Lista 1
Aula prática Assembly MIPS 2023.1
Ciência da Computação
Aula prática assembly MIPS

Atenção: o código de todas as questões deverá estar claramente comentado. Em caso contrário, a correção será muito dificultada e é de seu interesse colaborar com a correção!

Obs: O arquivo compactado deve obrigatoriamente ser nomeado Grupo {número do grupo} - Lista 01 substituindo a expressão demarcada pelas chaves com o número apropriado.

O supercomputador *Deep Thought* foi criado para calcular a resposta para a vida, o universo e tudo o mais. No entanto, após aproximadamente um bilhão de anos de tempo de processamento, uma erupção vulcânica destruiu o módulo responsável por compilar o código para assembly MIPS. Você foi convocado pelos programadores Loonquawl e Phouchg para realizar esta tarefa, já que o Dia da Resposta se aproxima rapidamente e é de suma importância que a cerimônia ocorra sem nenhum problema.

1. (1,0) Uma poderosa entidade de nome desconhecido e cor dourada em formato de triângulo equilátero tem ameaçado o processamento da resposta através de pactos faustianos com os programadores responsáveis. Para se proteger, é necessário implementar uma maneira de distinguir triângulos. Escreva um programa em linguagem de montagem do MIPS que receba três números inteiros armazenados na **memória** e verifica se aqueles três números podem ser comprimentos de lados de um triângulo, e caso sejam, defina qual tipo de triângulo ele é, armazenando o resultado numa string **s na memória**, “not” caso não sejam comprimentos de lados de um triângulo, “eq” para equilátero, “iso” para isósceles, “esc” para escaleno. Lembrando que para três números serem os comprimentos dos lados de um triângulo, cada lado deve ser menor que a soma dos outros dois.

2. (1,5) Ao considerar uma nova linha de pesquisa que eventualmente levaria ao desenvolvimento da nave *Heart of Gold*, *Deep Thought* inventou um gerador de improbabilidade infinita. No entanto, é necessário compilar um gerador de números aleatórios para iniciar

seu funcionamento. Codifique um programa que implementa um *linear congruential generator* e gera 10 valores, utilize o seguinte pseudo-código:

```
// Talvez uma syscall ajude :D
uint seed = rand();

const int mult_constant = 1664525;
const int add_constant = 1013904223;
for (int i = 0; i < 10; i++) {
     seed = seed * mult_constant + add_constant;
//utilize a instrução mulu para realizar multiplicação sem sinal e addu para realizar adição sem sinal
     printf("%d\n", seed);
}
```

3. (1,5) Foi descoberta uma conspiração para impedir que a resposta para a vida, o universo e tudo o mais fosse encontrada. Felizmente, os rebeldes utilizaram um protocolo de esteganografia extremamente inseguro. Para ser capaz de decodificar as mensagens, escreva um programa em linguagem de montagem do MIPS que recebe uma ***null-terminated* string da memória** e armazena **apenas as letras maiúsculas** em outra string. Em seguida, armazene a quantidade de letras maiúsculas no registrador v1 e encerre o programa.

4. (2,0) Após milênios no *backlog* de refatoração, a função de divisão inteira utilizada pelo *Deep Thought* será finalmente reescrita em assembly MIPS (a função anterior foi escrita em Malbolge por um *code golfer*). Escreva um programa em linguagem de montagem do MIPS que recebe dois números inteiros armazenados na memória e realize a divisão **inteira** dos dois números. Considere números positivos e negativos. A instrução **"`div`" não deverá ser utilizada** na implementação dessa questão (*Deep Thought* é alérgico). O resultado (quociente da operação) deverá ser armazenado em uma variável `RESULT` na memória e o resto da divisão deve ser armazenado em uma variável `REMAINDER` na memória.

5. (4,0) Implemente (**recursivamente**) na linguagem de montagem do MIPS, uma função que receba dois números (a e b) e retorne a mod b. Se a for negativo, armazene o valor 1 no registrador v1 e encerre o programa.

6. (0,01) O grupo de rebeldes mencionados na questão 5 retornou, mas dessa vez estão armados com criptografia AES256 e suas comunicações são indecifráveis! Para conseguir informação suficiente para frustrar seus planos, é necessário implementar uma função em assembly MIPS capaz de realizar fatoração primária em tempo polinomial.

a. (0,005) Implemente uma função que recebe um inteiro de 256 bits e retorna um array que contém todos os seus fatores primos em tempo polinomial.
b. (0,005) Prove que a função implementada está contida em P. Em seguida, escreva a *big O notation* da função.

Não entre em pânico, você não tem inimigos!