Sistemas Operacionais – Prof. Rafael R. Obelheiro

Exercícios — Fundamentos de SO

Processador hipotético

1. **[Stallings 1.1mod]** Considere um processador hipotético, semelhante ao usado no exemplo do slide 36, que possua os seguintes *opcodes*:

Opcode	Significado
0	AC $\leftarrow$ *num*
1	AC $\leftarrow$ *mem*
2	*mem* $\leftarrow$ AC
3	AC $\leftarrow$ AC + *mem*
4	AC $\leftarrow$ AC $-$ *mem*
5	desvie para *mem*
6	se AC=0, desvie para *mem*
7	lê número do dispositivo de entrada e coloca em AC
8	envia AC para o dispositivo de saída

   O operando de cada instrução é um número *num* (para o *opcode* 0), o endereço de memória *mem* (para os *opcodes* 1–6) ou o número do dispositivo de E/S (para os *opcodes* 7 e 8), que pode ser 0 para o teclado, 1 para o vídeo ou 2 para a interface de rede.

   Seguindo o formato ilustrado no slide 36, mostre a execução do seguinte programa:

   1: Leia um número do teclado e o coloque no acumulador (AC);
   2: Adicione o conteúdo da posição de memória 940;
   3: Imprima o conteúdo do acumulador no vídeo.

   Suponha que o número lido seja 3 e que a posição 940 contenha o valor 2.

2. Usando o processador do exercício 1, mostre a execução do programa abaixo, e explique o que ele faz. Os números à esquerda do sinal de dois pontos são endereços de memória, e os números à direita o conteúdo de cada endereço.

   Suponha que o valor inicial dos registradores da CPU sejam PC=100, AC=1234 e IR=9876. (PC é o contador de programa, AC é o acumulador e IR é o registrador de instrução.)

```
100: 0001
101: 2400
102: 8001
103: 2401
104: 8001
105: 3400
106: 2402
107: 1401
108: 2400
109: 1402
110: 2401
111: 5104
```

3. Usando o processador do exercício 1, escreva um programa que leia um número do teclado e imprima uma contagem regressiva, do número lido (inclusive) até zero. Use a instrução 9999 para indicar o fim do programa.

4. Usando o processador do exercício 1, escreva um programa que leia um número do teclado e imprima uma contagem progressiva, de zero até o número lido (inclusive).

5. Usando o processador do exercício 1, escreva um programa que leia dois números do teclado e imprima o produto entre eles.

Programação Assembly x86

6. Suponha que, em um processador x86, os registradores tenham atualmente os valores EAX=10, EBX=20, ECX=30. Determine o efeito produzido pela execução das seguintes instruções:

   (a) `mov $40, %edx`

   (b) `mov %ebx, %eax`

   (c) `mov %(ebx), %eax`

   (d) `add %ecx, %ebx`

   Considere que o conteúdo dos endereços absolutos de memória de 0 a 1999 seja a sequência de valores [1000, 2999], ou seja, *mem[addr]=addr+1000* (para $addr \in [0, 1999]$).

7. Determine o efeito produzido pela execução do seguinte trecho de código Assembly:

```
1   mov $1, %eax
2   movb $8, %cl
3   shl %cl, %eax
```

8. Determine o efeito produzido pela execução do seguinte trecho de código Assembly:

```
1   mov $10, %ecx
2   mov $1, %eax
3 inicio:
4   shl $1, %eax
5   loop inicio
```

9. Determine o efeito produzido pela execução do seguinte trecho de código Assembly (`x` é um rótulo na seção de dados):

```
1   mov x, %eax
2   dec %eax
3   mov %eax, x
```

10. Determine o efeito produzido pela execução do seguinte trecho de código Assembly (`x` e `y` são rótulos na seção de dados):

```
1   mov $250, %eax
2   add $10, %eax
3   mov %al, x
4   mov %ah, y
```

11. Determine o efeito produzido pela execução do seguinte trecho de código Assembly (`x` é um rótulo na seção de dados):

```
1   mov $532, %ax
2   add $10, %ah
3   mov %eax, x
```

