Arquitectura de Computadores 2009/10

Aula prática 5 – NASM e o ambiente de desenvolvimento

A. Programa exemplo

1. Considere o seguinte programa em nasm:

```
; exportar label de inicio do programa
global _start

section .data    ; seccao data - dados inicializados

; variaveis de 32 bits
x:   dd   2
y:   dd   2
res: dd   0

section .text    ; seccao text - codigo

_start:   nop            ;  dummy

; res = x + y
     mov  eax, [x]       ;  COLOCAR BREAKPOINT AQUI
     add  eax, [y]
     mov  [res], eax

; chamada ao sistema para terminar o programa
     mov  eax, 1         ; codigo de "exit"
     mov  ebx, 0         ; valor retornado ao sistema
     int  0x80           ; chamada ao sistema
```

2. Para a criação do executável e teste do programa, deve proceder à seguinte sequência de comandos:

   i. Assemblar o programa com o nasm

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

   ii. "Linkar" o programa

   ```
   ld -o prog prog.o
   ```

   iii. Executar o programa

   ```
   ./prog
   ```

   iv. Utilização do programa "**ddd**" – Data Display Debugger, para observar o comportamento do programa.

   ```
   ddd prog
   ```

Exemplo de sequência de passos de teste:

v. Coloque um *breakpoint* no programa na linha indicada. Esta opção surge deslocando o rato para o espaço em branco no inicio da linha e carregando no botão da direita.

vi. Inicie a execução do programa, seleccionando a opção de **RUN**, na janela de comandos.

vii. Observe os valores das variáveis. Pode efectuar esta observação de duas formas, colocando o rato sobre a variável que pretende analisar, ou fazendo **Display**. Para fazer **Display** deve carregar no botão da direita do rato, após seleccionar a variável.

viii. Para alterar os valores das variáveis de que fez **Display**, seleccione com o rato a caixa de display e carregue no botão direito. Seleccione de seguida a opção de **Set Value**.

ix. Observe os registos do CPU, seleccionado a opção **Register** no menu de **Status**.

x. Avance no programa seleccionado a opção **Step**, na janela de comandos.

B. Exercício – Apontadores

1. Declare agora duas variáveis, ptrx e ptry. Estas devem, no início do programa ser iniciadas com os endereços de x e y, respectivamente. Altere o resto do programa para efectuar a mesma operação mas agora usando estes apontadores. Será equivalente a implementar em *assembly* o seguinte código C:

```
ptrx = &x;
ptry = &y;
res = *ptrx + *ptry;
```

Use por exemplo, o endereçamento indirecto por registo. Verifique no *debugger* e compare a execução com a situação inicial.

C. Exercícios – Instruções aritméticas e lógicas

Considere x, y e res números sem sinal de 32 bits. Codifique em assembler as instruções C apresentadas de seguida.

Teste os seus programas nasm, observando o seu comportamento no *debugger* ddd.

1. `x = 2*(x+y) - 1;  //NÃO use a multiplicação`

2. `res = x & ~1;`

3. `x += x * y + 3;`

4.
```
if ( x<100)
    res -= x;
else
    res += y;
```

5.
```
if ( (x & 1) != 0 )  y++;
```

6.
```
if ( x>100 && x<200 )
        res = 'a';
    else
        res = 'b';
```

7.
```
// não utilize mul nem div
    if ( x < 10 )
        res += y/2;
    else if ( x < 20 )
        res += 2*y;
    else
        res += 3*y;
```

8.
```
// não utilize constantes
    // assuma que x tem sinal
    if ( x > 0 )
        x = 0;
    else
        x = -1;
```

9. Acrescente ao programa exemplo (parte A) uma subrotina que verifica se o conteúdo da variável x é par ou ímpar, para tal deixa em eax 1 se x for par e 0 se for ímpar. Efectue a sua chamada e afecte a variável `res` com o resultado da chamada.