プロ研 B「簡易 C コンパイラ」
言語・機能仕様書

1W233019
大戸暢丈

はじめに

本資料は C 言語のサブセットのコンパイラ「2cc」の言語・機能仕様書を記載する。

言語

1. 変数

変数はすべて 4 byte の int 型（-2^{31}〜$2^{31}+1$）で宣言時にも int という予約語は使用しない。関数外のグローバル変数と局所変数の両方に対応する。

2. 予約語

2cc がコンパイルする言語の予約語は次の通りである。

if, for, else, main, return, while

3. 識別子

英字で始まる文字列の並びにより構成される。

4. 式

2cc がコンパイルする言語の式は以下のとおりである。

+, -, *, /, %, =, ==, !=, <, <=, >, >=, (,), {, }, ;

5. 文法

expression:

assignment-expression

primary-expression:

identifier

constant

5.1. 単項演算子

Unary-expression:

Primary-expression

Unary-operator unary-expression

Unary-operator: +, -

5.2. 乗算演算子

Multiplicative-expression:

Unary-expression

Multiplicative-expression * Unary-expression

Multiplicative-expression / Unary-expression

Multiplicative-expression % Unary-expression

5.3. 加算演算子

Additive-expression:

Multiplicative-expression

Additive-expression + Multiplicative-expression

Additive-expression - Multiplicative-expression

5.4. 比較演算子

Relational-expression:

Additive-expression

Additive-expression < Additive-expression

Additive-expression <= Additive-expression

Additive-expression > Additive-expression

Additive-expression >= Additive-expression

5.5. 等号演算子

Equality-expression:

Relational-expression

Equality-expression == relational-expression

Equality-expression != relational-expression

5.6. 代入演算子

Assignment-expression:

Identifier = equality-expression

5.7. 宣言

Declaration:

Identifier;

5.8. 引数リスト

関数の引数のリストのこと

Parameter-list:

Parameter-declaration

Parameter-list, parameter-declaration

Parameter-declaration:

Identifier

5.9. if 文

if-statement:

if(expression) statement

if(expression) statement else statement

5.10. for, while 文

iteration-statement:

while(expression) statement

for(expression;expression;expression;) statement

5.11. return 文

return-statement:

return expression;

5.12. 関数

Function-declaration:

Identifier(parameter-list) statement

6. 組み込み関数

標準入出力に対応する簡易的な print と scan が利用できる。

機能

1. ソース言語

ソース言語は上述の C 言語のサブセットとする。

2. 生成する言語

生成する言語は Raspberry PI 64bit で動く ARM のアセンブリ言語である。実行バイナリへの変換は gcc に頼る。

3. コンパイラの実行

プロジェクトルートで Makefile を用意する。

4. 字句解析

ソース言語の軸解析を行う。認識するトークンは

記号: +, -, *, /, =, ==, !=, <, <=, >, >=, (,), {, }, ;

整数: 0-9

識別子: 英字で始まる文字列

予約語: if, for, else, main, return, while

5. 構文解析

ソースプログラムを言語仕様書で定めた構文規則に沿って解析し、抽象構文木を生成する。

6. 中間表現・抽象構文木

中間表現は構文解析木により構成され、抽象構文木のノードは次の通りに分類される。

ND_FUNCTION: 関数の定義

ND_RETURN

ND_EXPR_STMT:
ND_EMPTY_STMT: 空文（for 文の中など）

ND_IF
ND_WHILE
ND_FOR

ND_ASSIGN: 代入文

ND_LVAR: ローカル変数

ND_FUNCALL: 関数呼び出し

ND_NEG
ND_ADD
ND_SUB
ND_MUL
ND_DIV
ND_EQ
ND_NE
ND_LT
ND_LTE

ND_GT
ND_GTE

7. コード生成

ARM64 のアセンブリを生成する。