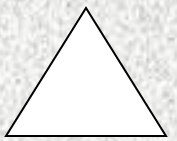
Projeto Orientado em Computação I

Implementação do Processador da Linguagem



Orientado
Eudes da Costa Cândido

Orientadora
Mariza Andrade da Silva Bigonha

Objetivo

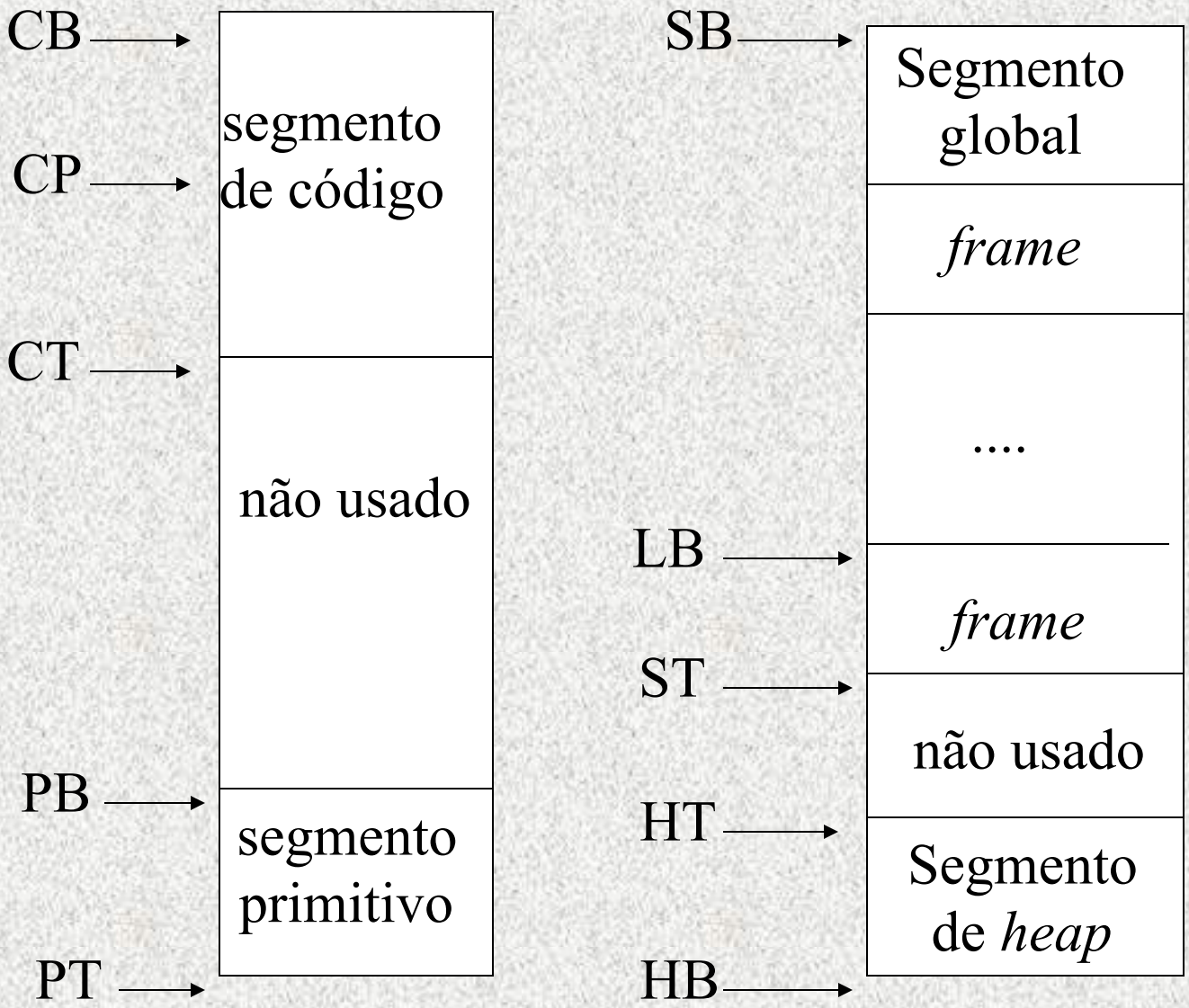
- Produto Principal: Implementação do processador da linguagem Triângulo, consistindo do interpretador, compilador, editor, montador, e desmontador.
- Sub-Produto: Disponibilização de uma ferramenta de estudo e avaliação para a disciplina Compiladores.

