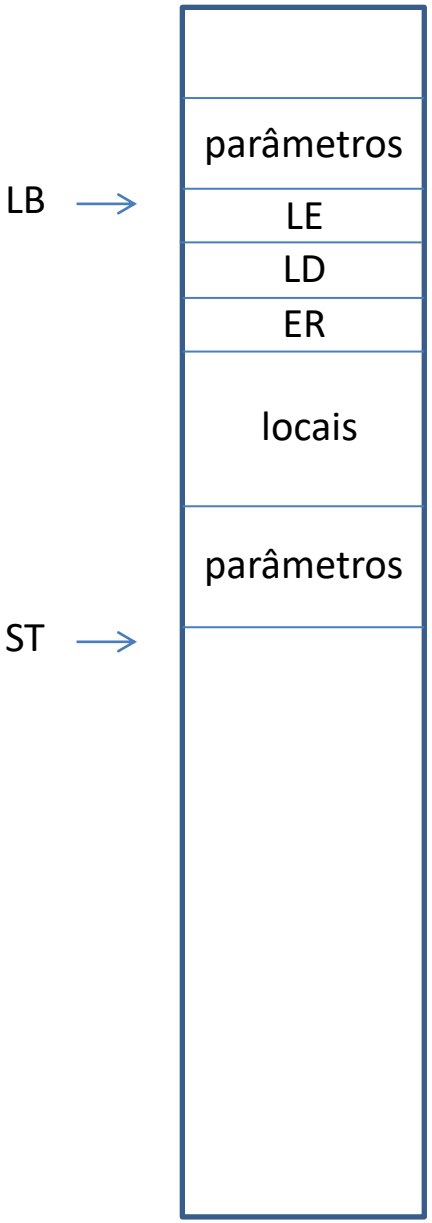
Detalhamento da manipulação de frames na chamada e retorno de subprogramas

Instruções CALL e RETURN

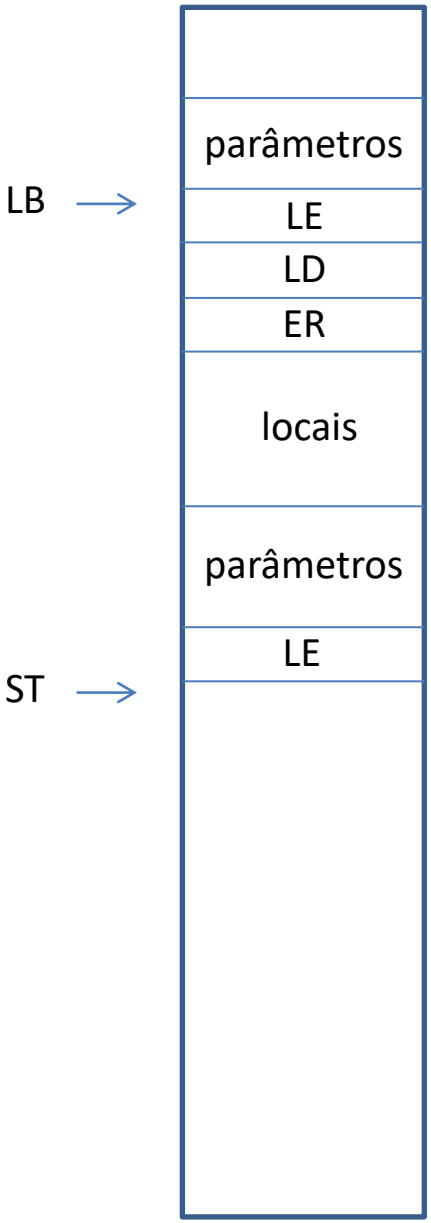
Baseado no Capítulo 6 de Programming Language Processors in Java, de Watt & Brown

