Section 9: Exam Review
Function Calls

CS164

Drill Question

Consider this program. Match each piece of data to its location relative to *entry's* starting rsp. * 6 points

```
(define (add3 x y z)
   (+ x (+ y z)))

(print (add3 1 2 3))
```

	rsp - 0	rsp - 8	rsp - 16	rsp - 24	rsp - 32	rsp - 40	rsp - 48
y	○	○	○	○	○	○	○
entry's return address	○	○	○	○	○	○	○
x	○	○	○	○	○	○	○
padding cell for stack alignment	○	○	○	○	○	○	○
add3's return address	○	○	○	○	○	○	○
z	○	○	○	○	○	○	○

Stack Review:

Draw a Stack Diagram that Illustrates a Function Call

(Feel free to consult lecture notes on Functions)

Name 2 differences between calling C functions vs. native functions.

Why do we need to align the stack pointer when calling a native function?

Practice Question

In this question you will complete a pseudocode description of how to compile function calls in our language (like in class).

Instructions: For each line in the pseudocode, circle exactly one of the alternatives separated by a slash in each set of { curly braces }.

How to compile `(foo e1 e2)` assuming that our stack index is currently set to stack_index:

1. Compile e1 with the stack index starting at stack_base { + / - } { 8 / 16 / 24 / 32 / 40 / 48 }. The resulting value will be stored in rax.
2. Store the resulting value on the { stack / heap }.
3. Compile e2 with the stack index starting at stack_base { + / - } { 8 / 16 / 24 / 32 / 40 / 48 }. The resulting value will be stored in rax.
4. Store the resulting value on the { stack / heap }.
5. { incremented / decremented } Rsp by { stack_base / (stack_base + 16) / (stack_base - 16) / stack_index / (stack_index + 16) / (stack_index - 16) }
6. { Save Rdi to the stack / Save Rdi to the heap / nothing }
7. Call the label for foo.
8. { incremented / decremented } Rsp by { stack_base / (stack_base + 16) / (stack_base - 16) / stack_index / (stack_index + 16) / (stack_index - 16) }
9. { Restore Rdi from the stack / Restore Rdi from the heap / nothing }

Practice Question

In this question you will complete a pseudocode description of how to compile function calls in our language (like in class).

Instructions: For each line in the pseudocode, circle exactly one of the alternatives separated by a slash in each set of { curly braces }.

How to compile `(foo e1 e2)` assuming that our stack index is currently set to stack_index:

1. Compile e1 with the stack index starting at stack_base { + / <u>-</u> } { 8 / <u>16</u> / 24 / 32 / 40 / 38 }. The resulting value will be stored in rax.
2. Store the resulting value on the { <u>stack</u> / heap }.
3. Compile e2 with the stack index starting at stack_base { + / <u>-</u> } { 8 / 16 / <u>24</u> / 32 / 40 / 38 }. The resulting value will be stored in rax.
4. Store the resulting value on the { <u>stack</u> / heap }.
5. { <u>incremented</u> / decremented } Rsp by { <u>stack_base</u> / (stack_base + 16) / (stack_base - 16) / stack_index / (stack_index + 16) / (stack_index - 16) }
6. { Save Rdi to the stack / Save Rdi to the heap / <u>nothing</u> }
7. Call the label for foo.
8. { incremented / <u>decremented</u> } Rsp by { <u>stack_base</u> / (stack_base + 16) / (stack_base - 16) / stack_index / (stack_index + 16) / (stack_index - 16) }
9. { Restore Rdi from the stack / Restore Rdi from the heap / <u>nothing</u> }

Note: This question is somewhat ill-defined. Exam questions will have one well defined answer!

Problems from last week

For each expression below, will it result in a **compile-time** error, a **run-time** error, or a **valid** result when run in the **class compiler**?

A. ```
   (let ((v1 (vector 1 true)))
     (let ((v2 (vector 1 true)))
       (vector-get v1 1)))
   ```
B. (vector-length (pair 1 2))
C. (list? (pair 1 2))
D. (vector-get (if (not (char? #\a)) (vector 1 true) (vector 2 true)) 1)
   ```

Problems from Last Week

The OCaml code below implements a **left** projection operation in the compiler. Write **three** s-expressions of the form **(left …)**. One that will result in a **run time** error, one that will result in a **compile time** error, and one that will result in the **number 22**.

```
| Lst [Sym "left"; e] ->
    compile_exp tab stack_index e
    @ ensure_pair (Reg Rax)
    @ [Mov (Reg Rax, MemOffset (Reg Rax, Imm (-pair_tag)))]
```

Problems from Last Week

What does the **interpreter symbol table** look like at each point in the code?

```
(let ((z false) (x 10)
        do (A_EXP)
            (pair 1
                (if z
                    (let (x (+ x 1)) (let (y 9) B_EXP))
                    (let (y 8) (C_EXP))
                )
            (x + 1)
            (D_EXP)))
```

A_EXP: __{z:false; x:10}_______ C_EXP: _________

B_EXP: ______________________ D_EXP: _________

Problems from Last Week

What does the **compiler symbol table** look like?
(var stack index in order of appearance)

```
(let ((z false) (x 10)
        do (A_EXP)
            (pair 1
                (if z
                    (let (x (+ x 1)) (let (y 9) B_EXP))
                    (let (y 8) (C_EXP))
                )
            (x + 1)
            (D_EXP)))
```

A_EXP: __{z:1; x:2}_______ C_EXP: __________
B_EXP: ___________________ D_EXP: __________

Questions