2023年编译原理实验任务

实验序号	实验内容	时间要求及分值
实验一	**编译工具链，Sysy 语言及其运行时库函数，目标平台 ARM 及 RISC-V** **共 4 个实训，前 2 个必选，后两个任选一个完成。**	截止日期：4-23 23:30 总分：15 分
1-1	编译工具链	8 分
任务	(1) gcc 的用法； (2) clang 的用法； (3) 交叉编译器： arm-linux-gnueabihf-gcc 和 riscv64-unknown-elf-gcc 虚拟机： qemu-arm 和 qemu-riscv64 等工具的使用； (4) make 的使用。	2 分 2 分 2 分 2 分
1-2	Sysy 语言及运行时库	4 分
任务	Sysy 语言程序： “买卖股票的最佳时机”程序	
1-3	ARM 平台及汇编	3 分
任务	ARM 汇编程序： 数组排序	(1-3 与 1-4 任选一个)
1-4	Risc-V 平台及汇编	3 分
任务	Risc-V 汇编程序： 数组排序	(1-3 与 1-4 任选一个)

实验序号	实验内容	时间要求及分值
实验二	Sysy 语言的词法分析 共 3 个实训，任选 1 个完成。	截止日期：4-30 23:30 总分：15 分
2-1	基于 flex 的词法分析器(C 语言实现)	15 分
任务	完善 flex 词法规则(.l)和对应动作，用 C 语言实现 Sysy 语言的词法分析 考核点：标识符识别，int/float 字面量的识别(含八进制，十六进制)，词法错误识别	
2-2	基于 flex 的词法分析器 (C++ 实现)	15 分
任务	完善 flex 词法规则(.l)和对应动作，用 C++语言实现 Sysy 语言的词法分析。 考核点：标识符识别，int/float 字面量的识别(含八进制，十六进制)，词法错误识别	
2-3	基于 antlr 的词法分析器 (C++实现)	15 分
任务	完善 antrl4 的词法规则(.g4)和词法分析主程序，在识别出 token 后输出规则定的词法信息。 考核点：标识符识别，int/float 字面量的识别(含八进制，十六进制)，词法错误识别	

实验序号	实验内容	时间要求及分值
实验三	**Sysy 语言的语法分析** **共 3 个实训，任选 1 个完成。**	截止日期：5-14 23:30 总分：40 分
3-1	基于 flex+Bison 的语法分析器(C 语言实现)	40 分
任务	完善 bison 的语法规则(.l)和对应动作，用 C 语言实现 Sysy 语言的语法检查和分析，构建 AST: (1) 正确报告语法错误； (2) 在没有语法错误的情形出，输出 AST。 考核点：Sysy2022 文法中的语句(Stmt)各产生式的表达和对应语义处理。	20 分 20 分
3-2	基于 flex+bison 的语法分析器 (C++实现)	40 分
任务	完善 bison 的语法规则(.l)和对应动作，用 C++语言实现 Sysy 语言的语法检查和分析，构建 AST: (1) 正确报告语法错误； (2) 在没有语法错误的情形出，输出 AST。 考核点：Sysy2022 文法中的语句(Stmt)各产生式的表达和对应语义处理。	20 分 20 分
3-3	基于 antlr 的词法分析器 (C++实现)	40 分
任务	完善 antlr4 的语法规则(.g4)和语法分析主程序，完成语法检查和分析，并构建 AST: (1) 正确报告语法错误； (2) 在没有语法错误的情形出，输出 AST。 考核点：Sysy2022 文法中的语句(Stmt)各产生式的表达和对应语义处理。	20 分 20 分

实验序号	实验内容	时间要求及分值
实验四	**Sysy 语言的 LLVM IR 中间代码生成** **共 2 个实训，任选 1 个完成。**	截止日期：5-28 23:30 总分：20 分
4-1	C 语言实现 Sysy 语言 LLVM IR	20 分
任务	将没有词法、语法、静态语义错误的 Sysy 语言程序翻译成 LLVM IR 中间代码.	
4-2	C++语言实现 Sysy 语言 LLVM IR	20 分
任务	将没有词法、语法、静态语义错误的 Sysy 语言程序翻译成 LLVM IR 中间代码.	

实验序号	实验内容	时间要求及分值
实验五	**Sysy 语言的目标代码生成** **共 2 个实训，任选 1 个完成。**	截止日期：6-10 23:30 总分：10 分
5-1	由 LLVM IR 生成 ARM 目标代码	10 分
任务	将符合 LLVM IR 规范的中间代码翻译成 ARM 目标代码. 不限制调用 LLVM 的 API。	
5-2	由 LLVM IR 生成 RISC-V 目标代码	10 分
任务	将符合 LLVM IR 规范的中间代码翻译成 RISC-V 目标代码. 不限制调用 LLVM 的 API。	

实验总成绩=上机成绩*0.8 + 实验报告*0.2