# PCS3616 - Sistemas e programação

# Projeto 1 - Chaveamento de processos


## Lucas Nimirio Maia Feng - 11915950
## Patrick Marques de Barros Costa - 11257550


O projeto consiste de 3 funções, a ROTINA1 que que gera números pseudo-aleatórios e salva eles; a ROTINA2, que faz uma verificação se os números do R1 são divisíveis por N2 escrevendo na tela ”SIM!” caso sejam e "NÃO!" caso não sejam e a função ROTINA3 que funciona como uma MAIN que importa as outras 2 funções e faz a ordem de chamada das funções na ordem adequada.

A ROTINA1 com base em N1 = 11915 funciona importando um número aleatório N e retorna o resto da divisão por N1 utilizando o conceito da divisão inteira, e subtraindo de novo o resultado multiplicado por N1 com N (algo bem semelhante com o conceito de % nas linguagens de programação).

A ROTINA2 com base em N2 = 550 funciona importando a ROTINA1 e analisando se a saída da ROTINA1 é divisível por N2, dependendo do resultado o código direciona a 2 sub-rotinas: NEGA que faz com que imprima “NÃO!” no monitor MVN e AFIRMA que faz com que imprima “SIM!” no monitor MVN

A ROTINA3 funciona importando a ROTINA1 e a ROTINA2 fazendo a formalização da chamada das 2 rotinas na ordem certa da proposta. Além disso, o grupo fez um comando de interrupção no meio da chamada das 2 rotinas (depois da primeira e antes da segunda).

# Respostas das perguntas

**1-** Os algoritmos poderiam ser mais eficientes por exemplo ao invés de ter que escrever a palavra SIM! ou NÃO! no monitor da MVN poderia ser algo que tivesse somente 1 ou 2 ASC (exemplo: “S” para sim e “N” para não) para que pudesse ser feito em uma única instrução PD e evitar uma possível interrupção indesejada no meio da escrita, mas a escolha dessa primeira opção da proposta faz com que o código se torne mais completo e informativo porém menos eficiente.

**2-** O processo para achar o resto da divisão nas 2 primeiras rotinas é o processo que mais tem tempo de execução prolongado, ou seja, o que mais deixa o programa menos eficiente. A melhor maneira de aumentar a eficiência seria realizar essa verificação para a escrita de “SIM!” e "NÃO!" de outra maneira, porém o grupo fez os códigos dessa maneira porque achamos que seria um pouco mais prático, simples e autoexplicativo para a proposta do projeto.

**3-** A maior dificuldade foi implementar a função main, principalmente na parte da interrupção. O grupo pesquisou saber mais sobre o processo de interrupção no assembly e decidimos implementar dessa maneira bem simples, que utiliza os 50 ciclos que são padrão.

**4-** Em nosso código, para comportar 3 rotinas sendo chaveadas, para realizar essa mudança, precisamos verificar se o número de instruções não ultrapasse o número de k. Pensando no caso para um número arbitrário K, o número de k é o máximo de ciclos, onde se o programa tiver um número de instruções maior do que numero de k, ocorrerá um travamento no programa, para adicionar mais instruções, o número de instruções não pode ser maior que o número de k.

**5-** Com a alteração do valor de NUM no código, consequentemente iria alterar o número total de instruções. O valor máximo não possui limite, e o valor mínimo seria o total de instruções das rotinas (29).