

Trabalho de Programação 2

Processador CESAR16i – Parte 1 – Programa de aplicação

1. Descrição Geral

Sua missão neste trabalho, que será desenvolvido em duas etapas, é implementar um programa para medir o tempo de reação e velocidade de digitação do usuário. O programa irá exibir sequências de caracteres formadas por 8 letras (maiúsculas e/ou minúsculas) e dígitos decimais e o usuário deverá digitar a mesma sequência no menor tempo possível, sem erros.

À semelhança do que ocorre em sistemas comerciais (Ex: Windows e Linux), o programa de aplicação deverá chamar “funções do kernel” para interagir com os periféricos (teclado, visor e timer) do Cesar sem necessidade de conhecer os detalhes de funcionamento destes periféricos e dos mecanismos de interrupção.

Assim, nesta primeira parte do trabalho você deverá implementar o programa de aplicação que valerá 50% da nota do trabalho. A segunda parte consistirá em implementar o kernel e suas funções: a rotina de inicialização do sistema de interrupções e os tratadores de interrupções do teclado e do timer, que também valerá 50% da nota.

Inicialmente, o programa de aplicação exibirá no visor a identificação do autor e instruções de uso (obrigatórias, formato definido por você), esperando que o usuário digite um **ENTER** após exibir cada mensagem antes de exibir a próxima. Em seguida, pedirá ao usuário que digite **ENTER** para iniciar o jogo. A partir daí, o programa irá limpar o visor e exibir no mesmo uma sequência de caracteres que deverá ser digitada pelo usuário seguida de um **ENTER**, como mostrado abaixo:



Figura 1

Na digitação da sequência de caracteres o programa deverá aceitar e processar corretamente o caractere BACKSPACE, para corrigir o que foi digitado, e também administrar um cursor (caractere ‘_’) que indicará a próxima posição onde será exibido o caractere digitado. O fim da digitação será indicado por um **ENTER**. Depois de recebida a sequência digitada, o programa informará ao usuário se a mesma está correta ou não e quantos segundos o usuário levou para digitar. Exemplos de saída após a digitação:

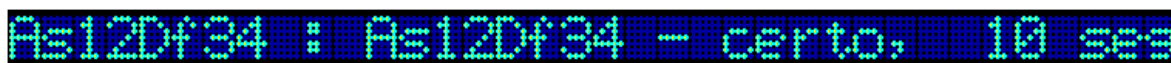


Figura 2

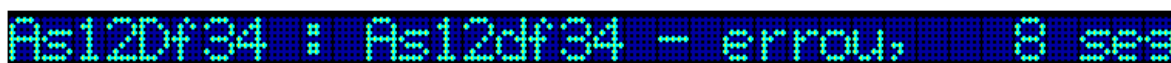


Figura 3

Em seguida, o programa deverá aguardar um **ENTER** para prosseguir e depois perguntar ao usuário se ele deseja jogar novamente ou encerrar o programa, com uma mensagem do tipo:

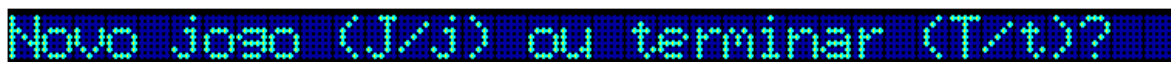


Figura 4

A entrega do trabalho será realizada em DUAS PARTES, cada uma em uma data diferente.

Na primeira parte deverá ser entregue **apenas** o programa de aplicação (APP). Para o desenvolvimento desta parte do trabalho, será colocado à disposição um arquivo fonte que deverá ser usado como base para o desenvolvimento (APP_REF.CED).

Também será disponibilizado um arquivo com a implementação do kernel (KERNEL_PROF.MEM). Esse arquivo deverá ser carregado no simulador antes que seja feita a carga parcial da sua solução, conforme instruções adiante.

2. Implementação.

O programa de aplicação deverá executar as etapas descritas a seguir. Para isso, sua APP deverá utilizar as funções do kernel. As funções disponíveis estão descritas na documentação do kernel ¹.

¹ Como toda a APP, o sistema de interrupções e do apontador da pilha não necessitam ser inicializados. Essas tarefas são de responsabilidade do kernel sobre o qual a APP está sendo executada.

1. Exibir no visor, obrigatoriamente, mensagens de identificação do autor (nome e número do cartão) e de instruções de uso da APP. Após exibir cada linha da mensagem, aguardar que seja digitado um caractere **ENTER** (código ASCII 13), antes de continuar.
2. Em seguida, colocar mensagem no visor perguntando se o usuário está pronto para iniciar as medições de tempo. Aguarde **ENTER** para continuar.
3. Então, limpar o visor e exibir uma sequência de caracteres (selecionada da tabela de sequências de caracteres – ver “Tabela de Sequências” a seguir) nas primeiras 8 posições, seguida de ' : ' e de um cursor para indicar onde será exibido o primeiro caractere digitado (ver Figura 1, acima). Nesse momento, a APP deve ler o timer, de forma a salvar seu valor no instante de início da medição do tempo.

