

# Ada Compiler

A compiler for a Ada subset that generates MIPS assembly code.

## Overview

The compiler accepts a single Ada procedure and is composed of a lexer + parser + AST builder + type checker + Intermediate Code Generator + Mips assembly generator. It accepts as input one Ada program and outputs: 1. AST representation 2. Whether it passed the type checking (if not it does not execute points 3. and 4.) 3. Three-address intermediate code 4. MIPS assembly code in another file

## Language Features

### Procedure Declaration

```
procedure Name is
    <declarations>
begin
    <statements>
end Name;
```

### Variable Declarations

- **Integer:** X : Integer; or X : Integer := 42;
- **String:** X : String := "ola"; or X : String := Get\_Line; or X : String (1 .. 255)
- Declarations must appear before begin

### Statements

#### Assignment

```
X := expr;
```

#### While Loop

```
while condition loop
    statements;
end loop;
```

#### If-Then-Else

```
if condition then
    statements;
else
    statements;
end if;
```

### **Single If**

```
if condition then
    statements;
end if;
```

### **If-Elsif-Else**

```
if condition then
    statements;
elsif condition then
    statements;
else
    statements;
end if;
```

### **Input**

`Get_Line(Var,Length);`

Reads a line from standard input.

`Get(var)`

Reads an integer from standard input.

### **Output**

`Put_Line(str);`

Prints a string or the contents of a string variable, and a new line at the end.

`Put(expr)`

Prints the result of expr.

## **Expressions**

### **Arithmetic Expressions**

- **Integer literals:** 42, -123
- **Variables**
- **Binary operators:** +, -, \*, /

- **Parentheses:** (expr)

### Boolean Expressions

- **Literals:** True, False
- **Logical operators:** and, or, not
- **Comparisons:** =, /=, <, <=, >, >=
- **Parentheses:** (bexpr)

### Strings

- **Literals:** hello
- **Identifiers**

## Build - Compilation

Requirements: Flex, Bison, GCC

make

## Usage

- ./prog tests/main?.adb (where ? is [1-13])

## Example of execution

make

./prog tests/main6.adb

--- Abstract Syntax Tree ---

```
mk_compound(
  mk_compound(
    mk_compound(
      mk_declaration(
        I,
        Integer,
        mk_int(1)
      ),
      mk_declaration(
        R,
        Integer,
        mk_int(1)
      )
    ),
    mk_declaration(
      N,
```

```

        Integer,
        NULL
    )
),
mk_compound(
    mk_compound(
        mk_get(N),
        mk_while(
            mk_bbinop(
                E_LESS,
                mk_ident(I),
                mk_ident(N)
            ),
            mk_compound(
                mk_assign(
                    R,
                    mk_op(
                        MULT,
                        mk_ident(R),
                        mk_ident(I)
                    )
                ),
            ),
            mk_assign(
                I,
                mk_op(
                    PLUS,
                    mk_ident(I),
                    mk_int(1)
                )
            )
        )
    )
),
mk_put(
    mk_ident(R)
)
)
)
--- Type Checking ---
Type checking passed

--- Intermediate Code ---
t0 := 1
t1 := 1
Read_Int t2
LABEL label0
t3 := t0
t4 := t2

```

```

COND t3 <= t4 THEN label1 ELSE label2
LABEL label1
t3 := t1
t4 := t0
t1 := t3 * t4
t3 := t0
t4 := 1
t0 := t3 + t4
JUMP label0
LABEL label2
t3 := t1
Print_Int t3

```

--- MIPS Assembly Code ---

(Writing to tests/main6.s)

Successfully generated tests/main6.s

===== Compilation Complete =====

**File tests/main6.s would look like this:**

```

.data

.text
.globl main
main:
    li $t0, 1
    li $t1, 1
    li $v0, 5
    syscall
    move $t2, $v0
label0:
    move $t3, $t0
    move $t4, $t2
    bgt $t3, $t4, label2
label1:
    move $t3, $t1
    move $t4, $t0
    mul $t1, $t3, $t4
    move $t3, $t0
    li $t4, 1
    add $t0, $t3, $t4
    j label0
label2:
    move $t3, $t1
    move $a0, $t3
    li $v0, 1
    syscall

```

## File Structure

```
.
├── lexer.l                # Flex lexer
├── parser.y               # Bison parser
├── ast.h / ast.c          # AST
├── code.h / code.c        # Intermediate code generator
├── genMIPS.h / genMIPS.c  # MIPS code generator
├── hashTable.h / hashTable.c # Symbol table (hash map)
├── typeChecker.h / typeChecker.c # Type checking
├── main.c                 # Main driver
├── Makefile               # Build script
├── README.md
└── tests/                 # Test Ada programs and
    respetives assembly files
```

## Authors

**COMP\_TP2\_GRUPO\_1** - Hugo Melo - 202304068 - Pedro Machado -  
202307196