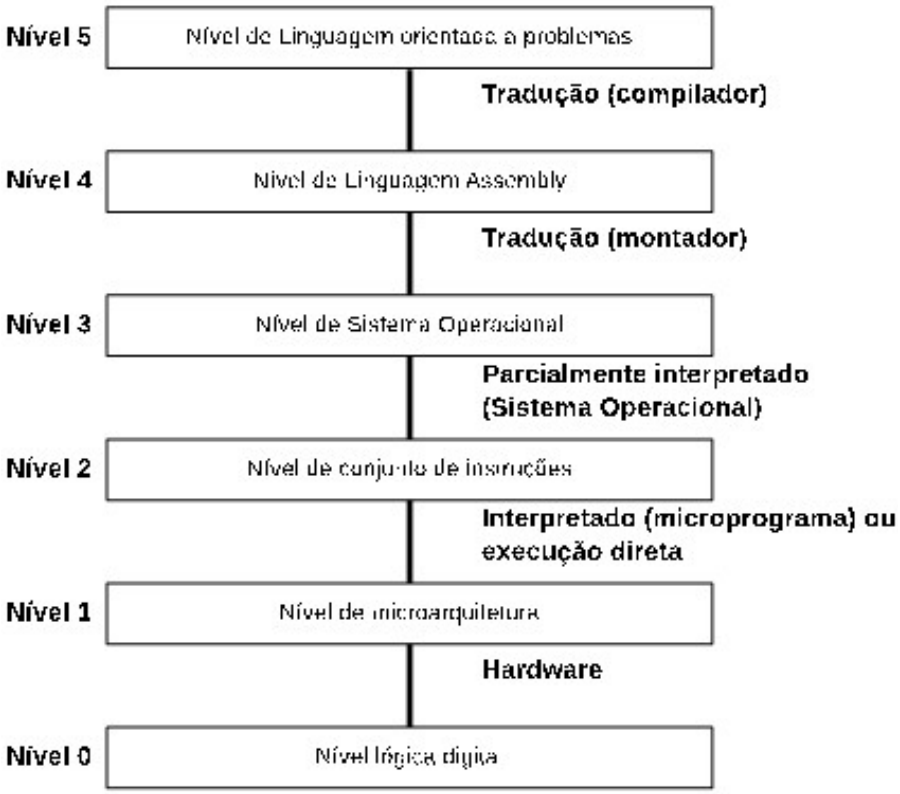
Questão 1: Com base na figura 1 comente o conceito de máquinas multiníveis.

Figura 1: Máquina Multiníveis



R: Computadores são projetados como uma série de níveis, cada um construídos sobre seus antecessores. Cada nível apresenta uma abstração distinta na qual estão presentes diferentes objetos e operações.

O nível 0 é o hardware verdadeiro da máquina, é o nível da lógica digital, os objetos interessantes são denominados portas.
O nível 1 é o verdadeiro nível de linguagem da máquina, nesse nível há definitivamente um programa, denominado microprograma, que interpreta as instruções de nível 2.
O nível 2 é chamado de nível de máquina convencional. Uma linguagem de nível 2 é definida por cada microprograma.
O nível 3 é geralmente um nível híbrido pois algumas de suas instruções são interpretadas pelo sistema operacional e outras são interpretadas diretamente pelo microprograma.
Os níveis 4 e 5, ao contrário dos anteriores, foram projetados para o uso direto pelo programador médio comum. Esses níveis são geralmente suportados por tradução enquanto os níveis 2 e 3 são interpretados

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.

O código é escrito em uma linguagem de alto nivel, traduzido para uma linguagem intermediaria até ser otimizada para uma linguagem de baixo nivel a ser interpretada por um montador e entendida pelo SO, então é otimizada para uma arquitetura X e é montada para uma linguagem de maquina.

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

R: Diferentemente de uma microarquitetura, as instruções são implementadas no nível de lógica digital.

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

R: ULA – em si efetua operações Aritméticas, adição, subtração e outras operações simples sobre as suas entradas, produzindo assim um resultado no REGISTRADOR DE SÁIDA. E também realiza operações lógicas AND/BOOLEAN.

UC - A UNIDADE DE CONTROLE é responsável por BUSCAR INSTRUÇÕES na MEMÓRIA PRINCIPAL e

determinar o seu tipo.

REGISTRADORES - A CPU contém uma pequena memória de alta velocidade usada para armazenar resultados temporários e para certo controle de informações. Essa memória é composta de uma quantidade de registradores, cada um deles com certo tamanho e função.

PC – É o REGISTRADOR mais importante que indica a próxima instrução a ser buscada para a execução e indica qual é a posição atual na sequência de execução de um processo.

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

R: Além da CPU os computadores usam co-processadores para realizar operações matemáticas a FPU, e o FPU é um hardware dedicado a executar operações matemáticas de dados representados em ponto flutuante em um computador.

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

R: Instruções sempre têm um OPcode que indica o que a instrução faz. Pode haver zero, um, dois ou três endereços presentes.
Existem quatro formatos comum de instrução: instrução sem endereço, instrução de um endereço, instrução de dois endereços, instrução de três endereços.
Cada instrução é executada como uma sequência de três fases pela UC:
1. acesso à memória (fetch)
2. decodificação (decode)
3. execução. (execute)
A instrução é inicialmente acessada na memória e transferida para o interior da CPU, mais especificamente num registrador especial da unidade de controle chamado de RI ("Registrador de Instrução"). Uma vez no RI, a instrução é interpretada por um circuito decodificador. Finalmente, ela é executada. A sequência apropriada de sinais é gerada pela unidade de controle, resultando nas transferências de dados e operações apropriadas.

