Group 14

Generating Compiler Phases from Specifications of Intermediate Representations (IRs)

Sponsored By : IIT Bombay

Internal mentor : Dr. Chhaya Gosavi
External mentor : Dr. Uday Khedkar (IITB Professor)

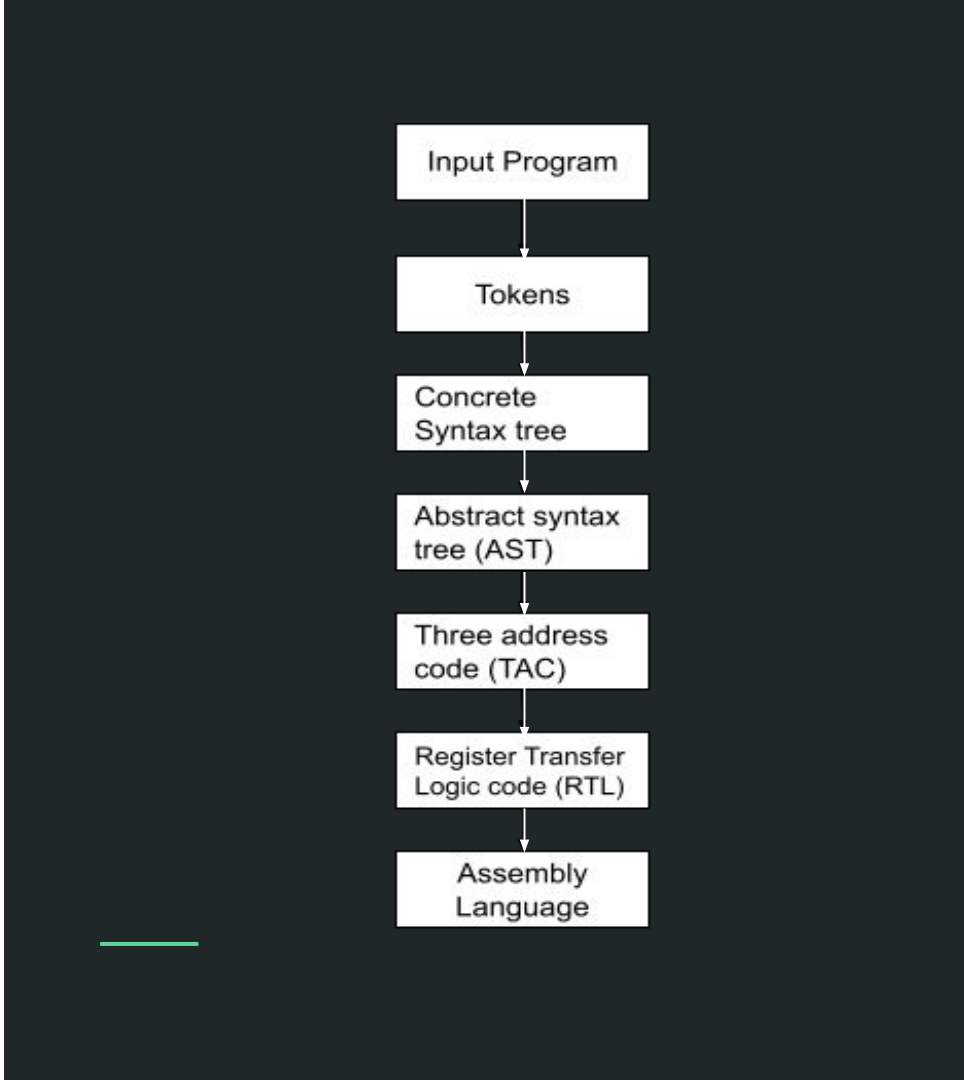
Group members - Sai Ghule (4332), Bhagyashree Rane (4371), Shravasti Deore (4916)

Agenda

- Problem Statement
- Motivation
- What is SCLP?
- Objectives for the Project
- System Architecture
- Test Cases
- UML diagrams
- System Requirement Specification
- Implementation Aspects
- Literature Survey
- Future Enhancements
- Documents and Process Followed
- References

Problem Statement

Generating Compiler Phases from Specifications of Intermediate Representations of SCLP

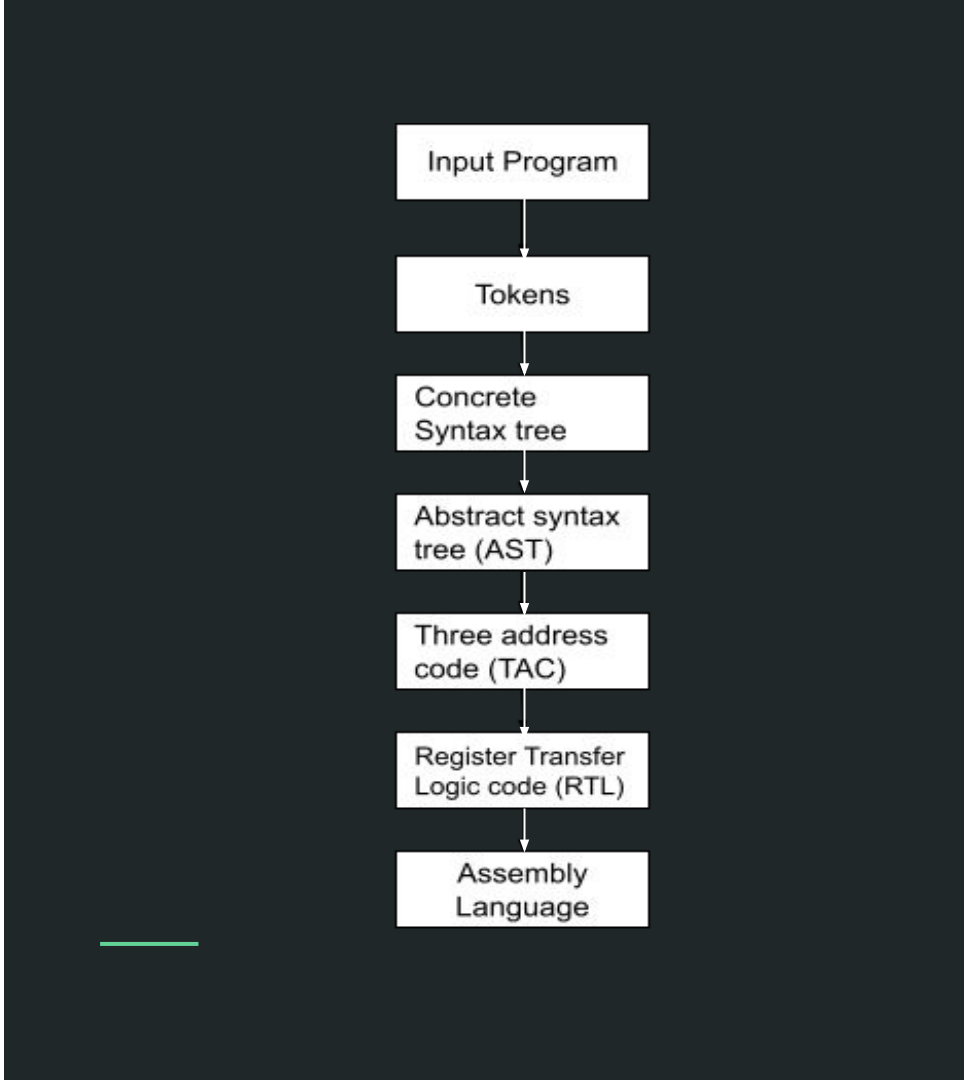


Motivation

- The motivation behind the implementation of SCLP has been to create a small well-crafted code base for a UG compiler construction lessons such that it can be enhanced systematically by the students.
- The focus is on incremental construction with some increments coming from language features and some coming from the phases in compilation.
- Generating Compiler Phases from Specifications of IRs is one such incremental construction.
- A secondary goal of sclp is to support interpretation to understand the relationship between compilation and interpretation first-hand.

