UNIVERSIDADE PAULISTA - UNIP

773Z Projeto Lógico de Computadores
Prof.: Renê de Souza Pinto

Lista de Exercícios para a Prova 2

Questão 1: Com base na figura 1 comente o conceito de máquinas multiníveis.

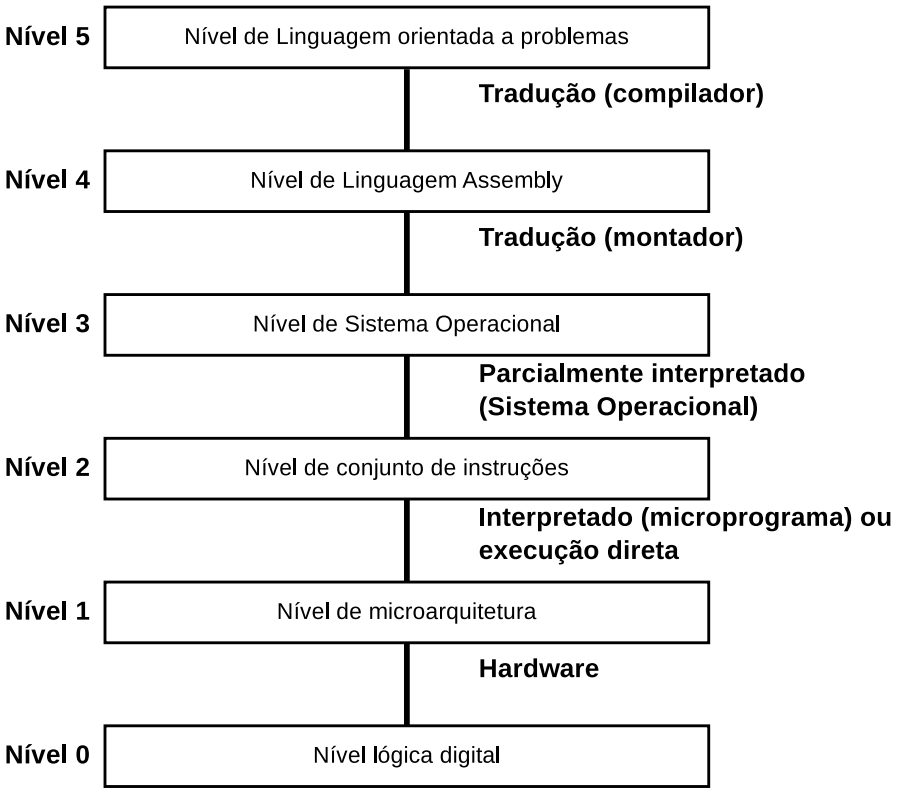

Figura 1: Máquina Multiníveis

Questão 2: Considere um software que foi escrito em linguagem C, compilado e o binário resultante executado em um Sistema Operacional atuando em uma máquina real, com processador de arquitetura Intel x86 (que possui microarquitetura). Comente cada uma das etapas no processo descrito considerando as camadas da figura 1.

Questão 3: Considerando a questão anterior, explique as mudanças no processo se a arquitetura escolhida não for baseada em microcódigo.

Questão 4: Comente a funcionalidade da **Unidade Lógica Arimética (ULA)**, **Unidade de Controle (UC)**, **Registradores** e **Contador de Programa (PC)** nos processadores.

Questão 5: Explique qual a função da FPU (Floating-point unit).

Questão 6: Explique de maneira geral como ocorre a decodificação de instruções efetuada pela Unidade de Controle (UC) da CPU. Comente sobre OP codes e o formato de instruções.

Questão 7: O que é um montador (*Assembler*)?

Questão 8: Explique como funciona um montador (*Assembler*) para uma determinada arquitetura.

Questão 9: Por que a linguagem Assembly é dependente de arquitetura?

Questão 10: Explique a diferença entre um arquivo de código objeto (compilado) e um executável (pronto para execução).

Questão 11: Qual a diferença entre bibliotecas de software estáticas e dinâmicas?