CALL (R) <id>

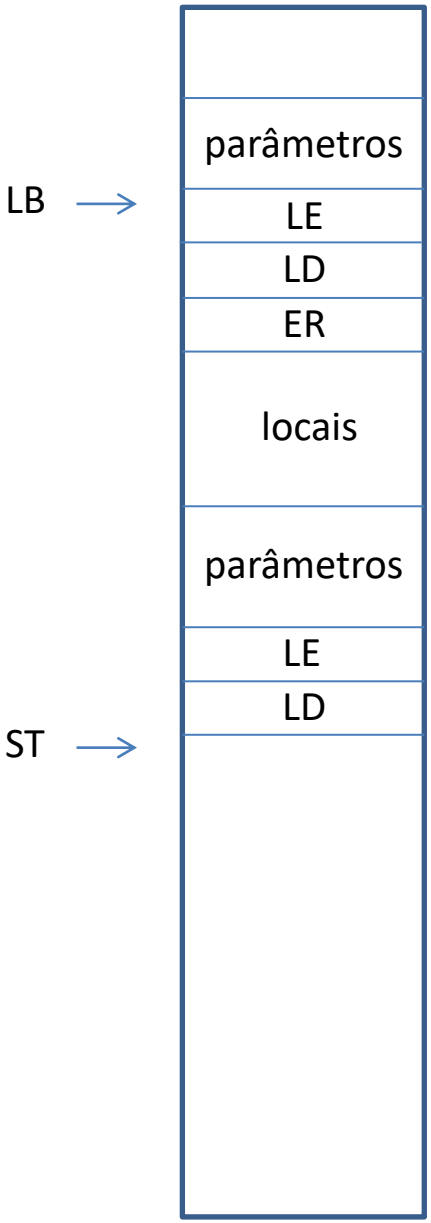
- R é o registrador que contém o LE do novo frame;
- <id> é o nome do subprograma que será chamado.



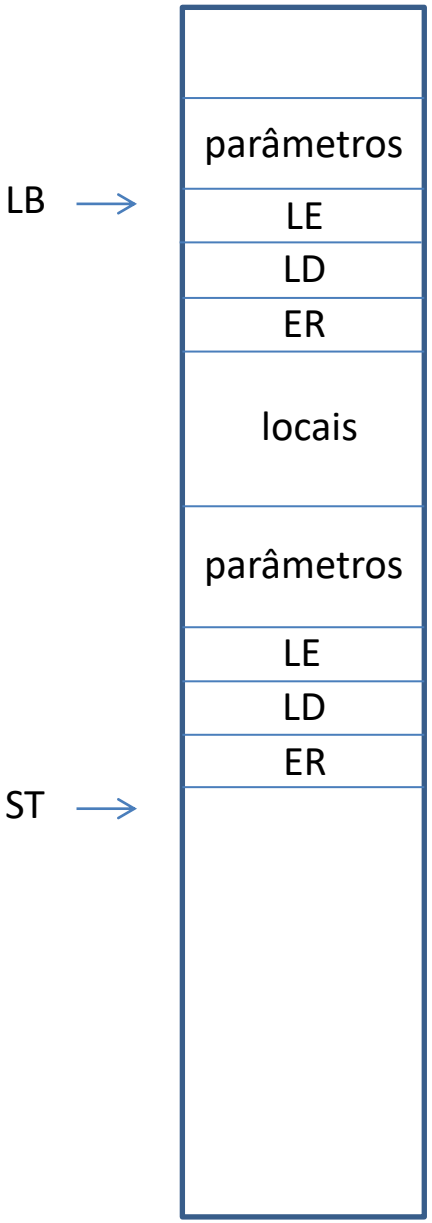
1. Os argumentos são avaliados e empilhados antes do CALL;



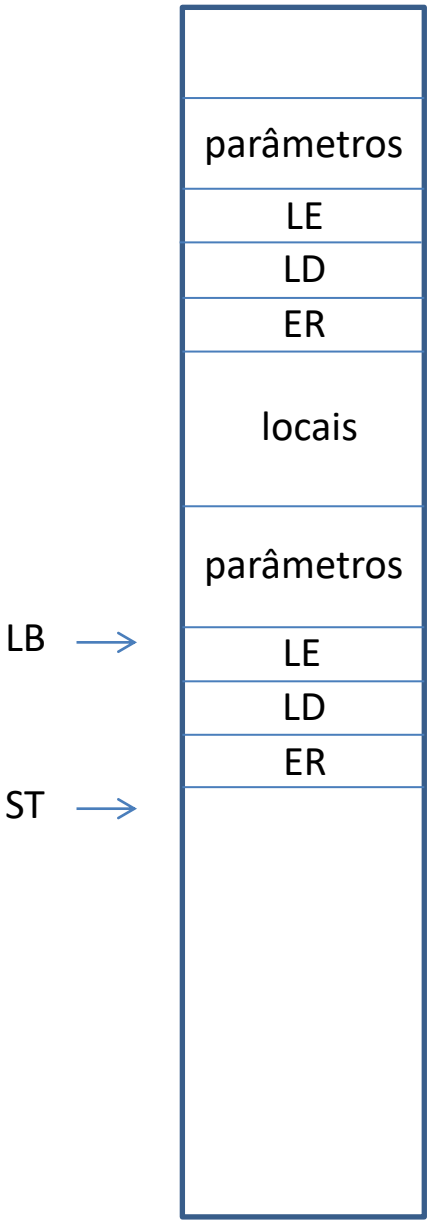
1. Os argumentos são avaliados e empilhados antes do CALL;
2. A execução da instrução CALL provoca os seguintes efeitos:
   a. Empilha o conteúdo de R (novo LE);



