COMP520 sample final questions

Ismail Badawi

November 22, 2012

Optimization

For each of the following proposed peephole optimizations, either argue that it is correct in all cases, or describe and explain an example situation where it is unsound.

(a)

`ldc`	x			
`ldc`	y	$\Rightarrow$	`ldc`	$x + y$
`iadd`				

(b)

`ldc`	x			
`isub`		$\Rightarrow$	`ldc`	$x - y$
`ldc`	y		`isub`	
`isub`				

(c)

`iload`	`1`			
`putstatic`	`foo.x`		`iload`	`1`
`getstatic`	`foo.x`	$\Rightarrow$	`ldc`	i_1
`ldc`	i_1		`iadd`	
`iadd`			`putstatic`	`foo.x`
`putstatic`	`foo.x`			

(d)

`ineg`	$\Rightarrow$	
`ineg`		

(e)

`istore`	n		
`iload`	n	$\Rightarrow$	`ireturn`
`ireturn`			

Garbage collection

1. For each of mark and sweep collection, reference counting, and stop and copy, briefly

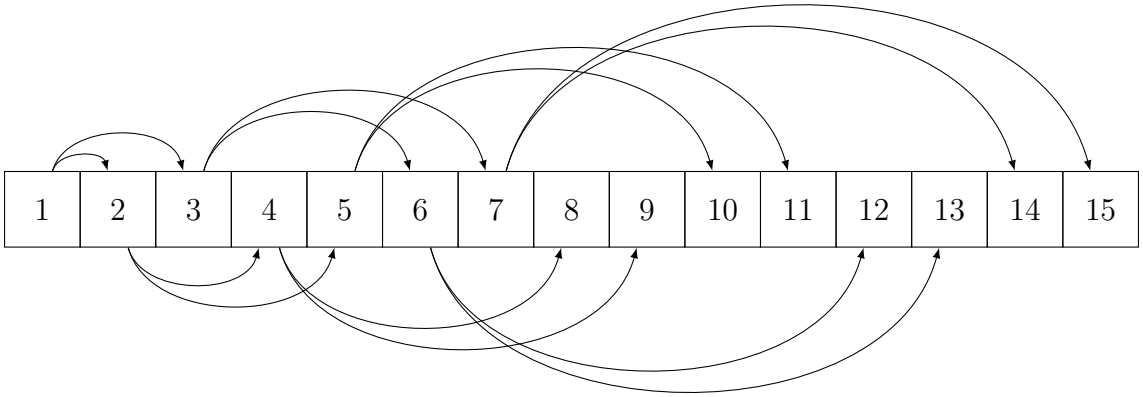
   (i) describe how the algorithm works

   (ii) state the primary strength of the algorithm

(iii) state the primary weakness of the algorithm, and possible ways to deal with it

2. A complete binary tree is created by allocating nodes starting from the root and following a breadth-first strategy (left to right).

   Suppose the tree created is of final size $2^4 - 1$, and is managed in a Java environment making use of garbage collection. The heap looks like this, where the nodes are labeled by order of creation; each square represents an object in the heap.



   Suppose that at some point after the tree is constructed, the right half of the tree becomes eligible for garbage collection (i.e. the pointer from 1 to 3 disappears).

   (a) Show what the heap would look like after collection by a mark and sweep garbage collector.

   (b) Show what the heap would look like after collection by a stop and copy collector. (Assume you have enough space in the heap.)

3. Consider this simple program:

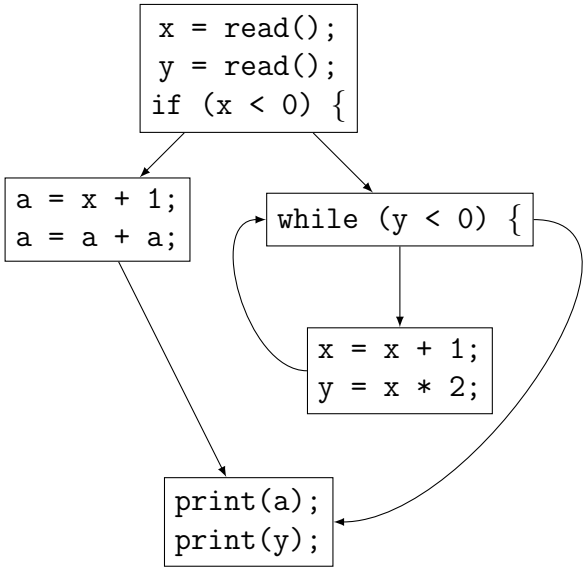
```
while (some_condition) {
    pointer = malloc(50 * 1024);
    do_something_with(pointer);
    // do_something_with has no side-effects
}
```

   Suppose you run this program with a heap size of 8 GB. Which garbage collection algorithm is preferable here (mark and sweep or stop and copy) and why?

Dataflow analysis

1. *Live variable analysis* is important to a variety of optimizations, most notably register allocation. Consider the following program and, on the right, its control flow graph:

```
x = read();
y = read();
if (x > 0) {
    a = x + 1;
    a = a + a;
} else {
    while (y < 0) {
        x = x + 1;
        y = x * 2;
    }
}
print(a);
print(y);
```



(a) Calculate live variable information on the above program. Show all dataflow equations and partial calculations.

(b) The above program uses 4 variables, a, b, x and y. Show how the liveness information can be used to improve basic block register allocation.

2. In class we briefly looked at available expressions analysis; a variant is *very busy expressions* analysis. An expression is *very busy* if it is computed on every path that leads from a program point, i.e. it's guaranteed that the expression will be computed at some time in the future. For example, in the following program:

```
S: x = 0;
   if (some_condition) {
       x = y+z;
   } else {
       w = y+z;
       x = 2*w;
   }
```

The expression `y+z` is very busy at program point S, because it's computed in every path starting from S, including both branches of the conditional.

(Computing very busy expressions at a program point is useful to do e.g. code hoisting; the `y+z` could be computed before the conditional and assigned to a temporary. (This is not quite common subexpression elimination as e.g. in the above example, `y+z` is not available at the start of either branch in the conditional). Hoisting can be useful to reduce code size, but its effect on runtime is not easy to predict in general.)

Describe how to compute the set of very busy expressions at every program point. In particular,

(i) Would this require a forwards or backwards analysis?

(ii) What is the merging operation? (This question can be stated many different ways: union vs intersection, may vs must, join vs meet, what is the merging operation, or confluence operation, etc. Just FYI)

(iii) What is the dataflow equation that should be used for a statement like `x = y + z;`?

Open ended

(This is an old final question. It's not really good preparation for the final, but feel free to come up with something and email it to me at `ismail.badawi@mcgill.ca` if you like.)

Describe an alternative to the WIG project that could be developed for some future version of this course. It should teach the same key lessons that the WIG project teaches, but may also cover more of the course syllabus. You should suggest a compiler for a domain specific language, one that may or may not currently exist; the JOOS project already provides an adequate example of a general purpose language. Your proposed project need not involve extensive program analysis and optimization.

This is an open-ended question. Your answer should be clearly written, the project should be feasible, and it should be interesting or "fun" in some way. You should provide examples of valid and invalid source code, grammar snippets, non-obvious (read: not C or Java-like) scope and type rules if there are any, and a description of how the back end (code generation and runtime system) works. Diagrams are welcome.

The important part is that your idea substitutes for the WIG project from a pedagogical perspective.