

Here is the program that generated the following code:

```
void f(int a, int b) {
    int x;

    x = a+b;
    write(x);
}

void main(void) {
    f(3, 4);
}
```

#####

```
.section .rodata
.WriteIntString: .string "%d "
.WriteLineString: .string "\n"
.WriteStringString: .string "%s "
.ReadIntString: .string "%d"
.text
.globl main
f:
```

```
    movq %rsp, %rbx
    sub $8, %rsp
    movq %rbx, %rax
    sub $8, %rax
    push %rax
    movq 16(%rbx), %rax
    push %rax
    movq 24(%rbx), %rax
    addl 0(%rsp), %eax
    addq $8, %rsp
    movq 0(%rsp), %rsi
    movq %rax, 0(%rsi)
    addq $8, %rsp
    movq -8(%rbx), %rax
    movl %eax, %esi
    movq $.WriteIntString, %rdi
    movl $0, %eax
    call printf
    add $8, %rsp
    ret
```

main:

```
    movq %rsp, %rbx
    sub $0, %rsp
    movl $4, %eax
    push %rax
    movl $3, %eax
    push %rax
    push %rbx
    call f
    pop %rbx
    add $16, %rsp
```

```
    #set up the frame pointer
    #allocate local variables
    #put the frame pointer into ac
    #local variable address
    #push the lvalue
    #param value
    #saving the left operand on the stack
    #param value
    #performing addition
    #popping the value saved on the stack
    #put the lvalue into rsi
    #assign
    #pop the lvalue from the stack
    #local variable value
    #value to print = arg2
    #printf string = arg1
    #clear the return value
    #call the C-lib printf function
    #deallocate local variables
    #return from the function
```

```
    #set up the frame pointer
    #allocate local variables
    #putting value into ac
    #pushing argument
    #putting value into ac
    #pushing argument
    #pushing the frame pointer
    #calling the function
    #retrieving the frame pointer
    #removing args from the stack
```

Header.
This is included in every program.

Enter function f

L-value of x

a+b

assignment

write statement

leave f

enter function main

call f(3, 4)

```
add $0, %rsp  
ret
```

```
#deallocate local variables  
#return from the function
```

return from main()
