Guião: **G-VI**

Exercícios adaptados do livro CSPP
Randal E. Bryant e David R. O'Hallaron

Apresentação

Este guião tem vista abordar os temas relacionados com o controlo de fluxo de instruções e a representação de variáveis estruturadas usando o jogo de instruções de IA32.

Exercício 1 *(Ciclo Do-While)***:** Considere o **trecho** de código, abaixo, resultante da compilação da função dw_loop com o ***gcc***

```
1  int dw_loop(int x, int y, int n)
2  {
3     do {
4      x += n;
5     y *= n;
6     n--;
7     } while ((n > 0) & (y < n)); /* O operador usado é o e lógico '&'
8           e não a conjunção '&&'*/
9     return x;
10 }
```

```
1     movl     8(%ebp),%esi
2     movl     12(%ebp),%ebx
3     movl     16(%ebp),%ecx
4     .p2align 4,,7                  /* alinha o código na memória para optimizar a cache */
5  .L6:
6     imull    %ecx,%ebx
7     addl     %ecx,%esi
8     decl     %ecx
9     testl    %ecx,%ecx
10    setg     %al
11    cmpl     %ecx,%ebx
12    setl     %dl
13    andl     %edx,%eax
14    testb    $1,%al
15    jne      .L6
```

a) Adicione em cada uma das linhas os comentários necessários à sua compreensão.
b) Construa uma tabela de utilização de registos.
c) Identifique a expressão de teste e o corpo da função no código fonte C, estabelecendo a correspondência com as linhas de código produzido por compilação.

Exercício 2 *(Ciclo While)***:** Para a função e código que seguem pretende-se uma resposta idêntica à requerida para o exercício anterior. Considere ainda a questão complementar, abaixo:

d) Que optimizações foram feitas pelo compilador?

```
1 int loop_while(int a, int b)
2 {
3     int i = 0;
4     int result = a;
5     while (i < 256) {
6         result += a;
7         a -= b;
8         i += b;
9     }
10    return result; }
```

```
1     movl     8(%ebp),%eax
2     movl     12(%ebp),%ebx
3     xorl     %ecx,%ecx
4     movl     %eax,%edx
5     .p2align 4,,7
6  .L5:
7     addl     %eax,%edx
8     subl     %ebx,%eax
9     addl     %ebx,%ecx
10    cmpl     $255,%ecx
11    jle      .L5
12    movl     %edx, %eax                ;       prepara retorno
```

Exercício 3 *(Apontadores)***:** Considere que o apontador para o início do *vector* S (tipo *integer short*) e o índice *i* (tipo *integer*) estão armazenados nos registos ***%edx*** e ***%ecx***, respectivamente.
Apresente, para cada uma das expressões abaixo: ***i)*** a respectiva declaração de tipo de dados; ***ii)*** uma fórmula de cálculo do valor; ***iii)*** uma instrução em IA32 que coloca aquele resultado, no registo %eax (tipo *) ou em alternativa no registo %ax (tipo *integer short*)

Expressão	Tipo de dados	Valor	Instrução
`S+1`			
`S[3]`			
`&S[i]`			
`S[4*i+1]`			
`S+i-5`			

Exercício 4 (*Estruturas***):** O procedimento `sp_init` (com algumas expressões omitidas) trabalha com um tipo de dados que obedece à declaração de tipo `struct prob` .:

```
struct prob {
      int *p;
      struct {
                int x;
                int y;
      } s;
struct prob *next;}
```

```
void sp init(struct prob *sp)
{
      sp->s.x =          ;
      sp->p =          ;
      sp->next = _________;}
```

a) Quantos octetos são necessários para representar aquela estrutura?
b) Qual o valor dos deslocamento em relação ao inicio do vector (em número de octetos) dos campos:

```
        p:
      s.x:
      s.y:
     next:
```

c) Considerando que após compilação de `sp_init` se obteve o código que segue para o corpo da função, preencha as expressões em falta (sublinhado) no código C da função.

```
1     movl      8(%ebp),%eax
2     movl      8(%eax),%edx
3     movl      %edx,4(%eax)
4     leal      4(%eax),%edx
5     movl      %edx,(%eax)
6     movl      %eax,12(%eax)
```