

Trabalho de Programação 2

Processador CESAR16i – Parte 1 – Programa Principal

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 reais, o programa principal deverá chamar “funções do sistema” para interagir com os periféricos (teclado, visor e timer) do Cesar sem necessidade de conhecer os detalhes de funcionamento destes periféricos e do sistema de interrupções.

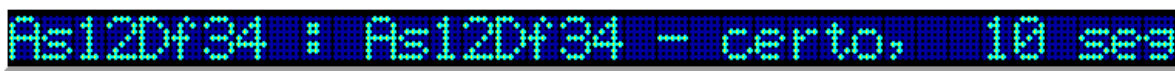
Assim, a primeira parte do trabalho consistirá em implementar o programa principal e valerá 50% da nota. A segunda parte consistirá em implementar as funções do sistema, a rotina de inicialização do sistema de interrupções e os tratadores de interrupções do teclado e do timer e também valerá 50% da nota.

Inicialmente, o programa principal 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:

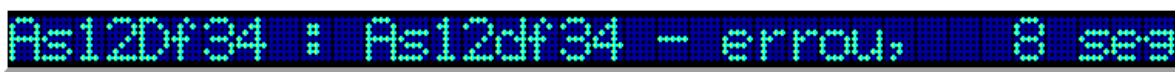


(1)

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:

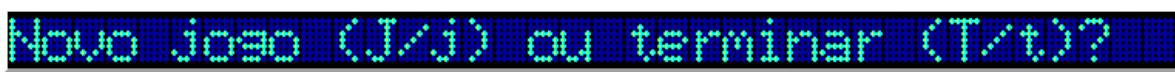


(2)



(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:



(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 principal. Para o desenvolvimento desta parte do trabalho, será colocado à disposição um código fonte que deverá ser usado como base para o desenvolvimento (REFERENCIA_PP.CED) e também um programa que deverá ser carregado no simulador antes de fazer a carga parcial da sua solução, para fins de teste, conforme instruções adiante (KERNEL_REF.MEM). Nesta primeira parte do trabalho, o programa de teste será responsável pela simulação do SISTEMA (que executa as funções e interage com os periféricos e o sistema de interrupções).

2. Implementação.

O programa principal deverá executar as etapas descritas a seguir. As funções do sistema referidas aqui (**KBHIT**, **GETCHAR**, **GETTIME**, **PUTCHAR** e **PUTMSG**) estão descritas no item 3 do enunciado. Considerar que a inicialização do sistema de interrupções e do apontador da pilha serão feitas pelo programa de teste que será colocado no Moodle.

1. Exibir no visor (usando a função do sistema **PUTMSG**) mensagens de identificação do autor (nome e número do cartão) e com instruções de uso (obrigatórias). Após exibir cada mensagem, usar a função do sistema **GETCHAR** para aguardar que seja digitado um caractere **ENTER** (código ASCII 13) antes de continuar, ignorando qualquer outro caractere digitado pelo usuário.
2. Perguntar ao usuário se está pronto para iniciar o jogo, aguardando **ENTER** para continuar.
3. Limpar o visor e exibir uma sequência de caracteres (selecionada de uma tabela – ver item 4 do enunciado) nas primeiras 8 posições, seguida de ' : ' e de um cursor para indicar onde será exibido o primeiro caractere digitado (ver mensagem 1, acima). Usar a função do sistema **GETTIME** para obter o tempo inicial.

Nesta mensagem, o caractere '_' indicará a posição do "cursor" que deverá ser administrada pelo programa para saber onde ecoar os caracteres lidos do teclado durante a leitura dos caracteres digitados. O programa deverá processar corretamente o caractere **BACKSPACE**, cuidando para não recuar sobre o texto inicial se for digitado um **BACKSPACE** quando nenhum outro caractere foi digitado.

Na leitura do que o usuário digitar (usando a função do sistema **GETCHAR**) deverão ser aceitos somente os caracteres previstos no enunciado (letras maiúsculas e minúsculas e dígitos de 0 a 9) e não deverão ser aceitos mais do que 8 caracteres, mas 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 fim da digitação. Qualquer outro caractere deverá ser ignorado (voltar a chamar **GETCHAR** para obter um novo caractere). Para "ecoar" no visor os caracteres digitados, deve ser usada a função do sistema **PUTCHAR**. Ao receber o **ENTER**, o programa deverá usar a função **GETTIME** para obter o tempo final da digitação. Depois de encerrada a digitação, os caracteres recebidos deverão ser comparados com os caracteres da sequência exibida pelo programa para verificar se a sequência digitada é igual à exibida pelo programa. Se for igual, o programa deverá exibir '**certo**' no visor, após a sequência digitada. Caso contrário, exibir '**errou**'. Em qualquer caso, exibir também o tempo, em segundos, decorrido entre a exibição da sequência pelo programa e a digitação do **ENTER** pelo usuário. O tempo deverá ser exibido com três dígitos, com eliminação de zeros não significativos. Ver mensagens 2 e 3 acima. Em seguida, usar a função do sistema **GETCHAR** para aguardar um **ENTER** (ignorar qualquer outro caractere) antes de prosseguir.