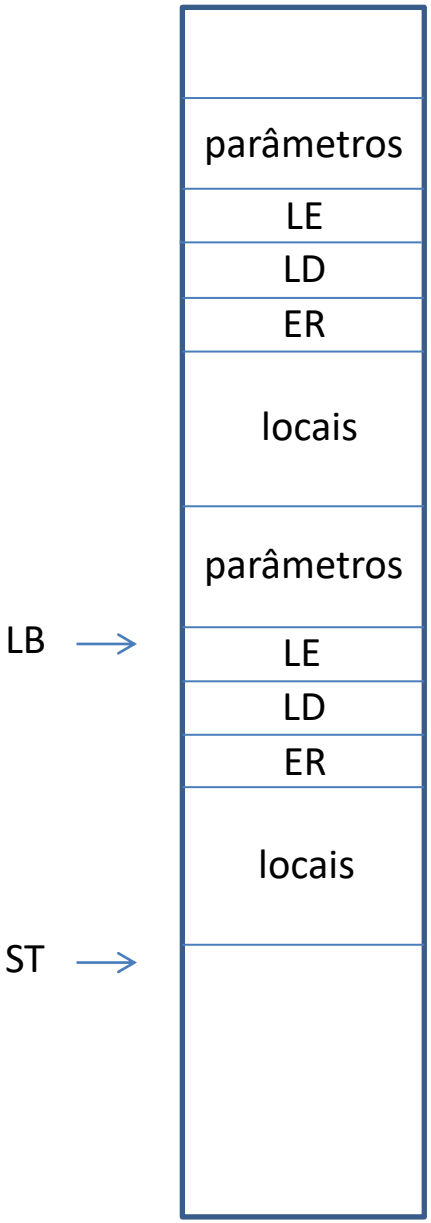
1. Os argumentos são avaliados e empilhados antes do CALL;
2. A execução da instrução CALL provoca os seguintes efeitos:
   a. Empilha o conteúdo de R (novo LE);
   b. Empilha o conteúdo do LB (novo LD);



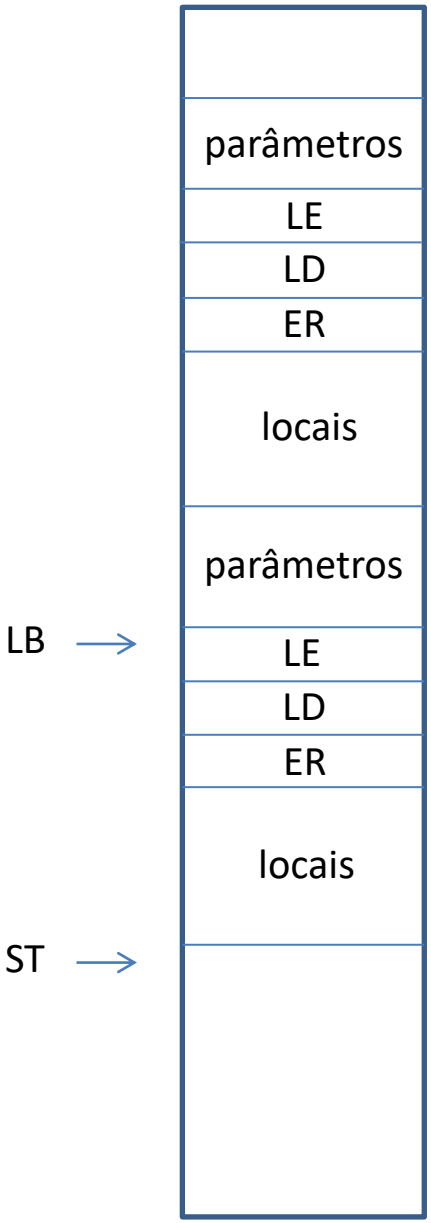
1. Os argumentos são avaliados e empilhados antes do CALL;
2. A execução da instrução CALL provoca os seguintes efeitos:
   a. Empilha o conteúdo de R (novo LE);
   b. Empilha o conteúdo do LB (novo LD);
   c. Empilha o endereço de retorno (novo ER);



