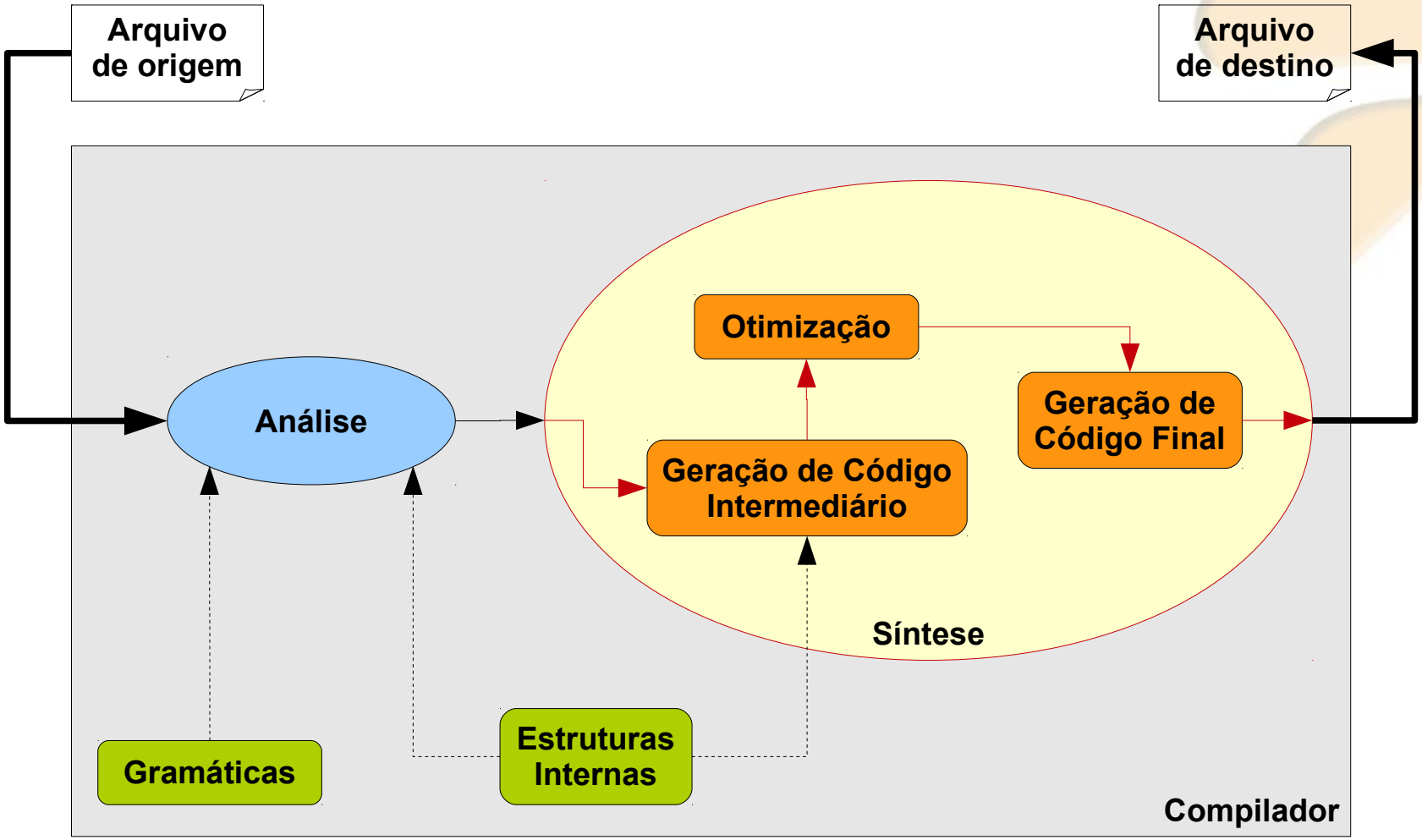
Compiladores

Gerência de Memória

Cristiano Lehrer, M.Sc.

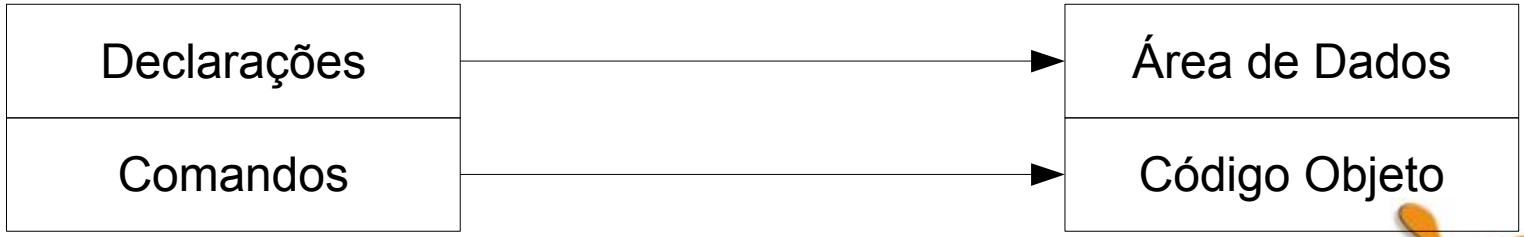
ybadoo

Atividades do Compilador



Introdução (1/2)

