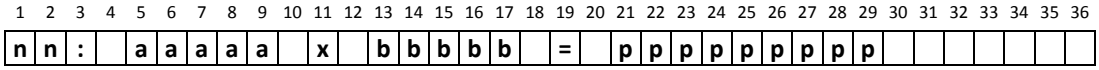
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA - DEPARTAMENTO DE INFORMÁTICA APLICADA

INF01108 - Arquitetura e Organização de Computadores I – semestre 2012/1 - Turmas A e B

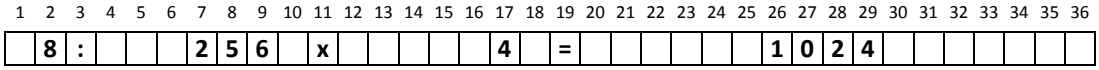
**Trabalho Prático 2 - Simulador CESAR (versão "14 bits.v2" do enunciado)**

## Definição

Escrever um programa para o Cesar que use um LFSR (*Linear Feedback Shift Register*) para calcular produtos de números inteiros sem sinal de 14 bits, gerados de forma pseudoaleatória, exibindo cada produto no visor de acordo com o leiaute mostrado na figura a seguir.




Sendo *nn* um inteiro de 1 a 99 que identifica sequencialmente os produtos calculados, *aaaaa* e *bbbbb* os fatores gerados aleatoriamente (inteiros sem sinal no intervalo [1, 16383]) e *ppppppppp* o produto obtido, que poderá variar de 1 a 268.402.689 (não é necessário exibir os pontos, mas os zeros não significativos devem ser eliminados em todos os 4 valores – ver exemplo abaixo).




Após exibir cada produto, o programa deverá aguardar que o usuário pressione a tecla [Enter] (código ASCII $13_{10}$), antes de exibir o próximo produto. Se, em vez de [Enter], o usuário pressionar a tecla [Esc] (código ASCII $27_{10}$), o programa deverá perguntar se deseja gerar uma nova sequência ou não. Se a resposta for não, exibir uma mensagem de encerramento e terminar.

## Especificações

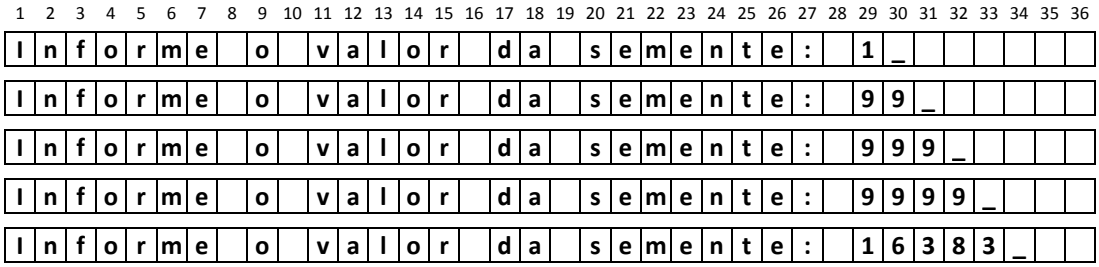
O programa deverá interagir com o usuário através do teclado e visor, e deverá seguir os seguintes passos:

1. Exibir no visor uma ou mais linhas, identificando o programa e o seu autor (nome e número de cartão). Após exibir cada linha, aguardar que o usuário pressione a tecla [Enter], antes de escrever a próxima (este procedimento vale para todas as exibições de *mensagens consecutivas* especificadas neste enunciado, e visa dar tempo ao usuário de ler cada mensagem antes de passar para a próxima).

