Assembly do IA-32 em ambiente Linux

Exercícios de Programação 2

(Adaptados do livro de Randal E. Bryant and David R. O'Hallaron)
Alberto J. Proença e António M. Pina

Avisos

Entrega **impreterível**: segunda 26-Nov-01, no Lab. da disciplina (não serão aceites trabalhos entregues depois deste prazo). Será aceite também uma nova entrega do exercício 1.8 (semana anterior).

Para mais informações, ler metodologia definida no enunciado dos Exercícios de Programação 1.

Introdução

A lista de exercícios que se apresenta estão directamente relacionados com a codificação de ciclos em assembly do IA-32 (segue o material apresentado nas aulas teóricas e práticas da semana 8 (ver sumários na página da disciplina na Web), requerendo os conceitos básicos adquiridos em aulas anteriores.

Exercício 2.1 *(Ciclo Do-While)*:

Considere que, para a seguinte porção de código em C:

```
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)); /* Note use of bitwise '&' */
8    return x;
9 }
```

GCC gera o seguinte código *assembly* :

```
  Initially x, y, and n are at offsets 8, 12, and 16 from %ebp
1    movl    8(%ebp),%esi
2    movl    12(%ebp),%ebx
3    movl    16(%ebp),%ecx
4    .p2align 4,,7                Inserted to optimize cache performance
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) Preencha a tabela de utilização de registos (semelhante ao exemplo do programa Fibonacci).

b) Identifique a expressão de teste (*`test-expr`*) e o corpo da função (*`body-statement`*) no bloco do código C, e enumere as linhas de código no programa em *assembly* que lhe são correspondentes.

c) Acrescente comentários ao programa em *assembly* (idênticos ao feito para o programa Fibonacci).

Exercício 2.2 *(Ciclo While)*:

Considere que, para a seguinte porção de código em C:

```
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;
11 }
```

GCC gera o seguinte código *assembly* :

```
  Initially a and b are at offsets 8 and 12 from %ebp
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
```

a) Preencha a tabela de utilização de registos (semelhante ao exemplo do programa Fibonacci).

b) Identifique a expressão de teste (`test-expr`) e o corpo da função (`body-statement`) no bloco do código C, e enumere as linhas de código no programa em *assembly* que lhe são correspondentes. Que optimizações foram feitas pelo compilador?

c) Acrescente comentários ao programa em *assembly* (idênticos ao feito para o programa Fibonacci).

d) Escreva uma versão do tipo `goto` (em C) da função, com uma estrutura semelhante ao do código *assembly* (tal como foi feito para o programa Fibonacci).

Exercício 2.3 *(Ciclo For)*:

A seguinte porção de código *assembly*:

```
  Initially x, y, and n are offsets 8, 12, and 16 from %ebp
1    movl    8(%ebp),%ebx
2    movl    16(%ebp),%edx
3    xorl    %eax,%eax
4    decl    %edx
5    js      .L4
6     movl   %ebx,%ecx
7    imull   12(%ebp),%ecx
8    .p2align 4,,7                  Inserted to optimize cache performance
9  .L6:
10   addl    %ecx,%eax
11   subl    %ebx,%edx
12   jns     .L6
13 .L4:
```

foi obtido pela compilação de código C que tinha a seguinte estrutura:

```
1 int loop(int x, int y, int n)
2 {
3    int result = 0;
4    int i;
5    for (i = ____; i ____ ; i = ___ ) {
6       result += _____ ;
7    }
8    return result;
9 }
```

Pretende-se completar o programa em C de modo a obter-se um programa equivalente ao obtido com o código *assembly*. De notar que o resultado da função é devolvido no registo `%eax`.
Para resolver este problema, sugere-se que

- arranje alguma "inspiração" para tentar acertar com os registos apropriados; deverá depois confirmar se a escolha efectuada faz sentido ou não;
- identifique/caracterize:
  - os registos que são alocados às variáveis `result` e `i`
  - o valor inicial de `i`
  - a condição a testar com `i`
  - o modo como a variável `i` é actualizada
  - a expressão em C que descreva o modo como a variável `result` é incrementada no ciclo.

a) O compilador retirou a expressão que incrementa a variável `result` do interior do ciclo. Porquê?

b) Complete as partes que faltam no código C.

Nº _____ Nome _____

Exercício 2.1 *(Ciclo Do-While)***:**

a)

Utilização dos Registos		
Registo	Variável	Valor inicial
	x	
	n	
	y	

b)

```
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)); /* Note use of bitwise '&' */
8   return x;
9 }
```

c)

```
1   movl  8(%ebp),%esi
2   movl  12(%ebp),%ebx
3   movl  16(%ebp),%ecx
4   .p2align 4,,7                Inserted to optimize cache
performance
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
```

Nº _____ Nome _____

Exercício 2.2 *(Ciclo While)***:**

a)

Utilização dos Registos		
Registo	Variável	Valor inicial
	a	
	b	
	i	

b)

```
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;
11 }
```

c)

```
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
```

d)

Nº _____ Nome _____

Exercício 2.3 *(Ciclo For)***:**

a) Alocação dos registos às variáveis `result` (_____) e `i` (_____)

Valor inicial de `i` _____

Condição a testar com `i` _____

Modo de actualizar a variável `i` _____

Expressão que incrementa `result` _____

O compilador retirou a expressão que incrementa a variável `result` do interior do ciclo. Porquê?

b)

```
1 int loop(int x, int y, int n)
2 {
3   int result = 0;
4   int i;
5   for (i = ____; i ____ ; i = ___ ) {
6     result += _____ ;
7   }
8   return result;
9 }
```