

Written Assignment 7

Due Thursday March 23, 2006

This assignment asks you to prepare written answers to questions on object layout and operational semantics. Each of the questions has a short answer. You may discuss this assignment with other students and work on the problems together. However, your write-up should be your own individual work. Remember that written assignments are to be turned in either at the start of lecture or in the CS164 homework box in 283 Soda by 12:30 PM on the due date.

Please write your name, your account name, your TA's name, and your section time on your homework! We need this information so that we can give you credit for the assignment and so that we can return it to you.

1. Consider the following Cool classes:

```
class A {
    attr1 : Int;
    attr2 : Int;
    method1() : Object { ... };
    method2() : Object { ... };
};

class B inherits A {
    attr3 : Int;
    method1() : Object { ... };
    method3() : Object { ... };
};
```

- (a) Draw a diagram that illustrates the layout of objects of type A and B, including their dispatch tables.
 - (b) Let `obj` be a variable whose static type is A. Assume that `obj` is stored in register `$a0`. Write MIPS code for the function invocation `obj.method2()`. You may use temporary registers such as `$t0` if you wish.
 - (c) Explain what happens in part (b) if `obj` has dynamic type B.
2. Suppose you wish to add arrays to Cool using the following syntax:

<code>let a:T[e₁] in e₂</code>	Create an array <i>a</i> with size <i>e</i> ₁ of <i>T</i> 's, usable in <i>e</i> ₂
<code>a[e₁] <- e₂</code>	Assign <i>e</i> ₂ to element <i>e</i> ₁ in <i>a</i>
<code>a[e]</code>	Get element <i>e</i> of <i>a</i>

Write the operational semantics for these three syntactic constructs. You may find it helpful to think of an array of type $T[n]$ as an object with n attributes of type T .

3. The operational semantics for Cool's `while` expression show that result of evaluating such an expression is always `void`. (See page 28 of the Cool manual.)

However, we could have used the following alternative semantics:

- If the loop body executes at least once, the result of the `while` expression is the result from the last iteration of the loop body.
- If the loop body never executes (i.e., the condition is false the first time it is evaluated), then the result of the `while` expression is `void`.

For example, consider the following expression:

```
while (x < 10) loop x <- x+1 pool
```

The result of this expression would be 10 if $x < 10$ or `void` if $x \geq 10$.

Write new operational rules for the `while` construct that formalize these alternative semantics.