# Main C Source

// Sean Batzel

// Assignment 2

// CMPS 250 Machine Organization and Assembly Programming

#include <stdio.h>

int gcd(int a, int b);

void DumpS(int n);

int main(){

int input1, input2, final;

printf("Enter a number: ");

scanf("%d", &input1);

printf("Enter another: ");

scanf("%d", &input2);

DumpS(64);

final = gcd(input1, input2);

DumpS(64);

printf("The GCD of %d and %d is %d.\n", input1, input2, final);

}

# Main Assembler Listing

.file "GCDRM.c"

.def \_\_\_main; .scl 2; .type 32; .endef

.section .rdata,"dr"

LC0:

.ascii "Enter a number: \0"

LC1:

.ascii "%d\0"

LC2:

.ascii "Enter another: \0"

LC3:

.ascii "The GCD of %d and %d is %d.\12\0"

.text

.globl \_main

.def \_main; .scl 2; .type 32; .endef

\_main:

pushl %ebp

movl %esp, %ebp

pushl %ebx

andl $-16, %esp

subl $32, %esp

call \_\_\_main

movl $LC0, %eax

movl %eax, (%esp)

call \_printf

movl $LC1, %eax

leal 24(%esp), %edx

movl %edx, 4(%esp)

movl %eax, (%esp)

call \_scanf

movl $LC2, %eax

movl %eax, (%esp)

call \_printf

movl $LC1, %eax

leal 20(%esp), %edx

movl %edx, 4(%esp)

movl %eax, (%esp)

call \_scanf

movl $64, (%esp)

call \_DumpS

movl 20(%esp), %edx

movl 24(%esp), %eax

movl %edx, 4(%esp)

movl %eax, (%esp)

call \_gcd

movl %eax, 28(%esp)

movl $64, (%esp)

call \_DumpS

movl 20(%esp), %ecx

movl 24(%esp), %edx

movl $LC3, %eax

movl 28(%esp), %ebx

movl %ebx, 12(%esp)

movl %ecx, 8(%esp)

movl %edx, 4(%esp)

movl %eax, (%esp)

call \_printf

movl -4(%ebp), %ebx

leave

ret

.def \_printf; .scl 2; .type 32; .endef

.def \_scanf; .scl 2; .type 32; .endef

.def \_DumpS; .scl 2; .type 32; .endef

.def \_gcd; .scl 2; .type 32; .endef

# Function C Source

// Sean Batzel

// Assignment 2

// CMPS 250 Machine Organization and Assembly Programming

void DumpS(int n);

int gcd(int a, int b){

int final;

DumpS(64);

if (a == b) final = a;

else final = gcd(b, a % b);

DumpS(64);

return final;

}

# Function Assembler Listing

.file "GCDRF.c"

.text

.globl \_gcd

.def \_gcd; .scl 2; .type 32; .endef

\_gcd:

pushl %ebp

movl %esp, %ebp

subl $40, %esp

movl $64, (%esp)

call \_DumpS

movl 8(%ebp), %eax

cmpl 12(%ebp), %eax

jne L2

movl 8(%ebp), %eax

movl %eax, -12(%ebp)

jmp L3

L2:

movl 8(%ebp), %eax

movl %eax, %edx

sarl $31, %edx

idivl 12(%ebp)

movl %edx, %eax

movl %eax, 4(%esp)

movl 12(%ebp), %eax

movl %eax, (%esp)

call \_gcd

movl %eax, -12(%ebp)

L3:

movl $64, (%esp)

call \_DumpS

movl -12(%ebp), %eax

leave

ret

.def \_DumpS; .scl 2; .type 32; .endef

# DumpS32 Assembler Listing

.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

# Runtime Stack and Output

batzels2@server1:/users/student/batzels2/Machine Org/Assignment2 % ./gcd

Enter a number: 5

Enter another: 5

------------------------------- FP=ffffdb04, 64 bytes

(EAX= 1),(EBX= 1),(ECX= 0),(EDX= -9520)

ffffdac4: 804863d( 134514237)

ffffdac8: 40( 64)

ffffdacc: ffffdaf8( -9480)

ffffdad0: 0( 0)

ffffdad4: ffffdb44( -9404)

ffffdad8: 0( 0)

ffffdadc: 281b9f04( 672898820)

ffffdae0: 40( 64)

ffffdae4: 1( 1)

ffffdae8: f( 15)

ffffdaec: 1( 1)

ffffdaf0: 10( 16)

ffffdaf4: 0( 0)

ffffdaf8: 5( 5)

ffffdafc: 5( 5)

ffffdb00: ffffdb44( -9404)

------------------------------- FP=ffffdac0, 64 bytes

(EAX= 5),(EBX= 1),(ECX= 0),(EDX= 64)

ffffda9c: 80486c6( 134514374)

ffffdaa0: 40( 64)

ffffdaa4: ffffdb00( -9472)

ffffdaa8: ffffdb44( -9404)

ffffdaac: ffffdb44( -9404)

ffffdab0: 40( 64)

ffffdab4: 0( 0)

ffffdab8: 5( 5)

ffffdabc: 5( 5)

ffffdac0: ffffdb04( -9468)

ffffdac4: 804864f( 134514255)

ffffdac8: 5( 5)

ffffdacc: 5( 5)

ffffdad0: 0( 0)

ffffdad4: ffffdb44( -9404)

ffffdad8: 0( 0)

------------------------------- FP=ffffdac0, 64 bytes

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

ffffda9c: 8048712( 134514450)

ffffdaa0: 40( 64)

ffffdaa4: ffffdb00( -9472)

ffffdaa8: 40( 64)

ffffdaac: ffffdb44( -9404)

ffffdab0: 40( 64)

ffffdab4: 5( 5)

ffffdab8: 5( 5)

ffffdabc: 5( 5)

ffffdac0: ffffdb04( -9468)

ffffdac4: 804864f( 134514255)

ffffdac8: 5( 5)

ffffdacc: 5( 5)

ffffdad0: 0( 0)

ffffdad4: ffffdb44( -9404)

ffffdad8: 0( 0)

------------------------------- FP=ffffdb04, 64 bytes

(EAX= 5),(EBX= 1),(ECX= 0),(EDX= 64)

ffffdac4: 8048666( 134514278)

ffffdac8: 40( 64)

ffffdacc: 5( 5)

ffffdad0: 0( 0)

ffffdad4: ffffdb44( -9404)

ffffdad8: 0( 0)

ffffdadc: 40( 64)

ffffdae0: 40( 64)

ffffdae4: 1( 1)

ffffdae8: f( 15)

ffffdaec: 1( 1)

ffffdaf0: 10( 16)

ffffdaf4: 5( 5)

ffffdaf8: 5( 5)

ffffdafc: 5( 5)

ffffdb00: ffffdb44( -9404)

The GCD of 5 and 5 is 5.

batzels2@server1:/users/student/batzels2/Machine Org/Assignment2 %