Practice Quiz 3

Question **1**

Determine the appropriate instruction suffix based on the operands.

mov q %rcx, (%rsp)

b w l q

Feedback

Your answer is correct.

Because this refers to 64-bit register values (e.g. **%rcx**), you would use **movq**.

The correct answer is:

Determine the appropriate instruction suffix based on the operands.

mov[q] %rcx, (%rsp)

Question **2**

Determine the appropriate instruction suffix based on the operands.

mov w (%eax), %dx

b w l q

Feedback

Your answer is correct.

Because this refers to 16-bit register values (e.g. **%dx**), you would use **movw**.

The correct answer is:

Determine the appropriate instruction suffix based on the operands.

mov[w] (%eax), %dx

Question **3**

Determine the appropriate instruction suffix based on the operands.

mov b $0xFF, %bl

b w l q

Feedback

Your answer is correct.

Because this refers to 8-bit register values (e.g. **%bl**), you would use **movb**.

The correct answer is:

Determine the appropriate instruction suffix based on the operands.

mov[b] $0xFF, %bl

Question **4**

Determine the appropriate instruction suffix based on the operands.

mov l (%rsp, %rdx, 8), %edx

b w l q

Feedback

Your answer is correct.

Because this refers to 32-bit register values (e.g. **%edx**), you would use **movl**.

The correct answer is:

Determine the appropriate instruction suffix based on the operands.

mov[l] (%rsp, %rdx, 8), %edx

Question **5**

Determine the appropriate instruction suffix based on the operands.

mov l %edi, (%rax)

b w l q

Feedback

Your answer is correct.

Because this refers to 32-bit register values (e.g. **%edi**), you would use **movl**.

The correct answer is:

Determine the appropriate instruction suffix based on the operands.

mov[l] %edi, (%rax)

Question **6**

The notation **M[x]** refers to the value of memory at address **x**, and **Reg[x]** refers to the value of register **x**.

What does this instruction mean? **addq 8(%rbp),%rax**

Select one:

a. Reg[rbp] = Reg[rbp] + Reg[rax]

b. Reg[rax] = Reg[rbp]+8 + Reg[rax]

c. Reg[rax] = Mem[Reg[rbp]]+8 + Reg[rax]

d. Reg[rax] = Mem[Reg[rbp]+8] + Reg[rax]

Feedback

Your answer is correct.

The correct answer is: Reg[rax] = Mem[Reg[rbp]+8] + Reg[rax]

Question **7**

The notation **M[x]** refers to the value of memory at address **x**, and **Reg[x]** refers to the value of register **x**.

What does this instruction mean?

**addq $0x11 , (%rax)**

a. Mem[Reg[rax]] = 17 + Mem[Reg[rax]]

b. Reg[rax] = 11 + Mem[Reg[rax]]

c. Mem[Reg[rax]] = 17 + Reg[rax]

d. Reg[rax] = 11 + Reg[rax]

Your answer is correct.

The correct answer is: Mem[Reg[rax]] = 17 + Mem[Reg[rax]]

Question **8**

The notation **M[x]** refers to the value of memory at address **x**, and **Reg[x]** refers to the value of register **x**.

What does this instruction mean?

**addq 16(%rbp),%rcx**

Select one:

a. Reg[rcx] = Reg[rcx] + Mem[Reg[rbp]] + 16

b. Reg[rcx] = Reg[rcx] + Mem[Reg[rbp] + 16]

c. Reg[rcx] = 16 + Mem[Reg[rbp]]

d. Mem[Reg[rbp]] = Reg[rcx] + Mem[Reg[rbp] + 16]

Your answer is correct.

The correct answer is: Reg[rcx] = Reg[rcx] + Mem[Reg[rbp] + 16]

Question **9**

For the following instruction:

addq 4(%rdx),%rax

What is the size of the addition results in bytes?

Select one:

a. 4 Bytes

b. 8 Bytes

c. 2 Bytes

d. 1 Byte

Your answer is correct.

The correct answer is: 8 Bytes

Question **10**

Suppose register %rax holds value 5 and %rcx holds value 10. Fill in the table below with formulas indicating the value that will be stored in register %rdx for each of the given assembly code instructions: (Answer in decimal)

Instruction Result

leal 6 (%rax), %rdx 11

leal (%rax, %rcx), %rdx 15

leal 0xC(%rax, %rcx, 4), %rdx 57

leal 8(, %rcx, 2), %rdx 28

Question **11**

Suppose register %rax holds value 5 and %rcx holds value 10. Fill in the table below with formulas indicating the value that will be stored in register %rdx for each of the given assembly code instructions: (Answer in decimal)

Instruction Result

leal 0xA (%rax), %rdx 15

leal (%rcx, %rcx), %rdx 20

leal 0xC(, %rax, 4), %rdx 32

leal 8(%rax, %rcx, 2), %rdx 33

Question **12**

Suppose register %rax holds value 10 and %rcx holds value 4. Fill in the table below with formulas indicating the value that will be stored in register %rdx for each of the given assembly code instructions: (Answer in decimal)

Instruction Result

leal (%rax, %rcx, 3), %rdx 22

leal 0xA (, %rax, 8), %rdx 90

leal 8 (%rax, %rcx), %rdx 22

leal (%rax, %rax), %rdx 20

Question **13**

Suppose register %rax holds value 5 and %rcx holds value 10. Fill in the table below with formulas indicating the value that will be stored in register %rdx for each of the given assembly code instructions: (Answer in decimal)

Instruction Result

leal (%rax, %rax, 4), %rdx 25

leal (, %rax, 2), %rdx 10

leal 8 (%rcx, %rcx), %rdx 28

leal (%rcx, %rax), %rdx 15

Question **14**

Suppose register %eax holds value 10 and %ecx holds value 4. Fill in the table below with formulas indicating the value that will be stored in register %edx for each of the given assembly code instructions:

Instruction Result

leal 0xA (%eax), %edx 20

leal (%ecx, %ecx), %edx 8

leal 0xC(, %eax, 4), %edx 52

leal 8(%eax, %ecx, 2), %edx 26

Question **15**

Suppose register %eax holds value 5 and %ecx holds value 10. Fill in the table below with formulas indicating the value that will be stored in register %edx for each of the given assembly code instructions:

Instruction Result

leal (%eax, %ecx, 3), %edx 35

leal 0xA (, %eax, 8), %edx 50

leal 8 (%eax, %ecx), %edx 23

leal (%eax, %eax), %edx 10

Question **16**

The assembly code on the right partially implements the C function shown on the left. Fill in the missing instruction to correctly implement the C function on the left.

|  |  |
| --- | --- |
| #include <stdio.h>   int main() {  long int a=5;     if (a >= 3) {  return 1;  }  else {  return 0;   } } | **main:  push %rbp  mov %rsp, %rbp   movq $0x5, -0x8(%rbp)  cmpq $0x2, -0x8(%rbp)   jle L1  mov $0x1, %eax  jmp L2  L1:  mov $0x0, %eax   L2:   pop %rbp  retq** |

Your answer is correct.

The correct answer is:

The assembly code on the right partially implements the C function shown on the left. Fill in the missing instruction to correctly implement the C function on the left.

Question **17**

The assembly code on the right partially implements the C function shown on the left. Fill in the missing instruction to correctly implement the C function on the left.

|  |  |
| --- | --- |
| int foo(long a) {   if (a > 0) {  return 0;  }  else {  return 1;   } | **testl %rdi,%rdi   jle L1  movl $0x0, %eax  jmp L2 L1:  movl $0x1, %eax  L2:   retq** |

Your answer is correct.

The correct answer is:

The assembly code on the right partially implements the C function shown on the left. Fill in the missing instruction to correctly implement the C function on the left.

Question **18**

The assembly code on the right partially implements the C function shown on the left. Fill in the missing instruction to correctly implement the C function on the left.

|  |  |
| --- | --- |
| int foo(a) {   if (a <= 5) {  return -10;  }  else {  return 10;   } } | **foo:  cmpl $0x5, %edi   jg L1  mov $0xfffffff6, %eax  jmp L2 L1:  mov $0xa, %eax  L2:   retq** |

Your answer is correct.

The correct answer is:

The assembly code on the right partially implements the C function shown on the left. Fill in the missing instruction to correctly implement the C function on the left.

Question **19**

The assembly code on the right partially implements the C function shown on the left. Fill in the missing instruction to correctly implement the C function on the left.

