Arquitectura de Computadores 2010/11

Aula prática 8 – Ligação C/assembly

A. Exemplo de ligação em que o programa principal está em assembler e chama uma rotina da biblioteca do C

Considere o seguinte ficheiro:

prf.asm

```
global main

extern printf

section .data
fmt: db 'num = %d', 10, 0
x    dd  20

section .text

main:
     push dword [x]
     push fmt
     call printf     ; printf("num = %d\n", x)
     add  esp, 8

     mov  eax,0
     ret
```

1. Edite e grave o ficheiro
2. "Assemble" o ficheiro print.asm:

```
nasm -f elf32 -g –o prf.o prf.asm
```

3. Ligue o ficheiro objecto print.o em C com a biblioteca do C:

```
gcc  -g -o prf prf.o
```

4. Execute o programa:

```
./prf
```

5. Modifique o programa anterior de forma a imprimir o seu nome, nº de aluno e clube de futebol preferido

B. Exemplo de ligação em que o programa principal está em C e chama uma 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
```

5. Edite e grave os ficheiros
6. "Assemble" o ficheiro quadrado.asm:

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

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

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

8. Execute o programa:

```
./prog
```

9. 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 (`mystrlen`) e da função `strlen` da biblioteca do C.

    b. Faça um programa completo em assembler que efectua a chamada à subrotina (`mystrlen`) 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. 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. Note que alguns destes programas foram propostos na aula passada.

    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 **factorial**, que corresponde ao seguinte código em C:

```
fact = 1;
for (i=1; i <= n; i++)
      fact = fact * i;
```

iii. Programe uma nova versão da função **factorial**, agora numa versão recursiva.

```
int factorial (int n){
      if (n == 1) return 1;
      else return n * factorial(n – 1);
}
```

iv. Programe a função **Combinacoes**, que retorna o número de subconjuntos de n elementos que se podem obter a partir de m elementos do conjunto::

$$C(m,n) = \frac{m!}{n!*(m-n)!}$$

Use a função factorial desenvolvida em ii ou iii.