UAlg FCT
UNIVERSIDADE DO ALGARVE
FACULDADE DE CIÊNCIAS E TECNOLOGIA

Compiladores

2020-2021

Projeto 4 – Geração de Código

Introdução

No 4.º Projeto pretende-se que os alunos explorem e aprendam a gerar código intermédio, na forma de código 3 endereços (3E), para uma parte da linguagem ALG.

Objetivos e requisitos

Este projeto divide-se em duas partes. Na primeira parte, os alunos deverão compreender, e integrar o código fornecido (o *visitor CodeGen* e as novas classes correspondentes às tabelas de símbolos, e símbolos), com a sua gramática e o código desenvolvido nos projetos anteriores.

Uma vez finalizada esta componente, os alunos terão de implementar uma tarefa adicional, que se concentra numa determinada particularidade da linguagem *alg*. Poderão consultar na tabela 1, qual a tarefa atribuída a cada grupo.

Para além do código fornecido, foi também disponibilizado um exemplo por cada tarefa (incluindo quer o ficheiro source quer o ficheiro resultante do processo de compilação. Os ficheiros compilados foram gerados por um compilador que implementa todas as tarefas, por isso é natural que a vossa geração não gere todo o código que está no exemplo.

Tarefa T1

A tarefa T1 consiste na geração de código 3E para declarações de variáveis com inicialização. No caso mais simples, como por exemplo int x = 4, a geração de código 3E corresponde simplesmente a usar uma instrução de cópia do endereço resultante da expressão direita para a variável do lado esquerdo.

No caso mais complexo, que corresponde a uma inicialização de um array (ex <int> a = [10]), 2 coisas podem acontecer. Se o valor do tamanho do array *n* for conhecido em tempo de compilação (compile-time), é reservado uma zona de memória correspondente ao tamanho necessário para guardar n objetos do tipo, sendo esta zona de memória reservada na memória estática da *stack frame*[1]. Caso o tamanho do array não seja conhecido em tempo de compilação, a reserva de memória será feito na componente dinâmica da *stack frame*. Para efectuar esta reserva, deverá ser usada a função interna _ST_ALLOC[2]. Esta função recebe um parâmetro como argumento, que é a quantidade de memória em bytes a reservar. A função retorna um ponteiro para o primeiro byte reservado.

Tarefa T2

Geração de código 3E para a expressão de extração de ponteiro ?. O operador ? aplicado a uma variável de tipo primitivo retorna um novo ponteiro para a zona de memória onde está guardada a variável. Isto pode ser facilmente implementado com uma instrução de atribuição com ponteiros em código 3E.

No caso em que se aplica o operador? a uma indexação de ponteiros, como por exemplo ?a[5], isto corresponde a gerar um ponteiro para a posição 5 do array. Esta operação pode ser vista como a

[1] A implementação do *scope* fornecida poderá ajudar a executar esta reserva.
[2] A função interna _ST_ALLOC (de Stack Alloc) reserva a quantidade recebida como argumento na área dinâmica da pilha e retorna um ponteiro para o primeiro byte da memória reservada

geração de um ponteiro para uma zona de memória que se obtém a partir de um deslocamento de 5 unidades[3] a partir do ponteiro a.

Tarefa T3

Geração de código 3E para uma expressão de indexação de ponteiro, quando esta regra é usada ou como expressão sozinha, ou como expressão de um lado direito de uma atribuição.

Neste caso, uma indexação de ponteiro (ex: a[5]) corresponde a devolver uma nova variável temporária com o valor que está guardado na posição 5 do array. Isto pode ser feito utilizando uma operação 3E de cópia indexada.

Tarefa T4

Geração de código 3E para declarações de funções (incluindo a função main). Uma declaração de função, corresponde a gerar uma etiqueta (label) com o nome da função, e na linha seguinte usar a instrução código 3E *BeginFunc n.* Esta instrução recebe como argumento o número de bytes a reservar para a zona estática de memória do stack frame. De seguida é gerado o código do corpo da função. No fim é adicionada a instrução código 3E EndFunc, que indica que a função deve terminar (e retornar o controlo para a função chamadora). No caso da função principal *alg,* esta não deve terminar com a instrução EndFunc, mas com a instrução *halt*, que indica o fim da execução do programa.

Não é necessário lidar com os vários tipos de blocos diferentes. Assuma que as suas funções terão apenas o corpo principal.

Tarefa T5

Geração de código 3E para invocações de funções. A invocação de uma função corresponde a gerar o código de todos os argumentos, e usar a instrução 3E *param* para colocar cada argumento no stack frame. De seguida é gerada a instrução "call fname, numparams", em que fname é o nome da função a invocar, e numparams é o número de parâmetros da stack frame que serão passados como argumento. Caso a função tenha um valor de retorno não nulo, deverá ser criada uma variável temporária para guardar o valor de retorno da função.

Tarefa T6

Geração de código 3E para instruções de atribuição. Uma instrução de atribuição em que o lado esquerdo corresponda a uma variável é muito simples, e pode ser implementado através de uma instrução 3E de cópia.

No entanto, uma instrução de atribuição em que o seu lado esquerdo seja uma expressão de indexação, deve ser tratada de forma diferente. Por exemplo a atribuição a[4] = 3 deve atribuir à posição 4 do array o valor 3. Isto pode ser feito através de uma instrução de cópia indexada em código 3E.

Tarefa T7

Geração de código 3E para *while*, *leave* e *restart*. Para além da geração de código while, vista nas teóricas, deverão ter em conta que é possível especificar uma instrução *finally*. Uma instrução *finally* é sempre executada após o ciclo *while* terminar.

[3] Em que cada unidade corresponde à memória necessária para guardar um elemento do array.

As instruções *leave* ou *restart* só podem ser usadas dentro de um ciclo while. O *leave* termina e sai do ciclo *while* mais próximo. A instrução *finally*, se existir, é executada depois do *leave*. A instrução *restart* reinicia o ciclo interior em que a instrução se encontrar, tal como a instrução *break* em C.

Grupo	Tarefa	Grupo	Tarefa
G01	7	G17	4
G02	5	G18	6
G03	4	G19	1
G04	3	G20	1
G05	2	G21	7
G06	7	G22	1
G07	1	G23	2
G08	6	G24	1
G09	7	G25	7
G10	4	G26	3
G11	4	G27	7
G12	4	G28	6
G13	6	G29	6
G14	5	G30	3
G15	2	G33	2
G16	1		

Tabela 1 – Tarefa a executar por grupo. As tarefas foram determinadas de forma aleatória.

Considerações adicionais

As tarefas aqui apresentadas não correspondem à totalidade de toda as características a implementar da linguagem Alg. Por exemplo, as instruções de leitura e escrita não foram aqui colocadas, pois correspondem a chamadas de funções internas de leitura e escrita, e foram consideradas pouco interessantes do ponto de vista pedagógico. Outro aspeto que foi simplificado foi o da definição de funções com múltiplos corpos (prólogo, corpo principal, e epílogo). No entanto, a lógica para trabalhar com 3 corpos não seria difícil. Bastaria criar *labels* para o início do corpo principal e do epílogo, e guardar uma variável temporária com o resultado do valor de retorno.

Condições de realização

O projeto deve ser realizado em grupo, de acordo com as inscrições em grupo do laboratório. Projetos iguais, ou muito semelhantes, originarão a reprovação na disciplina. O corpo docente da disciplina será o único juiz do que se considera ou não copiar num projeto.

O 4.º projeto vale 15% da nota final da componente prática. O projeto deverá ser entregue até às **23:59** do dia **04/06/2021**.