Universidade de Évora

Compiladores

2ª Frequência - 5 de Maio de 2015

(1,5) 1. O registo de activação contém (a ordem não interessa):

A. As variáveis locais, o program counter e os argumentos da função

B. Os argumentos da função, o return address, o stack pointer antigo e as variáveis locais

C. Um ou mais arrays com as variáveis locais, argumentos da função, stack pointer, frame pointer, return address e return value

D. Os argumentos da função, o frame pointer antigo, o return address e as variáveis locais

(1,5) 2. A geração de código deve ser feita:

A. Antes da análise semântica, quando a APT ainda mostra uma representação "fiel" do programa inicial

B. Depois da análise semântica, porque é aqui que a APT foi transformada numa Symbol Table

C. Depois da análise semântica, pois só nesta fase sabemos que o programa está correcto lexical, sintactica e semanticamente

D. Depois da APT estar ligada correctamente à Symbol Table

3. Considere o seguinte programa em *Ya!*:

```
1  f (n: int) : int {
2    r, i : int = 1;
3
4    while i <= n do {
5      r = r * i;
6      i = i + 1;
7    };
8
9    return r;
10 };
11
12 main () : void {
13   print(f(3));
14 };
15
```

(2) (a) Mostre uma representação da Symbol Table, no final da análise semântica da função `f()`.

(3) (b) Mostre uma representação da stack, durante a execução do programa, imediatamente antes do return da função `f()`. Considere que a stack começa no endereço 1000, cada célula ocupando 32 bits. Coloque anotações, para especificar o que cada célula da stack representa.

(3) (c) Proponha uma forma eficaz de guardar strings no Registo de Activação, focando-se na independência de plataforma. Ilustre com um exemplo prático.

(4) (d) Usando a abordagem de "máquina de pilha" sugerida nas aulas, mostre uma possível representação da função `f()`, em assembly MIPS.

(2) (e) Explique o motivo da inclusão do *Return Address* no Registo de Activação. Ilustre com um exemplo dos problemas que podem ocorrer se o *Return Address* não for guardado.

(3) (f) Suponha que pretende implementar estruturas (`structs`) na linguagem *Ya!*. Para este efeito, considere o formato "normal" das `structs` da linguagem C. Discuta as implicações desta nova componente da linguagem na análise semântica.