Compilation Course 046266 Winter 2016-17

General:

The compiler from C-- to RISKI Assembly.
Lexical analysis is done using Lex program. Parsing with syntactic and semantic analysis is done using Bison. The hole project is written in C++ language, using standard Makefile to create an executable named rx-cc. The rx-cc compiler takes C-- file as command line argument and creates an .rsk file. A few .rsk files can be linked together by the command rx-linker <file names>. The post linkage file generated has .e extension and can be run on the virtual machine by the command rx-vm <.e file>.

Files:

newParser.hpp:
Contains the main data structures (classes), their member functions implementation, the Stype definition, a list of helper functions (which are implemented in the .cpp file) and a nice drawings of the global data structures. We mainly used the standard C++ std libreries (like std::list, map, vector and so on).

newParser.cpp:
Contains the helper functions implementation and the main of the compiler.

Data Structures:

The global data structures are fully described in newParser.hpp by a drawing and a textual description.
The Stype is also located in the same file, and each field is annotated. We didn't have enough time to convert it into union (which is required) at the end.

We implemented two level type hierarchy:
types - first level in the hierarchy: Type represents a general type of C--. it can be primitive or complex. StructType derives from Type and holds a map of type fields (id, Type). These classes are instantiated during typedef declarations in C--.
Instances - second level: Variable represents a general variable of C--. it can be primitive or complex. Defstruct derives from Variable and holds a map of instance fields (id, Variable). These classes are instantiated during variable declarations in C--.

Block represents a scope in the C-- code. Each Block has it's symbol table and a pointer to it's parent Block. Function derives from Block and saves more information about saved registers, the function implementation, call lines for the linker and more.

Stack:

newParser.hpp contains a drawing of the stack. We decided that the caller is the only responsible for saving it's state before calling another function. We hold FP at I2 and SP on I1, while I0 is the machine defined register for JLINK instruction.