Original Decaf Code:

```
import printf;
void main ( ) {
  int a, b, c;
  int d, e;
  int g, h;
  a = 10;
  b = 20;
  c = 30;
  d = ( a + b );
  e = ( c * 3 );
  printf ( "%d %d\n", d, e );
  g = d % 16;
  h = e / 100;
  printf ( "%d %d\n", g, h );
}
```

Before Register Allocation:

```
.data
string_0:
       .string   "%d %d\n"
       .align 16
.text
.globl main
main:
       pushq  %rbp
       movq   %rsp, %rbp
       subq   $80, %rsp
       movq   $10, -8(%rbp)
       movq   $20, -16(%rbp)
       movq   $30, -24(%rbp)
       mov    -8(%rbp), %rax
       addq   -16(%rbp), %rax
       movq   %rax, -32(%rbp)      # d = a + b
       movq   $3, -64(%rbp)
       mov    -24(%rbp), %rax
       imulq  -64(%rbp), %rax
       movq   %rax, -40(%rbp)      # e = c * %0
       movq   -40(%rbp), %rdx
       movq   -32(%rbp), %rsi
       movq   $string_0, %rdi
```

```
	xorl	%eax, %eax
	callq	printf
	movq	$16, -72(%rbp)
	movq	-32(%rbp), %rax
	cqto
	idivq	-72(%rbp)
	movq	%rdx, -48(%rbp)		# g = d % tmp001
	movq	$100, -80(%rbp)
	movq	-40(%rbp), %rax
	cqto
	idivq	-80(%rbp)
	movq	%rax, -56(%rbp)		# h = e / tmp002
	movq	-56(%rbp), %rdx
	movq	-48(%rbp), %rsi
	movq	$string_0, %rdi
	xorl	%eax, %eax
	callq	printf
	jmp	.exit_main
.exit_main:
	xorl	%eax, %eax
	addq	$80, %rsp
	movq	%rbp, %rsp
	popq	%rbp
	ret
```

After Register Allocation:

```
.data
string_0:
	.string	"%d %d\n"
	.align 16
.text
.globl main
main:
	movq	$10, %rbx
	movq	$20, %r12
	movq	$30, %r13
	mov	%rbx, %rax
	addq	%r12, %rax
	movq	%rax, %r14	# d = a + b
	movq	$3, %r15
	mov	%r13, %rax
	imulq	%r15, %rax
	movq	%rax, %rdi	# e = c * %0
	movq	%rdi, %rdx
	movq	%r14, %rsi
```

```
	movq	$string_0, %rdi
	xorl	%eax, %eax
	callq	printf
	movq	$16, %rsi
	movq	%r14, %rax
	cqto
	idivq	%rsi
	movq	%rdx, %r8	# g = d % tmp001
	movq	$100, %r9
	movq	%rdi, %rax
	cqto
	idivq	%r9
	movq	%rax, %rdx	# h = e / tmp002
	movq	%rdx, %rdx
	movq	%r8, %rsi
	movq	$string_0, %rdi
	xorl	%eax, %eax
	callq	printf
	jmp	.exit_main
.exit_main:
	xorl	%eax, %eax
	ret
```