1)

------------------------------- FP=ffffdac4, 64 bytes

(EAX= 24),(EBX= 1),(ECX= 0),(EDX= -9564)

ffffda98: 8048635( 134514229)

ffffda9c: 40( 64)

ffffdaa0: ffffdac0( -9536)

ffffdaa4: 2817a99f( 672639391)

ffffdaa8: 1( 1)

ffffdaac: 40( 64)

ffffdab0: 40( 64)

ffffdab4: 1( 1)

ffffdab8: 11( 17)

ffffdabc: 18( 24)

ffffdac0: 4( 4)

ffffdac4: ffffdae8( -9496)

ffffdac8: 804851a( 134513946)

ffffdacc: 1( 1)

ffffdad0: ffffdb0c( -9460)

ffffdad4: ffffdb14( -9452)

4! is 24

server1#