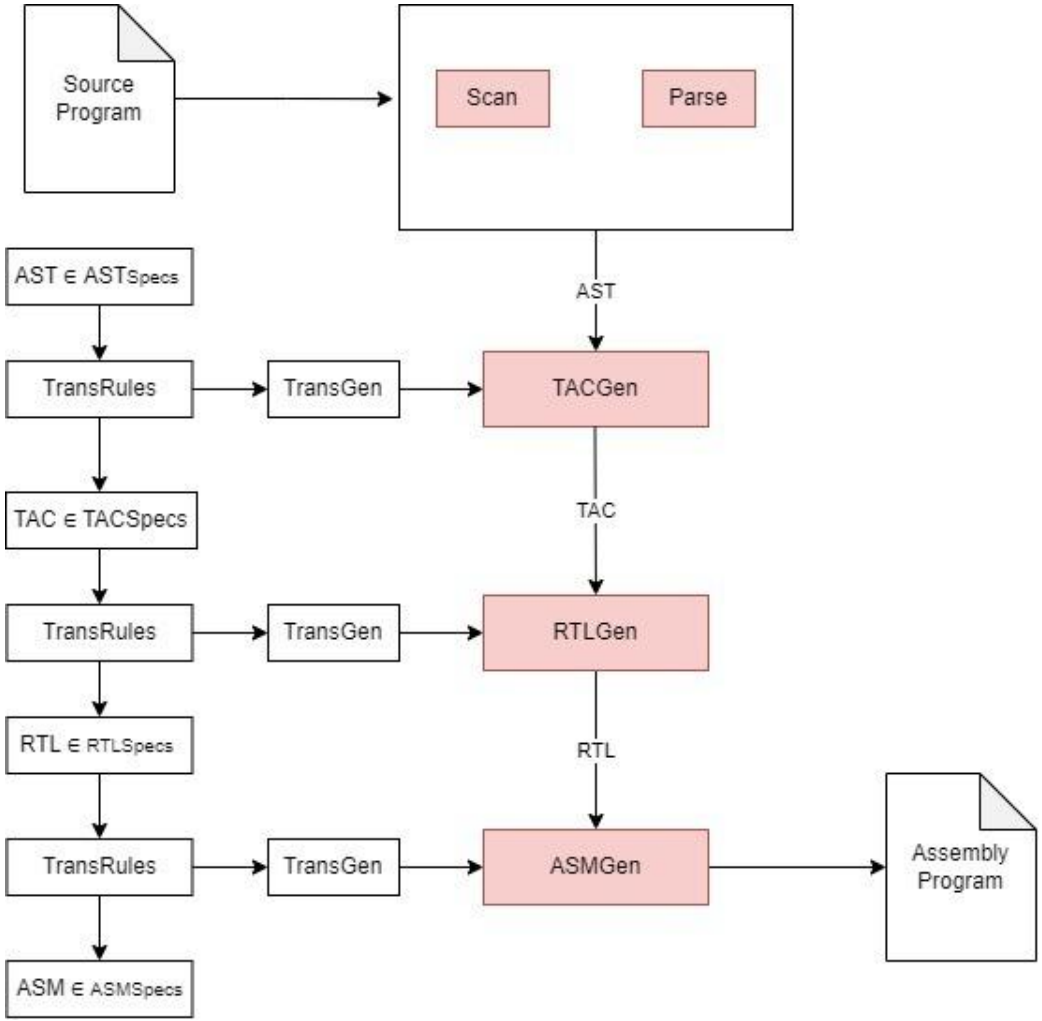
SCLP Compiler

SCLP is a language processor for a small C-like language implemented for UG courses at IIT Bombay.



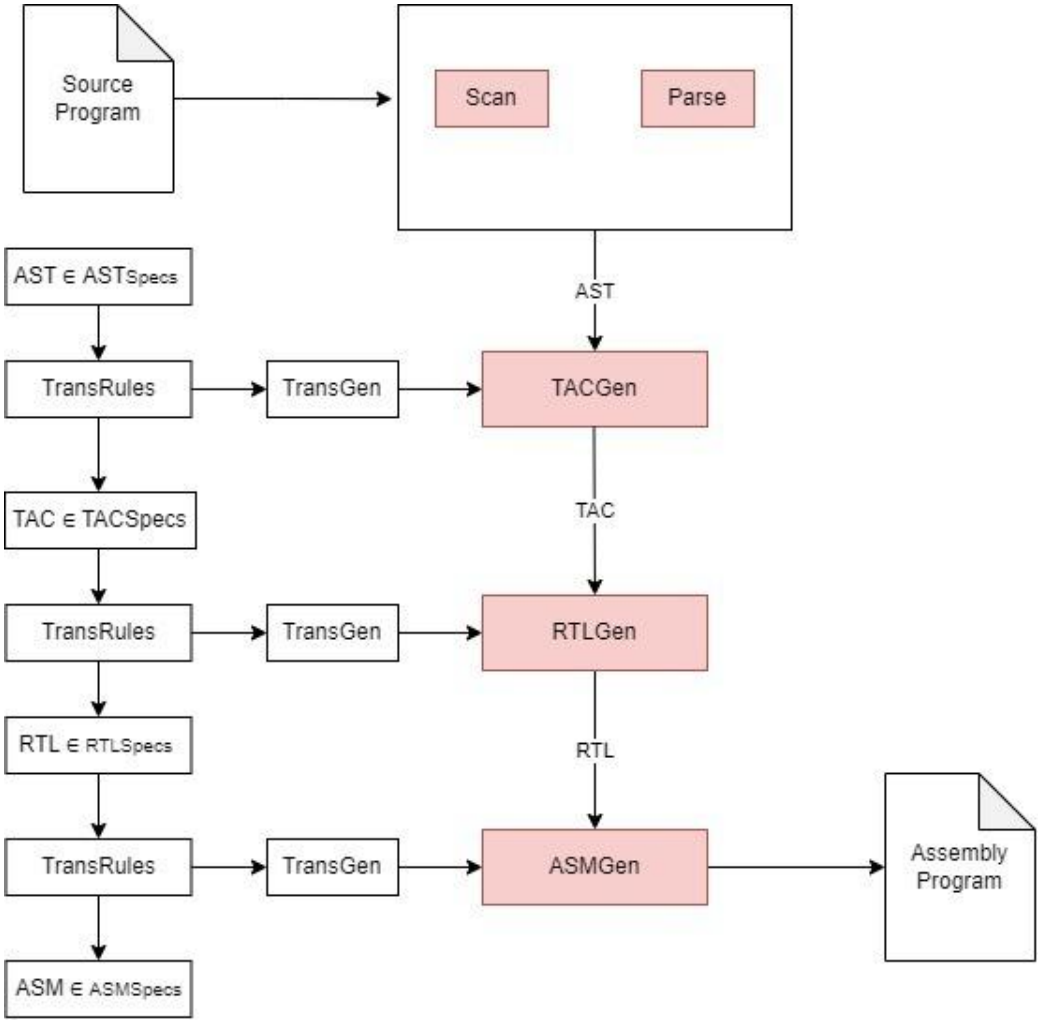
Objectives

- To make the already existing ASM specification documents cleaner.
- Create specification documents for AST and TAC.
- Generate Translation Rules from specifications of AST and TAC.
- Write TransGen for TACGen and RTLGen.
- ASMGen generated for a RISC processor using RIPES simulator.
- Integrate our work to current SCLP implementation.



