

Compiladores

Ambiente de suporte à execução

Suporte à Execução

- O Compilador gera código executável.
- Mas nem tudo está conhecido antes que o programa seja executado!
 - Valores de parâmetros e funções,
 - Memória dinamicamente alocada,
 - Dependendo do número de chamadas, qual endereço usar para achar o início de cada execução de um procedimento?
 - Etc...
- É preciso de um conjunto de rotinas (*run-time support package*) carregado junto com o código objeto gerado

Procedimentos

- Os comandos são organizados em procedimentos:
 - declaração com um nome associado que realiza uma dada tarefa
 - definição:
 - nome
 - variáveis
 - corpo

Procedimentos em ação (ativação)

- Fluxo de controle:
 - execução sequencial (seqüência de passos)
 - começa no início do corpo
 - termina no final do corpo
 - tempo de vida:
 - seqüência de passos executados
 - chamada de procedimentos:
 - desvio de execução
 - retorna o controle para o ponto imediatamente após o ponto de chamada

Exemplo: o Quicksort

```
program sort(input, output);
  var a : array [0..10] of integer;
  procedure readarray;
    var i: integer;
    begin
      for i:=1 to 9 do read(a[i]);
    end;
  function partition(y, z: integer): integer;
    var i, j, x, v: integer; begin ... end;
  procedure quicksort(m, n: integer);
    var i: integer;
    begin
      if ( n > m ) then begin
        i:= partition(m, n);
        quicksort(m, i-1);
        quicksort(i+1, n);
      end;
    end;
  begin
    a[0] := -9999; a[10] := 9999; readarray;
    quicksort(1, 9);
  end.
```

Árvores de Ativação

- Cada nó representa uma ativação de um procedimento
- A raiz representa a ativação do programa principal
- O nó de 'a' é pai de 'b' se e somente se o fluxo de controle muda de 'a' para 'b'
- O nó de 'a' está à esquerda de 'b' se e somente se a vida de 'a' ocorre antes de 'b'

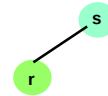
Árvores de ativação

execution begins



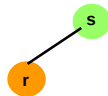
Árvores de ativação

execution begins
enter readarray



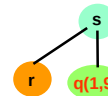
Árvores de ativação

execution begins
enter readarray
leave readarray



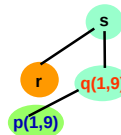
Árvores de ativação

execution begins
enter readarray
leave readarray
enter quicksort(1, 9)



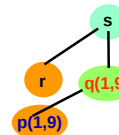
Árvores de ativação

execution begins
enter readarray
leave readarray
enter quicksort(1, 9)
enter partition(1, 9)



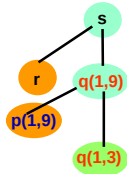
Árvores de ativação

execution begins
enter readarray
leave readarray
enter quicksort(1, 9)
enter partition(1, 9)
leave partition(1,9)



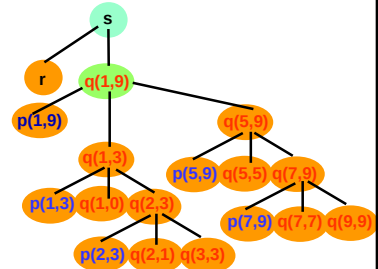
Árvores de ativação

execution begins
enter readarray
leave readarray
enter quicksort(1, 9)
enter partition(1, 9)
leave partition(1, 9)
enter quicksort(1, 3)



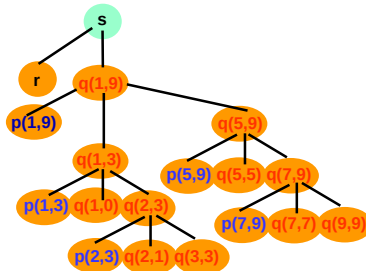
Árvores de ativação

execution begins
enter readarray
leave readarray
enter quicksort(1, 9)
enter partition(1, 9)
leave partition(1, 9)
enter quicksort(1, 3)
...
leave quicksort(5, 9)



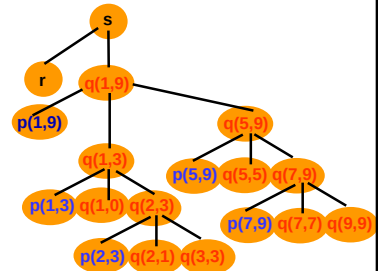
Árvores de ativação

execution begins
enter readarray
leave readarray
enter quicksort(1, 9)
enter partition(1, 9)
leave partition(1, 9)
enter quicksort(1, 3)
...
leave quicksort(5, 9)
leave quicksort(1, 9)



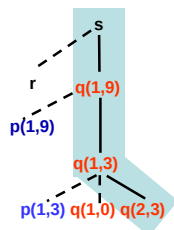
Árvores de ativação

execution begins
enter readarray
leave readarray
enter quicksort(1, 9)
enter partition(1, 9)
leave partition(1, 9)
enter quicksort(1, 3)
...
leave quicksort(5, 9)
leave quicksort(1, 9)
execution terminated



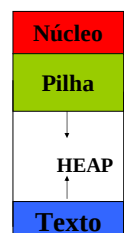
Pilhas de controle

- O fluxo de controle corresponde a uma **busca em profundidade** na árvore de ativação
- Usa-se uma pilha para controlar as ativações de procedimentos **ativos**



Organização de Memória

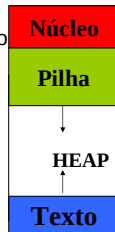
- Como a memória do programa é armazenada ?
 - código objeto gerado;
 - espaço para variáveis globais
 - área estática
 - pilha para ativação de procedimentos
 - espaço para memória dinâmica (heap).



