Arquitectura de Computadores I - Módulo 6

Optimização do código : IA32

1. Introdução

Pretende-se com esta aula prática que os alunos entendam alguns mecanismos de optimização do código gerado por um compilador de C. 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 {
       int idade;
       char nome[10];
} sociot;

sociot socios[1000];

void Maius (char * );

main ()
{
       int i;

       for (i=0; i<1000 ; i++)
         Maius (socios[i].nome);
}
```

```
void Maius (char *s)
{
       int j;

       for ( j =0; j< comp (s) ; j++)
              if (s[j]>='a' && s[j]<='z')
                     s[j]= s[j] + 'A' - 'a';
}

int comp (char *s)
{
  int z;

  for (z=0 ; s[z]!= '\0' ; z++);
  return (z);
}
```

3. Compilação sem optimização

Compile o programa `prog.c` usando o comando

`gcc -S prog.c`

Analisando o código *assembly* das funções `main()` e `Maius()` responda às seguintes questões:

Questão 1 – Identifique os blocos de instruções *assembly* correspondentes a cada instrução C.

Questão 2 – Onde se encontra armazenada a variável `i` da função `main()`? E a variável `j` da função `Maius()`? E o parâmetro `s`?

4. Compilação com optimização

Compile o programa `prog.c` usando o comando

`gcc –O2 -S prog.c`

Analisando o código *assembly* das funções `main()` e `Maius()` responda às seguintes questões:

Questão 3 – Como é testado em `main()` o fim do ciclo? Que aconteceu à variável `i`?

Questão 4 – Onde é mantido o valor do parâmetro `s` durante a execução de `Maius()`? E onde é mantida a variável `j`? Qual é a optimização que o compilador pretende com esta técnica?

Questão 5 – O teste da instrução condicional if na função Maius() é optimizado. Qual o objectivo do compilador com esta optimização?

5. Optimização do algoritmo

O desempenho de um programa depende, em grande parte, do algoritmo seleccionado pelo programador para implementar a funcionalidade pretendida. Uma análise rápida da função `Maius()` permite verificar que a função `comp()` é invocada em cada iteração do ciclo para calcular o comprimento de `s`, sem que este comprimento se altere. Basta, portanto, invocar `comp()` uma vez. Esta mudança simples diminui dramaticamente o tempo de execução do programa. A melhoria obtida aumenta com o tamanho de `s`.

Este tipo de optimização, designado por *code motion*, dificilmente será feito pelo compilador. Na verdade, este teria que analisar o código cuidadosamente, para se certificar que as alterações feitas a `s` não modificam nunca o resultado de `comp()`. No entanto, muitos compiladores aplicam automaticamente *code motion*, quando o resultado de uma expressão depende de variáveis que não são alteradas durante o período em que a expressão é utilizada.

Reescreva o programa em C utilizando o código apresentado na tabela abaixo.

progopt.c

```
void Maius (char *s)
{
       int j, l;

       l = comp (s);
       for ( j =0; j< l ; j++)
              if (s[j]>='a' && s[j]<='z')
                     s[j]= s[j] + 'A' - 'a';
}
```

6. Loop unrolling

Compile o programa `progopt.c` usando o comando

`gcc –O2 -S –funroll-all-loops progopt.c`

Analisando o código *assembly* da função `main()` e comparando com o código da secção 4 responda às seguintes questões:

Questão 6 – Qual a grande diferença relativamente ao código da secção 4?

Questão 7 – Qual o objectivo do compilador? Qual a desvantagem óbvia?