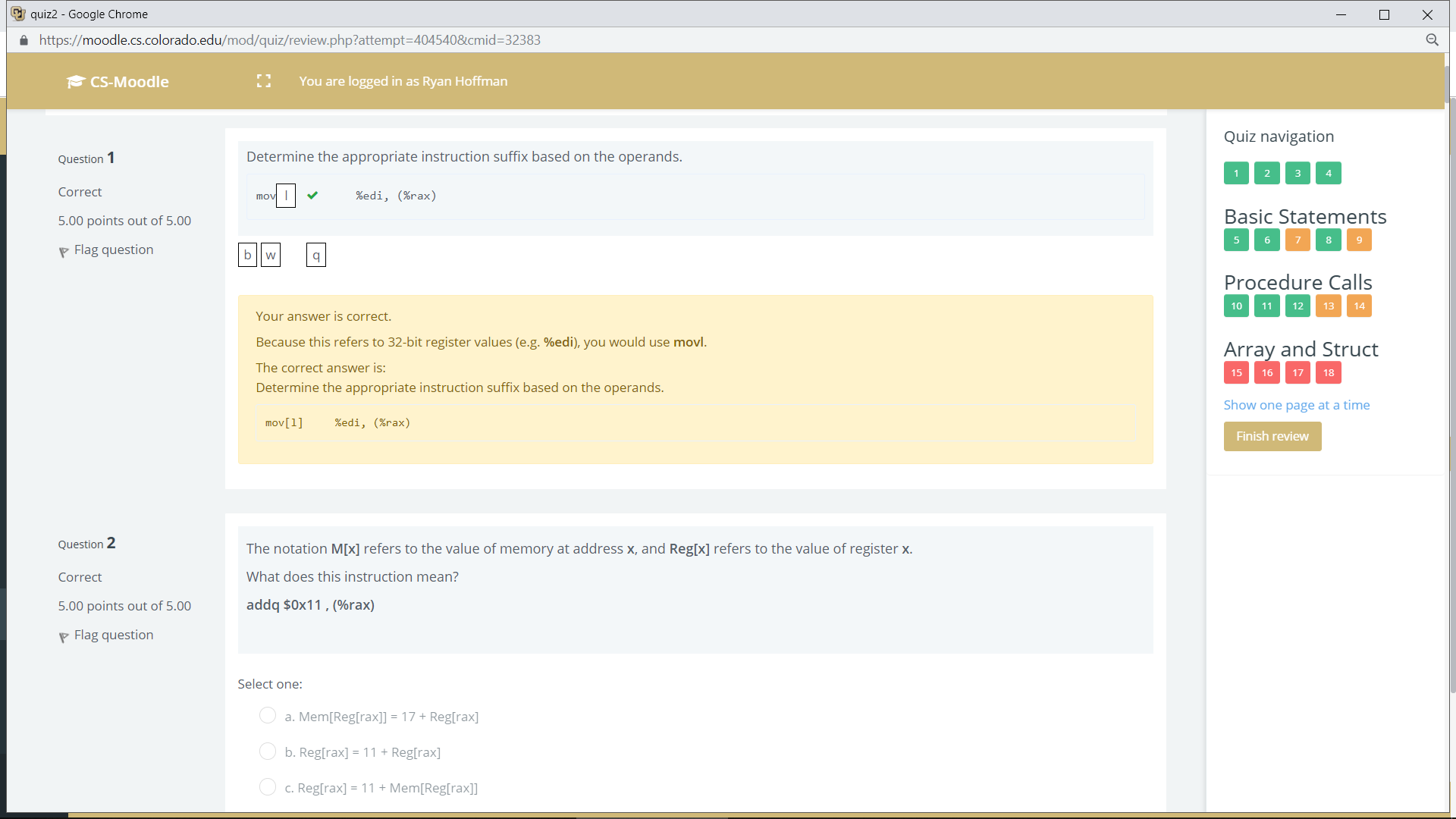
|  |  |
| --- | --- |
| #include <stdio.h>   int main() {  long int a=5;     if (a < 10) {  return 3;  }  else {  return 0;   } } | **main:  push %rbp  mov %rsp, %rbp   movq $0x5, -0x8(%rbp)  cmpq $0x9, -0x8(%rbp)   jg L1  mov $0x3, %eax  jmp L2 L1:  mov $0x0, %eax  L2:   pop %rbp  retq** |

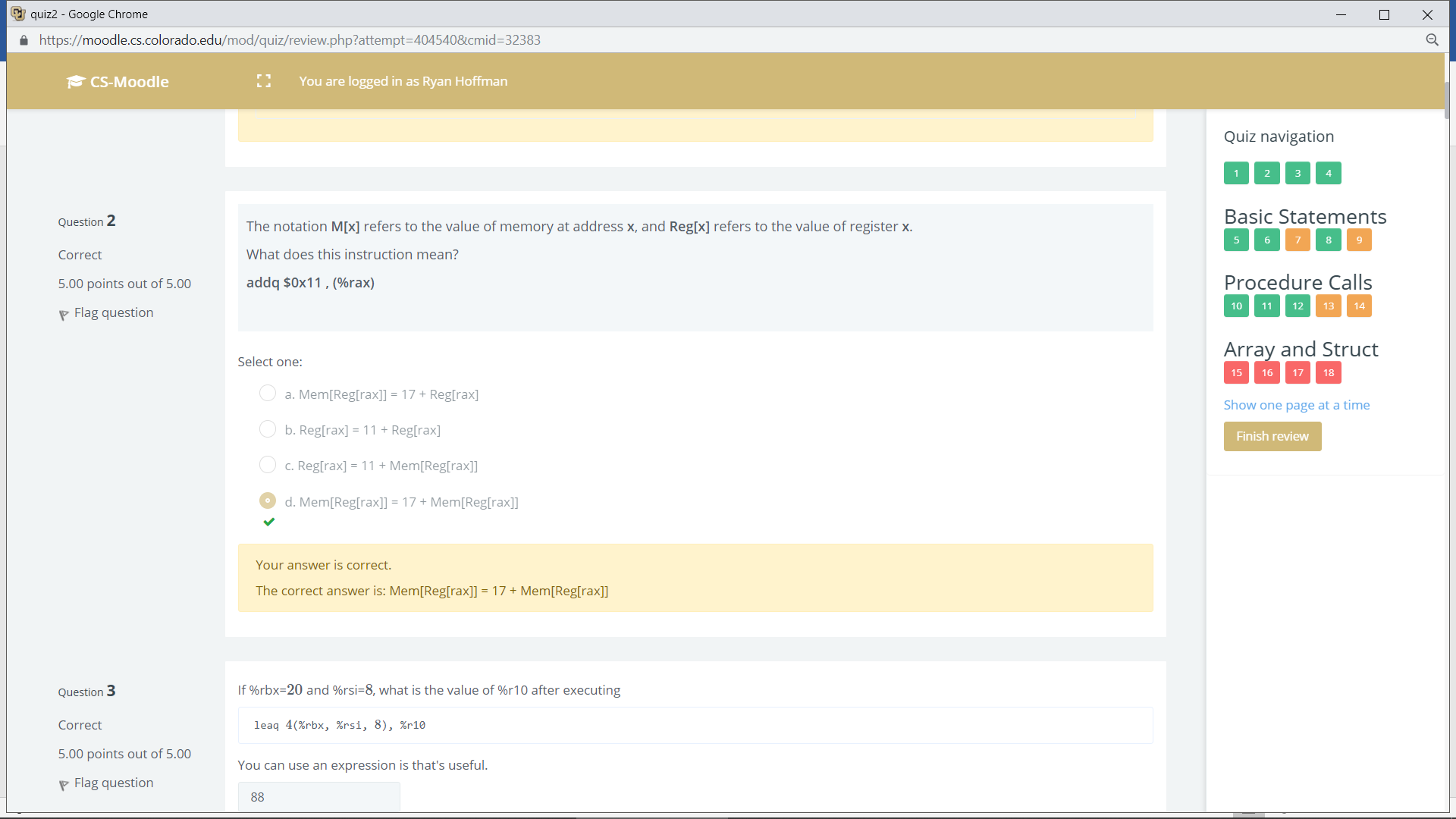
Your answer is correct.

