Arquitectura de Computadores 2009/10

Aula prática 7 – Vectores e Ligação C/assembly

A. Exemplo de ligação de programa em C com subrotina em assembler

Considere os seguintes ficheiros:

prog.c

```
#include <stdio.h>

extern int quadrado (int n );

int main(){
     int n;

     printf("Introduza n:" );
     scanf( "%d", &n );

     printf( "quadrado(%d) = %d\n", n, quadrado(n));

     return 0;
}
```

quadrado.asm

```
global quadrado

section .text

quadrado:
     push ebp
     mov ebp,esp

     mov eax, [ebp+8]
     mul eax

     pop ebp
     ret
```

1. Edite e grave os ficheiros
2. "Assemble" o ficheiro quadrado.asm:

```
nasm -f elf32 -g quadrado.asm
```

3. Compile o programa em C, ligando-o com o ficheiro anteriormente "assemblado":

```
gcc –Wall –g –o prog prog.c quadrado.o
```

4. Execute o programa:

```
./prog
```

5. Observe a execução do programa passo a passo com o debugger:

```
ddd prog &
```

B. Exercícios

Em todas as subrotinas (em assembly) destes exercícios deve seguir a convenção de passagem de parâmetros pela pilha utilizada pela linguagem C (consulte o documento *ref-c-asm-aulas.pdf*, que se encontra no CLIP na secção de *textos de apoio*).

1. Programe uma subrotina que calcula o comprimento de uma string, com os parâmetros a serem passados pelo stack.

    a. Ligue essa subrotina a um programa em C, que lê previamente a string do teclado e que no final imprime no ecrã o resultado da chamada da sua subrotina (my`strlen`) e da função `strlen` da biblioteca do C.

    b. Faça um programa completo em assembler que efectua a chamada à subrotina (my`strlen`) desenvolvida na alínea anterior. O seu programa deve chamar a subrotina, utilizando as convenções de passagem de parâmetros do C. Teste o seu programa utilizando o debugger ddd.

        O seu programa deve:

        - Colocar os parâmetros na pilha
        - Chamar a subrotina
        - Limpar a pilha (retirar o parâmetros)

2. Recorde a Cifra de César, programada em C no exercício 6 da aula prática 2. Desenvolva uma subrotina em assembly que realiza a codificação da Cifra de César.

    a. Implemente a subrotina de forma a poder ser chamada como uma função a partir de um programa em C.

    b. Teste a subrotina através da sua chamada a partir de

um programa em C. Experimente a codificar uma string usando a sua função em assembly e depois descodificar a string resultante usando a sua função em C. Deverá obter a string original.

c. Desenvolva agora o programa completo em assembler e teste-o utilizando o ddd.

3. Para cada um dos exercícios apresentados em baixo, realize os seguintes passos.

   a. Implemente a subrotina de forma a poder ser chamada como uma função a partir de um programa em C.

   b. Teste a subrotina através da sua chamada a partir de um programa em C.

   c. Desenvolva agora o programa completo em assembler e teste-o utilizando o ddd.

i. Programe uma subrotina em assembler para somar os elementos de um vector (`soma_vector`), que tem como parâmetros o endereço de um vector de inteiros e o número de elementos do vector.

```
int soma_vector (int *v, int size);
```

ii. Programe a função **ordenado**, que retorna verdadeiro ou falso conforme o vector **v** estiver ordenado por ordem crescente ou não.

```
int ordenado (int *v, int size);
```

Para testar o seu programa pode utilizar a seguinte declaração, que declara um vector de 5 posições, preenchido com valores de 1 a 5.

```
int v[] = {1,2,3,4,5};
```

iii. Programe a função **mytrunc**, que "corta" todos os caracteres da string **c** a partir da posição **n**. Tenha em conta que a string pode ter menos de **n** caracteres.

```
void mytrunc (char *c, int n);
```

Exemplo:

```
char x[] = “abcd”;

trunc(x,2);

// x fica com “ab”
```

iv. Programe a função **maiuscula**, que transforma todas as letras minúsculas da string **s** em maiúsculas.

```
void maiuscula (char *s);
```

v.