# Feedback — Quiz #5

Help

You submitted this quiz on **Mon 19 May 2014 8:55 PM PDT**. You got a score of **8.00** out of **8.00**.

## Question 1

Using the simple language and code generation technique described in the lecture to evaluate the following 2 expressions:

E1: ( ( 3 - 2 ) * 6 ) + 5

E2: 5 + ( 6 * ( 3 - 2 ) )

Which of the following statements are true?

[Choose all that apply]

| Your Answer | | Score | Explanation |
|---|---|---|---|
| ☑ E1 and E2 require the same number of stack pop actions. | ✔ | 0.17 | They all require 3 stack pop actions. |
| ☐ E1 generates fewer MIPS instructions than E2. | ✔ | 0.17 | They all generate 19 MIPS instructions. |
| ☑ The code for E2 uses more stack space than the code for E1. | ✔ | 0.17 | The MIPS instructions for E2 will push 3 numbers to the stack without pop actions. |
| ☐ E1 generates more MIPS instructions than E2. | ✔ | 0.17 | They all generate 19 MIPS instructions. |
| ☑ E1 generates 19 MIPS instructions. | ✔ | 0.17 | |
| ☐ E2 generates 21 MIPS instructions. | ✔ | 0.17 | They all generate 19 MIPS instructions. |
| Total | | 1.00 / 1.00 | |

**Question Explanation**

Both E1 and E2 evaluations require:
4 acc⟵imm actions, 3 push acc actions, 3 pop actions, and 3 acc⟵top_of_stack op acc actions.
Using the simple code generation technique, we will generate:
4 * 1 + 3 * 2 + 3 * 1 + 3 * 2 = 19 MIPS instructions.

# Question 2

Consider the following assembly language generated using the simple code generation strategy described in the lecture.

```
foo_entry:
    move $fp $sp
    sw $ra  0($sp)
    addiu sp $sp -4
    lw $a0 8($fp)
    sw $a0  0($sp)
    addiu $sp $sp -4
    lw $a0 4($fp)
    lw $t1  4($sp)
    addiu $sp $sp 4
    beq $a0 $t1 true1
false1:
    lw $a0 12($fp)
    b endif1
true1:
    li $a0 17
endif1:
   lw $ra 4($sp)
   addiu $sp $sp #1
   lw $fp 0($sp)
   jr $ra
```

One argument to an instruction (#1) is missing. And consider the following expression from which the code was generated, with missing subexpressions #2 and #3.

def foo(x,y,z) = if y = #2 then 17 else #3

What are the values for #1, #2 and #3?

| Your Answer | | Score | Explanation |
|---|---|---|---|
| #1 = 16, #2 = x, #3 = z | | | |
| #1 = 20, #2 = z, #3 = x | | | |
| #1 = 12, #2 = x, #3 = z | | | |
| #1 = 20, #2 = x, #3 = z | ✔ | 1.00 | |
| #1 = 12, #2 = z, #3 = x | | | |
| #1 = 16, #2 = z, #3 = x | | | |
| Total | | 1.00 / 1.00 | |

**Question Explanation**

#1 = 4 * the number of parameters + 8 = 4*3+8=20; see explanations in lectures
#2 = x; #2 refers to the value stored in 4($fp), that is the first parameter, x
#3 = z; #3 refers to the value stored in 12($fp), that is the third parameter, z

# Question 3

Consider the following Cool program:

```
class Main {
  main():Object {
    (new Bar).foo()
  };
};
class Foo inherits IO{
  a:Int;
  foo():Object{self};
};
class Bar inherits Foo {
```

```
  b:Int;
  foo():Object {
    let i:Int←0 in
      while i < 10 loop{
        (new Bar)@Foo.foo();
        i ← i+1;
      } pool
  };
};
```

Using the object layout described in lecture, in order to represent the objects that exist when this program terminates, how many pointers in memory will need to point to the code for some implementation of the method "foo", and how many pointers in memory will need to point to an Int that is some object's "a" attribute? Assume that all objects created still exist when the program terminates.

| Your Answer | | Score | Explanation |
|---|---|---|---|
| ○ 11 pointers to "foo" methods, 11 pointers to "a" attributes | | | |
| ○ 11 pointers to "foo" methods, 0 pointers to "a" attributes | | | |
| ○ 1 pointers to "foo" methods, 11 pointers to "a" attributes | | | |
| ○ 2 pointer to "foo" methods, 0 pointers to "a" attributes | | | |
| ◉ 2 pointer to "foo" methods, 11 pointers to "a" attributes | ✔ | 1.00 | |
| ○ 1 pointers to "foo" methods, 2 pointers to "a" attributes | | | |
| Total | | 1.00 / 1.00 | |

**Question Explanation**

There are 11 Bar objects in memory, (1 generated in main(), 10 generated in foo()). There are 11 pointers to some object's attribute a.
There are 2 classes containing foo() function defined (Foo, Bar). In each one's dispatch table, there is a pointer to some implementation of the method foo(). In total, there are 2 pointers to some implementation of the method foo().

# Question 4

Using the simple language and code generation technique from the lectures and putting temporaries on the stack, what will be the size (in bytes) of the activation record for the following function (assuming temporaries are stored in the activation record):

```
def f(x, y, z, w):
  if x = (y + z)
    then 5
  else if x = (if y = z then f(y, z, x, 1) else x)
    then x
  else x * (y + (z - x))
```

| Your Answer | | Score | Explanation |
|---|---|---|---|
| 16 | | | |
| 28 | | | |
| 24 | | | |
| 12 | | | |
| 32 | | | |
| 36 | ✔ | 1.00 | |
| 20 | | | |
| Total | | 1.00 / 1.00 | |

**Question Explanation**

In the activation record, there are 1 old frame pointer, 4 parameters, 1 return address and 3 temporaries, which in total requires 9*4=36 bytes.