- No contexto da compilação, o termo gerência de memória refere-se às atividades de alocação e liberação de memória para dados (variáveis e constantes), em tempo de compilação e em tempo de execução.
- A alocação e a liberação de memória para dados são gerenciadas por um conjunto de rotinas carregado junto com o código objeto gerado.
- Um programa objeto compõem-se de código e área de dados:
  - O código é produzido pelo gerador de código objeto.
  - A área para dados é reservada pelas funções do gerente de memória.

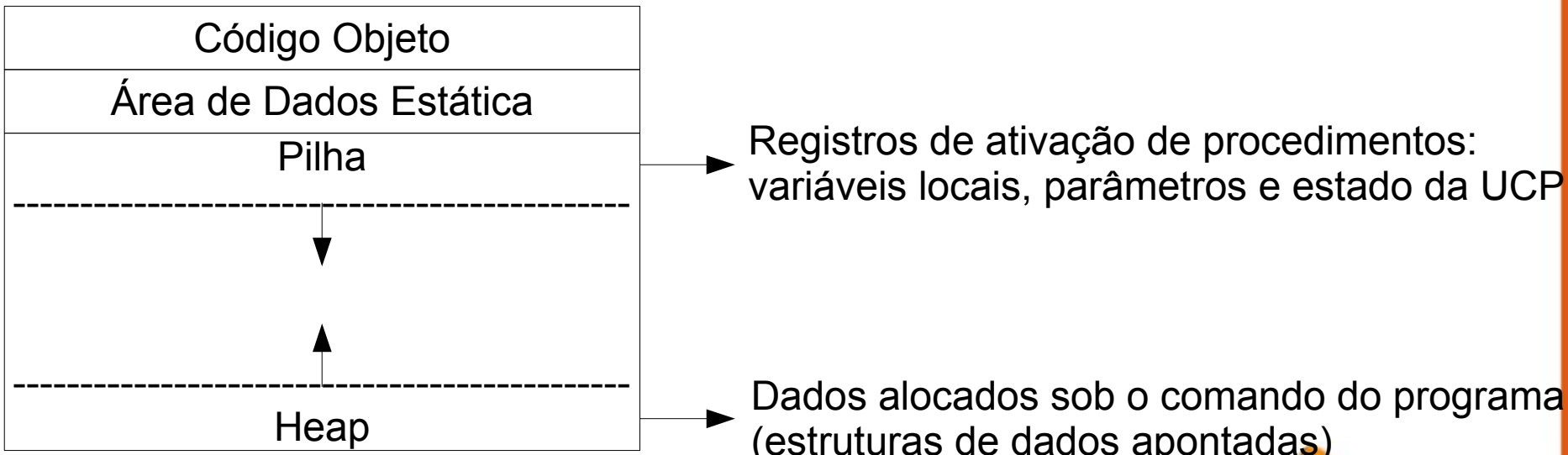


Introdução (2/2)

- O armazenamento de um dado na memória em tempo de execução depende do tipo desse dado:
  - Em geral, tipos de dados elementares como caracteres, inteiros e reais ocupam, cada um, uma palavra de memória.
  - Agregados como vetores, cadeias de caracteres e registros vão ocupar várias posições contíguas de memória.
- Se o tipo e o tamanho das variáveis são conhecidos em tempo de compilação, a reserva de espaço de memória é decidida na compilação.
  - Caso contrário, o espaço será alocado durante a execução do programa objeto.

Estratégias de Alocação de Memória (1/3)

- A memória para o programa compilado deve conter:
  - O código objeto gerado.
  - O espaço para as variáveis globais (área estática).
  - A pilha para ativação de procedimentos.
  - O espaço para memória dinâmica (*heap*).