A Máquina Abstrata TAM

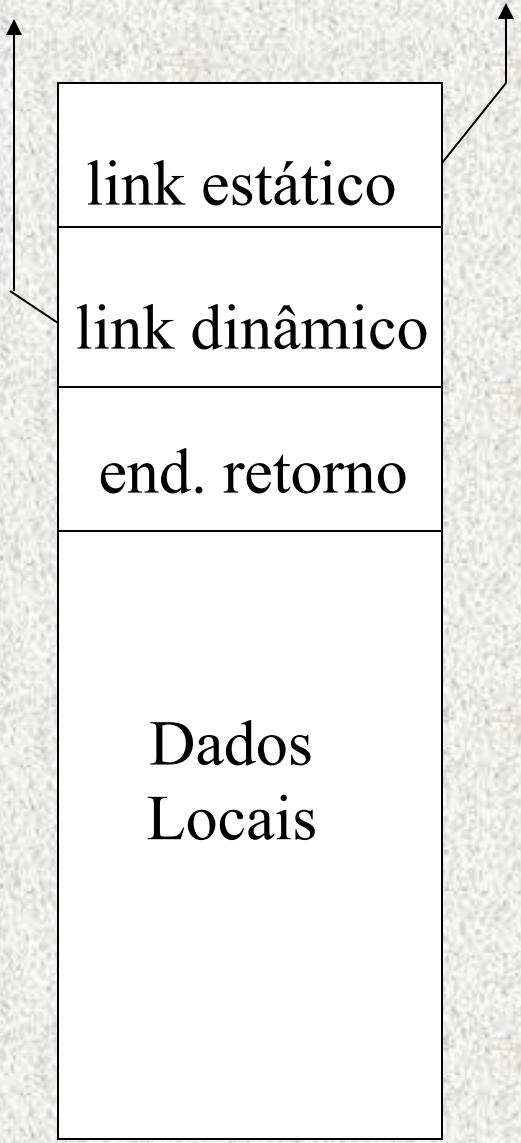
Linguagem Triângulo

- Desenvolvida por David A. Watt, da Universidade de Glasgow para estudo de Caso.
- Linguagem Pascal-*Like*