# Question 5

Given the following object definitions:

```
class A {
```

```
  x ⟵ Int;
  a() : Int { .. };
};
class B inherits A {
  y ⟵ Int;
  z ⟵ Int;
};
class C inherits B {
  c() : Int { .. };
};
```

What is the size of an object of class C in words using the COOL object layout strategy described in the video lectures?

| Your Answer | | Score | Explanation |
|---|---|---|---|
| 3 | | | |
| 5 | | | |
| 1 | | | |
| 7 | | | |
| 0 | | | |
| 6 | ✔ | 1.00 | |
| Total | | 1.00 / 1.00 | |

**Question Explanation**

There are 1 word for the class tag, 1 word for the object size, 1 word for the dispatch pointer, and 3 words for attributes.

# Question 6

Consider the following program.

```
Class F {{
  factorial(x:Int) : Int { if x = 1 then 1 else x * factorial(x-1) fi };
```

}

Class Main {

f:F ⟵ new F;

f.factorial(3);

}

Which of the following could be a valid step in the derivation of the dispatch to factorial in Main?

[Choose all that apply]

| Your Answer | | Score | Explanation |
|---|---|---|---|
| ☐ impl(F, factorial) = (x, x * factorial(x-1)) | ✔ | 0.25 | " x * factorial(x-1)" should be the body of the function factorial. |
| ☐ class(F) = (f:F ⟵ new F) | ✔ | 0.25 | "(f:F ⟵ new F)" should be "()" since class F doesn't have any attributes. |
| ☐ f, [x:l_x], [l_x ⟵ 3] \|- x = 1: Bool(true), [l_x ⟵ 3] | ✔ | 0.25 | "x = 1: Bool(true)" should be "x = 1: Bool(false)". |
| ☑ S_i+1 = S_i[2/l_x] | ✔ | 0.25 | It's a valid step when we bind the actual parameter 2 to the formal parameter x at calling function factorial(2). |
| Total | | 1.00 / 1.00 | |

# Question 7

Using the simple language, code generation technique described in the lecture to evaluate the following 2 expressions:

E1=e1+e2+e3+e4+e5

E2=e1+(e2+(e3+(e4+e5)))

Consider the conclusions about the number of temporaries(NT) needed in the evaluation of E1 and E2.

[Choose all that apply]

| Your Answer | Score | Explanation |
|---|---|---|

| | | |
|---|---|---|
| ☐ If all of e1, e2, e3, e4, and e5 are integer literals, NT(E1) > NT(E2). | ✔ | 0.14 |
| ☐ For any choice of e1, e2, e3, e4, and e5, NT(E1) > NT(E2). | ✔ | 0.14 |
| ☐ For different choices of e1, e2, e3, e4, and e5, the predicates NT(E1) < NT(E2), NT(E1) = NT(E2), and NT(E1) > NT(E2) can all be true. | ✔ | 0.14 |
| ☐ If all of e1, e2, e3, e4, and e5 are integer literals, NT(E1) = NT(E2). | ✔ | 0.14 |
| ☑ If all of e1, e2, e3, e4, and e5 are integer literals, NT(E1) < NT(E2). | ✔ | 0.14 |
| ☐ For any choice of e1, e2, e3, e4, and e5, NT(E1) < NT(E2). | ✔ | 0.14 |
| ☐ For any choice of e1, e2, e3, e4, and e5, NT(E1) = NT(E2). | ✔ | 0.14 |
| Total | | 1.00 / 1.00 |

**Question Explanation**

NT(E1) = max{NT(e1), NT(e2)+1, NT(e3)+1, NT(e4)+1, NT(e5)+1}
NT(E2) = max{NT(e1), NT(e2)+1, NT(e3)+2, NT(e4)+3, NT(e5)+4}
If all of e1, e2, e3, e4, and e5 are integer literals, NT(E1)=2 < 5=NT(E2)
For any choice of e1, e2, e3, e4 and e5, NT(E1) ≤ NT(E2)

# Question 8

Using the simple language, code generation technique described in the lecture to evaluate the following 2 expressions:

E1:

```
if e1=e2
  then if e3=e4 then e5 else e6
else
  if e3=e4 then e7 else e8
```

E2:

```
if e3=e4
  then if e1=e2 then e5 else e7
```

else

if e1=e2 then e6 else e8

Consider the conclusions about the number of temporaries(NT) needed in the evaluation of E1 and E2.Choose the correct one.

| Your Answer | | Score | Explanation |
|---|---|---|---|
| ○ For all choices of e1-e8, NT(E1) ＞ NT(E2). | | | |
| ◉ For all choices of e1-e8, NT(E1) = NT(E2). | ✔ | 1.00 | |
| ○ For all choices of e1-e8, NT(E1) ＜ NT(E2). | | | |
| ○ For difference choices of e1-e8, the predicates NT(E1) ＜ NT(E2), NT(E1) = NT(E2) and NT(E1) ＞ NT(E2) can all be true. | | | |
| Total | | 1.00 / 1.00 | |

**Question Explanation**

NT(E1) = max{NT(e1), NT(e2)+1, NT(e3), NT(e4)+1, NT(e5), NT(e6), NT(e7), NT(e8)}
NT(E2) = max{NT(e3), NT(e4)+1, NT(e1), NT(e2)+1, NT(e5), NT(e6), NT(e7), NT(e8)}
So for all choices of e1-e8, NT(E1) = NT(E2).