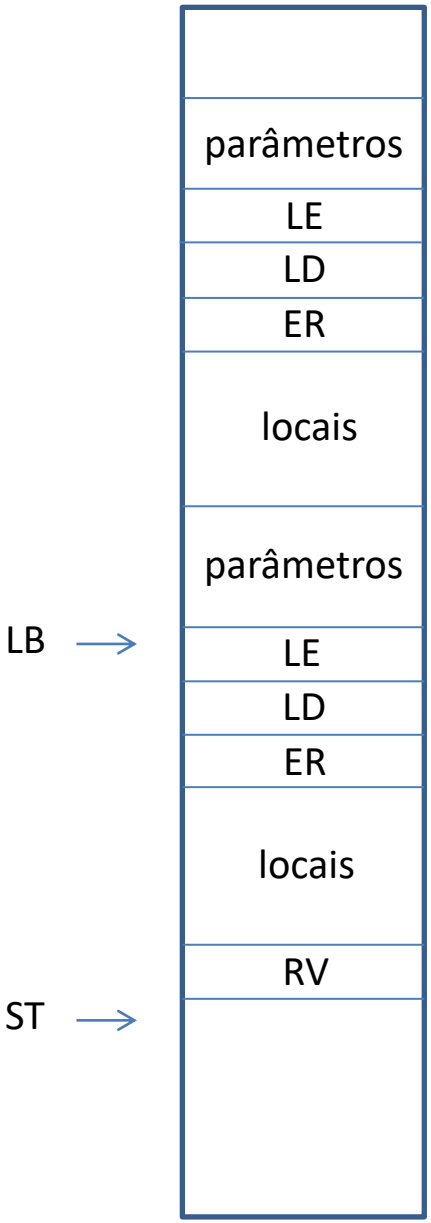
1. Os argumentos são avaliados e empilhados antes do CALL;
2. A execução da instrução CALL provoca os seguintes efeitos:
   a. Empilha o conteúdo de R (novo LE);
   b. Empilha o conteúdo do LB (novo LD);
   c. Empilha o endereço de retorno (novo ER);
   d. LB := ST - 3.



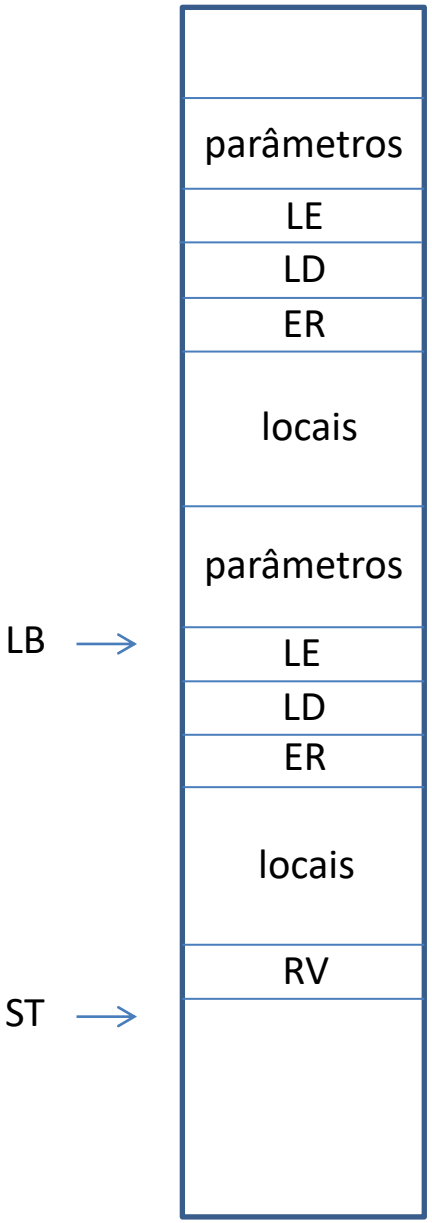
1. Os argumentos são avaliados e empilhados antes do CALL;
2. A execução da instrução CALL provoca os seguintes efeitos:
   a. Empilha o conteúdo de R (novo LE);
   b. Empilha o conteúdo do LB (novo LD);
   c. Empilha o endereço de retorno (novo ER);
   d. Copia o conteúdo do ST para o LB.
3. No início do subprograma são executados os PUSH para alocar as variáveis locais.