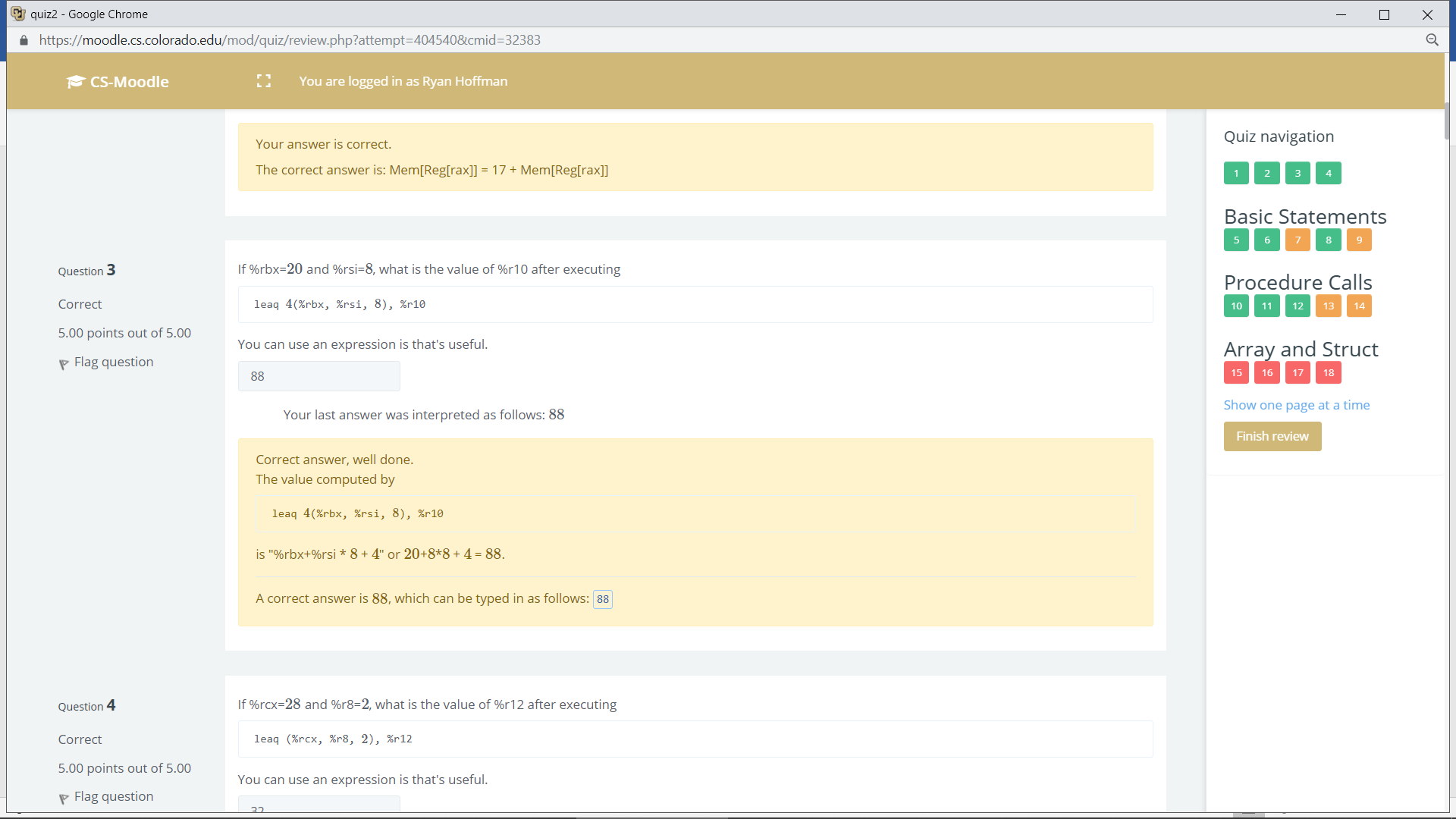
The correct answer is:

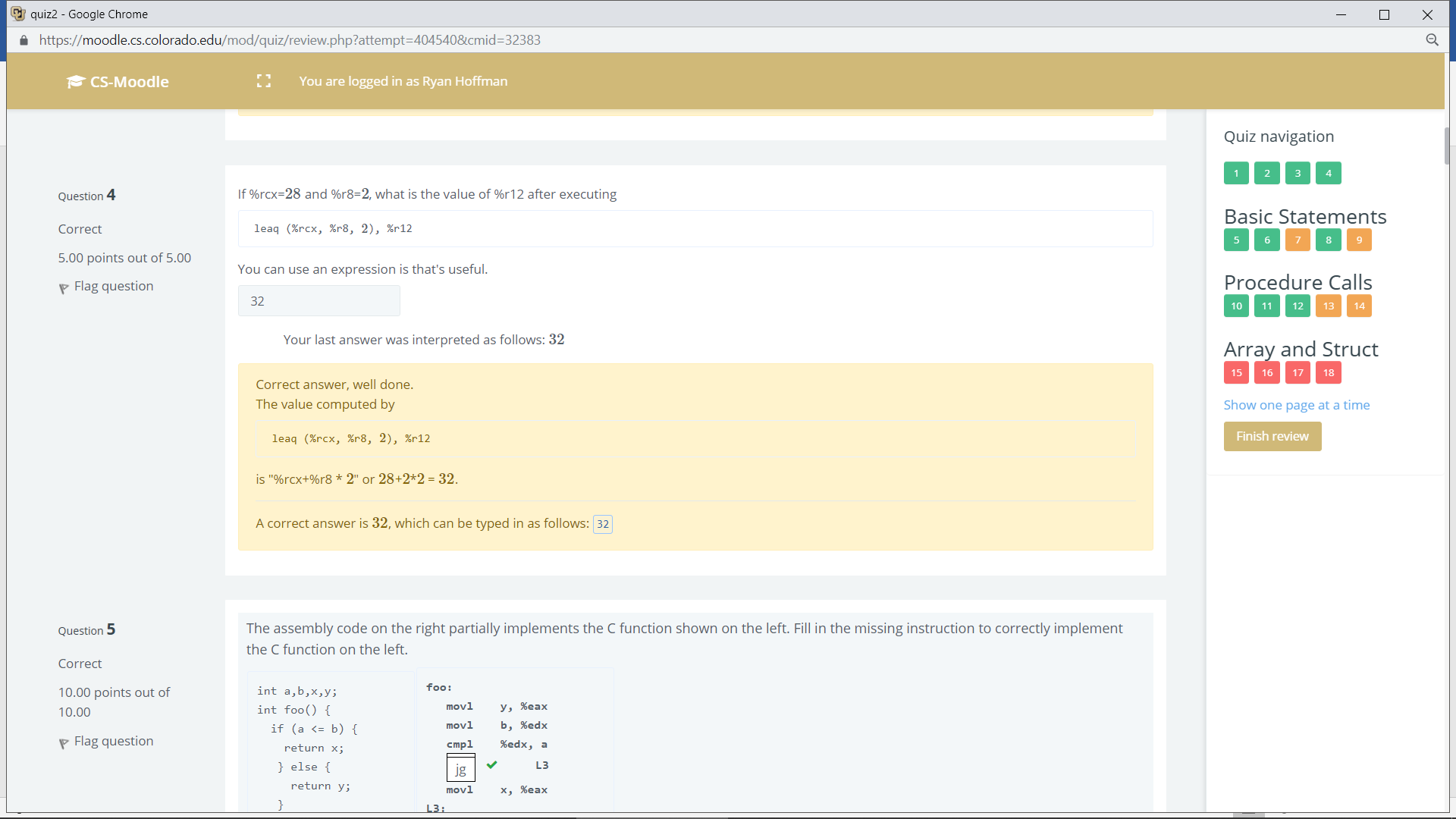
The assembly code on the right partially implements the C function shown on the left. Fill in the missing instruction to correctly implement the C function on the left.

