Ficha de Avaliação 2

17-Dez-02

Alberto José Proença

A ficha de avaliação compreende 2 componentes, ambas para serem resolvidas individualmente e respondidas nas folhas fornecidas. A primeira componente, a realizar durante a 1ª hora, corresponde à resolução em papel, sem recorrer a qualquer elemento de consulta, de problemas relacionados com a análise e optimização de funções. A segunda componente, a realizar durante a 2ª hora, irá ser preenchida com a validação e discussão dos resultados obtidos na 1ª parte, recorrendo a um dos postos de trabalho do laboratório.

1ª Parte

A. Análise de Funções

1. Considere as operações de manuseamento de um vector de ADT (analisado nas aulas) e apresentadas em baixo. Imediatamente antes da função `set_vec_element` ser invocada - em "**`Dados aqui`**" - os registos `%esp` e `%ebp` contêm, respectivamente, os valores `0xBFFFF40` e `0xBFFFF60`, enquanto o 1º argumento passado para a função tem o valor `0x804C000`.

```
typedef int data_t;

typedef struct {
    int len;
    data_t *data;
} vec_rec, *vec_ptr;

/* Código para inicializar o array */
volatile data_t sink;
vec_ptr vector;
void setup(int len)
{
    int i;
    vector = new_vec(len);
    /* Inicializa array  */
    for (i = 0; i < len; i++)
Dados aqui
      set_vec_element(vector, i, i+1);
    sink = (data_t) 0;
}
/*Atribuir um valor a um elemento e devolver 0 (index fora de limit) ou 1 (sucesso)*/
int set_vec_element(vec_ptr v, int index, data_t val)
{
Pergunta aqui
    if (index < 0 || index >= v->len)
     return 0;
    v->data[index] = val;
     return 1;
}

void main()
{  int dimensao = 1024;
   ...
   setup (dimensao);   //
   ...
   return;
}
```

a) Mostre o conteúdo das 8 células de memória cujo endereço começa em `0x804C000`, após a inicialização do vector.

Resposta:_____

b) Mostre o quadro de activação (*stack frame*) da função `set_vec_element` em "**`Pergunta aqui`**", com indicação clara dos endereços de todos os valores nele presentes e relevantes.

Nº	**Nome:**	**Turma:** Ter

Resposta:

B. Optimização de Desempenho

1. A função `mult_cte_vec1` multiplica um vector de ADT por uma constante. Usando funções já conhecidas das aulas, um possível algoritmo e respectiva codificação em C é apresentado de seguida:

```
void mult_cte_vec1(vec_ptr v, data_t *dest)
       {
          int i;

          for (i = 0; i < vec_length(v); i++) {
             v->data[i] *= *dest ;
          }
       }
```

   a) Pretende-se optimizar (em termos de desempenho) este código, usando apenas técnicas independentes da máquina utilizada. Escreva, em linguagem C, uma versão optimizada com esse objectivo (`mult_cte_vec2`).

   Resposta:

   b) Identifique os tipos de optimização efectuados e caracterize os métodos usados.

   Resposta:

Nº	**Nome:**	**Turma:** Ter

2ª Parte

Análise de Funções e Optimização de Desempenho

1. Use o código disponibilizado nas aulas TP e construa um programa que meça o valor de CPE (simplificado, o melhor de 5 medidas) para as funções `mult_cte_vec1` e `mult_cte_vec2`. Modifique o programa de modo a que ele inicialize o vector com os valores referidos no exercício **1.** da **1ª Parte**.

Compile esse programa com o nível de optimização 2, de forma a poder executá-lo através do depurador GDB.

a) Confime a estrutura do quadro de activação (*stack frame*) da função `set_vec_element` que obteve na resposta à pergunta **1.b)** da **1ª Parte**, usando comandos de visualização de dados e de código do GDB.

Resposta:

b) Para a função **`set_vec_element`** apresente o código simbólico (*assembly*) correspondente à instrução de atribuição no corpo da função: **`v->data[index] = val`.**

Resposta:

c) Execute o programa e apresente os melhores valores de CPE para ambas as funções.

Resposta:

CPE de `mult_cte_vec1:`_____ CPE de `mult_cte_vec2:`_____

d) Com a ajuda do código em *assembly*, justifique as melhorias obtidas no desempenho.

Resposta:

Nº	**Nome:**	**Turma:** Ter