PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

ESCOLA POLITÉCNICA

ALEXANDRE DOS SANTOS BINDO

CAROLINE JENUARIO RODRIGUES DA SILVA

MELQUISEDEQUE CASAGRANDE ALVES MENDES DIAS

RICARDO BERTOLIN

PROJETO COMPILADOR

CURITIBA

2023

ALEXANDRE DOS SANTOS BINDO

CAROLINE JENUARIO RODRIGUES DA SILVA

MELQUISEDEQUE CASAGRANDE ALVES MENDES DIAS

RICARDO BERTOLIN

PROJETO COMPILADOR

Projeto de Linguagens Formaise Compiladores da Graduação em Engenharia da Computação da Pontifícia Universidade Católica do Paraná, conferindo a primeira etapa.

Prof. Frank Coelho de Alcantara

CURITIBA

2023

SUMÁRIO

1 PROJETO COMPILADOR

1.1 CONTEXTO

O trabalho em questão é destinado ao desenvolvimento de um compilador em 3 etapas, envolvendo uma nova linguagem de programação. Será baseado no microcontrolador ATmega2560, utilizando ferramentas secundárias de softwares para demais etapas, como GitHub, Repl.it, sendo as demais definidas posteriormente, incluindo a parte de hardware, qual foi optado pelo Arduino Mega para execução.

1.2 DEFINIÇÃO DE LINGUAGEM

O código utilizará uma linguagem com a gramática definida abaixo, qual consiste em declarações definidas, considerando seu conjunto (qual contém), incluindo a definição para o lexema **basic**:

PROGRAM	*BLOCK*
BLOCK	*{DECLS STMTS}*
DECLS	*DECLS DECL |*
DECL	*TYPE ID ;*
TYPE	*TYPE [NUM] | BASIC*
STMTS	*STMTS STMT | E*

*Lexama **basic**, expressando tipos básicos:*

STMT	*IF (BOOL) STMT*
	IF (BOO) STMT ELSE STMT
	WHILE (BOOL) STMT
	DO STMT WHILE (BOOL);
	BREAK ;

*Considerando regras de precedência, será utilizado o não-terminal **factor** para lidar com expressões.*

BOOL	*BOOL || JOIN | JOIN*
JOIN	*JOIN && EQUALITY | EQUALITY*
EQUALITY	*EQUALITY == REL | EQUALITY != REL | REL*
REL	*EXPR < EXPR | EXPR <= EXPR | EXPR >= EXPR | EXPR > EXPR | EXPR*
EXPR	*EXPR + TERM | EXPR - TERM | TERM*
TERM	*TERM * UNARY | TERM / UNARY | UNARY*
UNARY	*UNARY | - UNARY | FACTOR*
FACTOR	*(BOOL) | NUM | REAL | TRUE | FALSE*

*Os lexemas **num** e **real** são usados para representar números inteiros e números de ponto flutuante de 16 bits, conforme serão necessários em futuras etapas.*

STMT	*TYPE ID = NUM | REAL | TRUE | FALSE*
	DIGITAL_READ (PIN)
	DIGITAL_WRITE (PIN, NUM)
	ANALOG_READ (PIN)
	ANALOG_WRITE (PIN, NUM)
	PIN_MODE (NUM, PIN_TYPE)
	DELAY (NUM)

1.3 RESTRIÇÕES DA LINGUAGEM

A linguagem elaborada não irá compreender estruturas de maior complexidade, como: Falta de Suporte a Arrays Multidimensionais, falta de estruturas de controle Avançadas (for, switch, foreach): características de recursividade, funções definidas.

O documento em questão não aborda, apesar da gramática estar bem definida, a forma como ela é usada no código, necessitando implementar um analisador sintático (em anexo) que segue essa gramática para verificar a validade dos programas. Também não é abordado diretamente nesta etapa, como a linguagem permite interação direta com hardware, além de como a linguagem lida com erros ou exceções.

1.4 EXEMPLOS DE APLICAÇÃO

Os seguintes exemplos básicos de aplicação para o Arduino Mega, demonstrando sua funcionalidade e aplicação.

Piscar um LED:

```
STMT TYPE ledPin = 13;
STMT TYPE setup() {
        PIN_MODE(ledPin, OUTPUT);
}
STMT TYPE loop() {
        DIGITAL_WRITE(ledPin, HIGH); // Liga o LED
        DELAY(1000); // Aguarda 1 segundo
        DIGITAL_WRITE(ledPin, LOW); // Desliga o LED
        DELAY(1000); // Aguarda 1 segundo
 }
```

Ler um Sensor de Temperatura (DHT22):

```
#include <DHT.h>
 #define DHTPIN 2
#define DHTTYPE
DHT22 DHT dht(DHTPIN, DHTTYPE);
STMT TYPE setup() {
        Serial.begin(9600);
        dht.begin();
 }
STMT TYPE loop() {
        FLOAT temperatura = dht.readTemperature(); // Lê a temperatura
        Serial.print("Temperatura: ");
        Serial.print(temperatura);
        Serial.println(" °C");
        DELAY(2000); // Aguarda 2 segundos entre as leituras
         }
```

Acende um LED com base na intensidade da luz:

```
STMT TYPE lightSensorPin = 3; // Pino onde o sensor de luz está conectado
STMT TYPE ledPin = 13; // Pino onde o LED está conectado
```

```
STMT TYPE setup() {
    PIN_MODE(ledPin, OUTPUT);
    SERIAL_BEGIN(9600); // Inicializa a comunicação serial
}

STMT TYPE loop() {
    INT lightValue = ANALOG_READ(lightSensorPin); // Lê o valor do sensor de luz

    // Verifica se a intensidade da luz é maior que um limite
    IF (lightValue > 500) {
        DIGITAL_WRITE(ledPin, HIGH); // Liga o LED
    } ELSE {
        DIGITAL_WRITE(ledPin, LOW); // Desliga o LED
    }

    DELAY(1000); // Aguarda 1 segundo antes de verificar novamente
}
```

1.5 RESULTADO DA ETAPA

O presente documento descreve a entrega da primeira fase, qual será composta da apresentação da linguagem definida, destacando as regras de produção criadas para interação com o Hardware. Além disso, em anexo, será incluso exemplos de código utilizando a sua linguagem e que apresentem todas as funcionalidades, assim como a segunda parte da primeira entrega consiste na criação de um analisador léxico, capaz de validar todos os possíveis lexemas da linguagem, usando máquinas de estado finito, implementadas em Python, qual para teste, inclui arquivos de texto que contenham todos os lexemas da linguagem, com saída sendo um string. Finalmente, procura-se garantir que a linguagem está corretamente definida atende todos os requisitos especificados, também, o analisador léxico analisar a linguagem indicando os erros existentes.

REFERÊNCIAS

AHO, Alfred. LAM. S. Monica. SETHI. Ravi. ULLMAN, D. Jeffrey
Compiladores: Princípios, Técnicas e Ferramentas. 30 de Outubro de 2007.

Documentação do Arduíno Mega.https://docs.arduino.cc/hardware/mega-2560.

https://frankalcantara.com/construcao-interpretadores/