QUIZ 2

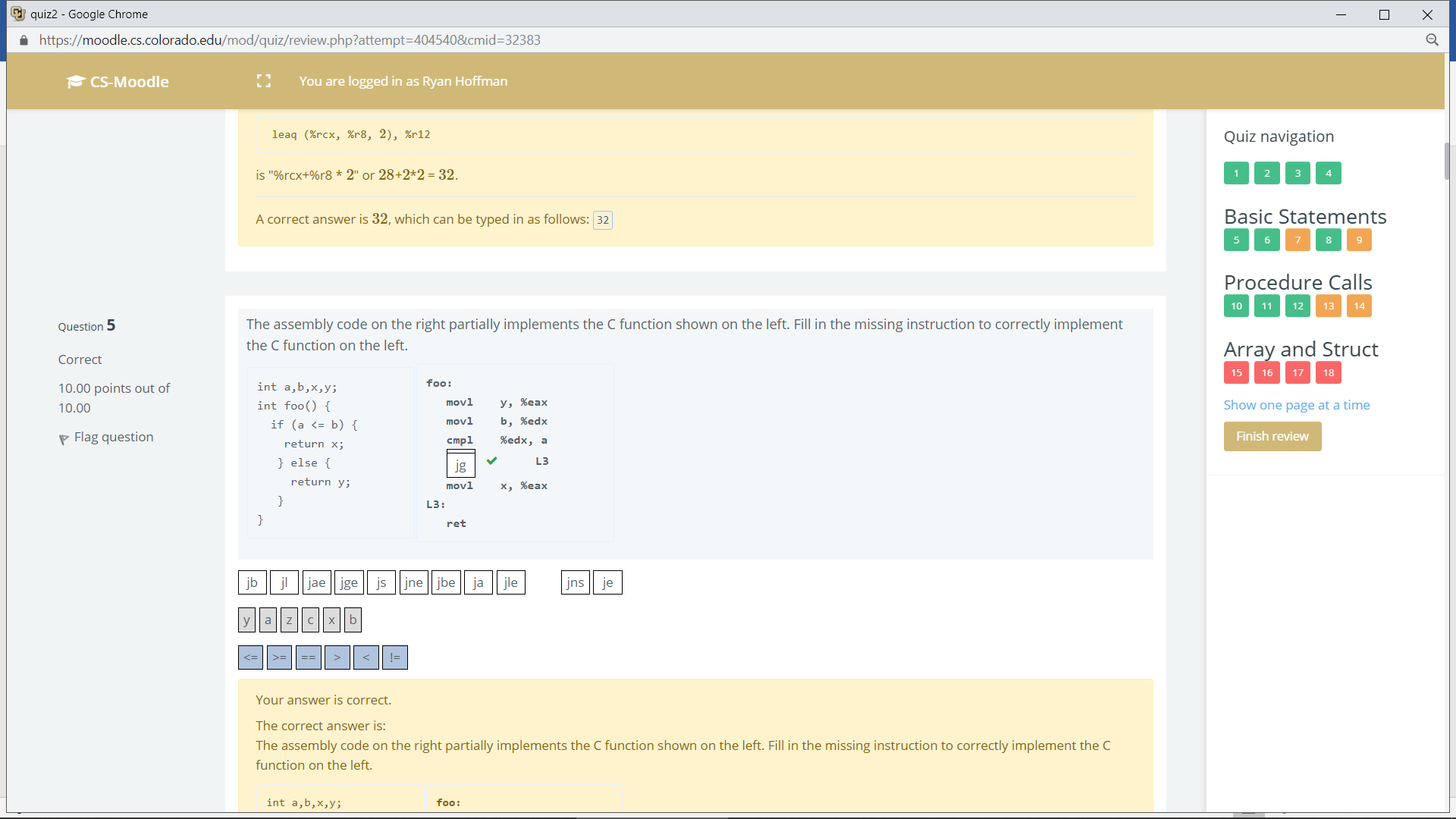




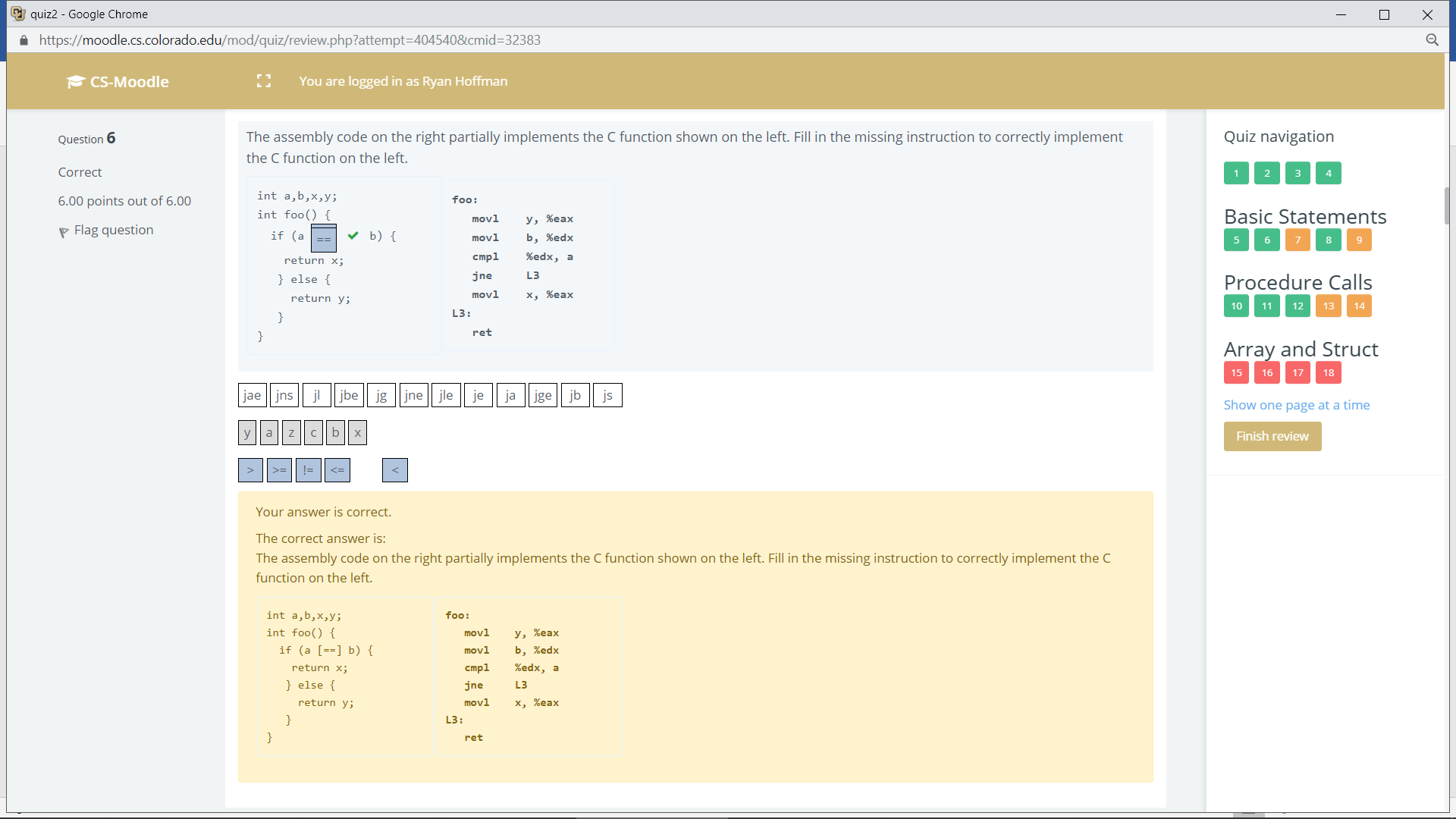




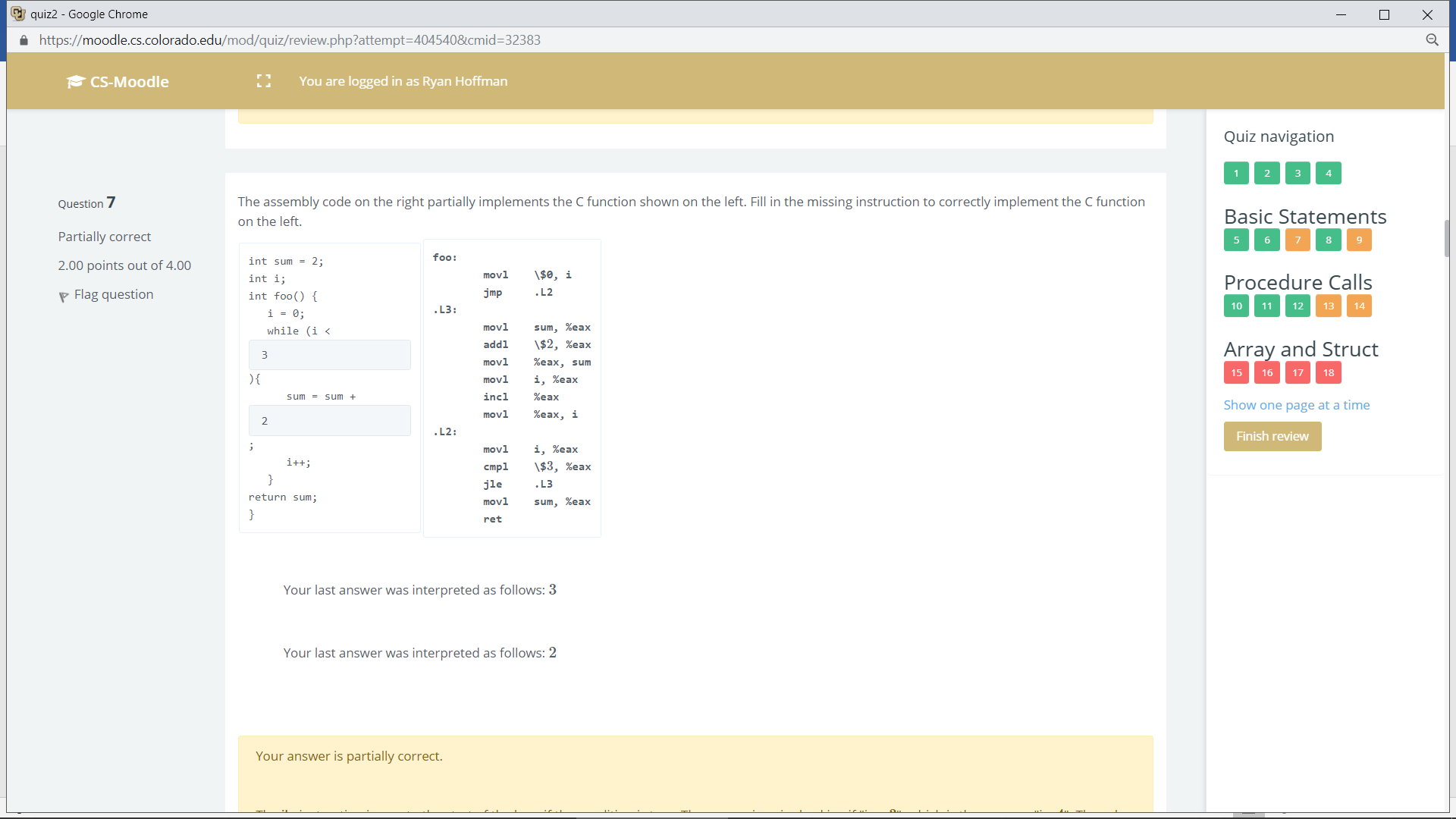
Q 5:

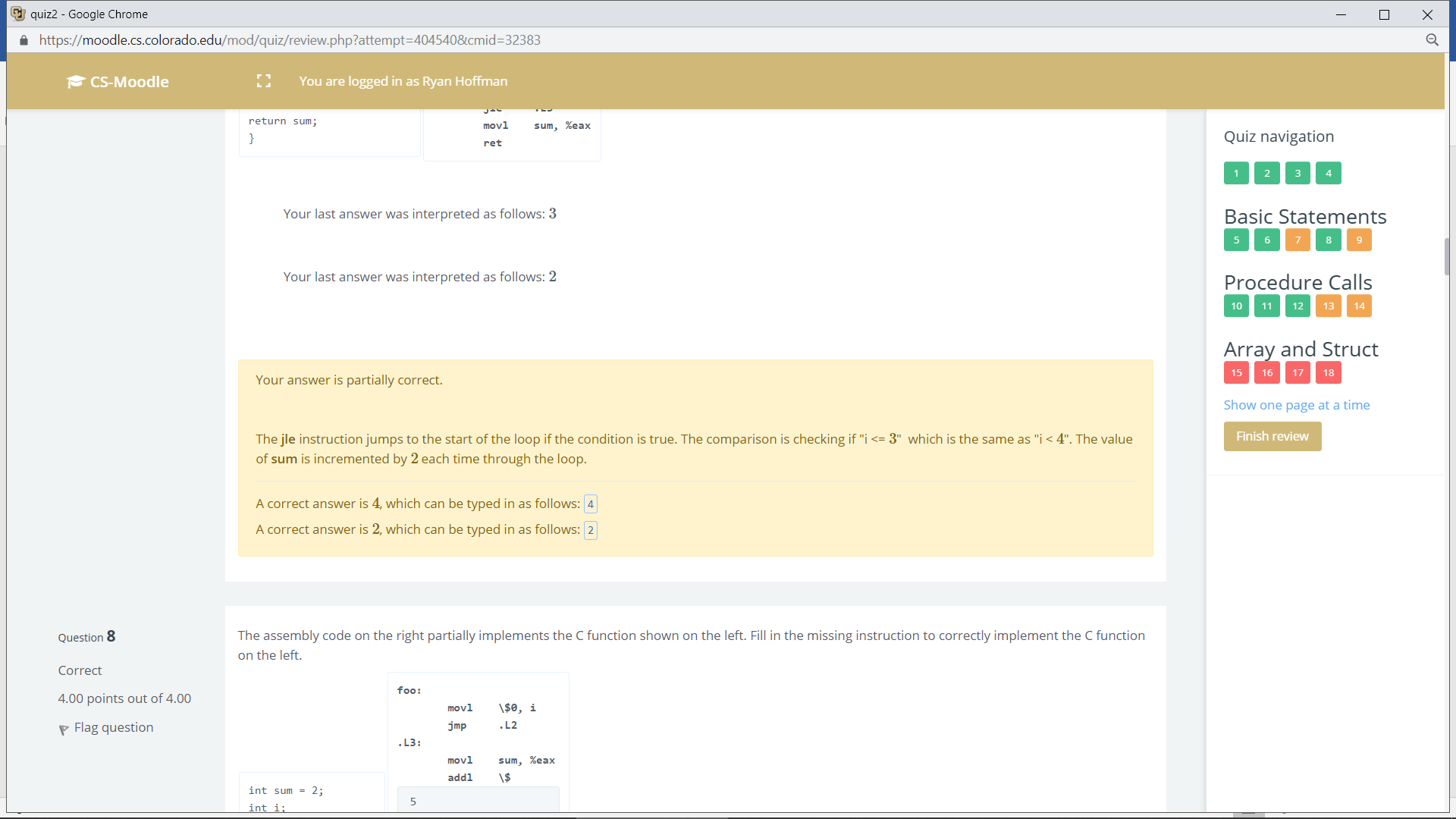


Q6

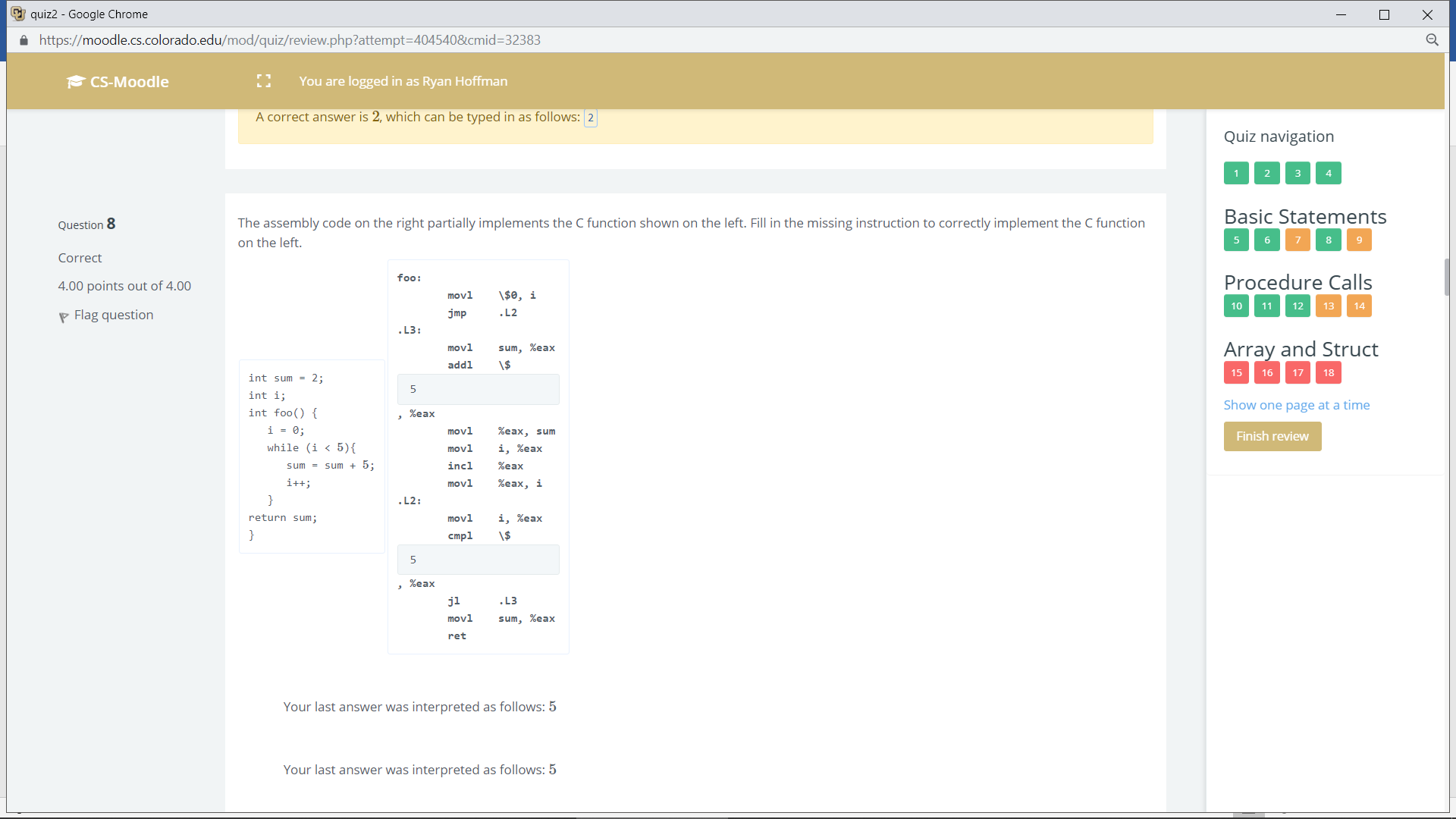