Questão 12: Explique como funcionam bibliotecas de software dinâmicas, implementadas através de arquivos .DLL no Windows ou .SO no Unix/Linux.

Questão 13: Com base na figura 2, que contém uma ULA simplificada (1 bit), responda:

a) Quais operações são suportadas por esta ULA?

b) Qual a função do inversor de bit do operando B?

c) É possível estender esta ULA para operar em 32 bits? Como?

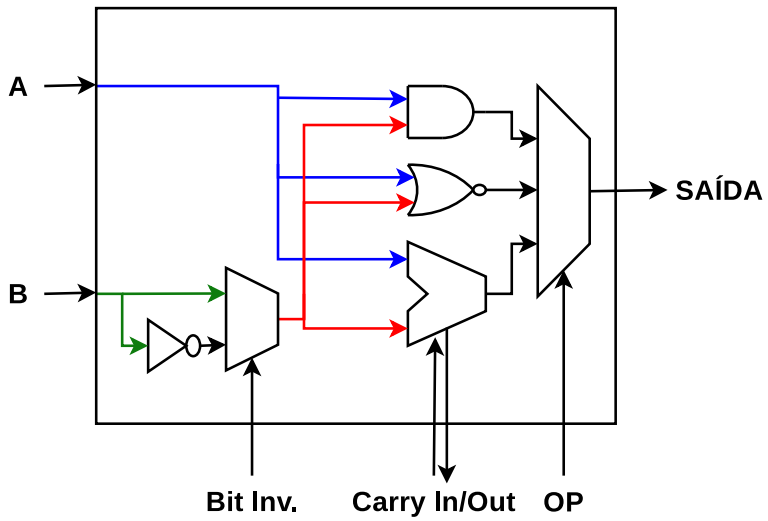

Figura 2: ULA simplificada de 1 bit.

Questão 14: A figura 3 contém um trecho de código assembly para arquitetura Intel x86 (32 bits). Explique o que o código faz e qual será o valor final dos registradores **EAX** e **EBX**.

```
1 mov eax, 0x0F
2 mov ebx, 0x04
3 add eax, ebx
```

Figura 3: Código assembly para arquitetura Intel x86 (32 bits).

Questão 15: Considere o programa completo da Listagem 1. Pesquise as instruções utilizadas e tente prever a saída do programa.

Listagem 1: Programa escrito em Assembly (Intel x86 32 bits)

```
1 ;;
2 ; Para compilar:
3 ;       nasm -felf32 sort.asm
4 ;       gcc -m32 sort.o -o sort
```

```
5  ;
6  extern printf, puts, exit
7  global main
8
9  section .data
10 msg: db '%d ', 0x00
11 cr:  db 0x0A, 0x00
12 vet: dw 3,5,7,9,8,2,1,4,6,10
13
14 section .text
15 print_vet:
16         mov ecx, 0
17 loop:
18         xor eax, eax
19         mov  ax, [ecx*2+vet]
20         push ecx
21         push eax
22         push msg
23         call printf
24         pop eax
25         pop eax
26         pop ecx
27         inc ecx
28         cmp ecx, 10
29         jb loop
30         ret
31
32 sss:
33         mov ecx, 0
34 l2:
35         push ecx
36         xor eax, eax
37         mov ax, [ecx*2+vet]
38         mov ecx, 0
39 l3:
40         xor ebx, ebx
41         mov bx, [ecx*2+vet]
42         cmp ax, bx
43         jb swap
44         jmp cont
45 swap:
46         mov [ecx*2+vet], ax
47         pop edx
48         mov [edx*2+vet], bx
49         mov ax, bx
50         push edx
51 cont:
52         inc ecx
53         cmp ecx, 10
54         jb l3
55         pop ecx
56         inc ecx
57         cmp ecx, 10
58         jb l2
59         ret
60
61 main:
62         call print_vet
63         push cr
64         call puts
```

```
65          call sss
66          call print_vet
67          call puts
68          pop eax
69          push 0x00
70          call exit
```