Estratégias de Alocação de Memória (2/3)

- A reserva de memória para dados pode ser feita de três maneiras:
  - Alocação estática:
    - Se o tipo (ou comprimento) do dado é conhecido em tempo de compilação e não é modificado durante a execução do programa, a reserva de memória é feita durante a compilação, de forma estática.
  - Alocação em pilha:
    - Áreas para dados locais de procedimentos (subrotinas ou funções) devem ser alocadas dinamicamente.
    - Embora o tamanho seja, muitas vezes, conhecido tem tempo de compilação, a alocação de espaço de memória somente pode ser realizada em tempo de execução, porque a ordem de chamadas é determinada pela execução do programa.
    - Essas áreas são, em geral, alocadas numa estrutura em pilha (pilha de ativação de procedimentos), a qual pode ser a própria pilha de execução da máquina.

Estratégias de Alocação de Memória (3/3)

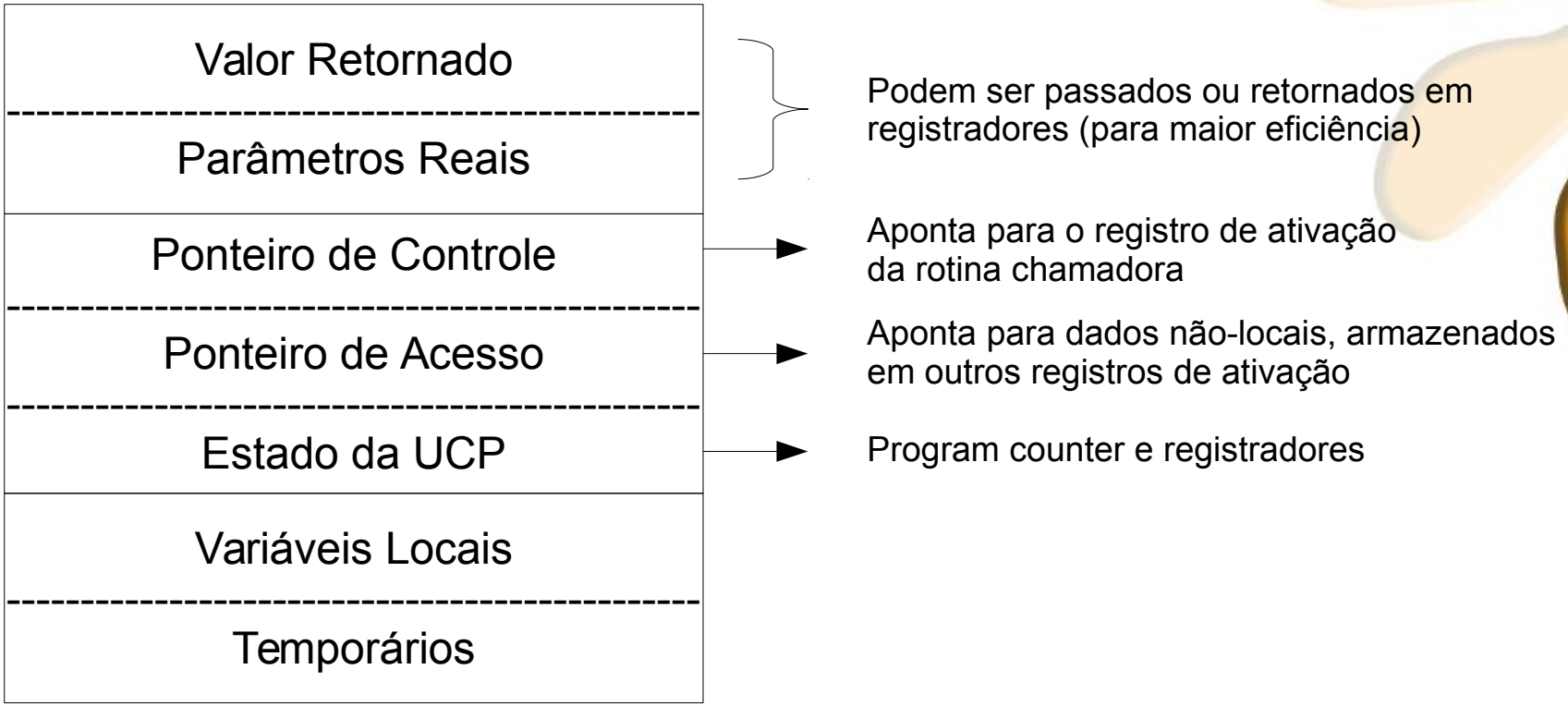
- Alocação dinâmica:
  - Para as estruturas de dados referenciadas através de ponteiros, as áreas também são reservadas dinamicamente.
  - Essas áreas são alocadas e liberadas, sob o controle do programa, por comandos específicos (por exemplo, `new` e `dispose`, em Pascal).
  - Normalmente, essas estruturas são alocadas na área denominada *heap*, que cresce no sentido contrário ao da pilha.