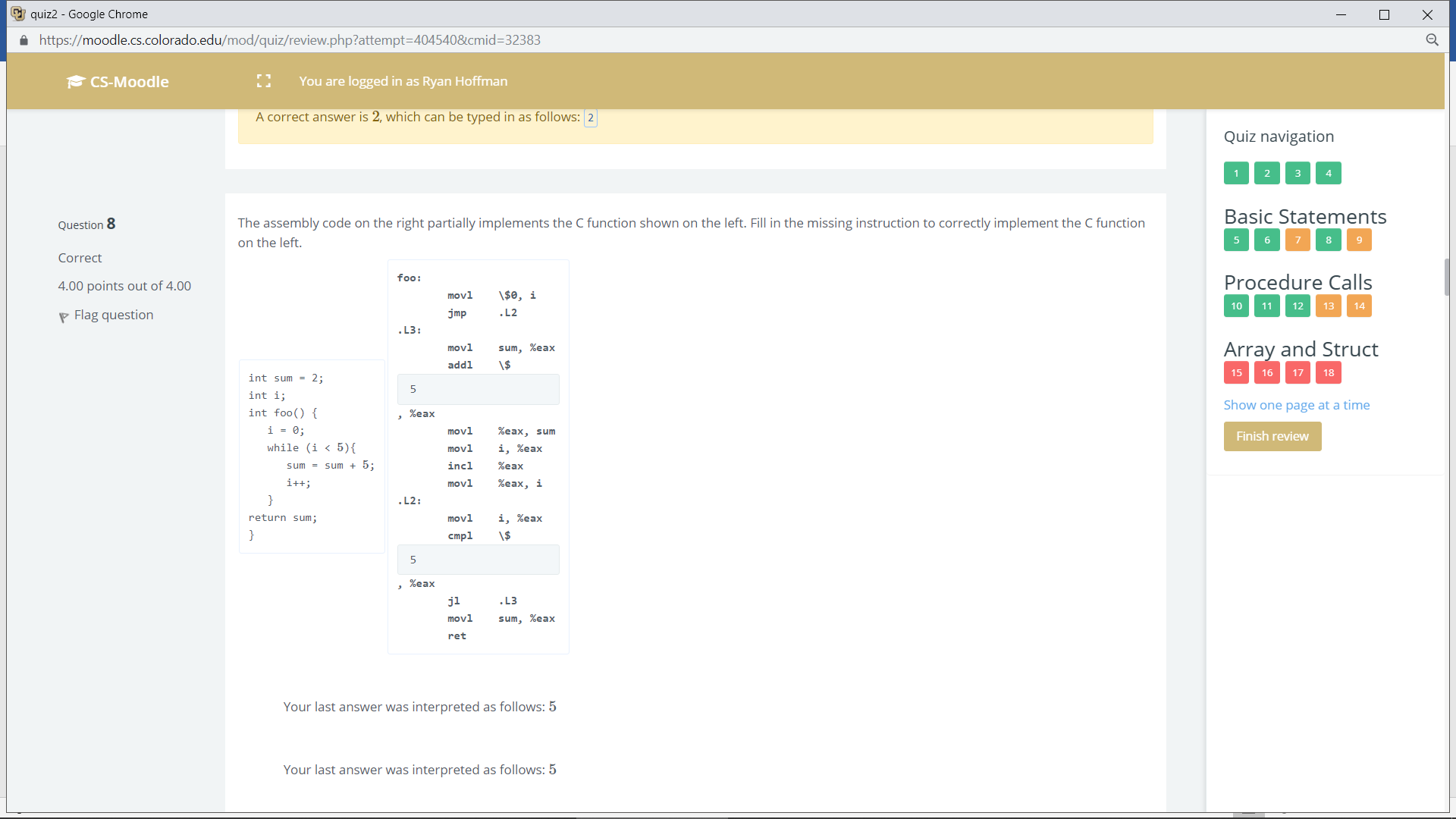
Q7

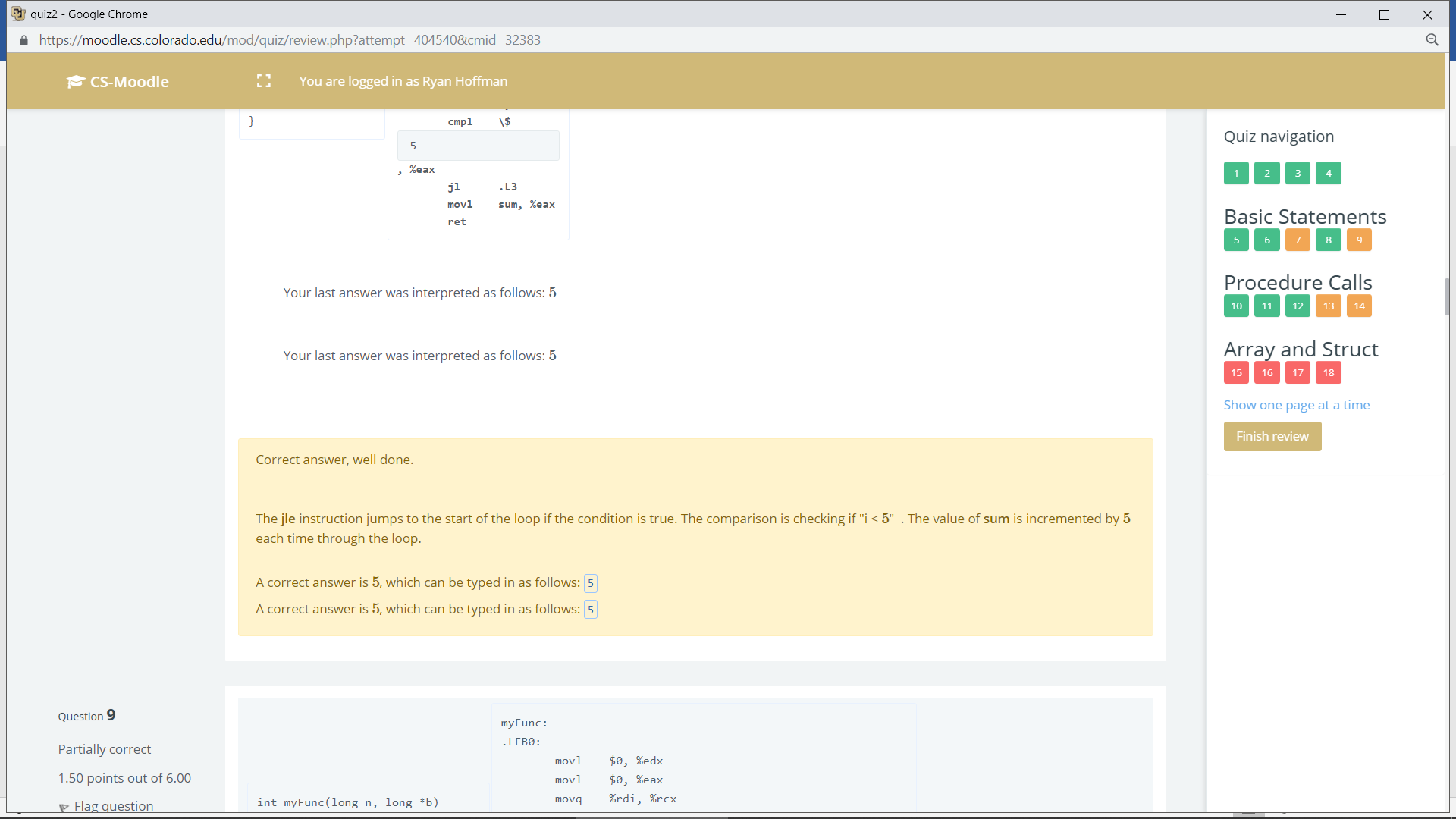




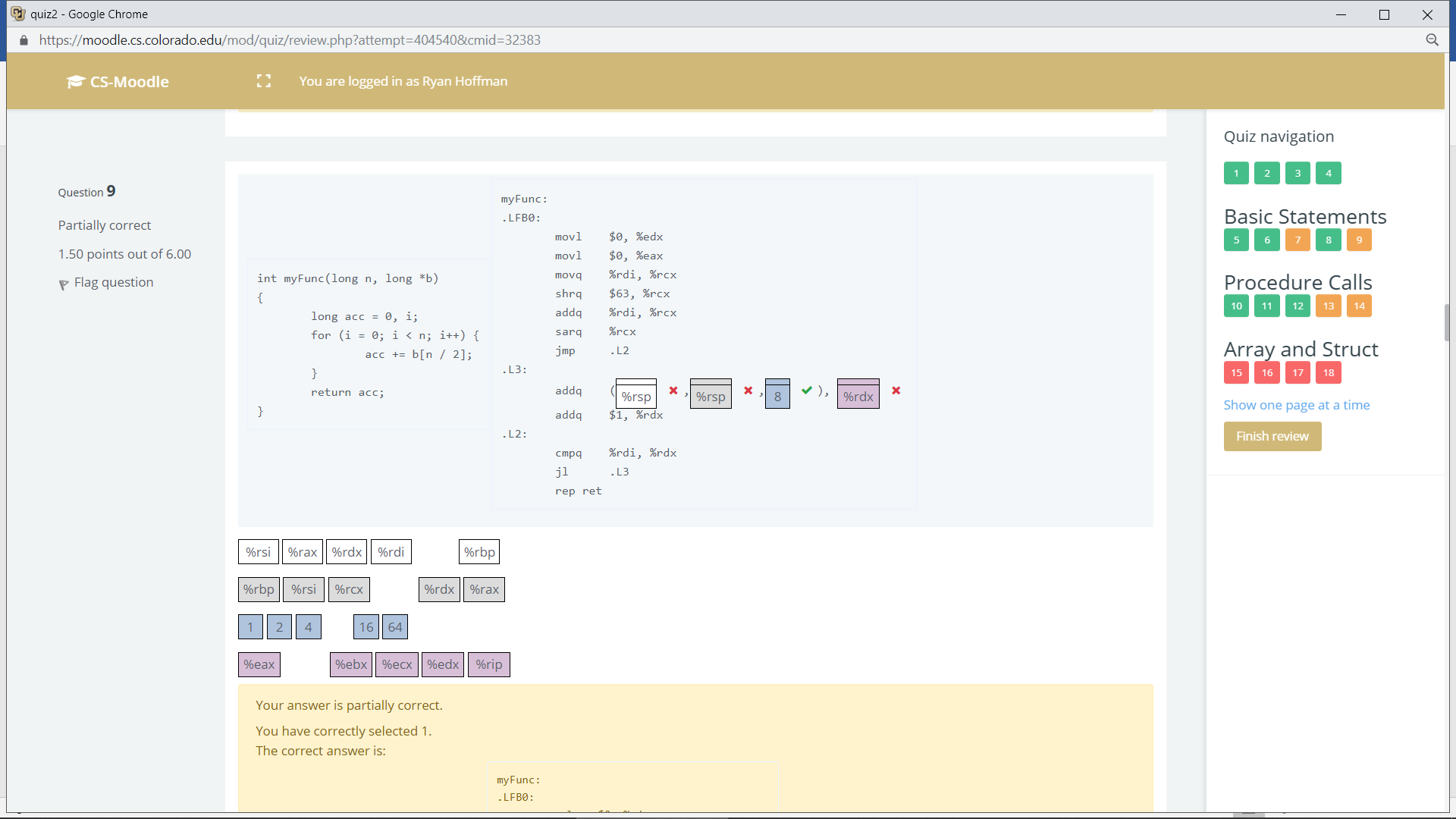