Alocação de memória

Alocação estática:

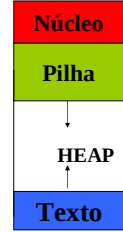
- reserva de memória é feita durante a compilação, de forma estática.
- Tipo (ou comprimento) do dado é conhecido em tempo de compilação
- Comprimento não é modificado durante a execução do programa



Alocação de memória

Alocação dinâmica (HEAP):

- estruturas de dados referenciadas através de ponteiros, as áreas também são reservadas dinamicamente.
- áreas são alocadas e liberadas, sob o controle do programa(dor)
- alocadas na área de "heap", que cresce no sentido contrário ao da pilha.

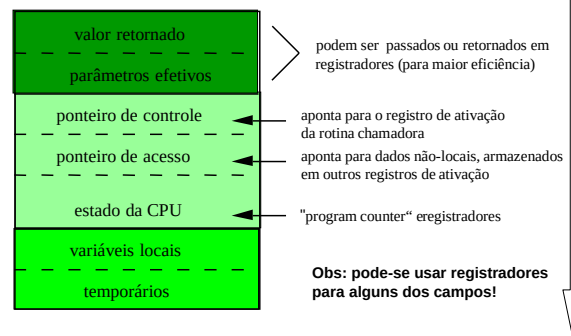


Alocação de memória

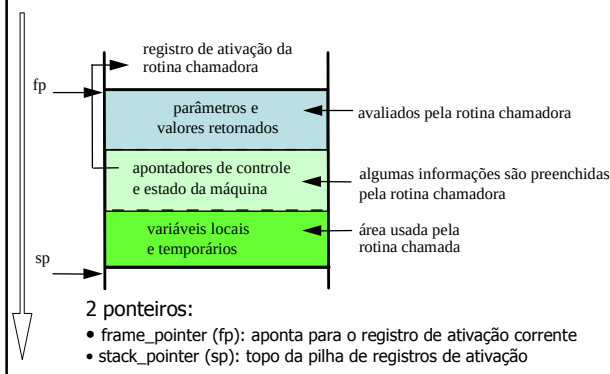
Alocação em pilha (STACK):

- Áreas para dados locais de procedimentos (subrotinas ou funções): devem ser alocadas dinamicamente.
- 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.
- Áreas são alocadas numa estrutura em pilha de ativação de procedimentos
 - Liberação quando retorna!
- Na pilha entram (e saem) **registros de ativação**.

Registro de Ativação



Ponteiros de Acesso



Código de Chamada

rotina chamadora:

- avalia os parâmetros efetivos e os coloca na pilha
- Registradores em uso pelo chamador são salvos em memória
- armazena o endereço de retorno e o valor antigo do frame_pointer no registro de ativação da rotina chamada e atualiza o valor do frame_pointer;

rotina chamada:

- salva valores de registradores e outras informações do estado da máquina;
- inicializa variáveis locais e começa sua execução.

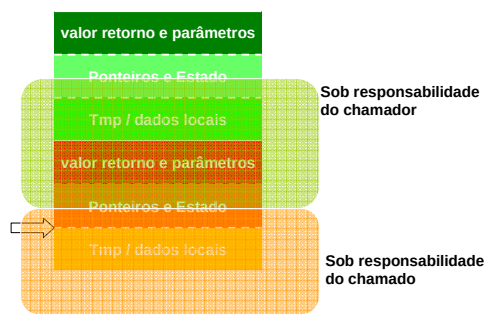
Código de Retorno

- **rotina chamada:**
 - armazena o valor de retorno logo após o registro de ativação da rotina chamadora;
 - restaura o apontador topo_ra e os registradores da máquina e desvia para o endereço de retorno dentro da rotina chamadora;
- **rotina chamadora:**
 - copia o valor retornado no seu próprio registro de ativação

Seqüência de chamada



Seqüência de chamada



Seqüência de retorno

- O chamado coloca um valor de retorno no devido lugar...
 - Ou seja, logo depois do registro do chamador!
- Recupera o antigo valor de 'topo' no campo de estado
 - E também os outros registradores...
- Desvia para o antigo PC
- Importante: o chamador ainda pode acessar o que está acima de 'topo'
 - ... Questão: o que está neste lugar?

Passagem de Parâmetros

- **Passagem de parâmetros por valor**
 - Método mais simples de passagem de parâmetros
 - Parâmetros são avaliados, e seus valores são passados para o procedimento chamado.
 - Implementação:
 - um parâmetro formal é tratado exatamente como um nome local, de maneira que a memória para os parâmetros formais é reservada no registro de ativação do procedimento chamado;
 - o procedimento chamador avalia os parâmetros reais e armazena seus valores na memória reservada para os parâmetros formais.

Passagem de Parâmetros

- **Passagem de parâmetros por endereço**
 - procedimento chamador passa o endereço de cada parâmetro real.
 - se um parâmetro real é:
 - Identificador:
 - seu endereço é passado;
 - Expressão:
 - expressão é avaliada num temporário, e o endereço desse temporário é passado;
 - Referências a parâmetros formais, no procedimento chamado, são feitas de forma indireta.

Próxima aula...

- Fim do suporte ao runtime:
 - Passagem de parâmetros
 - Escopo de variável
 - Tabela de símbolos com escopo...