2. Exibir uma mensagem solicitando que o usuário digite um número para ser usado como "semente" para a geração dos números pseudoaleatórios a multiplicar (ver exemplos abaixo). O valor será digitado sem zeros não significativos e deve ser um inteiro sem sinal de 1 a 5 dígitos, no intervalo [1, 16383] (mas talvez o usuário não saiba isto ...). O valor lido deve ser convertido de ASCII para binário e não pode ser menor do que 1 nem maior do que 16.383. Se estiver correto, o valor binário deve ser armazenado em uma variável denominada "semente", que será usada pela subrotina de geração de números pseudoaleatórios (ver adiante).
   Na digitação e leitura do valor da semente, observar as seguintes regras:
   - cada caractere digitado deve ser ecoado no visor
   - deve ser apresentado (e controlado) um cursor, representado pelo caractere *Underscore* ("_", código ASCII $95_{10}$)
   - deve ser permitido o uso da tecla [Backspace] (código ASCII $8_{10}$) para corrigir erros de digitação (cuidado com o tratamento do [Backspace] **antes** do primeiro caractere ser digitado ou depois de ter apagado todos os caracteres digitados!)

- o programa deve considerar que a entrada de todos os dígitos terminou quando for pressionada a tecla [Enter], mas deve permitir que todos os dígitos (inclusive o último) sejam corrigidos com o uso de [Backspace].
- portanto, nesta etapa o programa deverá aceitar como entrada somente os códigos ASCII dos dígitos de 0 a 9 ($48_{10}$ a $57_{10}$) e dos valores gerados pelas teclas [Backspace] ($8_{10}$) e [Enter] ($13_{10}$).

Exemplos de entradas válidas:




3. Escrever no visor a mensagem "Sequencia de produtos calculados:" (o conjunto ASCII padrão não inclui caracteres acentuados, como '**ê**') e aguardar que o usuário pressione a tecla [Enter].

4. Usar a subrotina PRandom (explicada no final do enunciado) para gerar dois inteiros positivos entre 1 e 16.383. Usar as sub-rotinas da BibCesar para multiplicar estes dois valores, obtendo o produto (um valor inteiro positivo de 32 bits), e para converter o valor binário do produto para ASCII.

5. Exibir os fatores e o produto no visor, no formato mostrado na definição do trabalho, e aguardar que o usuário pressione uma tecla. Nesta etapa, não aceitar nenhuma tecla que não seja [Enter] ou [Esc]. Se for pressionada [Enter], voltar à etapa 4. Se for pressionada [Esc], ir para a etapa 6.

6. Perguntar ao usuário se ele deseja gerar uma nova sequência de produtos. Se a resposta for 's' ou 'S', voltar à etapa 3 (ou seja, a nova sequência não pedirá um valor novo para semente, e usará o último valor gerado por PRandom para a sequência anterior). Se a resposta for 'n' ou 'N', passar para a etapa 7. Se a resposta não for nenhuma destas quatro, informar ao usuário que a resposta é inválida e repetir a pergunta. O programa deverá permitir que o usuário mude de idéia ao digitar sua resposta, ou seja, ele deve poder mudar a resposta, usando a tecla [Backspace] e digitando uma nova letra, antes de pressionar a tecla [Enter].

7. Exibir no visor uma mensagem de encerramento e terminar a execução do programa.

Não esqueça de que, antes de armazenar no *buffer* do teclado os caracteres digitados pelo usuário, o Cesar converte os mesmos para ASCII. Portanto, tais caracteres podem ser enviados para o visor neste mesmo formato. No entanto, para uso em operações aritméticas os valores devem ser convertidos para binário. Da mesma forma, o resultado da multiplicação devolvido pela subrotina da BibCesar estará em binário, devendo ser convertido para ASCII para escrever no visor (dica: cuidado com estouros na divisão neste processo ...).
Conforme as definições acima, na leitura dos números devem ser previstos casos de erros de digitação (letras e outros símbolos que não sejam dígitos ou as teclas de controle). Além de valores acima de 16.383, os números com mais de 5 dígitos também devem ser tratados como casos de erro.
O formato das diversas mensagens não especificadas no enunciado fica a cargo do programador.
O trabalho deve ser desenvolvido usando o montador Daedalus e, no início do código fonte, devem ser colocados comentários com a identificação do autor (nome, número do cartão e turma pelo menos). Ao longo do código fonte também devem ser usados comentários, para descrever a finalidade dos principais blocos do programa, a fim de permitir que um programador que no futuro vá estudá-lo ou fazer sua manutenção possa entender a lógica do mesmo.