Q8

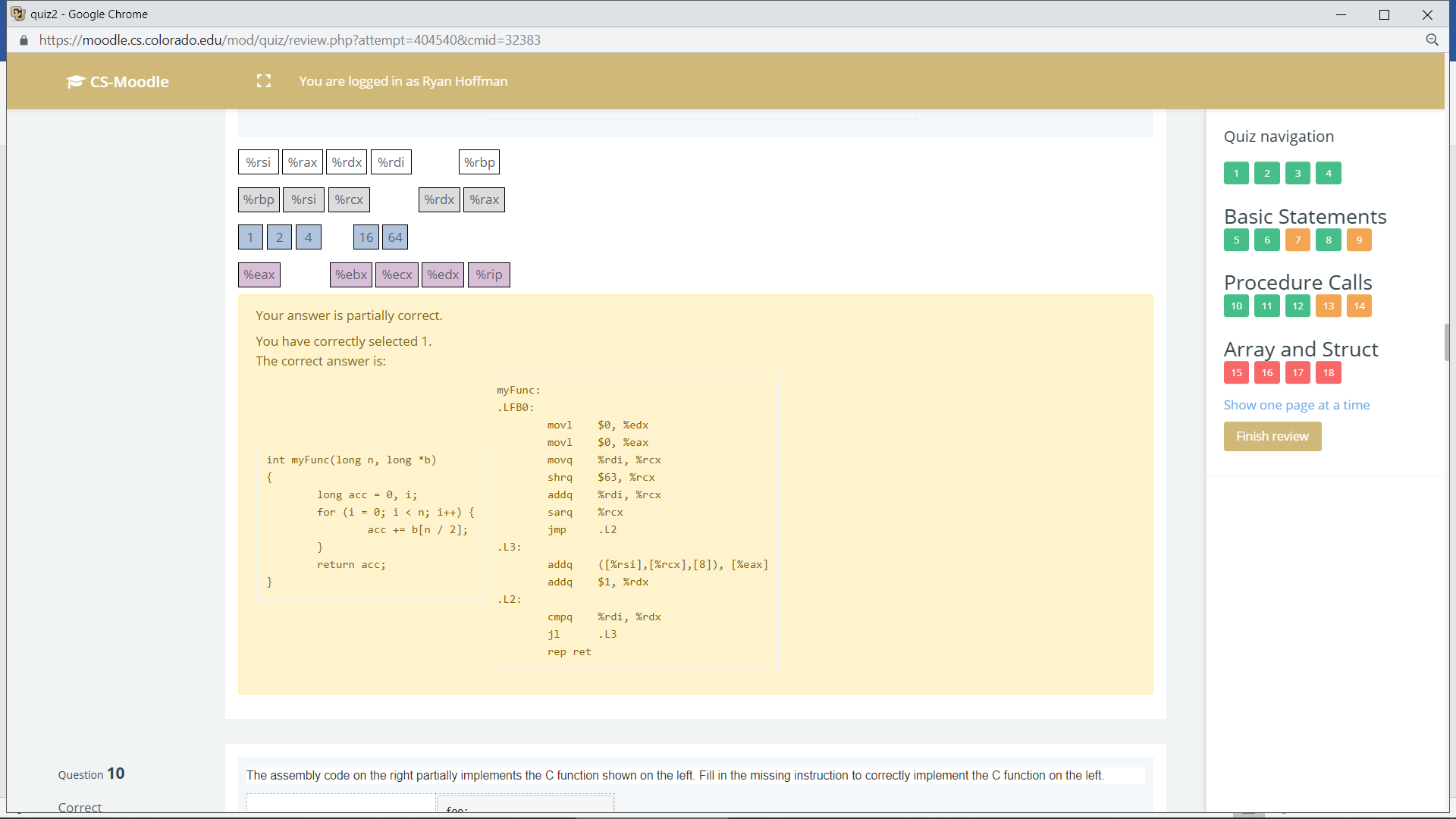


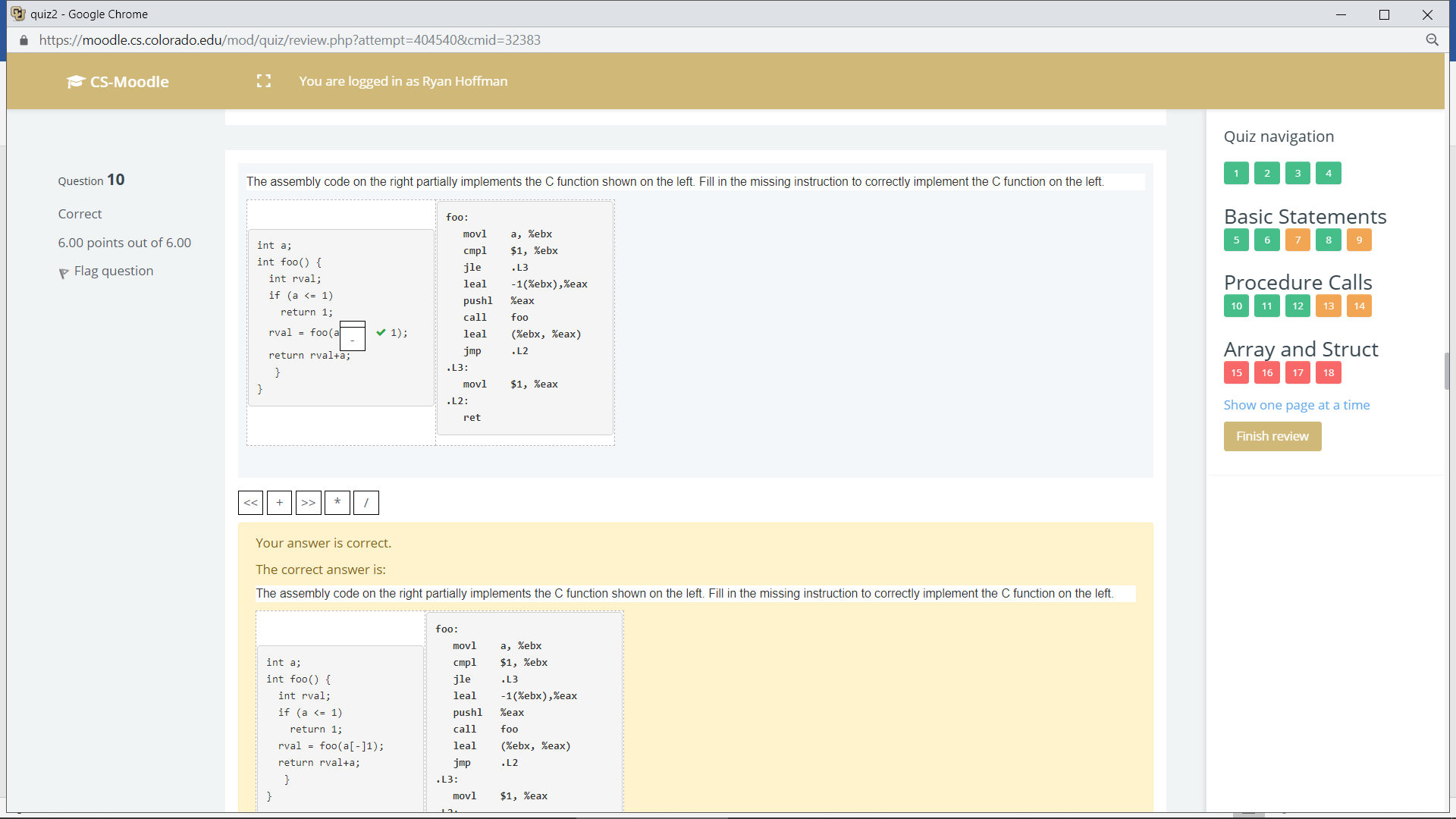