Alocação em Memória de Pilha (1/2)

- A memória de pilha é usada para armazenamento das informações relativas às chamadas de procedimentos.
- Para cada chamada, é empilhado um registro de ativação (RA), o qual contém informações sobre o estado da UCP no momento da chamada, uma área de dados para as variáveis locais, uma área para parâmetros e alguns ponteiros de controle da própria pilha.
- Em cada retorno de procedimento, é desempilhado um registro de ativação.

ybadoo

Alocação em Memória de Pilha (2/2)



Passagem de Parâmetros (1/2)

- Modelos Semânticos de Passagem de Parâmetros:
  - Modo entrada (*in mode*).
  - Modo saída (*out mode*).
  - Modo de entrada/saída (*inout mode*).
- Modelos Conceituais de Transferência:
  - Um valor real é transferido fisicamente.
  - Um caminho de acesso é transmitido.

ybadoo

Passagem de Parâmetros (2/2)

- Modelos de Implementação de Passagem de Parâmetros:
  - Passagem por Valor.
  - Passagem por Resultado.
  - Passagem por Valor-Resultado.
  - Passagem por Referência.
  - Passagem por Nome.

Passagem por Valor

- Passagem por Valor (*in mode*):
  - O valor do parâmetro real é usado para inicializar o parâmetro formal correspondente, que, então age como uma variável local no subprograma.
  - Normalmente implementada pela transferência de dados real, mas pode também utilizar o caminha de acesso.
  - Desvantagens do acesso pelo método do caminho:
    - O valor deve estar numa célula protegida contra a escrita.
    - Acesso mais complicado.
  - Desvantagens do método de cópia física:
    - Requer mais armazenamento.
    - Custo para transferir fisicamente o parâmetro.

Passagem por Resultado

- Passagem por resultado (*out mode*):
  - Valores locais são passados de volta ao chamador.
  - Transferência física é usualmente utilizada.
  - Desvantagens:
    - Se o valor é passado, tempo e espaço.
    - Em ambos os casos, a dependência da ordem pode ser problema:
      - Exemplo na sintaxe do Pascal:

```
procedure sub1 (y : integer, z : integer);

…

sub1 (x, x);
```

  - O valor de x no chamador dependerá da ordem dos assinalamentos no retorno.

Passagem por Valor-Resultado

- Passagem por valor-resultado (*inout mode*):
  - Transferência física, em ambos os modos.
  - Também chamado de passagem por cópia.
  - Desvantagens:
    - Os referentes ao passagem por resultado.
    - Os referentes ao passagem por valor.

ybadoo

Passagem por Referência (1/2)

- Passagem por referência (*inout mode*):
  - Passagem pelo método de caminho de acesso.
  - Também conhecido como passagem por compartilhamento.
  - Vantagens:
    - Processo de passagem de parâmetros é eficiente.

Passagem por Referência (2/2)

- Desvantagens:
  - Acesso lento.
  - Pode permitir *aliasing*:
    - Colisão de parâmetros – exemplo na sintaxe do Pascal:

```
procedure sub1(a: integer, b: integer);

…

sub1(x, x);
```

    - Colisão de elementos de array – exemplo na sintaxe do Pascal:

```
sub1 (a[i], a[j]; /* se i = j */

sub2 (a, a[i]);
```

    - Colisão entre parâmetros formais e globais.

Passagem por Nome (1/3)

- Passagem por Nome (múltiplos modos):
  - Parâmetros passados por nome, o parâmetro real é, com efeito, textualmente substituído para o parâmetro formal correspondente em todas as suas ocorrências no subprograma.
  - Parâmetros formais são vinculados a valores ou a endereços reais no momento da chamada ao subprograma.
  - Propósito:
    - Flexibilidade na vinculação tardia.
  - Resultados semânticos:
    - Se o parâmetro real for uma variável escalar, será passado por referência.
    - Se o parâmetro real for uma expressão constante, será passado por valor.
    - Se o parâmetro real for um elemento de *array*, será passado diferente de qualquer em dos métodos estudados.

Passagem por Nome (2/3)

- Exemplo, em ALGOL:

```
procedure BIGSUB;
    integer GLOBAL;
    integer array LIST[1 : 2];
    procedure SUB(PARAM);                    begin
        integer PARAM;                          LIST[1]
:= 2;
        begin                                   LIST[2]
:= 2;
                PARAM := 3;
GLOBAL := 1;
                GLOBAL := GLOBAL + 1;
SUB(LIST[GLOBAL]);
```

Passagem por Nome (3/3)

- Desvantagens da passagem por nome:
  - Referência muito ineficientes.
  - São difíceis de implementar e podem confundir tanto leitores como os escritores de programas.

ybadoo