1. Os argumentos são avaliados e empilhados antes do CALL;
2. A execução da instrução CALL provoca os seguintes efeitos:
   a. Empilha o conteúdo de R (novo LE);
   b. Empilha o conteúdo do LB (novo LD);
   c. Empilha o endereço de retorno (novo ER);
   d. Copia o conteúdo do ST para o LB.
3. No início do subprograma são executados os PUSH para alocar as variáveis locais.
4. O subprograma é executado;

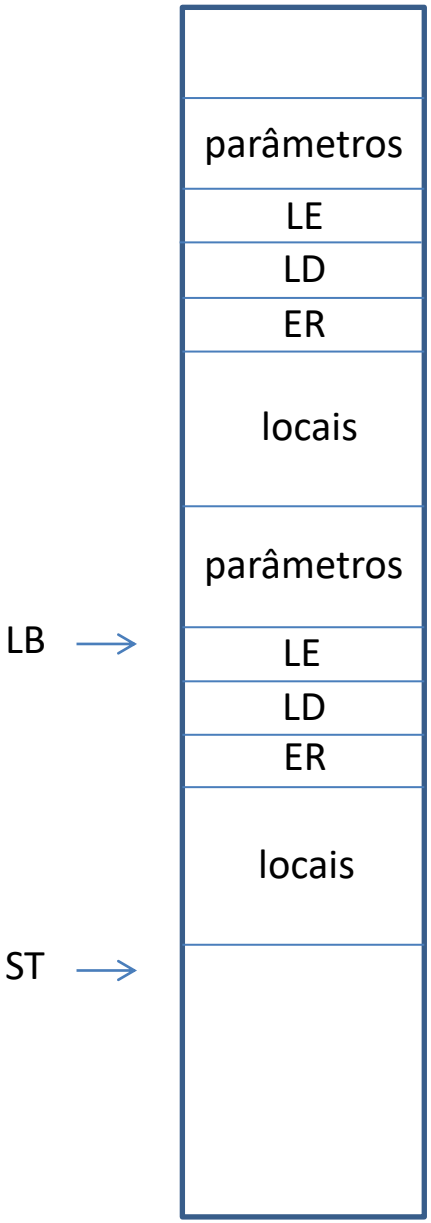


1. Os argumentos são avaliados e empilhados antes do CALL;
2. A execução da instrução CALL provoca os seguintes efeitos:
   a. Empilha o conteúdo de R (novo LE);
   b. Empilha o conteúdo do LB (novo LD);
   c. Empilha o endereço de retorno (novo ER);
   d. Copia o conteúdo do ST para o LB.
3. No início do subprograma são executados os PUSH para alocar as variáveis locais.
4. O subprograma é executado;
5. Se o subprograma for uma função, haverá um valor de retorno no topo da pilha (RV).