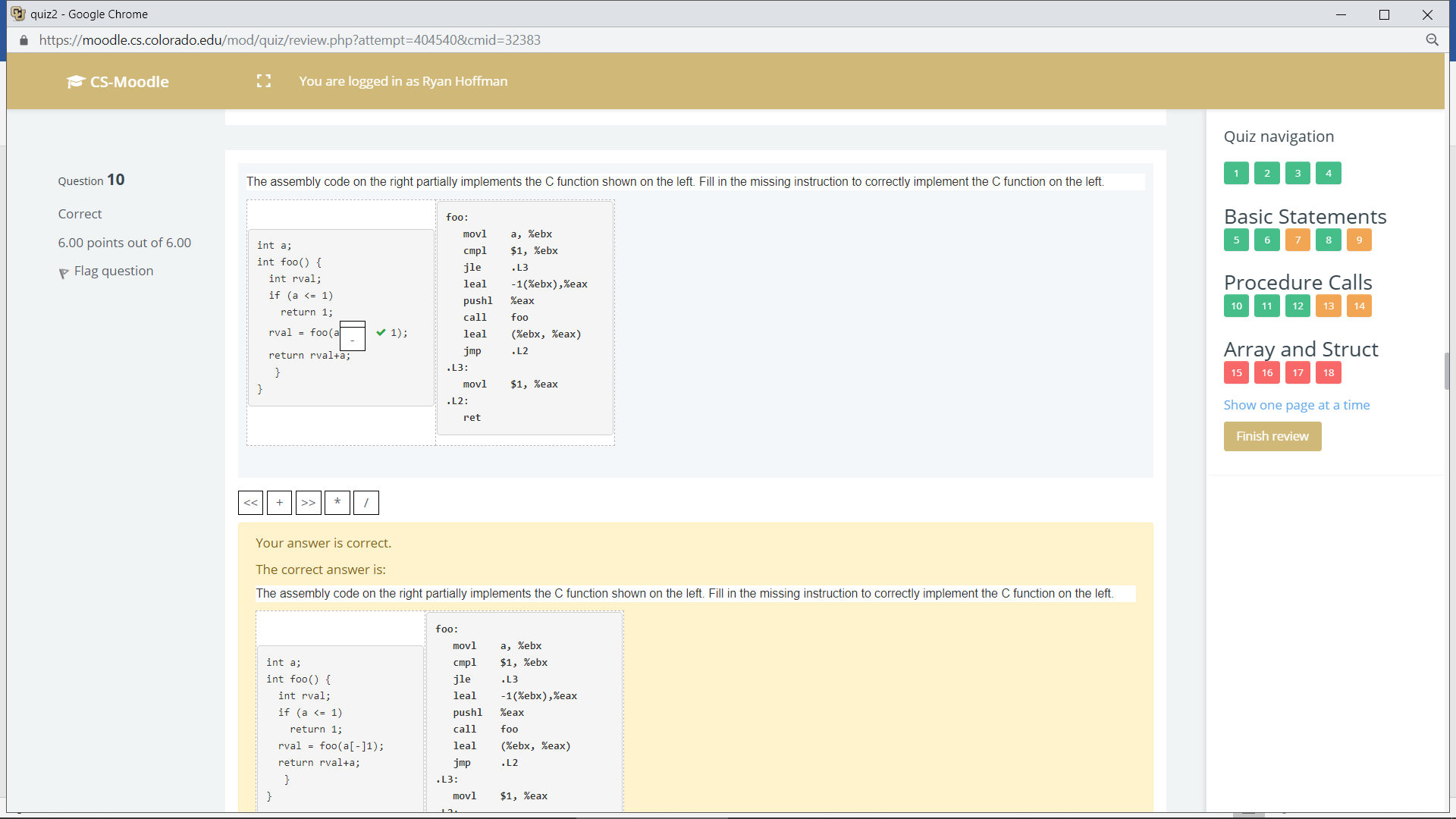


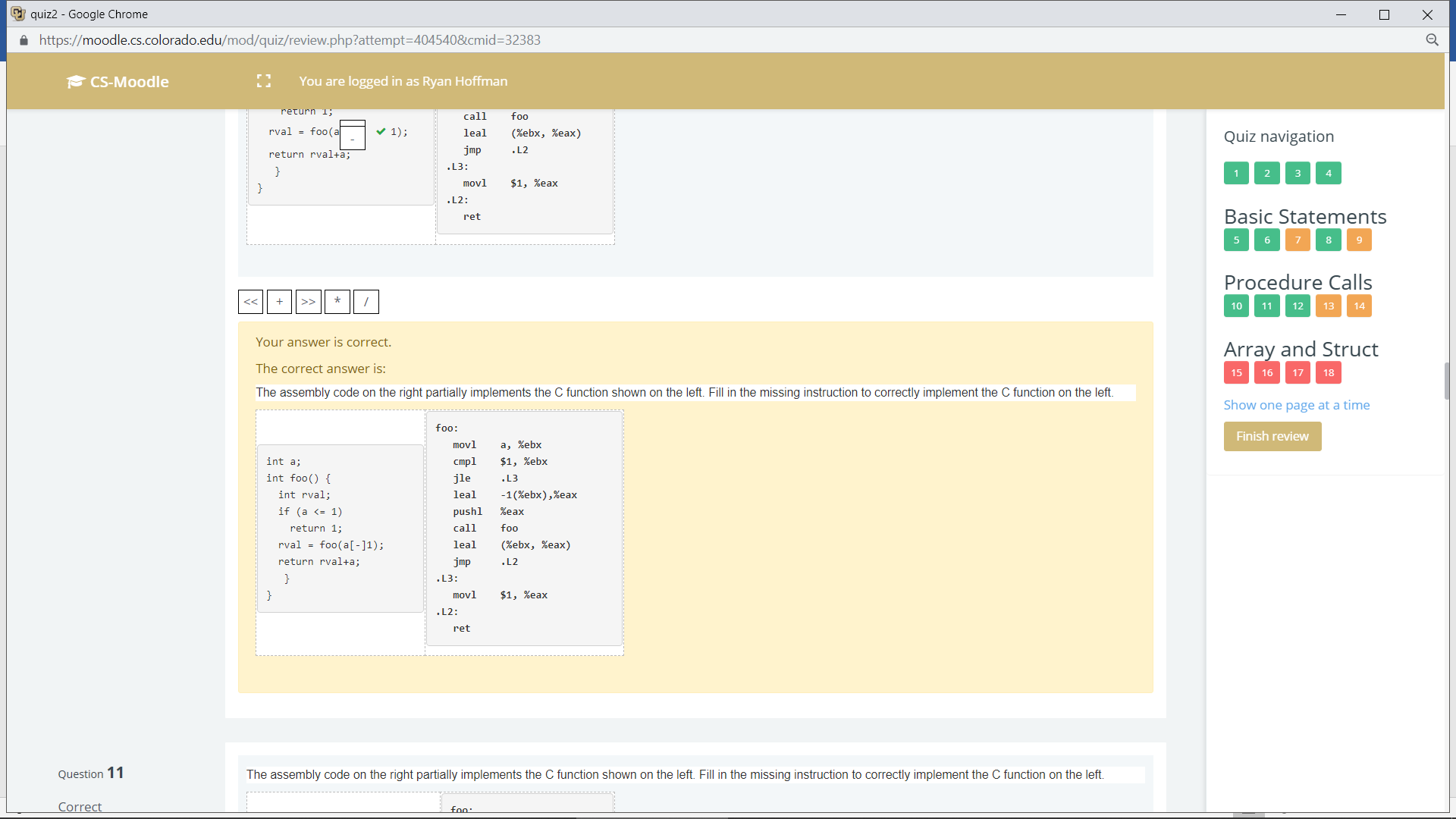
Q9











Q11

