

# CMSC 430 (Spring 2009)

## Practice Problems 4

Use the following 3-address code and Java stack code instructions for answering code generation questions.

3-addr Instruction	Effect
load R1 x	$R1 \leftarrow x$
store x R1	$x \leftarrow R1$
add R1 R2 R3	$R1 \leftarrow R2 + R3$
sub R1 R2 R3	$R1 \leftarrow R2 - R3$
mult R1 R2 R3	$R1 \leftarrow R2 * R3$
neg R1 R2	$R1 \leftarrow -(R2)$

Java Stack Code	Effect
nop	none
ldc.int c	push constant $c$ onto stack
iload index(x)	push local variable $X$ onto stack
istore index(x)	pop stack, store in local variable $X$
iadd	pop 2 elems off stack, add, push
isub	pop 2 elems off stack, subtract, push
imult	pop 2 elems off stack, multiply, push
ineg	pop stack, negate, push
goto L	jump to handle $L$
ifeq L	pop stack, jump to handle $L$ if zero
if_icmpeq L	pop 2 elems, jump to $L$ if equal
if_icmpgt L	pop 2 elems, jump to $L$ if 1st greater
dup	duplicate top of stack
pop	pop top of stack
swap	swap top two positions of stack

### 1. Run-time environment

- What is a frame used for?
- Name six items of of run-time information stored in a frame. For each item, identify whether its value is set before the procedure is called, during procedure execution, or right before procedure return.
- Name one type additional type of information only stored in a frame for Java programs.
- When can storage for a variable be allocated in a frame?
- What advantage is obtained when allocating variables in a frame?
- Name two advantages of managing memory allocation manually in the code.
- Name two advantages of managing memory allocation automatically in the run-time system.
- Name two methods of managing memory allocation automatically in the run-time system.

### 2. Intermediate representations.

Consider the statements:

- $x := a + (b * a)$
- $x := a - ((b + a) * c)$

- Translate each into an AST
- Translate each into 3-address code
- Translate each into Java stack code
- Which representation is the most compact? Why?
- Which representation is easy to manipulate? Why?
- Which representation is hard to manipulate? Why?
- Which representation is closest to the input program? Why?

### 3. Code generation.

You are generating code for a Java stack machine. You are given the following grammar attributes and helper functions:

Attribute	Holds
AstNode.code	list of instructions
Function	Effect
genInst( $X$ )	create new instruction $X$ returns handle to instruction
append( ... )	concatenates lists of instructions

- What grammar actions needed to generate code for a C-style IF statement in the following production?  

$$\text{stmt} \rightarrow \text{IF} ( \text{exp} ) \text{stmtList} ;$$

$$\{ \text{stmt.code} = ??; \}$$
- What grammar actions needed to generate code for a C-style FOR loop in the following production?  

$$\text{stmt} \rightarrow \text{FOR} ( \text{stmt} ; \text{exp} ; \text{stmt} ) \text{stmt} ;$$

$$\{ \text{stmt.code} = ??; \}$$
- Write grammar actions needed to generate control code for an AND expression in the following production, using numerical value representation of booleans. Use *short circuiting*.  

$$\text{exp} \rightarrow \text{exp}_1 \text{ AND } \text{exp}_2$$

$$\{ \text{exp.code} = ??; \}$$
- Write grammar actions needed to generate control code for an NOR expression in the following production, using numerical value representation of booleans. Use *short circuiting*.  

$$\text{exp} \rightarrow \text{exp}_1 \text{ NOR } \text{exp}_2$$

$$\{ \text{exp.code} = ??; \}$$
- Write grammar actions needed to generate control code for an  $\geq$  (GEQ) expression in the following production, using numerical value representation of booleans.  

$$\text{exp} \rightarrow \text{exp}_1 \text{ GEQ } \text{exp}_2$$

$$\{ \text{exp.code} = ??; \}$$

---

4. Complex code generation.

- (a) Name two issues and solutions to generating code for function calls in C.
  - (b) Name two issues and solutions to generating code for array references in C.
  - (c) What code must the compiler generate for the code  
int i, a[ 100 ] ;  
...  
a[ i + 5 ] = 4 ;
  - (d) What code must the compiler generate for the code  
int foo ( int x ) ;  
...  
x = foo( i + 2 ) ;
    - i. Assuming foo() is call-by-value?
    - ii. Assuming foo() is call-by-reference?
-