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

R: O Assembler atua com um tradutor do código escrito na linguagem Assembly em linguagem de máquina, substituindo as instruções, variáveis pelos códigos binários e endereços de memória correspondentes. Os compiladores de várias linguagens de alto nível fazem a compilação dos programas em duas etapas, na primeira transformando o código fonte em código Assembly e em seguida gerando o binário com a ajuda de um Assembler.

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

R: A maioria dos assemblers é de duas passagens. A passagem um é dedicada a montar uma tabela de símbolos

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

R: Porque alguns processadores diferem em seus conjuntos de instruções, portanto para cada arquitetura há uma linguagem de montagem e essas linguagens de montagem diferem no número e tipo de operações que suportam. Também têm diferentes tamanhos e números de registradores, e diferentes representações dos tipos de dados armazenados.

Questão 10: Com base na figura 2, que contém uma ULA simplificada (1 bit), responda:

Quais operações são suportadas por esta ULA?

R: AND, NOR, SOMADOR/SUBTRATOR

Qual a função do inversor de bit do operando B?

R: Permitir a função de subtração utilizando o somador invertendo o bit

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

R: Fazendo 32 ULAs de 1 bit utilizando o modelo cascata

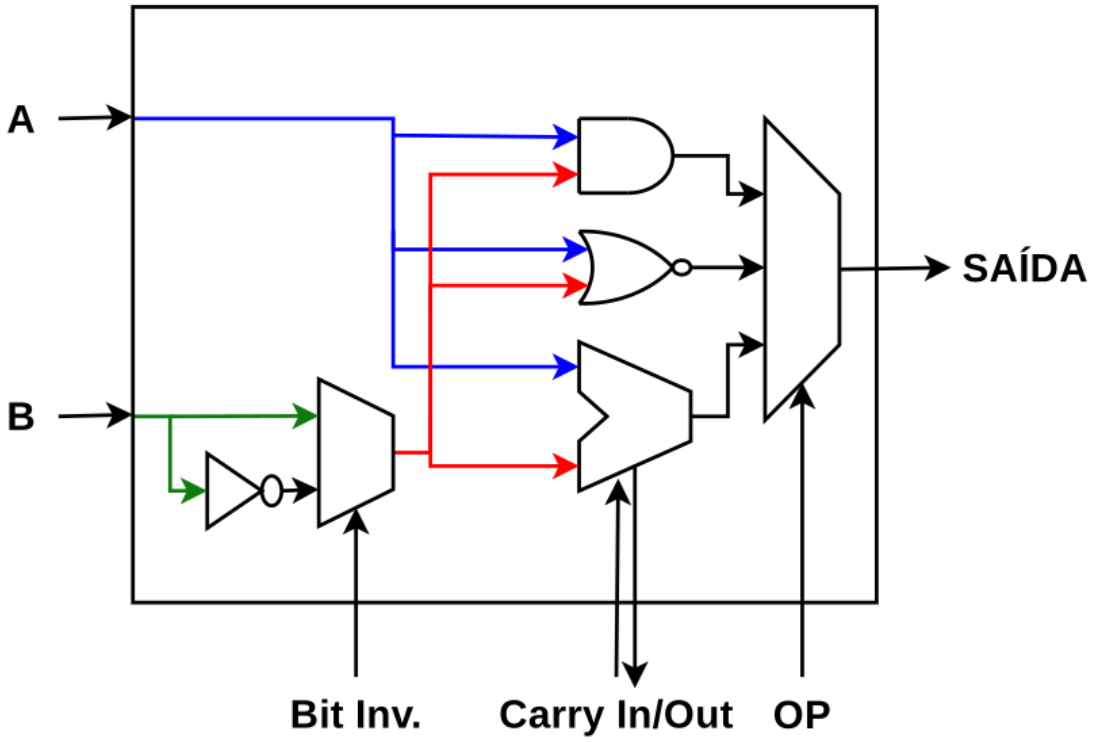


Figura 2: ULA simplificada de 1 bit.

Questão 11: A figura 3 contém um trecho de código assembly para arquitetura Intel x86 (32 bits). Explique o que o código faz e qual ser á 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).

R: É uma instrução de soma, move o registrador EAX e EBX e realiza um ADD para somar os registradores a saída será o valor 13:

mov eax , 0x0F [0x0F(16) =15(10)]

mov ebx, 0x04 [0x04(16)=4(10)]

add eax, ebx ([eax=eax+ebx]=[0x0F+0x04(base16)]=[15+4=19(base10)])

eax=13

mov – movimenta registradores de um endereço para o outro

add – adição de dois operandos, armazenando o result no primeiro operando

call – chama uma sub-rotina, adicionando-a na pilha de execução

ret – implementa um retorno na subrotina, retornando para o local do salto antes da chamada

pop – operador para remover elemento da pilha na memória

push – operador para empilhar operando no topo da pilha na memória

xor – executa operação lógica no operando, nesse caso é feito um xor, zera os operandos quando são iguais

cmp – instrução de comparação, assim como um if. Ex cmp jf,00 = if(jf==00)

jmp – transfere o fluxo do programa para o conteudo do operando local, levando as operções aritimeticas junto

inc – incrementa conteúdo no operando, adicionando 1

loop – transfere o fluxo da execução para uma subrotina, que sera executada até determinada condição