Os trabalhos serão corrigidos manualmente e devem ser observadas rigorosamente as seguintes especificações:

- O código do programa deve iniciar no endereço 0 da memória.
- A primeira instrução executável deve estar no endereço 0.
- O programa deverá executado para gerar tantas sequências de produtos quanto o usuário desejar.
- Não há garantia de que os dados digitados como valor da semente estão corretos. Portanto, o programa deve verificar a consistência desses dados.

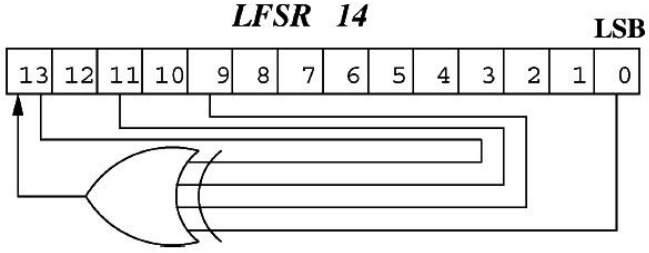
O trabalho deverá ser desenvolvido de forma **individual** e entregue via Moodle, contendo um arquivo fonte comentado (.ced ou .txt) e um executável (.mem). Para os nomes dos arquivos, utilize a inicial de seu primeiro nome seguida de seu número de cartão. Assim, por exemplo, o aluno João José da Silva, com cartão numero 00123456, nomearia seu arquivos como J123456.mem, J123456.ced e J123456.zip.

Os testes serão realizados com a opção "Atualizar Registradores" desligada (ver Menu "Executar" no simulador). Leve isto em conta caso o seu programa utilize rotinas que dependam de tempo (como um cursor piscante, por exemplo).

**Definição da subrotina PRandom (detalhes vistos em aula nos dias 23 e 24 de abril)**

Esta rotina deverá simular o funcionamento de um LFSR (*Linear Feedback Shift Register*) para gerar números inteiros sem sinal de 14 bits, no intervalo [1, 16383], utilizando como semente o valor fornecido pelo usuário na etapa 2 do algoritmo. Ela será chamada duas vezes durante o cálculo de cada produto (etapa 4 do algoritmo), para gerar os dois fatores. Note que a cada nova sequência de produtos não será solicitada uma nova semente.

A figura abaixo mostra o leiaute do LFSR, com a porta XOR de 4 entradas necessária. Note que esta figura(*) , ao contr5ário da usada no enunciado anterior, mostra os bits do registrador na mesma ordem usada em aula, com o menos significativo (bit 0) à direita e o mais significativo (bit 13) à esquerda. Portanto, a cada chamada da subrotina, o conteúdo do LFSR é girado para a direita em 1 bit e o bit mais significativo ($b_{13}$) é substituído pelo valor de ($b_{13}$ XOR $b_{11}$ XOR $b_9$ XOR $b_0$), onde $b_i$ indica o bit de ordem $i$ do registrador ($0 \leq i \leq 13$).




(*) Fonte: http://www.see.ed.ac.uk/~s0571365/Files/Articles/Communication/On%20Algorithmic%20Rate-Coded AER%20Generation_Linares-Barranco_06.pdf

A critério do professor, aqueles trabalhos que funcionarem corretamente nos testes e forem além dos requisitos mínimos definidos no enunciado poderão receber uma bonificação de até 10% na nota. Exemplos de requisitos adicionais são: a exibição dos valores com leiaute variável (suprimir os espaços em branco correspondentes a zeros não significativos), colocação de pontos no lugar certo entre os dígitos exibidos e o uso de cursor piscante na leitura do valor da semente (etapa 2). Por exemplo:

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
8 : | 1 . 2 5 6 | x | 4 . 0 0 0 | = | 5 . 0 2 4 . 0 0 0 |

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
9 9 : | 1 6 . 3 8 3 | X | 1 6 . 3 8 3 | = | 2 6 8 . 4 0 2 . 6 8 9 |
```

**Dia e hora limites para entrega: 13/05/2012, 23h55, via Moodle**