Nesta mensagem, o caractere ' _ ' indicará a posição do “cursor” que deverá ser administrada pela APP, para indicar onde ecoar (escrever) os caracteres obtidos do teclado durante a leitura dos caracteres digitados. A APP deverá processar corretamente o caractere **BACKSPACE**, cuidando para não recuar além da posição do primeiro caractere.

A APP só deve aceitar a digitação dos caracteres previstos: letras maiúsculas e minúsculas e dígitos de 0 a 9. Também não deverão ser aceitos mais do que 8 caracteres. Entretanto, também deverão ser aceitos e processados corretamente o caractere **BACKSPACE** (código ASCII 8), para correção do que foi digitado, e o caractere **ENTER**, para indicar o fim da digitação. Qualquer caractere não previsto deverá ser ignorado.

Ao receber o **ENTER**, o programa deverá ler o timer novamente, para obter o tempo final da digitação. Além disso, a APP deverá comparar os caracteres da sequência exibida com a sequência digitada, verificando se são iguais. Se forem iguais, o programa deverá escrever '**certo**' no visor (ver Figura 2). Caso contrário, deverá escrever '**errou**' (ver Figura 3). Em qualquer caso, a APP também deve exibir o tempo, em segundos, transcorrido desde o início da medição de tempo até a digitação do **ENTER**.

O tempo deverá ser exibido com três dígitos, com eliminação de zeros não significativos. Ver Figuras 2 e 3 acima. Em seguida, a APP deve aguardar um **ENTER** (ignorar qualquer outro caractere), para então prosseguir.

4. Colocar no visor mensagem de pergunta se o usuário deseja realizar nova medição ou encerrar o programa (ver mensagem 4 acima). A APP só deve aceitar os seguintes caracteres: 'J' ou 'j' para realizar nova medição– neste caso, voltar à etapa 3 – ou 'T' ou 't' para terminar o programa – neste caso, passar à etapa 5.
5. Exibir uma mensagem de encerramento e terminar a execução do programa.

3. Tabela de sequências

A APP deverá usar uma tabela com 16 sequências de caracteres, cada uma com 8 caracteres. A APP deverá selecionar uma sequência que será exibida no visor, conforme descrito na etapa 3 (ver “Implementação”). Para escolher a sequência a ser exibida a cada jogada, deverá ser lido o tempo atual do kernel e utilizados apenas os 4 bits menos significativos desse valor, para decidir qual das sequências a ser usada (os 4 bits representam o índice da tabela a ser utilizado). Você pode escolher qualquer sequência que desejar. Entretanto, sugere-se mesclar sequências simples e complexas, como no exemplo abaixo:

```
tab_seqs: dab 'abcdefgh' ; índice 0
          dab '01234567' ; índice 1
          dab 'A1B2C3D4' ; índice 2
          dab 'ab01CD23' ; índice 3
          dab 'Alfabeto' ; índice 4
          dab 'Numerais' ; índice 5
          dab 'WXYZ9876' ; índice 6
          dab 'INF01108' ; índice 7
          ...
          dab 'novembro' ; índice 14
          dab '0a1B2c4D' ; índice 15
```

4. Divisão do espaço de endereçamento (alocação de memória)

Para a implementação do trabalho o espaço de endereçamento do CESAR será dividido da seguinte forma:

Faixa de Endereços	Descrição
H0000 a H00FF	Esta área não deve ser acessada nem modificada pelo programa de aplicação. Ela será preenchida quando for carregado o kernel (KERNEL_PROF.MEM).
H0100 a H7FFF	Área para o programa de aplicação a ser desenvolvido na primeira parte do trabalho. Ele será carregado usando uma carga parcial do arquivo onde está sua solução, para os endereços H0100 a H7FFF.

H8000 a HFFFF	Esta área não deve ser acessada nem modificada pelo programa de aplicação. Ela será preenchida quando for carregado o kernel (KERNEL_PROF.MEM).
---------------	---

5. Correção e Entregáveis

A correção desta primeira parte do trabalho será feita usando um programa de teste especialmente desenvolvido para esta finalidade, que será carregado no simulador antes de fazer uma carga parcial do código de sua solução nos endereços H0100 a H7FFF da memória.

Deve ser entregue um arquivo fonte (arquivo .CED) com a solução correspondente, escrito em linguagem simbólica do CESAR16i usando o montador Daedalus. O código do programa fonte deverá conter comentários descritivos da implementação.

Esta parte do trabalho deverá ser entregue até a data prevista indicada no sistema Moodle. Não serão aceitos trabalhos entregues além do prazo estabelecido. Trabalhos não entregues até a data prevista receberão nota zero.

6. Observações

Recomenda-se a troca de ideias entre os alunos. Entretanto, a identificação de cópias de trabalhos acarretará na aplicação do Código Disciplinar Discente e a tomada das medidas cabíveis para essa situação (**tanto o trabalho original quanto os copiados receberão nota zero**).

O professor da disciplina reserva-se o direito, caso necessário, de solicitar uma demonstração do programa, onde o aluno será arguido sobre o trabalho como um todo. Nesse caso, a nota final do trabalho levará em consideração o resultado da demonstração.