COMPILADORES 2019/2020

aula 0x06 - Análise Semântica - Registo de Activação

02/04/2020

Pedro Patinho <pp@di.uevora.pt>

Universidade de Évora - Departamento de Informática

SUMÁRIO

RECAPITULANDO...

RECAPITULANDO...

1. Análise lexical
   - produção de 'tokens'
2. Análise sintáctica
   - definição da gramática
   - produção da 'APT'

     definição das 'classes' para os nós da APT
3. Análise semântica

   3.1 nomes, tipos

     'symbol table'

     extensão das 'classes' da APT

   3.2 Preparação para tempo de execução

     Registo de activação (ou 'stack frame')

A STACK

A STACK

- Contém os argumentos da função chamada
  - Colocados pelo chamador
- Contém as variáveis locais da função
  - Colocadas pela função chamada
- Contém o valor de retorno
  - Colocado por quem?
  - E onde?

VARIÁVEIS LOCAIS

VARIÁVEIS LOCAIS NA STACK

- Dependem da linguagem
- Linguagens "não reentrantes"
  - Funções/procedimentos não podem ser recursivos (ex: Fortran, COBOL, etc.)
  - Recursos específicos de uma activação de função ficam em local fixo, associado à função.
- Outras linguagens
  - Várias activações em simultâneo (ex: C, Pascal, YaLang, etc.)

    Chamadas recursivas
    Activações assíncronas (com SIGNALs, etc)
  - Recursos específicos de uma activação de função ficam em local associado à activação.

ACTIVAÇÃO/DESACTIVAÇÃO

- Política LIFO (Last in, First out)
- Corresponde a uma Stack
- Só se pode desactivar uma activação A quando for a última activação na Stack
- Limitação: quando a função devolve outra função criada localmente
  - Linguagens funcionais (CaML, Haskell, etc.)
  - Requer acesso à activação criadora, depois de desactivada
  - Neste caso, não se pode usar a Stack

REGISTOS

STACK FRAME

- Encarado como uma implementação do TAD “pilha”
- Implementação proposta
  - Vector de “células”
  - Registo especial SP (Stack Pointer)

    Endereço da última célula empilhada

    Convenção: PDL (push-down list, cresce para baixo)

STACK FRAME

- Registo de activação
  - Zona contígua da pilha na qual se encontra a informação relativa a uma activação em particular
  - Terminologia convencionada: designa-se por RA (registo de activação), ou pelos termos em inglês: Activation Record ou Stack Frame (ou apenas Frame).
- Registo auxiliar: FP (frame pointer)
  - Endereço do início da Frame
  - Dimensão do espaço é variável
  - Indica o local onde estão os valores presentes à entrada da função: o valor de retorno e os parâmetros.

FRAME POINTER

- Designa-se por FP o registo cuja função é a de representar o valor de SP à entrada da função
  - Registo de conveniência
  - SP pode variar ao longo de uma activação
    - Resultados intermédios
    - Alocação dinâmica
  - Variação de SP torna difícil o acesso ao conteúdo do RA
- Este registo nem sempre é explicitamente representado
  - Exemplo: `gcc -fomit-frame-pointer`
  - Ao omitirmos FP, perde-se capacidade de debug

REGISTOS DE HARDWARE

- Arquitecturas modernas definem um número relativamente grande de registos de uso geral
- Típico: 16 a 32
  - no x86 são só 8
  - no amd64 já são 16
- Podem considerar-se memórias de acesso rápido
- É desejável que o compilador faça uso dos registos sempre que possível.
- Número limitado: implica fazer escolhas de uso

USO PARA REGISTOS

- Cálculos e operações intermédias
- Os registos são únicos (sempre os mesmos)
  - O seu conteúdo deve ser salvaguardado sempre que se chama uma função e reposto quando esta retorna
  - A responsabilidade desta operação pode caber:
    - à função chamadora (“caller-save”)
    - à função chamada (“callee-save”)
  - Esta decisão é objecto de convenção para cada compilador, não é habitualmente imposta pelo hardware.
  - Pode haver registos preservados (SP e FP) ou voláteis (nunca são salvaguardados nem repostos)

PASSAGEM DE PARÂMETROS

- Alocados na stack?
  - Esta abordagem tem sido preterida a favor do uso de registos particulares para os parâmetros
  - Precauções a tomar:
    - Podem não existir registos suficientes
    - Os registos podem ter de ser reutilizados, se a função chamar outras funções
  - Não sendo obrigatória a alocação de espaço na stack, torna-se possível a optimização deste recurso
    - Condiciona-se a decisão de guardar um parâmetro na stack à existência de certas propriedades (por exemplo, a invocação de funções suplementares)

ENDEREÇO DE RETORNO

- Tradicionalmente, o endereço de retorno era guardado na stack
- Actualmente, prefere-se a abordagem de guardar o endereço de retorno num registo e, **só se necessário**, guardá-lo na pilha, evitando assim acessos desnecessários à memória.

CONTEÚDO DO R.A.

RESUMO DO CONTEÚDO DO RA

- Ligação ao RA anterior
- Ligação ao RA de mesmo nível lexical anterior
- Argumentos
- Endereço de retorno
- Espaço para o valor de retorno
- Valores de registos salvaguardados
- Variáveis locais