System Architecture

High Level Design



Low level Design

- Change the simulator from spim to ripes
  - Specifications for SPIM
  - Specifications for RIPES
  - Specifications for RTL (need to study how RTL will change for ripes, can we keep the same RTL?)
  - Translation rules from RTL to ASM
  - Machine readable form of the specifications and translation rules
  - Implementation of the translator generator

Low level Design

- Generate TAC generator
  - Specifications for AST
  - Specifications for TAC
  - Translation rules from AST to TAC
  - Machine readable form of the specs and translation rules
  - Implementation of the translator generator

Low level Design

- Generate RTL generator
  - Specifications for TAC
  - Specifications for RTL
  - Translation rules for converting TAC to RTL
  - Machine readable form of the specs and translation rules
  - Implementation of the translator generator

Test Cases

Test Cases

Test Case Name	Purpose of the Test	Input	Expected Output
Test .c programs	To check if RIPES assembly code generates output	Level 1 to Level 6 .c programs	The programs successfully executed with desired outputs
Test .c programs for floating point calculations	To check if RIPES target generates output	Any .c program with floating point operation	The program should generate error
RTL to ASM	To check if RTLSpecs correct translation rules	Three address code	Should match with the already produced results

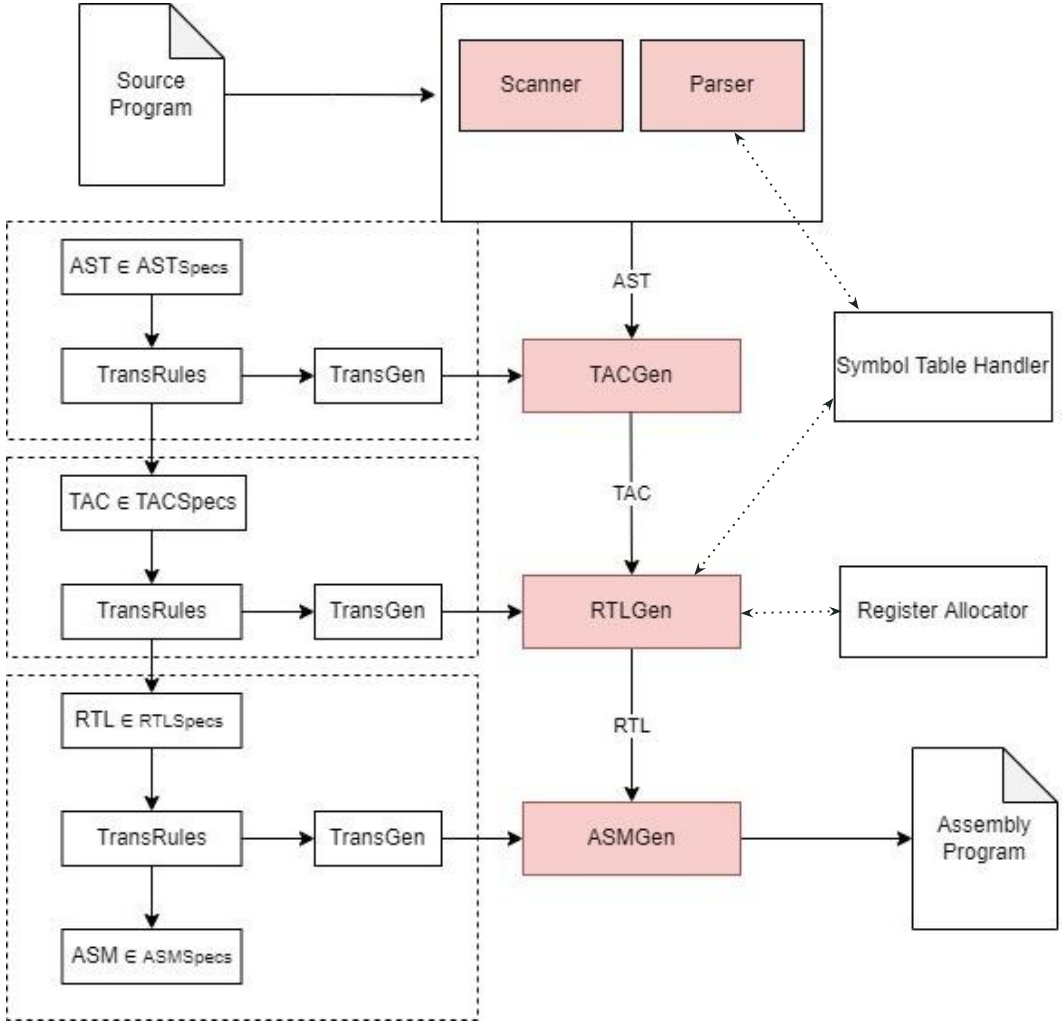
Test Cases

- Tested Level 1 to Level 6 valid and invalid codes.
- Test Cases for .spim target assembly for SPIM would be different to the .s assembly for RIPES
- The one and the only difference between the two types of test cases would be the use of floating point numbers.
- The test cases for RIPES is such that the programs would not contain floating point operations.
- The test cases for translating RTL to ASM would include RTL codes generated from the above mentioned source codes.
- Similarly, the test cases for translating AST to TAC would include AST statements generated from the same source codes.

