

Lista de Exercícios I - Monitoria

Arquitetura de Computadores

Para realizar os exercícios a seguir, instale o simulador SimuS. Consulte também a referência da arquitetura utilizada.

Exercício 1: Implemente um gerador da sequência de Fibonacci sem utilizar instruções de pilha (POP/ PUSH).

Requisitos mínimos:

- inicializar os dois primeiros termos da sequência ($F(0)$ e $F(1)$);
- calcular termos subsequentes iterativamente sem usar POP/ PUSH;
- armazenar o termo atual em uma variável acessível em memória.

Saída esperada: o programa deve produzir os termos da sequência de Fibonacci em ordem crescente na memória, um termo por iteração; não é necessário implementar limite superior; o programa deve progredir iterativamente.

Exercício 2: Implemente um programa que some dois valores inteiros armazenados em memória.

A variável `NUM1` contém o primeiro operando e `NUM2` contém o segundo operando. Some os dois valores e armazene o resultado em `RESULTADO`.

Requisitos mínimos:

- ler `NUM1` e `NUM2`;
- executar a operação de soma;
- armazenar o resultado em `RESULTADO`;

Saída esperada: após execução, a variável `RESULTADO` conterá a soma de `NUM1` e `NUM2`.

Exercício 3: Implemente um programa que verifique se um valor inteiro armazenado em memória é par ou ímpar.

O número de entrada está na variável `NUMERO`. Se `NUMERO` for par, armazene 1 em `RESULTADO`; caso contrário, armazene 0.

Requisitos mínimos:

- ler um número;

- utilizar operações lógicas (por exemplo AND com 1) ou aritméticas para determinar paridade;
- usar desvios condicionais (JZ/JN) para controlar o fluxo;
- escrever o resultado em RESULTADO (1 = par, 0 = ímpar).

Saída esperada: após execução, RESULTADO conterá 1 se NUMERO for par ou 0 se for ímpar.

Exercício 4: Implemente um programa que conte de 1 até 5 utilizando um loop e armazene o valor atual em memória.

Requisitos mínimos:

- inicializar a variável CONTADOR com 1;
- incrementar CONTADOR a cada iteração utilizando instruções aritméticas;
- empregar comparações e desvios condicionais (por exemplo JN/JZ) para controlar o término do loop.

Saída esperada: ao final da execução, a variável CONTADOR deve conter o valor 5.

Exercício 5: Implemente um programa que utilize endereçamento indireto para copiar um valor de memória.

A variável PONTEIRO contém o endereço de memória onde está o valor de origem. Use o modo de endereçamento indireto para carregar o valor apontado por PONTEIRO e copie-o para a variável DESTINO.

Requisitos mínimos:

- definir um valor origem em memória;
- definir PONTEIRO apontando para esse endereço;
- usar instrução com endereçamento indireto para ler o valor e em seguida armazená-lo em DESTINO.

Saída esperada: após execução, a variável DESTINO deve conter a cópia do valor de origem.

Exercício 6: Implemente um programa que aplique uma máscara de bits por meio de operação lógica AND.

O operando está na variável NUMERO e a máscara está em MASCARA. Armazene o resultado da operação em RESULTADO.

Requisitos mínimos:

- ler NUMERO e MASCARA do espaço de dados;
- aplicar a operação lógica bit a bit AND entre NUMERO e MASCARA;

- armazenar o resultado em RESULTADO.

Saída esperada: após execução, RESULTADO conterá o valor de NUMERO AND MASCARA.

Exercício 7: Implemente um programa que utilize deslocamento lógico à esquerda para multiplicar um valor por 2.

O valor de entrada está na variável NUMERO. Aplique a instrução SHL e armazene o resultado em RESULTADO.

Requisitos mínimos:

- ler NUMERO do espaço de dados;
- aplicar instrução de deslocamento lógico à esquerda (SHL) para multiplicar por 2;
- armazenar o resultado em RESULTADO;

Saída esperada: após execução, RESULTADO conterá NUMERO multiplicado por 2. Desconsidere overflow.

Exercício 8: Implemente um programa que realize subtração com borrow utilizando a instrução SBC.

Calcule NUM1 - NUM2 considerando o estado inicial do carry/borrow conforme a arquitetura; defina o carry/borrow inicial antes de executar SBC.

Requisitos mínimos:

- carregar NUM1 e NUM2;
- definir o carry/borrow inicial conforme necessário;
- executar a instrução SBC para efetuar a subtração com borrow;
- armazenar o resultado em RESULTADO.

Saída esperada: após execução, RESULTADO conterá NUM1 - NUM2 (com borrow aplicado).

Exercício 9: Implemente um programa que utilize a operação lógica XOR para alternar bits de um valor.

O operando está na variável NUMERO e a máscara está em MASCARA_XOR. Aplique NUMERO XOR MASCARA_XOR e armazene o resultado em RESULTADO.

Requisitos mínimos:

- ler NUMERO;
- aplicar a operação XOR entre NUMERO e MASCARA_XOR;
- armazenar o resultado em RESULTADO.

Saída esperada: após execução, RESULTADO conterá NUMERO XOR MASCARA_XOR.

Exercício 10: Implemente um programa que demonstre o uso da pilha por meio das instruções PUSH e POP.

Empilhe o valor armazenado em VALOR_A, carregue VALOR_B no acumulador, em seguida desempilhe o valor de VALOR_A e some-o ao acumulador.

Requisitos mínimos:

- utilizar instruções PUSH e POP para preservar e restaurar valores na pilha;
- carregar VALOR_B no acumulador antes de desempilhar VALOR_A;
- efetuar a soma e armazenar o resultado em RESULTADO.

Saída esperada: após execução, RESULTADO conterá $\text{VALOR_A} + \text{VALOR_B}$.