Homework #7: CMPT-379

Distributed on Mon, Nov 20; Due on Mon, Nov 27

Anoop Sarkar – anoop@cs.sfu.ca

(1) **Code Generation (Second Steps)**:

In this section of the homework, you make the second step towards full code generation in **Decaf**. You will extend your previous code generation implementation to deal with method definitions (especially recursive methods) and method calls as well as global variables and global array variables (which are the only kind of arrays allowed in **Decaf**). As before, the target of the code generation step will be MIPS R2000 assembly language. You should augment your code generator for the fragment of **Decaf** from the last assignment to cover the following fragment of **Decaf** syntax.

⟨program⟩	→	**class** ⟨class-name⟩ **'{'** ⟨field-decl⟩ * ⟨method-decl⟩ * **'}'**
⟨class-name⟩	→	**id**
⟨field-decl⟩	→	⟨type⟩ { **id** | { **id** **'['** **intConstant** **']'** } }$^{+}$**,** **';'**
	|	⟨type⟩ **id** **'='** ⟨constant⟩ **';'**
⟨method-decl⟩	→	{ ⟨type⟩ | **void** } **id** **'('** [{ ⟨type⟩ **id** }$^{+}$**,**] **')'** ⟨block⟩
⟨block⟩	→	**'{'** ⟨var-decl⟩ * ⟨statement⟩ * **'}'**
⟨var-decl⟩	→	⟨type⟩ { **id** }$^{+}$**,** **';'**
⟨type⟩	→	**int** | **bool**
⟨statement⟩	→	⟨assign⟩ **';'** | ⟨method-call⟩ **';'** | ⟨block⟩ | **return** [⟨expr⟩] **';'**
⟨assign⟩	→	⟨lvalue⟩ **'='** ⟨expr⟩
⟨method-call⟩	→	⟨method-name⟩ **'('** [{ ⟨expr⟩ }$^{+}$**,**] **')'**
	|	**callout** **'('** **stringConstant** [{ **','** { ⟨callout-arg⟩ }$^{+}$**,** }] **')'**
⟨method-name⟩	→	**id**
⟨callout-arg⟩	→	⟨expr⟩ | **stringConstant**
⟨lvalue⟩	→	**id** | **id** **'['** ⟨expr⟩ **']'**
⟨expr⟩	→	⟨lvalue⟩ | ⟨method-call⟩ | ⟨constant⟩
	|	⟨expr⟩ ⟨bin-op⟩ ⟨expr⟩
	|	**'-'** ⟨expr⟩
	|	**'!'** ⟨expr⟩
	|	**'('** ⟨expr⟩ **')'**
⟨bin-op⟩	→	⟨arith-op⟩ | ⟨rel-op⟩ | ⟨eq-op⟩ | ⟨cond-op⟩
⟨arith-op⟩	→	**'+'** | **'-'** | **'*'** | **'/'** | **'<<'** | **'>>'** | **'%'** | **rot**
⟨rel-op⟩	→	**'<'** | **'>'** | **'<='** | **'>='**
⟨eq-op⟩	→	**'=='** | **'!='**
⟨cond-op⟩	→	**'&&'** | **'||'**

⟨constant⟩ → **intConstant** | **charConstant** | ⟨bool-constant⟩

⟨bool-constant⟩ → **true** | **false**

You should attempt to implement register spilling. However, I will not enforce this strictly, if your code generation step runs out of registers to use, your program can exit with an error message.

Save the MIPS assembly program to file `filename.mips` and run the simulator `spim` as follows:

```
spim -file <filename.mips>
```

Create the entire compiler pipeline which accepts **`Decaf`** code and produces MIPS assembly that can then be run using `spim` as shown above.