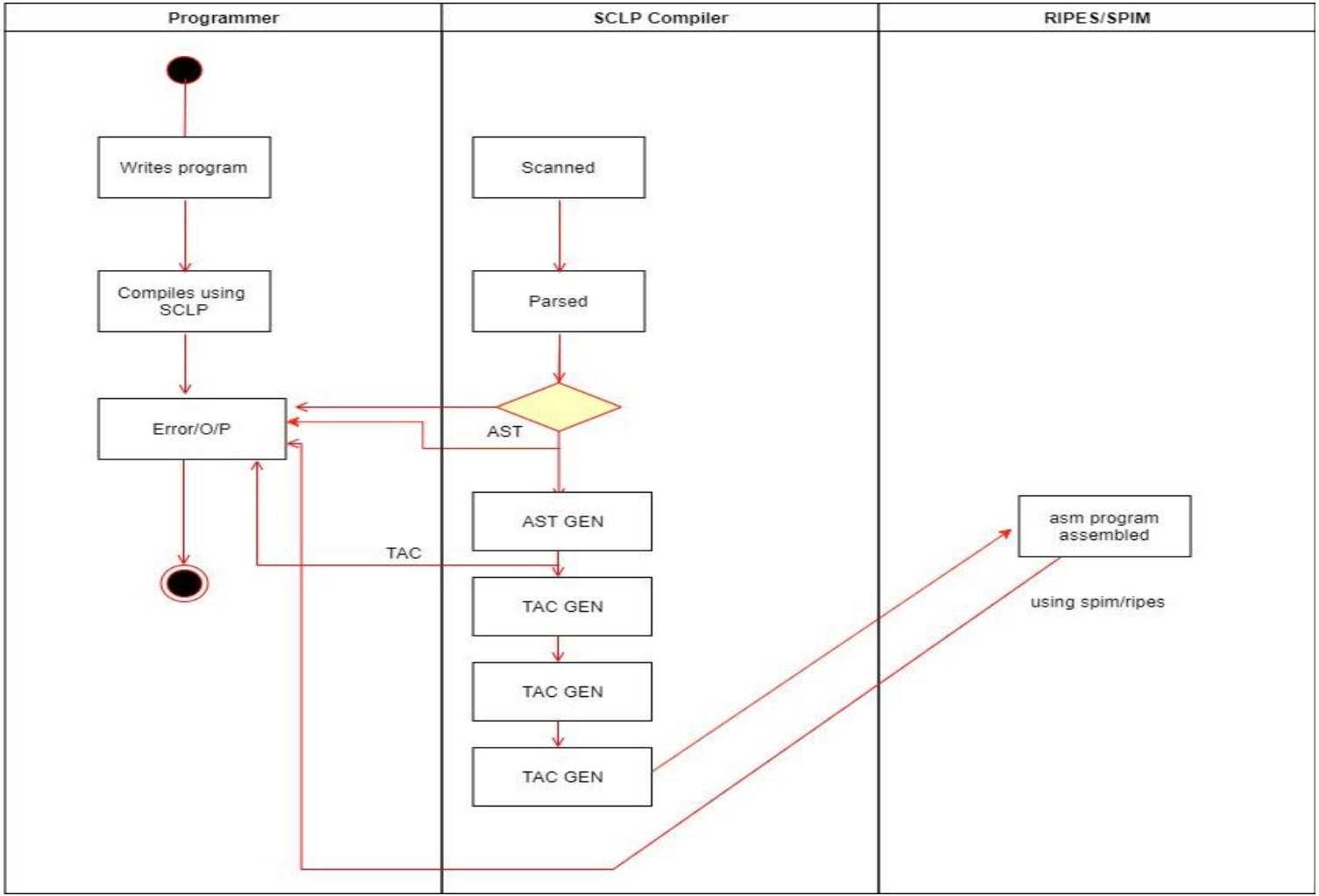
UML Diagrams

Component Diagram

The dotted boxes are the Specification Translators

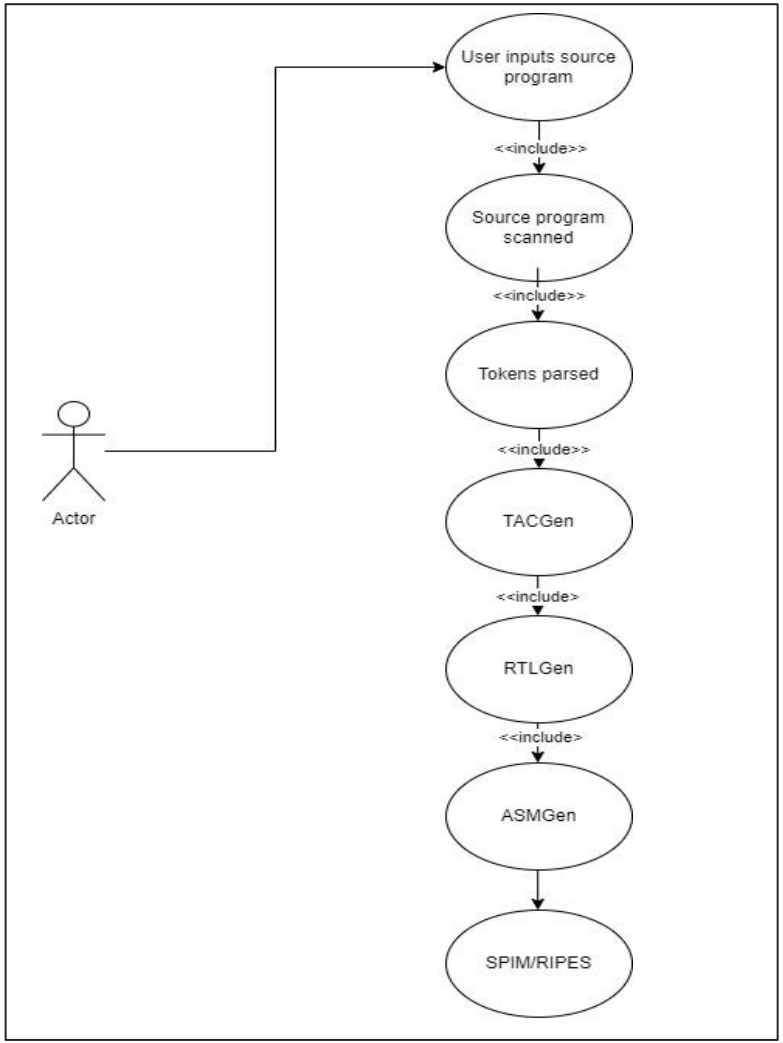


Activity Diagram

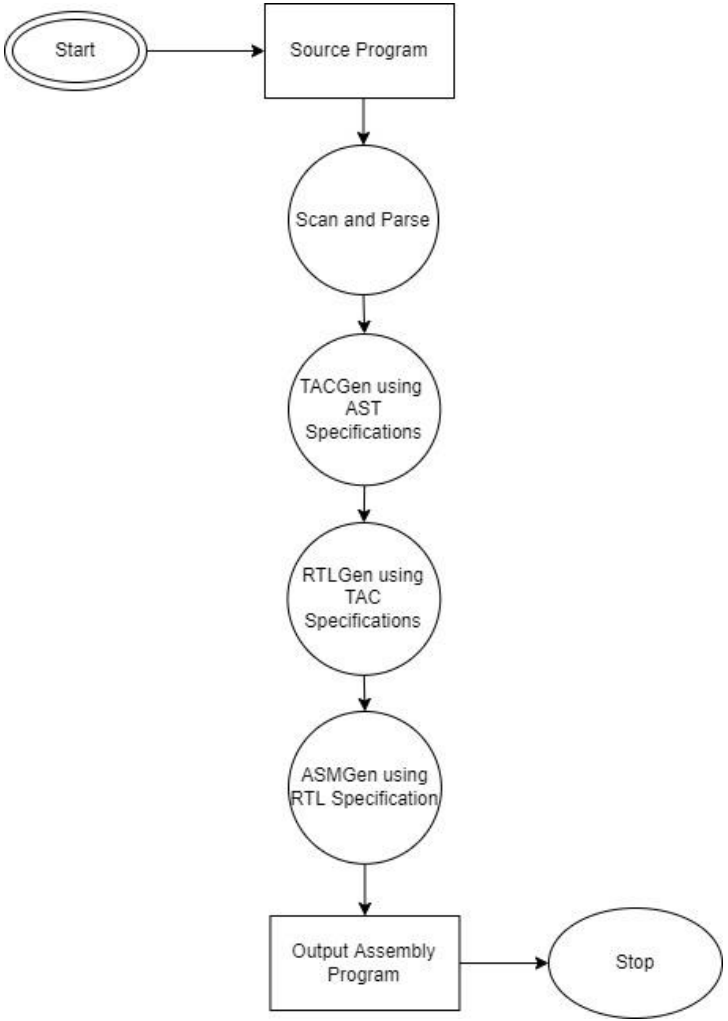


SCLP Use Case Diagram

Name	SCLP compiler
Description	Source program compiles through auto generated IRs
Pre Condition	The source program should be in the C-like language defined
Sequence	• Source code written, scanned and Parsed • TACGen, RTLGen, ASMGen generated from the specifications provided internally • Target assembly generated
Post Condition	The output of the source program generated

