Arquitectura de Computadores I - Módulo 5

Funções e procedimentos : IA32

1. Introdução

Pretende-se com esta aula prática que os alunos entendam o código gerado por um compilador de C para um programa com variáveis estruturadas e funções, utilizando diferentes níveis de optimização. Para atingir este objectivo o aluno deve desenvolver um pequeno programa em C, compilá-lo e interpretar o código gerado.

2. Linguagem de alto nível

Escreva em C, usando o editor de texto que considerar mais adequado, o seguinte programa:

prog.c

```
typedef struct {
       char nome[20];
       int idade;
       char genero;
} elemento;

int conta_gen (elemento lista[], int n)
{
       int ret=0;

       for ( ; n>0 ; n--)
              if (lista[n-1].genero == 'M')
                     ret++;
       return (ret);
}
```

3. Compilação sem optimização

Compile o programa `prog.c` usando o comando

`gcc –S prog.c`

Analisando o código *assembly* responda às seguintes questões:

Questão 1 – Identifique o deslocamento de cada uma das variáveis e parâmetros relativamente a `%ebp`.

Questão 2 – Qual o factor de escala associado ao vector `lista`? Porquê?

Questão 3 – Descreva em detalhe a forma como é calculado o endereço de `lista[n-1].genero`.

Questão 4 – Descreva em detalhe a forma como é decrementada a variável `n`. Consegue pensar numa forma de o fazer apenas com uma instrução?

Questão 5 – Qual o mecanismo utilizado para implementar `return (ret)`?

4. Compilação com optimização

Compile o programa `prog.c` usando o comando

`gcc –O1 -S prog.c`

Analisando o código *assembly* responda às seguintes questões:

Questão 6 – Identifique o deslocamento de cada uma das variáveis e parâmetros relativamente a `%ebp`. Qual a localização da variável local `ret`?

Questão 7 – O código apenas acede aos parâmetros uma vez no início da função. Porquê?

Questão 8 – Descreva em detalhe a forma como é calculado o endereço de `lista[n-1].genero`.

Questão 9 – Qual o mecanismo usado para realizar o teste do ciclo?

Questão 10 – Porque é que `%ebx` é guardado na *stack* e restaurado no fim da função?