Trabalho da 1ª unidade

Cada questão a seguir possui um algoritmo correspondente na linguangem C. Traduza esses algoritmos para o assembly do MIPS. Use os registradores que achar necessário.

1 – Escreva um algoritmo que calcule a divisão inteira e o resto da divisão de dois números.

Entrada:
Ler dois números inteiros positivos correspondentes ao dividendo e o divisor.
Saída:
Consiste no resultado da divisão inteira (quociente) e o resto da divisão:

Exemplo de entrada:
10
3

Saída correspondente:
Quociente = 3
Resto = 1

```
#include <stdio.h>
int main(){
    int dividendo, divisor, resto, quociente;

    scanf("%d", &dividendo);
    scanf("%d", &divisor);

    resto = dividendo;
    quociente = 0;

    while(resto >= divisor){
        resto = resto - divisor;
        quociente++;
    }

    printf("Quociente = %d\n", quociente);
    printf("Resto = %d\n", resto);

    return 0;
}
```

2 – Depois de desenvolver o algoritmo que calcula o resto de uma divisão, utilize-o como procedimento para um programa que verifica se um número é par ou ímpar.

Entrada:
Corresponde a um número inteiro positivo.
Saída:
Para cada caso, mostre uma mensagem indicando se o número é ímpar ou par.

Exemplo de entrada:
13
Saída correspondente:
Impar

```
#include <stdio.h>
int resto_divisao(int dividendo, int divisor){
    int resto, quociente;
    resto = dividendo;
    quociente = 0;
    while(resto >= divisor){
        resto = resto - divisor;
        quociente++;
    }
    return resto;
}
int main(){
    int numero, resto;
    scanf("%d", &numero);
    resto = resto_divisao(numero, 2);
    if(resto == 0)
        printf("Par\n");
    else
        printf("Impar\n");

    return 0;
}
```

3 – Escreva um procedimento recursivo que calcule a exponencial de um número.

Entrada:

Consiste de dois números inteiros positivos, a base e o expoente respectivamente.

Saída:

O número que corresponde ao resultado dessa operação:

Exemplo de entrada:

2
8

Saída correspondente:

256

```
#include <stdio.h>
int power(int base, int expoente){
    int resultado;
    if(expoente == 0)
        resultado = 1;
    else
        resultado = base * power(base,
expoente-1);
    return resultado;
}

int main(){
    int base, expoente, resultado;

    scanf("%d", &base);
    scanf("%d", &expoente);

    resultado = power(base, expoente);

    printf("%d\n", resultado);

    return 0;
}
```

4 – Escreva o procedimento do programa ao lado que ordena um vetor com N posições e mostre os elementos desse vetor em ordem crescente.

Entrada:

A primeira linha da entrada consiste em um inteiro positivo N, que indica a quantidade de elementos que o vetor possui. As N linhas seguintes correspondem aos valores de cada elemento do vetor.

Saída:

Consiste de duas linhas, a primeira mostra os números do vetor na sequência que foi lida e a segunda, os números em ordem crescente.

Exemplo de entrada:

5
3
7
4
8
1

Saída correspondente:

3 7 4 8 1
1 3 4 7 8

```
#include <stdio.h>
#include <stdlib.h>

void printVetor(int *a, int len){
    for(int i = 0; i < len; i++)
        printf("%d ", a[i]);

    printf("\n");
}
void ord(int p, int *a, int len){
    if(len > p){
        for(int i = p; i < len; i++){
            if(a[p] > a[i]){
                int aux = a[i];
                a[i] = a[p];
                a[p] = aux;
            }
        }
        ord(p+1, a, len);
    }
}

int main(){
    int p, len, *a;
    p = 0;

    scanf("%d", &len);

    a = (int*)malloc(sizeof(int)*len);

    for(int i = 0; i < len; i++){
        scanf("%d", &(a[i]));
    }

    printVetor(a, len);
    ord(p, a, len);
    printVetor(a, len);

    return 0;
}
```