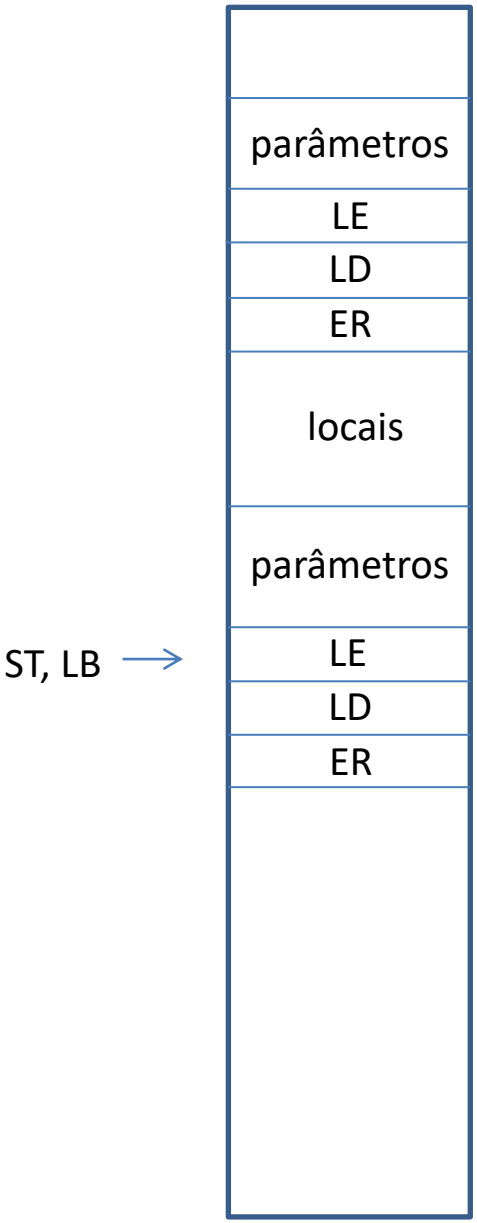


RETURN (n) d

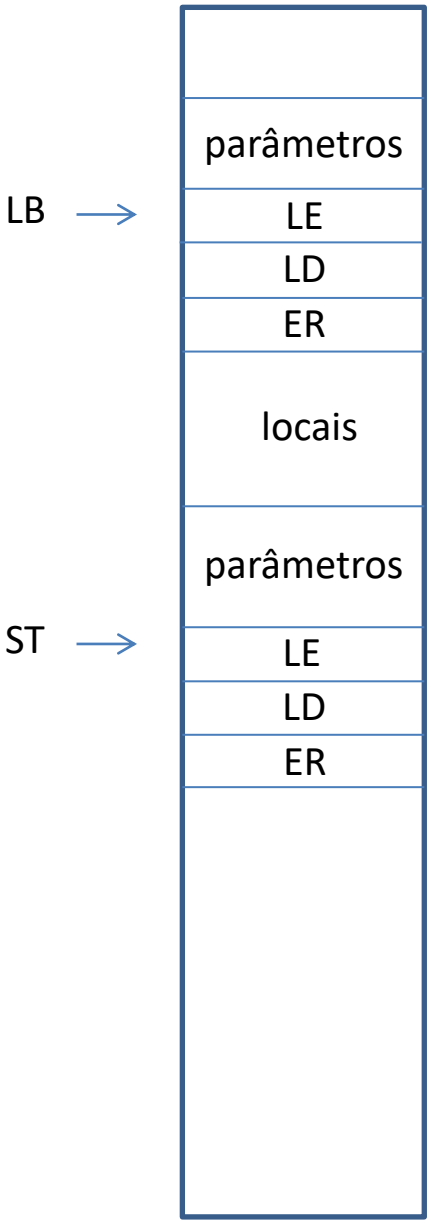
- “n” é o número de posições de memória usadas para o valor de retorno (RV, que está no topo da pilha);
- “d” é o número de posições de memória usadas pelos parâmetros.



