Universidade do Minho

Módulo 6

Estrutura de Activação

1. Introdução

Pretende-se com esta sessão teórico-prática que os alunos acompanhem a execução de uma função recursiva instrução a instrução, examinando a estrutura de activação para cada instância da função. Para este efeito a depuração do programa, ao nível maquina, será feita usando o GDB. O texto anexo "**Introdução ao GDB *debugger***", contém informação pertinente ao funcionamento da sessão laboratorial, e é uma sinopse ultra-compacta do manual; a versão integral está disponível no site da GNU.

NOTA: A QUESTÃO ASSINALADA COM TPC DEVE SER RESOLVIDA **ANTES DA SESSÃO TP** E ENTREGUE AO DOCENTE NO INÍCIO DA MESMA. A FOLHA ANEXA A ESTE MÓDULO, ASSINALADA COMO TPC, DESTINA-SE A ESSE FIM.

2. Linguagem de alto nível

Escreva em C, usando o editor de texto que considerar mais adequado, o seguinte programa:

prog.c

```
#include <stdio.h>
typedef struct {
  char nome[10];
  int idade;} elemento;

elemento BD[6]={{"",10},{"",35},{"",22},{"",51},{"",73},{"",5}};

main () {
  int soma;

  soma = s_idades (6, BD);
  printf ("A soma das idades e' %d\n", soma);
}

int s_idades (int n, elemento *a) {
  int ret, i;

  i = a[n-1].idade;
  if (n==1)    ret = i;
  else         ret = i + s_idades (n-1, a);
  return (ret);   }
```

Compile o programa `prog.c` usando o comando

`gcc –g –S prog.c`

e visualize o código *assembly* gerado. Gere o executável usando o comando

`gcc –g –o prog prog.c`

3. Depuração

Para depurar o programa vamos usar o GDB (consulte o anexo para lista de comandos):

`gdb prog`

Visualize o código *assembly* da função `s_idades()` usando o comando

```
gdb: disas s_idades
```

TPC : Questão 1 – Desenhe o esquema da estrutura de activação da função `s_idades()`, representando e identificando os vários campos e respectivos tamanhos, assim como os deslocamentos relativamente ao apontador para a estrutura.

Questão 2 – Desenhe a estrutura do vector, incluindo os elementos e campos dentro de cada elemento.

Questão 3 – Identifique as várias instruções responsáveis pelo cálculo do endereço de `a[n-1].idade`, justificando cada uma. Qual o factor de escala?

Coloque um *breakpoint* na primeira instrução *assembly* correspondente à primeira instrução em C da função `s_idades()`. Execute o programa (`run`) até a execução parar neste ponto.

Questão 4 – Analise a informação que o GDB disponibiliza sobre a estrutura de activação, usando o comando `info frame`. Qual o endereço de retorno? Qual o endereço para onde aponta `%ebp`?

Questão 5 – Examine o conteúdo da zona da pilha correspondente à estrutura de activação. Uma possibilidade é pedir para examinar 8 palavras (de 32 *bits*) a partir de `%ebp` menos 16 (`x/8w $ebp-16`). Qual o valor de n? Qual o endereço base do vector?

Questão 6 – Examine o conteúdo do vector, confirmando se os valores em memória correspondem efectivamente aos valores indicados em C. O endereço base (`base_addr`) pode ser obtido através da informação apresentada nas duas alíneas anteriores. Este vector tem 24 palavras de tamanho, logo escreva `x/24w <base_addr>`.

Questão 7 – Qual o deslocamento a somar ao endereço base para aceder a a[n-1]? Verifique com o GDB, executando as instruções a passo (`stepi`), que este é o valor calculado.

Questão 8 – Qual o endereço de `a[n-1].idade`? Verifique, com o GDB, que este é o endereço efectivamente calculado.

Questão 9 – Prossiga com o programa (`continue`) até que o GDB pare na 2ª invocação da função. Represente no papel a estrutura de activação das duas instâncias da função, escrevendo e comentando os valores dos parâmetros, antigo `%ebp` e endereços de retorno.

Anexo: Introdução ao GNU *debugger*

O GNU *debugger* `GDB` disponibiliza um conjunto de funcionalidades úteis na análise e avaliação do funcionamento de programas em linguagem máquina, durante a sua execução; permite ainda a execução controlada de um programa, com indicação explícita de quando interromper essa execução – através de *breakpoints*, ou em execução passo-a-passo - e possibilitando a análise do conteúdo de registos e de posições de memória, após cada interrupção.

A tabela/figura seguinte (de CSAPP) ilustra a utilização de alguns dos comandos mais comuns para o IA32.

Command	Effect
Starting and Stopping	
quit	Exit GDB
run	Run your program (give command line argum. here)
Breakpoints	
break sum	Set breakpoint at entry to function `sum`
*break *0x80483c3*	Set breakpoint at address `0x80483c3`
delete 1	Delete breakpoint 1
delete	Delete all breakpoints
Execution	
stepi	Execute one instruction
stepi 4	Execute four instructions
nexti	Like `stepi`, but proceed through function calls
continue	Resume execution
finish	Run until current function returns
Examining code	
disas	Disassemble current function
disas sum	Disassemble function sum
disas 0x80483b7	Disassemble function around address `0x80483b7`
print /x $eip	Print program counter in hex
Examining data	
print $eax	Print contents of `%eax` in decimal
print /x $eax	Print contents of `%eax` in hex
print 0x100	Print decimal representation of `0x100`
print /x ($ebp+8)	Print contents of `%ebp` plus 8 in hex
*print *(int *) 0xbffff890*	Print integer at address `0xbffff890`
x/2w 0xbffff890	Examine 2(4-byte) words starting at addr 0xbffff890
x/20b sum	Examine first 20 bytes of function `sum`
Useful information	
info frame	Information about current stack frame
info registers	Values of all the registers
help	Get information about GDB

Figure 3.27 (de CSAPP e Prof. Alberto Proença): **Example GDB Commands.** These examples illustrate some of the ways GDB supports debugging of machine-level programs.

TPC	
Número:	Nome:

Questão 1 – Desenhe o esquema da estrutura de activação da função `s_idades()`, representando e identificando os vários campos e respectivos tamanhos, assim como os deslocamentos relativamente ao apontador para a estrutura.