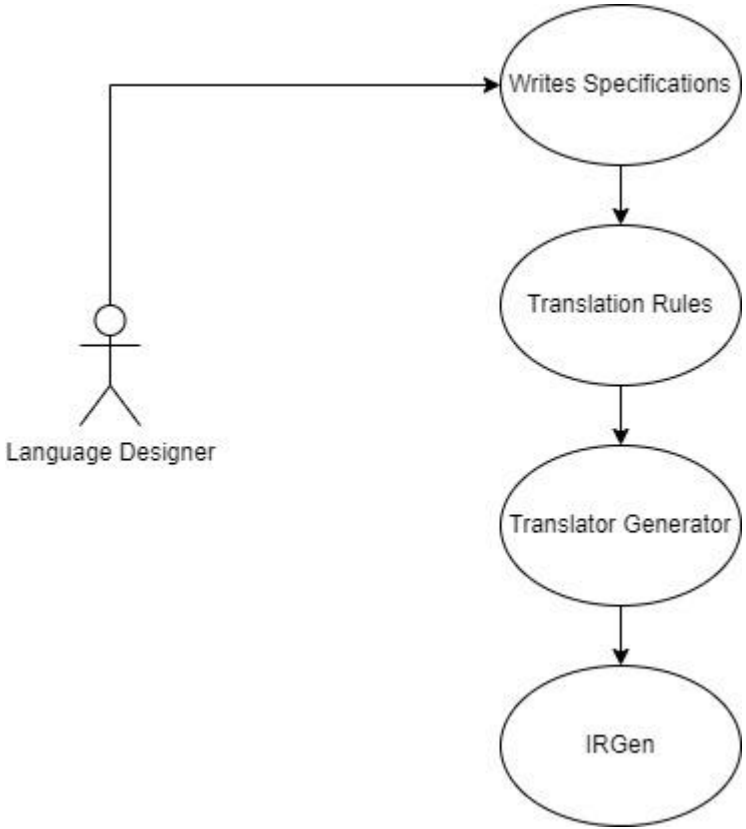


Data Flow Diagram

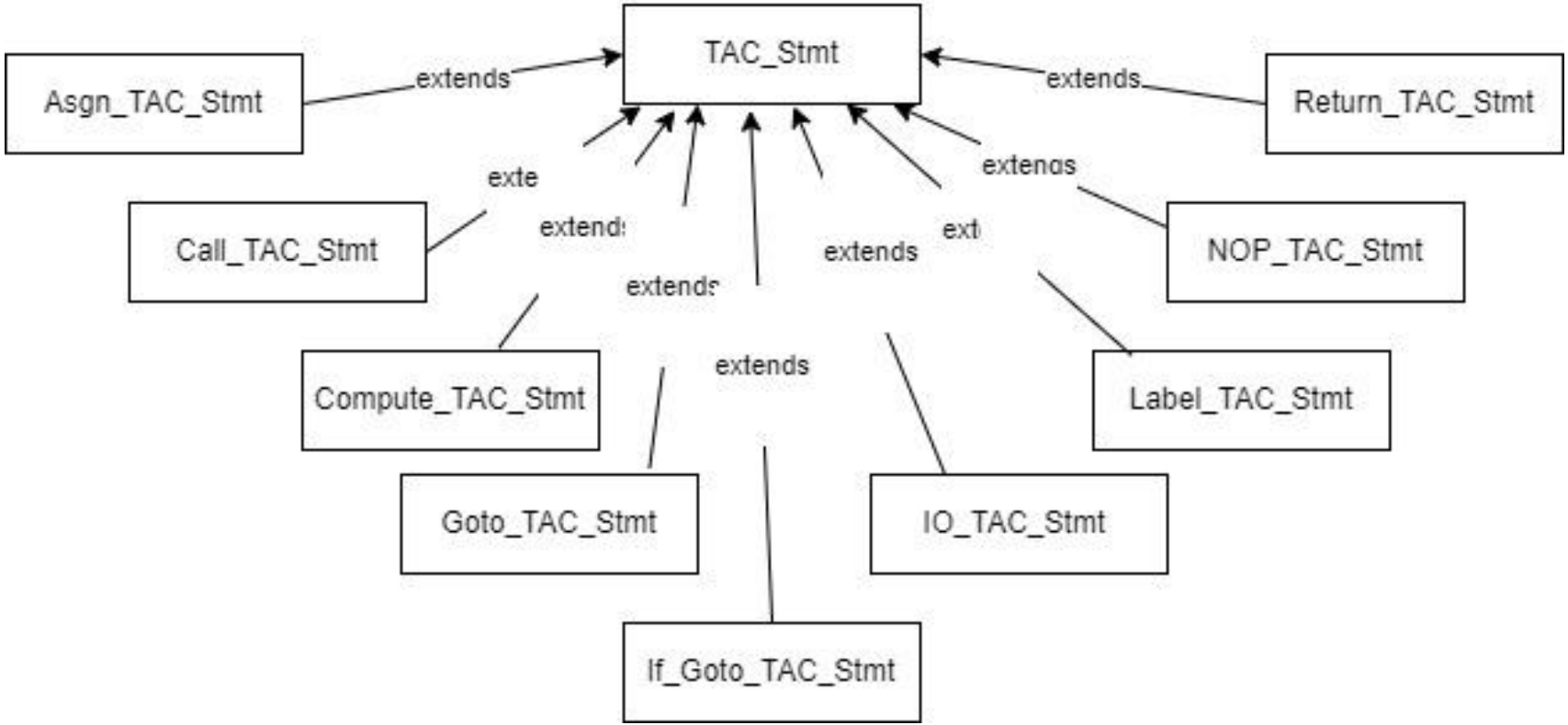


IR Generator Use Case Diagram

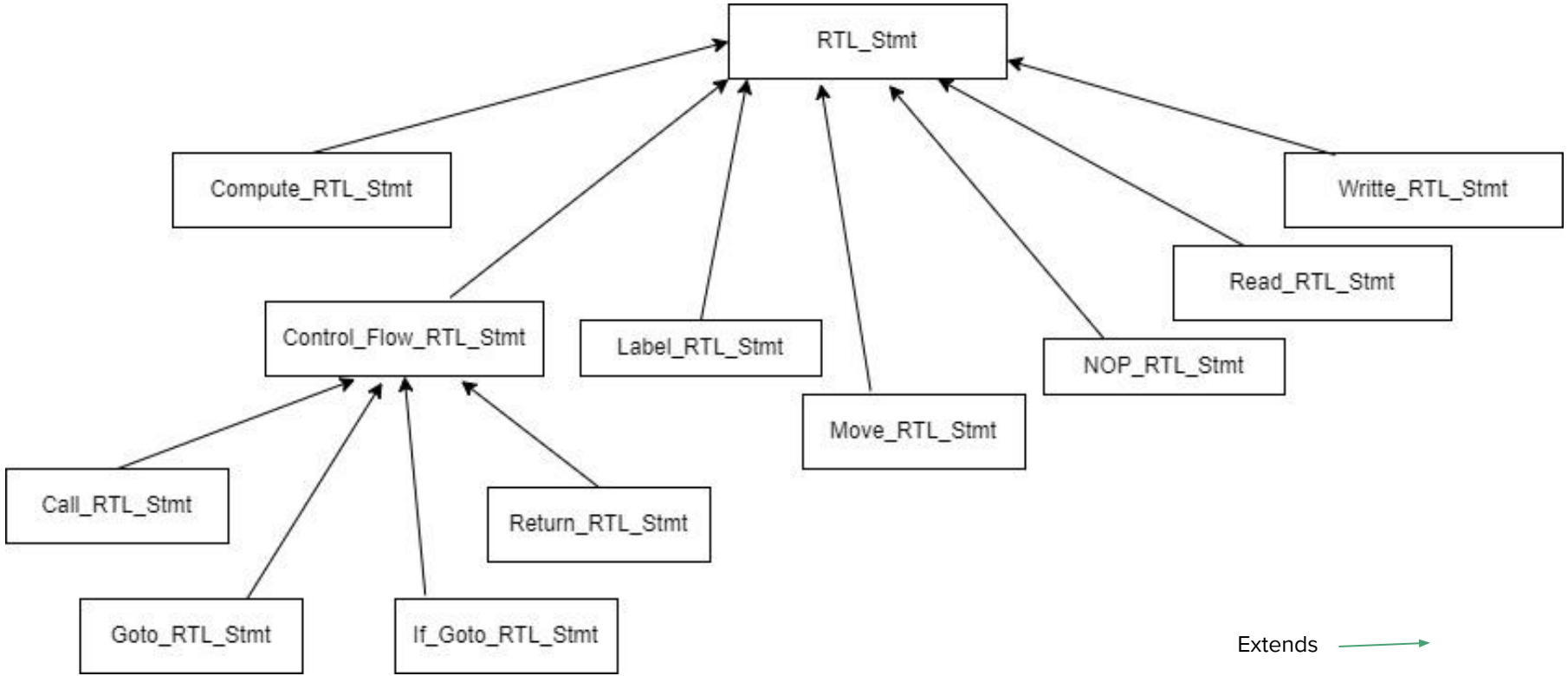
Name	Generation of IRs
Description	Generate autogenerated IRs from language specifications
Pre Condition	Specifications and Translation rules must be defined
Sequence	• Input scanned • Input goes through the IRGen produced from the machine readable form of the specs and translation rules
Post Condition	The output of the source program generated



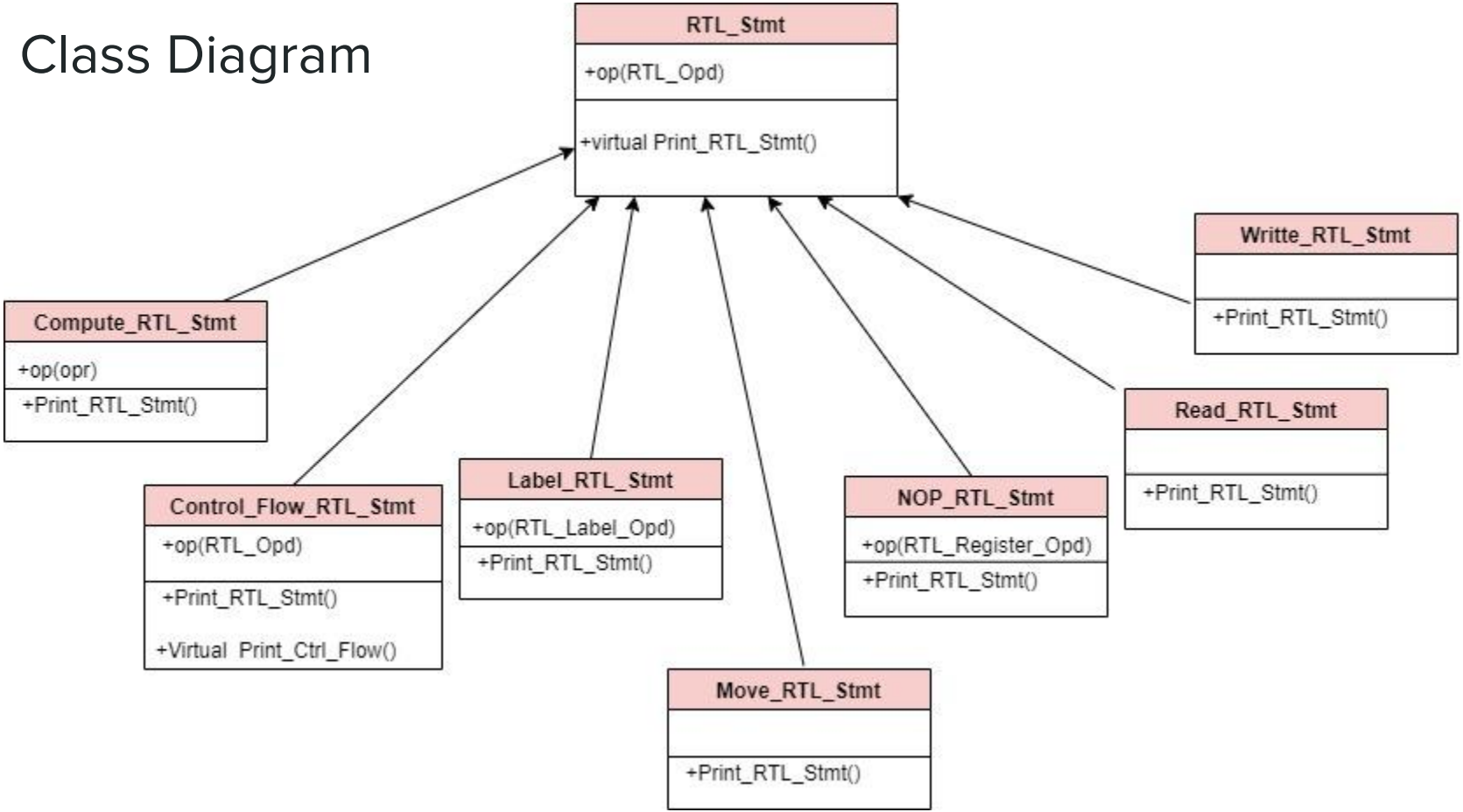
Class Diagram for TAC



Class Diagram for RTL



Class Diagram



Software Requirements Specification(SRS)

- Scope
  - Generation of ASM generator for SPIM by writing specifications for SPIM and RTL
  - Generation of ASM generator for RIPES by writing specifications for RIPES and RTL
  - Generation of TAC generator by writing specifications for AST and TAC
- Not in Scope
  - Implementation of generators that will construct interpreters
- Functional Requirements
  - The IR generator should be generated using the Translator Generator using the machine readable form of the specifications and translation rules.
  - The compiler should be able to read an input .c program and generate the expected outputs.
  - SCLP should be able to generate AST, TAC, and RTL files as the outputs of the .c program.
  - SCLP should be able to generate subsequent IR files for IR inputs.