Memória e Registradores



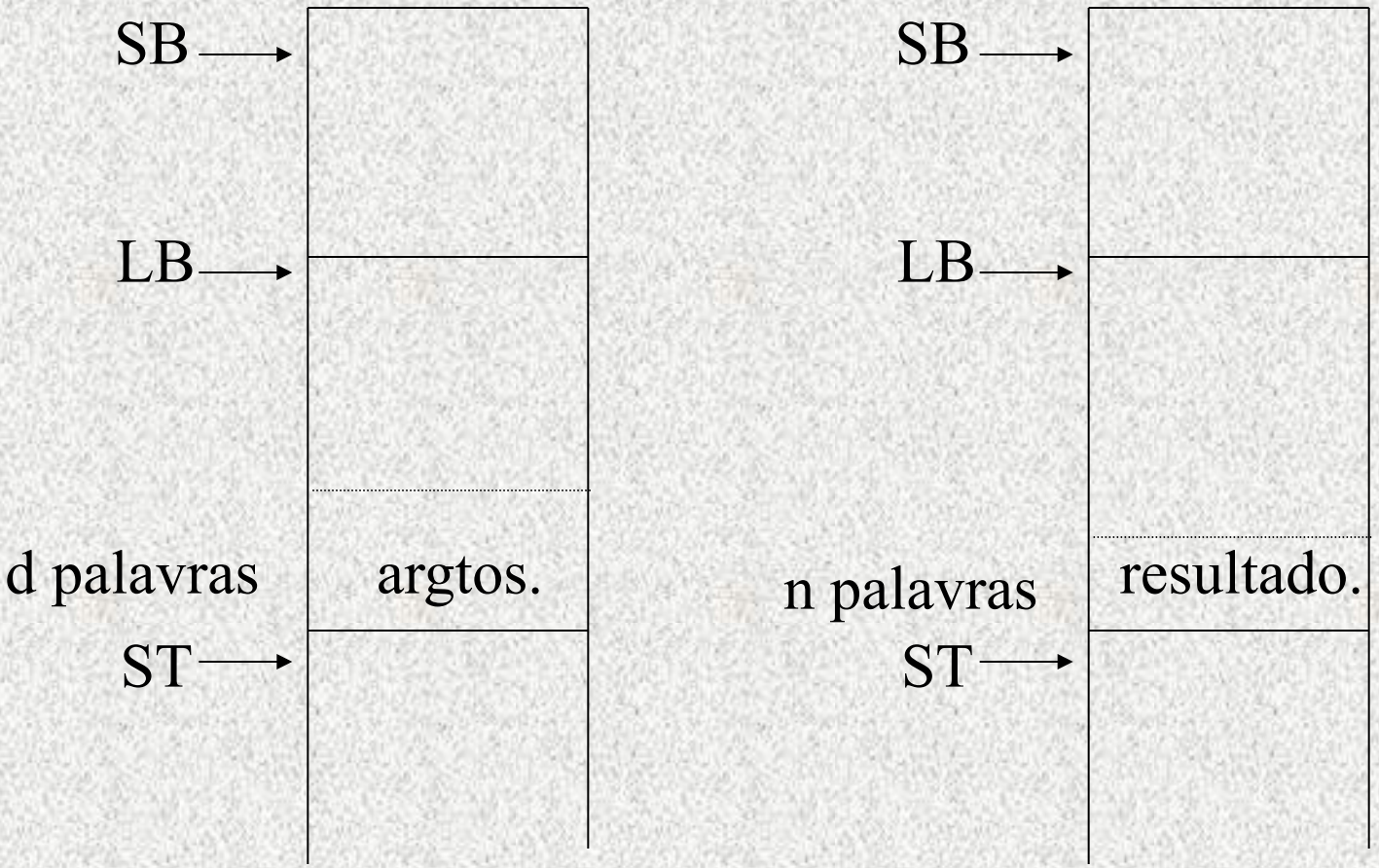
Frame



Instruções

OpCode 4 bits	Reg. 4 bits	n 4 bits	d 16 bits (com sinal)

Rotinas



Instruções

- **Load (n) d[r]**
- **LoadA d[r]**
- **LoadI (n)**
- **LoadL d**
- **Store (n) d[r]**
- **Call (n) d[r]**
- **Return (n) d**
- **Push d**
- **Pop (n) d**
- **Jump d[r]**
- **JumpI**
- **Halt**

Rotinas Primitivas

- id
- add, sub, mult, div, mod, pred, succ, neg
- and, or, not
- lt, le, ge, gt, eq, ne
- get, put
- geteol, puteol
- getint, puting
- new, dispose

Máquina de Pilha

- Pilha: Memória LIFO
- Operações principais: Push e Pop
- Não necessita de endereçamento
- Muito útil na chamada de subrotinas ou funções

Exemplo

Seja, por exemplo, a avaliação da expressão: B + C - D

Em uma máquina de Pilha:

B + C - D
B C + D - (forma posfixada)
push val B, push val C, add, push val D, sub

Em uma máquina de registradores:

```
load r0, B
load r1, C
add r0, r1
load r2, D
sub r0, r2
```

O Interpretador

- Implementação efetuada em Delphi 3, sistema operacional Windows 95/NT.
- Entrada: Instruções em hexadecimal
- Saída: Simulação da Máquina TAM

Exemplo de Teste

- A*b + (1-c*2))

LoadL 7	7
LoadL 2	7, 2
LoadL 3	7, 2, 3
Load (1) 0(SB)	7, 2, 3, 7
Load (1) 1(SB)	7, 2, 3, 7, 2
Call mult	7, 2, 3, 14
LoadL 1	7, 2, 3, 14, 1
Load (1) 2(SB)	7, 2, 3, 14, 1, 3,
LoadL 2	7, 2, 3, 14, 1, 3, 2
Call mult	7, 2, 3, 14, 1, 6
Call sub	7, 2, 3, 14, -5
Call add	7, 2, 3, 9

Conclusão

Nesta primeira etapa do projeto, concluímos parte das capacidades do Processador da Linguagem Triângulo, que são o interpretador, editor e desmontador.

Para a segunda parte, no POC II, pretendemos implementar um Compilador bottom-up para a linguagem Triângulo que complementará o trabalho resultante desta primeira parte.