

Linguagem ESTEIRA APS

Contexto da APS

Objetivo da APS:

- Criar uma linguagem de programação completa (variáveis, condicionais, loops)
- Implementar análise léxica e sintática com Flex e Bison
- Compilar para uma VM de destino (MicrowaveVM)

Linguagem criada: ESTEIRA — simula uma esteira ergométrica / equipamento de treino

VM utilizada: MicrowaveVM (VM minimalista estilo Minsky Machine)

Registradores principais: TIME, POWER, TEMP, WEIGHT

Motivação

Por que criar a linguagem ESTEIRA?

- Aproximar programação de um contexto de “máquina real”
- Modelar um dispositivo com tempo, potência e sensores de treino
- Explorar compilação para um alvo extremamente restrito (MicrowaveVM)

Motivação acadêmica:

- Exercitar linguagens formais, análise léxica/sintática e compilação
- Demonstrar Turing-completude em um cenário não convencional

Visão Geral da Linguagem ESTEIRA

Estrutura básica:

```
esteira "nome" { // declarações e comandos }
```

Suporte a:

- Variáveis: var
- Tipos: int, float, bool, string
- Condicionais: se, senao
- Loops: enquanto
- Ações: ligar, desligar, iniciar, parar, definir
- Entrada/Saída: mostrar(...), bip
- Tempo: esperar 3s, esperar 500ms

Principais Construções da Linguagem

Variáveis:

```
var tempo:int = 5;  
var potencia:int = 60;
```

Condicionais:

```
se (tempo > 0) { mostrar(tempo); } senao { bip; }
```

Loops:

```
enquanto (tempo > 0) { mostrar(tempo); tempo = tempo - 1; }
```

Ações:

```
ligar; desligar; parar; definir velocidade = 10;
```

Tempo:

```
esperar 3s; esperar 500ms;
```

EBNF (Resumo da Gramática)

program ::= esteira string { stmt | var_decl }

var_decl ::= var id_list [: type] [= expr] ;

type ::= int | float | bool | string

stmt ::= assign | if_stmt | while_stmt | action_stmt |

in_out_stmt | esperar time_val | bloco

if_stmt ::= se (expr) bloco [senao bloco]

while_stmt ::= enquanto (expr) bloco

action_stmt ::= ligar | desligar | iniciar | parar | definir alvo =
dim_val

in_out_stmt ::= mostrar(expr...) | bip

Arquitetura do Compilador

Pipeline do compilador:

1. Fonte (.est) — linguagem ESTEIRA
2. Análise léxica (Flex / lexer.l) → gera tokens
3. Análise sintática (Bison / parser.y) → valida gramática
4. Ações semânticas e geração de código C
5. Gera .mwasm (Microwave Assembly)
6. Execução na VM → python3 main.py programa.mwasm

MicrowaveVM (VM de Destino)

VM minimalista (estilo Minsky Machine)

Registradores de escrita: TIME, POWER

Registradores leitura: TEMP, WEIGHT

Instruções principais:

- SET R n • INC R • DECJZ R label
- GOTO label • PRINT • HALT

TEMP é atualizado a cada instrução (modelo térmico)

Mapeamento ESTEIRA → MicrowaveVM

Variáveis especiais:

- tempo:int → registrador TIME
- potencia:int → registrador POWER

Exemplo: tempo = 5 → SET TIME 5

Loop exemplo:

L0: DECJZ TIME L1

PRINT

DECJZ TIME L2

L2: GOTO L0

L1:

Exemplos de Código

Exemplo 1 – Programa Básico:

```
esteira "teste_basico" { var tempo:int=3; var potencia:int=50;  
ligar; mostrar(tempo); desligar; parar; }
```

Exemplo 2 – Loop de Contagem:

```
esteira "loop_tempo" { var tempo:int=5; var potencia:int=60;  
ligar; enquanto(tempo>0){ mostrar(tempo); tempo=tempo-1; }  
desligar; parar; }
```

Resultados dos Testes na VM

Teste 01 – Básico:

TIME:3 → BEEEEEP! → Final: TIME=3, TEMP=15

Valida: variáveis, ligar/desligar, mostrar(tempo)

Teste 02 – Loop:

TIME:4 → TIME:2 → TIME:0 → BEEEEEP!

Valida: enquanto(tempo>0), tempo=tempo-1, saída correta

Teste com Esperar

Teste 03 – Esperar:

TIME:0 repetido, TEMP:246°C

Uso do comando esperar (s, ms, min)

Mostra interação entre ESTEIRA e modelo térmico da VM

TIME é usado como contador interno de 'ticks'

Conclusão

A APS implementa:

- Linguagem própria (ESTEIRA)
- Análise léxica com Flex e sintática com Bison
- Backend para VM minimalista (MicrowaveVM)
- Testes reais executando na VM

Aprendizados:

Integração entre teoria e prática

Compilação para máquina extremamente limitada

ESTEIRA dá 'vida' a um modelo teórico

Obrigado pela atenção!