12. Determine o efeito produzido pela execução do seguinte trecho de código Assembly (`x` e `y` são rótulos na seção de dados):

```
  mov $255, %ax
  inc %al
  mov %al, x
  mov %ah, y
```

13. Determine o efeito produzido pela execução do seguinte trecho de código Assembly (`x` é um rótulo na seção de dados):

```
  mov $10, %eax
  mov x, %ebx
  cmp %ebx, %eax
  jl  r1
  mov %eax, x
r1: ...
```

14. Determine o efeito produzido pela execução do seguinte trecho de código Assembly (`x` é um rótulo na seção de dados):

```
  mov $10, %eax
  mov x, %ebx
  cmp %ebx, %eax
  jg  r1
  mov %eax, x
r1: ...
```

15. Escreva trechos de código Assembly que implementem o trechos de código C abaixo. Considere que as variáveis em C são inteiros **com sinal** de 4 bytes (à exceção de `ptr`, que é um ponteiro, também de 4 bytes), e que existem no programa Assembly rótulos de mesmo nome na seção de dados para representá-las.

    (a) `y = 3*x + 1;`

    (b) `delta = b*b - 4*a*c;`

    (c) `if (x > y) max = x; else max = y;`

    (d) `if (a >= b) ptr = &a; else ptr = &b;`

    (e) `while (x > 0) { y = y*2; x--; }`

16. Um dos exemplos em Assembly disponíveis no Moodle é `parimp2.s`, que lê um número inteiro e informa se ele é par ou ímpar. Seguindo as orientações encontradas no início do código fonte, monte o programa e teste o seu funcionamento.

17. Usando `parimp2.s` como base, escreva um programa em Assembly que leia um número inteiro e informe se ele é maior que, menor que, ou igual a 51.

18. Escreva um programa em Assembly que leia dois números inteiros e informe se o primeiro é maior que, menor que, ou igual ao segundo.

19. Escreva um programa em Assembly que exiba a média entre dois números inteiros lidos da entrada padrão.

20. Escreva um programa em Assembly que mostre a soma e a diferença de dois números inteiros lidos da entrada padrão. A diferença deve ser zero (caso os dois números sejam iguais) ou positiva (o maior subtraído do menor).

21. Escreva um programa em Assembly que leia dois números inteiros x e y calcule x^y usando

    (a) multiplicação

    (b) apenas somas.

22. Uma subrotina em Assembly tem três argumentos (parâmetros de entrada), `a`, `b` e `c`, e duas variáveis locais, `i` e `j`. Tanto os parâmetros quanto as variáveis locais são inteiros de 4 bytes.

    (a) Seguindo o modelo da Figura 7 (página 20 do material), mostre como ficaria a pilha durante a execução dessa subrotina. Associe os argumentos e variáveis locais aos respectivos endereços relativos a EBP, e indique onde ficaria ESP.

    (b) Mostre como ficaria essa pilha após a execução do seguinte conjunto de instruções:

```
1  push $200
2  push $250
3  pop %esi
4  push $300
```

    Qual o conteúdo do registrador ESI?

23. Escreva uma subrotina `print3` em Assembly que imprima três números passados como parâmetro na pilha, equivalente ao código C

```
void print3(int x, int y, int z) {
   printf("x=%d, y=%d, z=%d\n", x, y, z);
}
```

24. Escreva uma subrotina em Assembly que implemente a função `strlen()`, retornando em EAX o número de caracteres de uma *string* passada como parâmetro na pilha.

25. Escreva uma subrotina em Assembly que calcule a média aritmética de quatro números passados como parâmetro na pilha. A média calculada deve ser retornada em EAX.

26. Escreva uma subrotina em Assembly que calcule a média aritmética de uma quantidade arbitrária de números passados como parâmetro na pilha. O primeiro parâmetro deve ser a quantidade de números a processar. Por exemplo, para calcular a média de 4, 9 e 5, a chamada em C equivalente seria `media(3, 4, 9, 5)`. A média calculada deve ser retornada em EAX.