Ganna Shulika, Michael Noppinger, Benjamin Scherer

Compiler Construction Class
SS2016
Assignment N° 12 – FINAL ASSIGNMENT

N°	What and how we implemented it	Works?
1.	- implemented the bitwise operations *sll, srl, sllv, srlv* and we made *nop* equal to *sll* - used the selfie library functions to implement the bitwise operations. Fully implemented and working.	+
2.	- the scanner properly distinguishes < from << and > from >> - implemented in getSymbol(){...} - invented unique tokens for the new symbols SYM_LEFTSHIFT = , SYM_RIGHTSHIFT = - added a new production rule for the new symbols to the C* grammar: shiftExpression = simpleExpression { (">>" | "<<") simpleExpression } . - added a procedure for parsing the new symbols to starc according to the new production rule, used a copy of the procedure for parsing simple expressions as a template. Fully implemented and working.	+
3.	Changed the implementation of the leftShift and rightShift procedures using << and >> instead of * and /. The original semantics of both procedures is preserved. As required we also avoided calling the twoToThePowerOf procedure in our solution. Fully implemented and working.	+
4.	Implemented constant folding in arithmetic expressions. (eg. 2+2 is merged to 4) Introduced a grammar attribute that represents the value of an expression if available. Added methods createAttribute(); where we allocated the 2 addresses for the type and the value; added getter and setter methods, also added resetAttribute(attribut) method to put zeros there. Added a method loadConstantBeforeNonconstant() Used the attribute to evaluate constant expressions at compile time and delay code generation for arithmetic expressions accordingly. Represented the attribute by a call-by-reference parameter to the involved parsing procedures. Tested our code by including code snippets in main that are subject to constant folding. Fully implemented and working.	+
5.	Did the implementation for arrays and extended the grammar just like Niklaus Wirth in Compiler Construction. Fully implemented and working.	+
6.	Adapted grammer and parser to recognize two dimensional arrays. Adapted how to calculate the address accordingly. Fully implemented and working.	+
7.	Adapted Scanner to recognize structs and added symbol for structs.	~
8.	Implemented structs and struct access - BUT struggling with demonstration of self-compilation when implementing the symbol table with structs. We are still struggling with it although we know the theoretical background and how to calculate the addresses where the structs are stored. So we reverted back to a version without struct access to become self compiling again.	-
9.	We implemented the &&, || in gr_expression. But we did not implement ! and lazy evaluation.	~
10.	Lack of time and motivation.	-
11.	If the size of entry to be added is equal to symbol table entry size we reuse the memory. Fully implemented and working.	+

Additionally we reduced a lot of repeated code to make the gr_methods more readable and understandable, commented a lot the states of the machine while taking the new symbol.