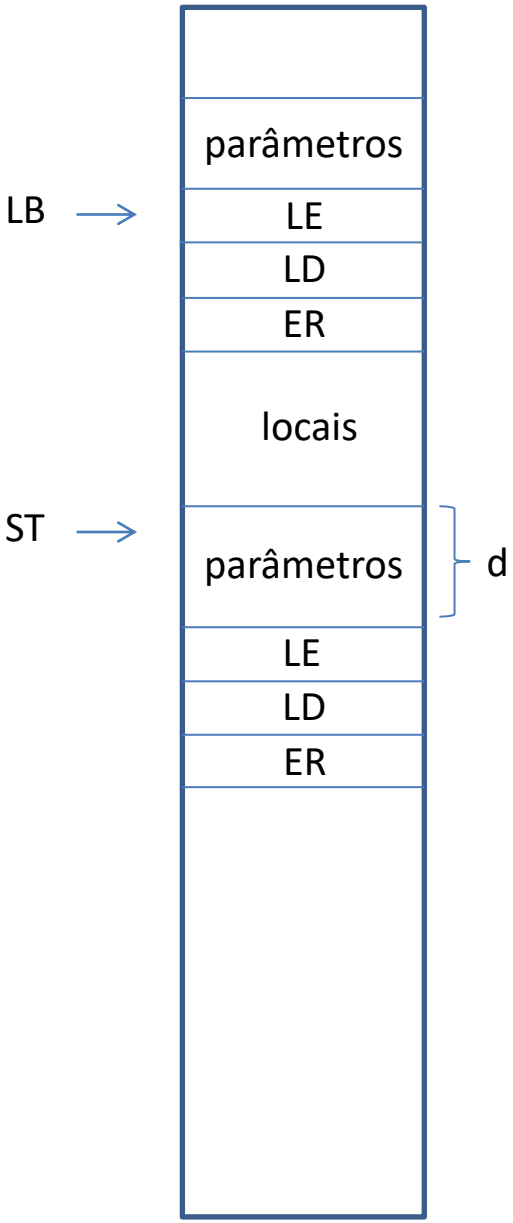
1. Desempilha o valor de retorno (RV, “n” posições do topo da pilha) e armazena num registrador temporário T_1;



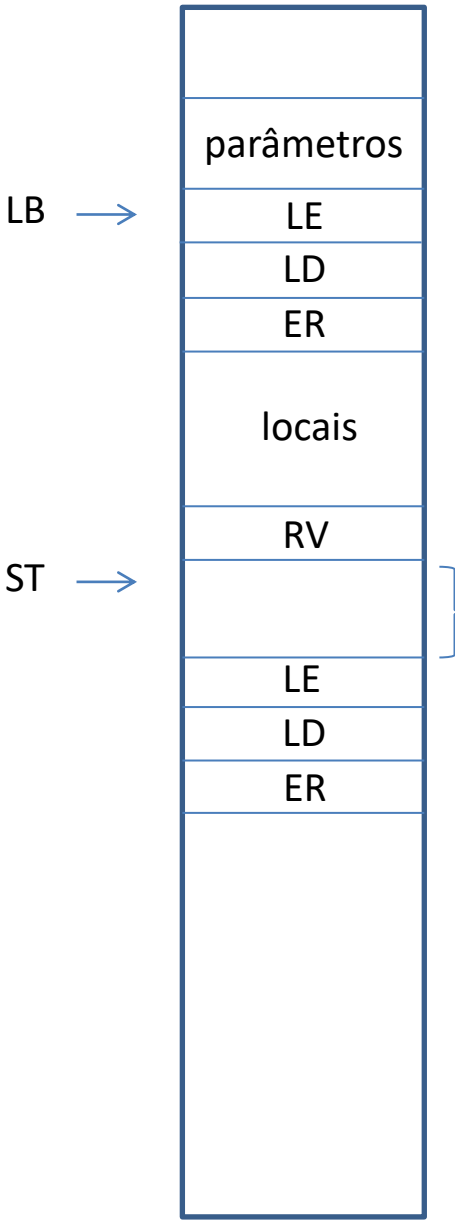
1. Desempilha o valor de retorno (RV, “n” posições do topo da pilha) e armazena num registrador temporário T_1;
2. ST:=LB;



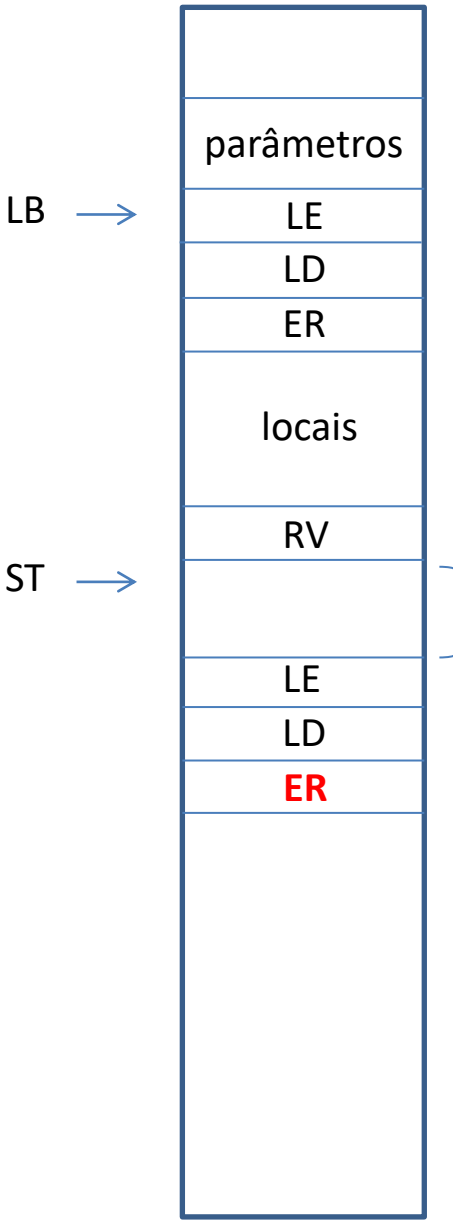
1. Desempilha o valor de retorno (RV, “n” posições do topo da pilha) e armazena num registrador temporário T_1;
2. ST:=LB;
3. LB:=(ST)+1;