Software Requirements Specification(SRS)

- Non-Functional Requirements
  - Availability- The generators should perform the tasks it is assigned to perform.
  - Reliability- All levels of .c programs should be readable and accurately executed.
  - Flexibility- The system should be flexible for future development.
  - Usability- The system should be easily usable for the UG courses.
- System Requirements
  - Software Requirements-
    - Linux - Ubuntu(Terminal), Python, gcc(GNU Compiler Collection), RIPES - RISCV simulator, SPIM - MIPS simulator, LEX & YACC
  - Hardware Requirements-
    - Operating System - Linux distribution, Ubuntu
    - RAM - 4GB

Technology

Languages to write the translator generators:

- C++
- Python

Lexical Analysers and Parser Generators:

- Lex and Yacc
- Flex & Bison

Simulators:

- SPIM MIPS simulator
- RIPES RISC-V simulator

Implementation Aspects

- Studied the SCLP compiler, various IRs and levels.
- Delivered bug reports, analysed SCLP and the SCLP web page and thus suggested changes.
- Studied the formalised ASM instructions written by the previous group.
- Suggested ways to improve the said formalization, making it more precise.
- Changed the specifications syntax of instructions to our suggested syntax.
- Studies RIPES simulator
- Researched on the differences between RISCV and MIPS processor which helped in studying the differences between spim and ripes simulator.
- Researching on the ways to write RTL specifications using Ripes.

Specifications Document

- ASM specification for SPIM:

$$[\textit{Load Integer Immediate}]\frac{S(li,o1,o2):\tau^S_{mv} \qquad o1:\tau^S_{gpr} \qquad o2:\tau^S_{int}}{H;D;C;R\vdash S(li,o1,o2)\rightarrow H;D;C;R[o1\mapsto o2] \qquad R:\tau^S_{gpr}}$$

- ASM specification for RIPES:

$$[\textit{Load Integer Immediate}]\frac{S(li,o1,o2):\tau^S_{mv} \qquad o1:\tau^S_{gpr} \qquad o2:\tau^S_{int}}{H;D;C;R\vdash S(li,o1,o2)\rightarrow H;D;C;R[o1\mapsto o2] \qquad R:\tau^S_{gpr}}$$

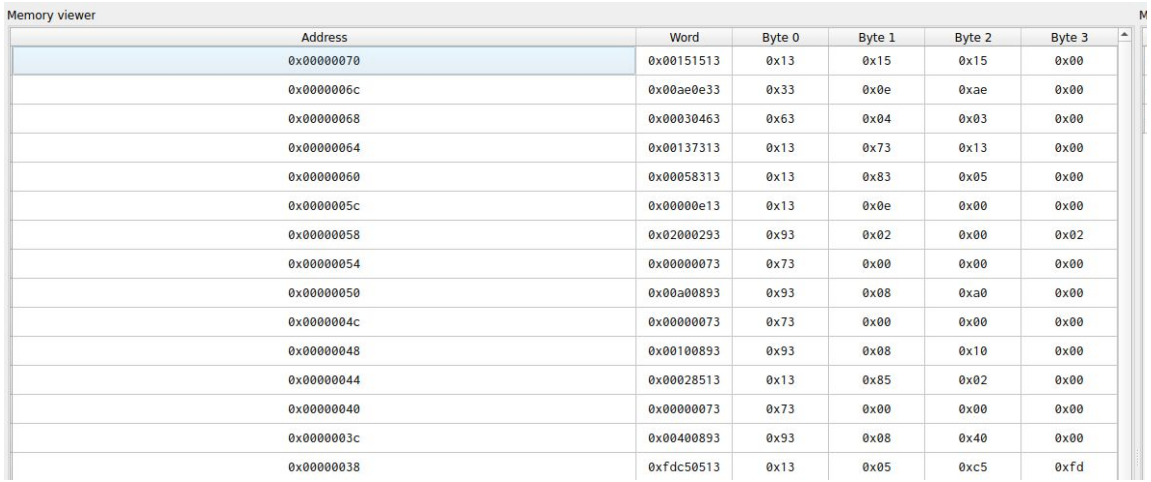
- RTL specification:

$$[\textit{Load Integer Immediate}]\frac{R(iLoad,o1,o2):\tau^R_{mv} \qquad o1:\tau^R_{reg} \qquad o2:\tau^R_{int}}{R\vdash S(iLoad,o1,o2)\rightarrow R[o1\mapsto o2] \qquad R:\tau^S_{gpr}}$$

Thorough study of RISC-V instructions for RIPES

Studied the instructions and compared them with the MIPS instructions. Discussed the mapping of MIPS register and RISCV registers.

Adjoining image shows the memory structure of RIPES.


Memory viewer

Address	Word	Byte 0	Byte 1	Byte 2	Byte 3
0x00000070	0x00151513	0x13	0x15	0x15	0x00
0x0000006c	0x00ae0e33	0x33	0x0e	0xae	0x00
0x00000068	0x00030463	0x63	0x04	0x03	0x00
0x00000064	0x00137313	0x13	0x73	0x13	0x00
0x00000060	0x00058313	0x13	0x83	0x05	0x00
0x0000005c	0x00000e13	0x13	0x0e	0x00	0x00
0x00000058	0x02000293	0x93	0x02	0x00	0x02
0x00000054	0x00000073	0x73	0x00	0x00	0x00
0x00000050	0x00a00893	0x93	0x08	0xa0	0x00
0x0000004c	0x00000073	0x73	0x00	0x00	0x00
0x00000048	0x00100893	0x93	0x08	0x10	0x00
0x00000044	0x00028513	0x13	0x85	0x02	0x00
0x00000040	0x00000073	0x73	0x00	0x00	0x00
0x0000003c	0x00400893	0x93	0x08	0x40	0x00
0x00000038	0xfdc50513	0x13	0x05	0xc5	0xfd

