Sistemas Operacionais – Prof. Rafael R. Obelheiro

Exercícios — Fundamentos de SO — Respostas

1. Sem resposta

2. O programa exibe a sequência de Fibonacci:

$$1, 1, 2, 3, 5, 8, 13, 21, \ldots,$$

onde cada termo é a soma de seus dois antecessores:

$$F_n = F_{n-1} + F_{n-2}$$
$$F_1 = 1 \qquad F_2 = 1$$

(Estritamente falando, a sequência de Fibonacci inicia com $F_0 = 0$, mas o programa começa com F_1. Fica como exercício modificá-lo para iniciar com F_0.)

3. O algoritmo implementado pelo programa é o seguinte:

```
1 read(n);
2 do {
3    print(n);
4    if (n != 0)
5       n--;
6 } while (n != 0);
```

A posição de memória 600 armazena a constante 1, para poder decrementar o acumulador, e a posição de memória 601 armazena a variável n.

Programa:

```
100: 0001    ; AC = 1
101: 2600    ; [600] = AC (1)
102: 7000    ; AC = teclado
103: 2601    ; [601] = AC (N)
104: 8001    ; video = AC
105: 6109    ; JZ 109
106: 4600    ; AC -= [600]
107: 2601    ; [601] = AC
108: 5104    ; JMP 104
109: 9999    ; HALT
```

4. O algoritmo implementado pelo programa é o seguinte:

```
1 read(n1);
2 n2 = 0;
3 do {
4    print(n2);
5    n2++;
6    n1--;
7 } while (n1 != 0);
8 print(n2);
```

A memória é usada de acordo com a seguinte tabela:

Endereço	Variável
700	1 (const)
701	n1
702	n2

Programa:

```
100: 0001    ; AC = 1
101: 2700    ; [700] = AC (1)
102: 7000    ; AC = teclado
103: 2701    ; [701] = AC (N1)
104: 0000    ; AC = 0
105: 2702    ; [702] = AC (N2)
106: 8001    ; video = AC
107: 3700    ; AC += [700]
108: 2702    ; [702] = AC
109: 1701    ; AC = [701]
110: 4700    ; AC -= [700]
111: 2701    ; [701] = AC
112: 6115    ; JZ 115
113: 1702    ; AC = [702]
114: 5106    ; JMP 106
115: 1702    ; AC = [702]
116: 8001    ; video = AC
117: 9999    ; HALT
```

5. O algoritmo implementado pelo programa é o seguinte:

```
1 read(n1);
2 read(n2);
3 n3 = 0;
4 while (n2 != 0) {
5     n3 += n1;
6     n2--;
7 }
8 print(n3);
```

A memória é usada de acordo com a seguinte tabela:

Endereço	Variável
500	1 (const)
501	n1
502	n2
503	n3

Programa:

```
100: 0001    ; AC = 1
101: 2500    ; [500] = AC (1)
102: 7000    ; AC = teclado
103: 2501    ; [501] = AC (N1)
104: 7000    ; AC = teclado
105: 2502    ; [502] = AC (N2)
106: 0000    ; AC = 0
107: 2503    ; [503] = AC (N1*N2)
108: 1502    ; AC = [502]
109: 6117    ; JZ 117
110: 1503    ; AC = [503]
111: 3501    ; AC += [501]
112: 2503    ; [503] = AC
113: 1502    ; AC = [502]
114: 4500    ; AC -= [500]
115: 2502    ; [502] = AC
116: 5109    ; JMP 109
117: 1503    ; AC = [503]
118: 8001    ; video = AC
119: 9999    ; HALT
```

6. Os dois principais objetivos de um SO são gerenciar os recursos de *hardware* e fornecer abstrações para que aplicações e usuários acessem o *hardware* de forma mais conveniente.

7. (a), (c), (d)

8. Não é possível construir um SO seguro usando um processador sem níveis de privilégio, pois, sem a separação de níveis, não há como impedir que um programa de usuário sobrescreva regiões de memória do sistema operacional, desabilitando ou contornando qualquer mecanismo de segurança aplicado pelo SO.

9. Uma instrução *trap* chaveia o modo de execução de uma CPU de modo usuário para modo núcleo (privilegiado). Essa instrução permite que um programa de usuário invoque funções do núcleo do sistema operacional (ou seja, chamadas de sistema).

10. Uma chamada de sistema permite que um processo de usuário invoque funções do núcleo do sistema operacional. Os programas de usuário empregam chamadas de sistema para invocar serviços do SO.