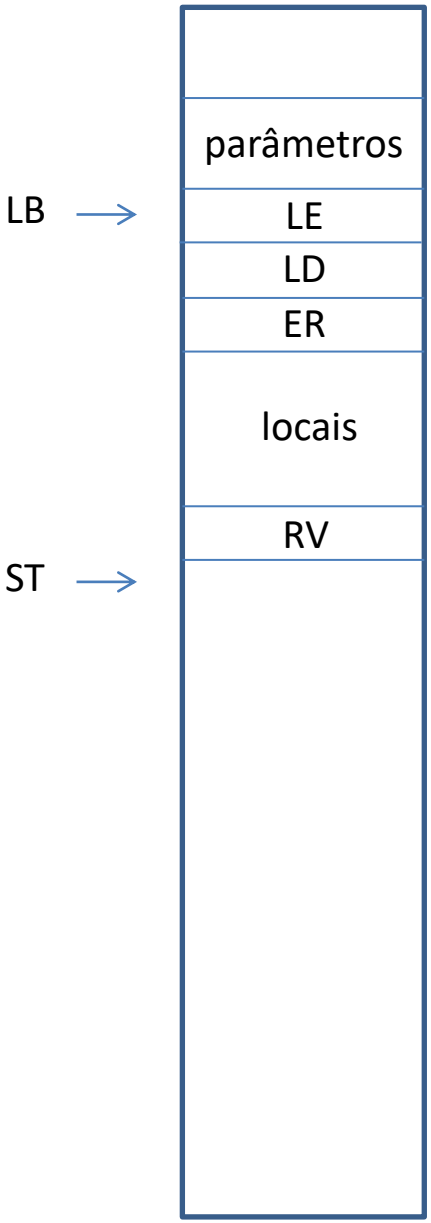
1. Desempilha o valor de retorno (RV, “n” posições do topo da pilha) e armazena num registrador temporário T_1;
2. ST:=LB;
3. LB:=(ST)+1;
4. ST:=(ST)-d;



1. Desempilha o valor de retorno (RV, “n” posições do topo da pilha) e armazena num registrador temporário T_1;
2. ST:=LB;
3. LB:=(ST)+1;
4. ST:=(ST)-d;
5. Empilha T_1 (RV, com “n” posições de memória);



1. Desempilha o valor de retorno (RV, “n” posições do topo da pilha) e armazena num registrador temporário T_1;
2. ST:=LB;
3. LB:=(ST)+1;
4. ST:=(ST)-d;
5. Empilha T_1 (RV, com “n” posições de memória);
6. [Localização do ER]: Desvia para o endereço de retorno (ER) armazenado em (ST)+d-size(RV)+2;



1. Desempilha o valor de retorno (RV, “n” posições do topo da pilha) e armazena num registrador temporário T_1;
2. ST:=LB;
3. LB:=(ST)+1;
4. ST:=(ST)-d;
5. Empilha T_1 (RV, com “n” posições de memória);
6. [Localização do ER]: Desvia para o endereço de retorno (ER) armazenado em (ST)+d-size(RV)+2;
7. A situação é idêntica à inicial, exceto pela eventual presença do valor de retorno (RV) no topo da pilha.

- Não é necessário fazer POP para desalocar as variáveis locais;
- Presume-se que size(RV)<=d+size(LE)+size(LD).