4. Perguntar ao usuário se deseja jogar novamente ou encerrar o programa (ver mensagem 4 acima), aceitando apenas os quatro caracteres especificados: 'J' ou 'j' para jogar de novo – 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. Funções do sistema que serão chamadas pelo programa

Todas as operações envolvendo acesso aos periféricos do Cesar (teclado, visor e *timer*) serão executadas através de chamadas para funções do "sistema", sendo vedado o acesso direto aos periféricos pelo programa principal. Para executar as funções do sistema serão utilizadas chamadas para uma sub-rotina (**JSR R7, _SISTEMA**), com passagem de parâmetros específicos para cada função, conforme mostrado na tabela a seguir. O endereço de **_SISTEMA** estará definido como uma constante no programa fonte que deverá ser usado como base para o desenvolvimento da solução (**REFERENCIA_PP.CED**).

Função	Descrição	Código (R5)	Parâmetros
KBHIT	Devolve 0 no registrador R0 se nenhuma tecla foi pressionada e outro valor em caso contrário. Retorna ao programa chamador sem esperar digitação.	0	Devolve: R0 = 0 ou diferente de 0
GETCHAR	Obtém o código ASCII de um caractere digitado no teclado e devolve o mesmo no registrador R0. Não "ecoar" o caractere lido no visor. Enquanto não for digitado nenhum caractere, não retorna ao programa chamador.	1	Devolve: R0 = código ASCII do caractere digitado
GETTIME	Obtém no R0 o contador de tempo decorrido mantido pelo sistema. Este contador é zerado na inicialização do sistema de interrupções, incrementado a cada segundo a partir daí, e volta a 0 quando passados 65535 segundos.	2	Devolve: R0 = valor do contador de tempo
PUTCHAR	Escreve um caractere no visor, na posição informada pelo programa chamador através do registrador R1. O código ASCII do caractere a ser escrito deve estar nos bits menos significativos do R0. Os bits 8 a 15 do R0 são ignorados.	3	R0 = código ASCII do caractere a escrever no visor (bits 0 a 7) R1 = número da posição do visor (0 a 35) onde escrever o caractere
PUTMSG	Escreve no visor uma mensagem formada por uma sequência de caracteres ASCII terminada por um byte com valor zero (formato ASCIIZ), a partir da posição no visor informada pelo programa chamador através do registrador R1. Se o espaço até o fim do visor for menor do que o tamanho da mensagem, continua escrevendo a partir do início do visor até terminar a mensagem.	4	R0 = endereço da mensagem a escrever no visor R1 = número da posição do visor (0 a 35) onde deve começar a escrever a mensagem

4. Tabela de sequências de caracteres a exibir

O programa deverá usar uma tabela com 16 sequências de caracteres com 8 caracteres cada uma para selecionar a sequência que será exibida no visor na etapa 3 (ver item 2 - Implementação). Para escolher a sequência a ser exibida a cada jogada, deverá ser usada uma chamada da função **GETTIME** e os 4 bits menos significativos do valor devolvido por ela no R0 deverão ser usados

como índice (0 a 15) para buscar uma das sequências nesta tabela. Para tornar mais interessante o jogo, sugere-se mesclar sequências simples e mais complexas, como no exemplo abaixo (formato Daedalus):

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

5. Arquivo base (REFERENCIA_PP.CED) e entrega do trabalho

Para a realização do trabalho você receberá um arquivo chamado REFERENCIA_PP.CED, que deverá utilizar como base para o seu desenvolvimento. Nesse arquivo estarão definidos os endereços que deverão ser usados na comunicação entre o programa principal e o “núcleo do sistema” (KERNEL_REF.MEM).

A entrega do trabalho será realizada em DUAS PARTES, cada uma em uma data diferente. Na primeira parte deverá ser entregue **apenas** o programa principal.

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

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

Faixa de Endereços em decimal (e hexadecimal)	Descrição
0 a 255 (0000 ₁₆ a 00FF ₁₆)	Área para inicialização do sistema. Esta área não deve ser acessada nem modificada pelo programa principal. Ela será carregada a partir do código de teste do programa principal (KERNEL_REF.MEM).
256 a 32767 (0100 ₁₆ a 7FFF ₁₆)	Área para o programa principal a ser desenvolvido nesta parte do trabalho. Ele será carregado usando uma carga parcial dos endereços 256 a 32767 a partir do arquivo contendo sua solução.
32768 a 65535 (8000 ₁₆ a FFFF ₁₆)	Área para colocar a rotina de inicialização do sistema de interrupções, a implementação das funções do sistema, os tratadores de interrupções do teclado e do timer (que serão implementados na segunda parte do trabalho) e a pilha. Ela será carregada a partir do código de teste do programa principal (KERNEL_REF.MEM).

7. 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 256 a 32767 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.

8. 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.