

Assembly do IA-32 em ambiente Linux

TPC8 e Guião laboratorial

Objetivo e notas

Os exercícios propostos no TPC8 introduzem o **suporte a tipos de dados estruturados em C**, no IA-32. Estes exercícios devem ser realizados no servidor remoto (máquina sc.di.uminho.pt).

A resolução deverá ser entregue **impreterivelmente** no início da sessão PL, com a presença do estudante durante a sessão PL para que o TPC seja contabilizado na avaliação por participação. Não serão aceites trabalhos entregues fora da PL.

Vetores de tipos simples

1. a) **(TPC)** Crie um ficheiro `vectorInt.c` com o seguinte código e execute a sua compilação para *assembly*, usando o comando `gcc -O2 -S vectorInt.c`.

```
#include <stdio.h>

#define N 100

void init(int *vec) {
    int i;
    for(i=0; i<N; i++)
        vec[i]=i;
}

int soma(int *vec) {
    int i, s = 0;
    for (i=0 ; i<N ; i++)
        s += vec[i];
    return(s);
}

int main() {
    int v[N];

    init(v);
    int s = soma(v);
    printf("soma=%d\n", s);
}
```

- b) **(TPC)** Identifique as instruções responsáveis pelo ciclo `for` na função `soma`. Qual seria a diferença se a constante `N` fosse um argumento da função?
- c) **(TPC)** Quantos *bytes* ocupa o vetor? Em que zona de memória está alocado o vetor e qual é a instrução no *assembly* gerado que reserva espaço para o vetor? Indique a instrução *assembly* que liberta esse espaço?
- d) Aumente sucessivamente o `N` 100 vezes (i. é., acrescente dois zeros de cada vez) e execute o programa até a execução gerar um "Segmentation fault". Utilize o `gdb` para identificar o ponto do programa onde ocorreu o erro e identificar a origem do problema.
- e) **(TPC)** Identifique as instruções responsáveis pelo cálculo do endereço de `vec[i]` na função `soma`. Qual seria a diferença se os elementos do vetor fossem do tipo `double`?

Vetores de estruturas

2. a) **(TPC)** Crie um ficheiro `vectorStr.c` com o seguinte código e execute a sua compilação para *assembly*, usando o comando `gcc -O2 -S vectorStr.c`.

```
#include <stdio.h>

#define N 100

struct S {
    char s[4];
    int a;
};

void init(struct S *vec) {
    int i;
    for(i=0; i<N; i++) {
        vec[i].s[0] = '\\0';
        vec[i].a = i;
    }
}

int soma(struct S *vec) {
    int i, soma = 0;
    for (i=0 ; i<N ; i++)
        soma += vec[i].a;
    return(soma);
}

int main() {
    struct S v[N];
    init(v);
    int s = soma(v);
    printf("soma=%d\\n",s);
}
```

- b) **(TPC)** Identifique e explique as instruções responsáveis pelo cálculo do endereço de `vec[i].a`. Compare com a resposta à alínea e) da questão 1.
- c) Qual o espaço ocupado por cada elemento do vetor. Qual será o espaço ocupado se o tamanho do campo `s` da estrutura aumentar para 5 caracteres.
- d) Verifique qual o espaço efetivamente alocado para cada elemento do vetor quando o campo `s` da estrutura tem 5 caracteres. Para isso, modifique no código C para imprimir o tamanho da estrutura (i.é., `printf("Size of struct %d\\n",sizeof(struct S));`).
- e) Identifique e explique as instruções responsáveis pelo cálculo do endereço de `vec[i].a` no caso anterior e compare com a resposta na alínea b).

Nº

Nome:

Turma:

Resolução dos exercícios (deve ser redigido manualmente)**1. Vetores de tipos simples**

b) Identifique aqui as instruções responsáveis pela implementação do ciclo `for` na função `soma` e qual seria a diferença se a constante `N` fosse um argumento da função?

c) Indique quantos *bytes* ocupa o vetor; em que zona de memória está alocado e transcreva as instruções no *assembly* gerado que reservam e libertam o espaço para o vetor.

e) Identifique aqui as instruções responsáveis pelo cálculo do endereço do `vec[i]` na função `soma` e indique qual seria a diferença se os elementos do vetor fossem do tipo `double`?

2. Vetores de estruturas

Indique aqui e explique as instruções responsáveis pelo cálculo do endereço de `vec[i].a`.