Project Phase 2: Updates, Clarifications, and Suggestions

Note: In cases where this document and the phase 1 or 2 project descriptions disagree, the updates in this document supersede the original details.

Updates

2.21.14.a.
For semantic checking, you may safely ignore our previous rule that comparison operators have no association. There is no need to check this during your semantic checks.

2.21.14.b.
For IR generation, the following updates to Tiger need to be taken into account:

1. Tiger is 0-indexed.
2. For-loop control flow should work as it does in languages like python: The upper-bound is *exclusive*. For example, `for i := 0 to 10 do` should execute for i = 0, 1, 2, ..., 9.

3.22.14.a.
The intermediate representation for binary operations has changed. Part 4 has been slightly modified as a result. Use the following updated form:

```
op, x, y, z
```

This reads as, “Do operation op to the values y and z, and assign the new value to x.” A simple example of this can be given with the following:

```
2 * a + (b - 3)
```

The IR code for this expression would be the following:

```
sub, t1, b, 3
mult, t2, a, 2
add, t3, t1, t2
```

Here is the updated binary operation table:

Binary operation: (op, x, y, z)

Op	Example source	Example IR
add	a + b	add, t1, a, b
sub	a - b	sub, t1, a, b
mult	a * b	mult, t1, a, b
div	a / b	div, t1, a, b
and	a & b	and, t1, a, b

or	a | b	or, t1, a, b

3.23.14.a.
You may use the following additional IR instruction for initialization arrays:

Array Assignment: (op, x, size, value)

Op	Example source	Example IR
assign	var X : ArrayInt := 10; /* ArrayInt is an int array of size 100 */	assign, X, 100, 10