Formalized ASM specifications for RIPES

Created specifications document for RISC-V instructions for RIPES

Adjoining image shows a few compute instructions

Compute Instructions

$$[Add\ ints]\frac{S(add,o1,o2,o3):\tau^S_{cmp} \quad o1:\tau^S_{gpr} \quad o2:\tau^S_{gpr} \quad o3:\tau^S_{gpr}}{H;D;C;R\vdash S(add,o1,o2)\rightarrow H;D;C;R[o1\mapsto R[o2]+R[o3]] \quad R:\tau^S_{gpr}} \tag{9}$$

$$[Add\ ints]\frac{S(addi,o1,o2,o3):\tau^S_{cmp} \quad o1:\tau^S_{gpr} \quad o2:\tau^S_{gpr} \quad o3:\tau^S_{int}}{H;D;C;R\vdash S(addi,o1,o2)\rightarrow H;D;C;R[o1\mapsto R[o2]+o3] \quad R:\tau^S_{gpr}} \tag{10}$$

$$[Sub\ ints]\frac{S(sub,o1,o2,o3):\tau^S_{cmp} \quad o1:\tau^S_{gpr} \quad o2:\tau^S_{gpr} \quad o3:\tau^S_{gpr}}{H;D;C;R\vdash S(sub,o1,o2,o3)\rightarrow H;D;C;R[o1\mapsto R[o2]-R[o3]] \quad R:\tau^S_{gpr}} \tag{11}$$

$$[Sub\ ints]\frac{S(sub,o1,o2,o3):\tau^S_{cmp} \quad o1:\tau^S_{gpr} \quad o2:\tau^S_{gpr} \quad o3:\tau^S_{int}}{H;D;C;R\vdash S(sub,o1,o2,o3)\rightarrow H;D;C;R[o1\mapsto R[o2]-o3] \quad R:\tau^S_{gpr}} \tag{12}$$

$$[Multiply\ ints]\frac{S(mul,o1,o2,o3):\tau^S_{cmp} \quad o1:\tau^S_{gpr} \quad o2:\tau^S_{gpr} \quad o3:\tau^S_{gpr}}{H;D;C;R\vdash S(mul,o1,o2,o3)\rightarrow H;D;C;R[o1\mapsto R[o2]\times R[o3]] \quad R:\tau^S_{gpr}} \tag{13}$$

$$[Divide\ ints]\frac{S(div,o1,o2,o3):\tau^S_{cmp} \quad o1:\tau^S_{gpr} \quad o2:\tau^S_{gpr} \quad o3:\tau^S_{gpr}}{H;D;C;R\vdash S(div,o1,o2,o3)\rightarrow H;D;C;R[o1\mapsto R[o2]\div R[o3]] \quad R:\tau^S_{gpr}} \tag{14}$$

Specifications document for RTL

Currently working on the specifications document for RTL for SPIM and for RIPES.

Adjoining image shows a li(Load Immediate) instruction equivalent of RTL

$$[Load\ Integer\ Immediate]\frac{R(iLoad, o1, o2) : \tau^R_{mv} \qquad o1 : \tau^R_{reg} \qquad o2 : \tau^R_{int}}{R \vdash S(iLoad, o1, o2) \rightarrow R[o1 \mapsto o2] \qquad R : \tau^S_{gpr}}$$

Mapping of the ASM instructions for SPIM and RIPES

The mapping of both the target assembly codes should be a minimal work.

SPIM instruction seq to compare if R[o2] and o3 are equal

$$[If\ equal\ to]\frac{S(seq, o1, o2, o3) : \tau^S_{cmp} \qquad o1 : \tau^S_{gpr} \qquad o2 : \tau^S_{gpr} \qquad o3 : \tau^S_{int}}{H; D; C; R \vdash S(seq, o1, o2, o3) \rightarrow H; D; C; R[o1 \mapsto (R[o2] = o3?1 : 0)] \qquad R : \tau^S_{gpr}}$$

Equivalent RIPES instructions for the same seq SPIM instruction

$$[Sub\ ints]\frac{S(sub, o1, o2, o3) : \tau^S_{cmp} \qquad o1 : \tau^S_{gpr} \qquad o2 : \tau^S_{gpr} \qquad o3 : \tau^S_{int}}{H; D; C; R \vdash S(sub, o1, o2, o3) \rightarrow H; D; C; R[o1 \mapsto R[o2] - o3] \qquad R : \tau^S_{gpr}}$$

$$[If\ equal\ to\ zero]\frac{S(seqz, o1, o2) : \tau^S_{cmp} \qquad o1 : \tau^S_{gpr} \qquad o2 : \tau^S_{gpr}}{H; D; C; R \vdash S(sne, o1, o2) \rightarrow H; D; C; R[o1 \mapsto (R[o2] = 0?1 : 0)] \qquad R : \tau^S_{gpr}}$$

Division of work done

Sai Ghule	Studied and compiled a report on: - TAC - Command Line arguments for SCLP - SPIM Specifications for Load, Store and Move Instructions - RIPES Specifications for Arithmetic Instructions Studied RTL for Specs
Bhagyashree Rane	Studied and compiled a report on: - AST - Bug Reports - SPIM Specifications for Branch and Compare Instructions - RIPES Specifications for Branch and Compare Instructions - RTL Specifications for Load Instructions
Shravasti Deore	Studied and compiled a report on: - RTL - Bug Reports - SPIM Specifications for Arithmetic Instructions - RIPES Specifications for Load, Store and Move Instructions Compared RTL and ASM

Literature Survey

A. V. Aho, Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, J. D. Ullman, 1986. Compilers: Principles, Techniques, and Tools. Addison-Wesley.	Standard textbook about compiler construction for programming languages. Has in depth coverage of topics like lexical analysis, parsing, generation of intermediate code, etc. Also, describes development of a mini compiler.
Tofte. M. 2012 compiler generators : what they can do , what they might do and what they will never do. Springer Science & Business Media.	Reports on the Compiler Generator CERES. The concept of Compiler Generator is described along with the comparison of CERES with other systems. Although, it does not say anything about intermediate representations.
Lee, P.. Realistic Compiler Generation. MIT Press.	Formal Specifications, their need and high-level descriptions, and generation of realistic compilers is primarily treated. Used denotational semantics.

Gap Analysis

- Compilation Models seldom use compiler generators i.e. the compiler is handwritten
- The reviewed models of compilation did not emphasize the importance of Intermediate Representations

Future Enhancements

Using Specifications with program state to:

- Implement generators that will construct interpreters
- Thus, providing virtual machines for AST, TAC, and RTL.

Documents and Process Followed

Documents Created so far

1. SRS (System Requirement Specification)
2. Bug Reports with 14 bugs
3. Design Critique Report
4. Specification report for ASM of SPIM
5. Specification report for ASM of RIPES

Software Engineering

Agile Project:

1. Iterative Process:

- Thoroughly tested the SCLP compiler using existing test suite.
- SCLP has gone through a lot of versions each of which has been tested.

2. Flexible requirements :

- Project has grown.
- Literature survey has been from the very beginning.

Software Engineering

- The project is being developed using the Agile methodology.
- The requirements and solutions evolve through collaboration between the external mentor and the team every 10 days.
- The external mentor also checks if the requirements and the solutions are met after every iteration.
- We have weekly meetings with our external mentor Khedkar sir.
- We also have 2 internal weekly meetings to discuss and distribute work and to make sure that each of us are on the same page.
- Practices such as using collaboration tools like Overleaf is used.
- We have also started with the NPTEL Course on compiler construction.

Work Plan

	7 Feb	14 Feb	21 Feb	27 Feb	6 March	13 March
specs for spim						
specs for ripes						
specs for RTL						
translation rules from RTL to ASM						
machine readable form of the specs and translation rules						
implementation of the translator generator						

References

A. V. Aho, Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, J. D. Ullman, 1986. Compilers: Principles, Techniques, and Tools. Addison-Wesley.

Tofte. M. 2012 compiler generators : what they can do , what they might do and what they will never do. Springer Science & Business Media.

A Complete Specification of a Simple Compiler

Sclp: A Language Processor for a Small C-like Language

Programmed Introduction to MIPS Assembly Language

SPIM Quick Reference

MIPS IV Instruction Set

SPIM Documentation

Instruction Set Architecture

The RISC-V Instruction Set Manual

Thank You