Avaliação de desempenho na execução de aplicações

Trabalho para Casa: **TPC9**

Alberto José Proença

Prazos

Entrega **impreterível**: semana de 09-Dez-02, na hora e local da sessão Teórico-Prática.

Não serão aceites trabalhos entregues depois deste prazo.

Metodologia

Idêntica aos trabalhos anteriores.

Introdução

Os exercícios que se apresenta segue directamente o material apresentado nas aulas teóricas das semanas 10 e 11 (ver sumários na página da disciplina na Web), requerendo os conceitos básicos adquiridos em aulas anteriores e um ficheiro com código C (ver *link* nos sumários).

Contexto

Transformação sistemática de um programa noutro mais eficiente (cont.)

O último exercício proposto no trabalho anterior tinha como objectivo calcular, de forma simplificada, o CPE da função `combine1`, compilada sem optimização e para um vector de 1000 elementos, para 4 casos de "`combine`": (i) soma de inteiros, (ii) produto de inteiros, (iii) soma de reais e (iv) produto de reais. O programa que permite obter o valor desse CPE para o caso (i) está apresentado na resolução do TPC anterior; a adaptação para os restantes casos é imediata.

Com este TPC pretende-se complementar esse trabalho de medições e de análise, com as actividades apresentadas a seguir.

Exercícios

1. As seguintes funções em C (disponibilizado em formato electrónico conjuntamente com este enunciado) são variantes de optimização da função original `combine1`.

a) Usando como "*template*" a resolução do TPC anterior, escreva programas em C para cada uma das funções apresentadas. Compile com optimização `-O2` esses programas, e execute 5 vezes cada um deles; considere os melhores valores de CPE obtidos para cada versão optimizada, e preencha a coluna apropriada da tabela fornecida.

b) Modifique o programa fornecido para `combine1x_prod_int`, `combine1x_sum_float` e `combine1x_prod_float`. Complete a tabela referida na alínea anterior.

c) Identifique os tipos de optimização efectuados (se dependentes ou não da máquina) e caracterize os métodos usados.

d) Considere apenas as optimizações independentes da máquina. Identifique ao nível do *assembly* as optimizações introduzidas.

```
void combine11(vec_ptr v, data_t *dest)
{
   int i;
   int length = vec_length(v);

   *dest = IDENT;
   for (i = 0; i < length; i++) {
         data_t val;
         get_vec_element(v, i, &val);
         *dest = *dest OPER val;
   }
}

void combine12(vec_ptr v, data_t *dest)
{
   int i;
   int length = vec_length(v);
   data_t *data = get_vec_start(v);

   *dest = IDENT;
   for (i = 0; i < length; i++) {
         *dest = *dest OPER data[i];
   }
}


void combine13(vec_ptr v, data_t *dest)
{
   int i;
   int length = vec_length(v);
   data_t *data = get_vec_start(v);
   data_t x = IDENT;

   *dest = IDENT;
   for (i = 0; i < length; i++) {
         x = x OPER data[i];
   }
   *dest = x;
}


void combine14(vec_ptr v, data_t *dest)
{
   int length = vec_length(v);
   data_t *data = get_vec_start(v);
   data_t *dend = data+length;
   data_t x = IDENT;

   for (; data < dend; data++) {
         x = x OPER *data;
   }
   *dest = x;
}


void combine15(vec_ptr v, data_t *dest)
{
   int length = vec_length(v);
   int limit = length-1;
   data_t *data = get_vec_start(v);
   data_t x = IDENT;
   int i;

   /* Combine 2 elements at a time */
   for (i = 0; i < limit; i+=2) {
         x = x OPER data[i] OPER data[i+1];
   }

      /* Finish any remaining elements */
   for (; i < length; i++) {
         x = x OPER data[i];
   }
   *dest = x;
}


void combine16(vec_ptr v, data_t *dest)
{
   int length = vec_length(v);
   data_t *data = get_vec_start(v);
   int over = length%2;
   data_t *dend = data+length-over;
   data_t x = IDENT;

   while (data < dend) {
         x = x OPER data[0];
         x = x OPER data[1];
         data += 2;
   }
   dend += over;
   while (data < dend) {
         x = x OPER *data;
         data ++;
   }
   *dest = x;
}


void combine17(vec_ptr v, data_t *dest)
{
   int length = vec_length(v);
   int limit = length-1;
   data_t *data = get_vec_start(v);
   data_t x0 = IDENT;
   data_t x1 = IDENT;
   int i;

   /* Combine 2 elements at a time */
   for (i = 0; i < limit; i+=2) {
         x0 = x0 OPER data[i];
         x1 = x1 OPER data[i+1];
   }

   /* Finish any remaining elements */
   for (; i < length; i++) {
         x0 = x0 OPER data[i];
   }
   *dest = x0 OPER x1;
}
```

Nº	**Nome:**	**Turma:** Seg1 - Seg2 - Ter

Resolução dos exercícios

1. Tempos de execução de `combine` e variantes

a) Cálculo de CPE's (versão simplificada)

Função	**Método**	**Inteiro** *soma*	**Inteiro** *produto*	**Real** *soma*	**Real** *produto*
`combine1`	ADT, não optimizado				
`combine1`	ADT, `-O2`				
`combine11`					
`combine12`					
`combine13`					
`combine14`					
`combine15`					
`combine16`					
`combine17`					

b) Identificação das optimizações ao nível do *assembly*