**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ**
**ESCOLA POLITÉCNICA**

**DIEGO BARRETO PEDROSO SIMÕES**
**MURIEL TRAMONTIN VON LINSINGEN**
**VINICIUS RAMOS GARCIA**
**WILLIAM HOEFLICH WOINAROWSKI**

# PROJETO DE COMPILADOR

**CURITIBA**
**2023**

**DIEGO BARRETO PEDROSO SIMÕES**
**MURIEL TRAMONTIN VON LINSINGEN**
**VINICIUS RAMOS GARCIA**
**WILLIAM HOEFLICH WOINAROWSKI**

# PROJETO DE COMPILADOR

Projeto apresentado à disciplina de Linguagens Formais e Compiladores da Graduação em Engenharia da Computação da Pontifícia Universidade Católica do Paraná, como requisito parcial de avaliação.

Orientador: Prof. Frank Coelho de Alcantara

**CURITIBA**
**2023**

**SUMÁRIO**

# 1 PROJETO DE COMPILADOR

## 1.1 CONTEXTO

Este projeto tem como objetivo o desenvolvimento de um compilador para uma nova linguagem de programação a ser estruturada, e seu foco será nas plataformas de sistemas embarcados baseados no microcontrolador ATmega2560. Para o desenvolvimento do compilador serão utilizados alguns recursos de *software*, como a plataforma de desenvolvimento online Replit, o Visual Studio Code e o Hercules para a gravação do Assembly gerado no microcontrolador, e de hardware, que será composto pela placa Arduino Mega, onde todos os exemplos gerados pela linguagem serão executados.

## 1.2 DEFINIÇÃO DA LINGUAGEM

A linguagem a ser utilizada neste compilador está baseada na gramática fornecida a seguir, onde um programa consistirá em um bloco de declarações definidos pela tabela:

| **BLOCO/DECLARAÇÃO** | **CONTÉM** |
|---|---|
| PROGRAM | BLOCK |
| BLOCK | {DECLS STMTS} |
| DECLS | DECLS DECL \| ϵ |
| DECL | TYPE ID |
| TYPE | TYPE [NUM] \| BASIC |
| STMTS | STMTS STMT \| ϵ |

O lexema **BASIC** expressa os tipos básicos da linguagem:

| | |
|---|---|
| STMT | IF (BOOL) STMT |
| | IF (BOOL) STMT ELSE STMT |
| | WHILE (BOOL) STMT |
| | DO STMT WHILE (BOOL) |
| | BREAK |

| | |
|---|---|
| | BLOCK |

Cada bloco ou declaração contém alguns parâmetros, que também podem ser blocos ou declarações, e representam o acervo de palavras restritas aceito pela linguagem.

As produções para as expressões tratam da associatividade e precedência de operadores. Elas usam um não-terminal para cada nível de precedência e um não-terminal, **FACTOR**, para as expressões entre parênteses, identificadores, referências de arranjos e constantes:

| | |
|---|---|
| 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 |

O lexema **NUM** indica números inteiros, o lexema **REAL** indica números de ponto flutuante com 16 bits segundo o padrão IEEE-754, conhecido como meia precisão. Todas as operações matemáticas, serão realizadas com a precisão definida no padrão IEEE – 754 para 16bits.

Para a interação com o hardware foram propostas as seguintes regras a serem associadas ao lexema **BASIC**:

| | |
|---|---|
| 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) |

O lexema **NUM** indica números inteiros, e o lexema **PIN_TYPE** indica a configuração de leitura ou escrita do pino do microcontrolador:

| | |
|---|---|
| PIN_TYPE | OUTPUT \| INPUT |

A partir desta gramática podemos criar programas com interação direta com o *hardware* para a execução de diversas atividades, porém, com algumas restrições a serem abordadas no próximo tópico.

## 1.3 RESTRIÇÕES DA LINGUAGEM

Está linguagem não irá compreender estruturas mais complexas como:

- Estrutura de repetição “FOR”
- Estrutura condicional composta “IF stmt ELSEIF stmt ELSE stmt”
- Estrutura condicional composta “SWITCH stmt CASE expr”
- Recursividade
- Funções

## 1.4 EXEMPLOS DE APLICAÇÃO

A seguir estão disponíveis 3 exemplos de aplicação da linguagem de programação contendo interação com o hardware do microcontrolador ATMega2560:

1. Piscar um led (pisca o led em intervalos de 1 segundo):

```
NUM PIN_10 = 10

PIN_MODE (PINO_10, OUTPUT)

WHILE (TRUE) {
        DIGITAL_WRITE (PINO_10, TRUE)
        DELAY (1000)
        DIGITAL_WRITE (PINO_10, FALSE)
        DELAY (1000)
}
```

2. Liga led com botão (acende o led somente com o botão pressionado):

```
NUM PIN_9 = 9
NUM PIN_10 = 10

PIN_MODE (PINO_9, INPUT)
PIN_MODE (PINO_10, OUTPUT)

WHILE (TRUE) {
        IF (DIGITAL_READ (PINO_9)) {
                DIGITAL_WRITE (PINO_10, TRUE)
        }
}
```

3. Indicador de intensidade de um potenciômetro (aciona os leds conforme ocorre alteração no potenciômetro):

```
NUM PIN_9 = 9
NUM PIN_10 = 10
NUM PIN_11 = 11

PIN_MODE (PINO_9, INPUT)
PIN_MODE (PINO_10, OUTPUT)

WHILE (TRUE) {
        IF (ANALOG_READ (PIN_9) < 256){
                DIGITAL_WRITE (PINO_10, FALSE)
                DIGITAL_WRITE (PINO_11, FALSE)
        }
        IF (ANALOG_READ (PIN_9) >= 256){
                DIGITAL_WRITE (PINO_10, TRUE)
        }
        IF (ANALOG_READ (PIN_9) >= 512){
                DIGITAL_WRITE (PINO_10, FALSE)
                DIGITAL_WRITE (PINO_11, TRUE)
        }
}
```

## 1.5 PROJETO DO COMPILADOR

Todo o projeto do compilador será realizado em Python através da IDE Replit e do Visual Studio Code, passando pela análise léxica, sintática e semântica. Para a gravação do Assembly gerado será utilizado o *software* Hercules

## REFERÊNCIAS

ATMega2560. Disponível em: <https://www.microchip.com/en-us/product/ATMEGA2560>. Acesso em: 16 abr. 2023.

Análise léxica, sintática e semântica. Disponível em: <https://autociencia.blogspot.com/2019/03/automatos-lexico-sintaxe-semantica.html>. Acesso em: 16 abr. 2023.

IEEE-754. Disponível em: <https://www.mathworks.com/help/coder/ug/what-is-half-precision.html>. Acesso em: 16 abr. 2023.