Linguagens de Programação

Fabio Mascarenhas - 2017.2

http://www.dcc.ufrj.br/~fabiom/lp

Exceções e a pilha

- Quando abandonamos cada escopo a pilha retorna para o ponto onde estava quando entramos naquele escopo
- Mas o mecanismo atual só faz isso se a execução chegou até o final do escopo
- Quando introduzimos exceções em MicroC isso não é mais verdade: as exceções vazam memória!
- Podemos corrigir isso se *trycatch* fizer esse retorno da pilha caso alguma exceção tenha acontecido
- Para isso quebramos a linearidade do *stack pointer*, isso quer dizer que uma implementação imperativa precisa guardar o valor atual de *sp* antes de um bloco *try*

Passagem por referência

- A passagem por referência é um caso restrito da passagem por nome:

```
fun troca(_a, _b)  -- a e b são por ref
  let tmp = _a in
    _a := _b;
    _b := tmp
  end
end
let x = 1, y = 2 in
  -- troca os valores x e y!
  troca(x, y);
  x - y
end
```

- O corpo de *troca* é avaliado como se estivéssemos substituindo *_a* e *_b* por *x* e *y*, onde *x* e *y* mantém as associações do escopo onde foram definidos

Passagem por referência

- Com a substituição do corpo de *troca*, avaliamos a seguinte expressão:

```
let tmp = x in
  x := y;
  y := tmp
end
```

- É fácil ver que isso realmente troca os valores das variáveis *x* e *y*!

- A passagem por referência adiciona uma restrição onde os argumentos precisam ser *lvalues*: apenas expressões que podem aparecer do lado esquerdo de uma atribuição

- Os parâmetros por referência são ponteiros dereferenciados implicitamente em cada uso