

Trabalho de Programação 2

Processador CESAR16i – Parte 1 – Programa de aplicação

1. Descrição Geral

Sua missão nesse trabalho é controlar a posição de uma câmera, capaz de capturar imagens dos ladrilhos de uma parede, de maneira a totalizar os ladrilhos, conforme seu estado: em boas condições ou ladrilhos defeituosos e os locais onde não existem ladrilhos.

Para controlar a posição da câmera, você deve enviar comandos de movimento. Mas, a câmera só informa a situação dos ladrilhos quando tiver parado de se mover. Portanto, antes de obter as informações dos ladrilhos que estão sendo visualizados pela câmera, você deve aguardar que encerre o movimento da mesma.

O seu programa deve ser escrito para o processador CESAR16i, e deve seguir o diagrama abaixo:

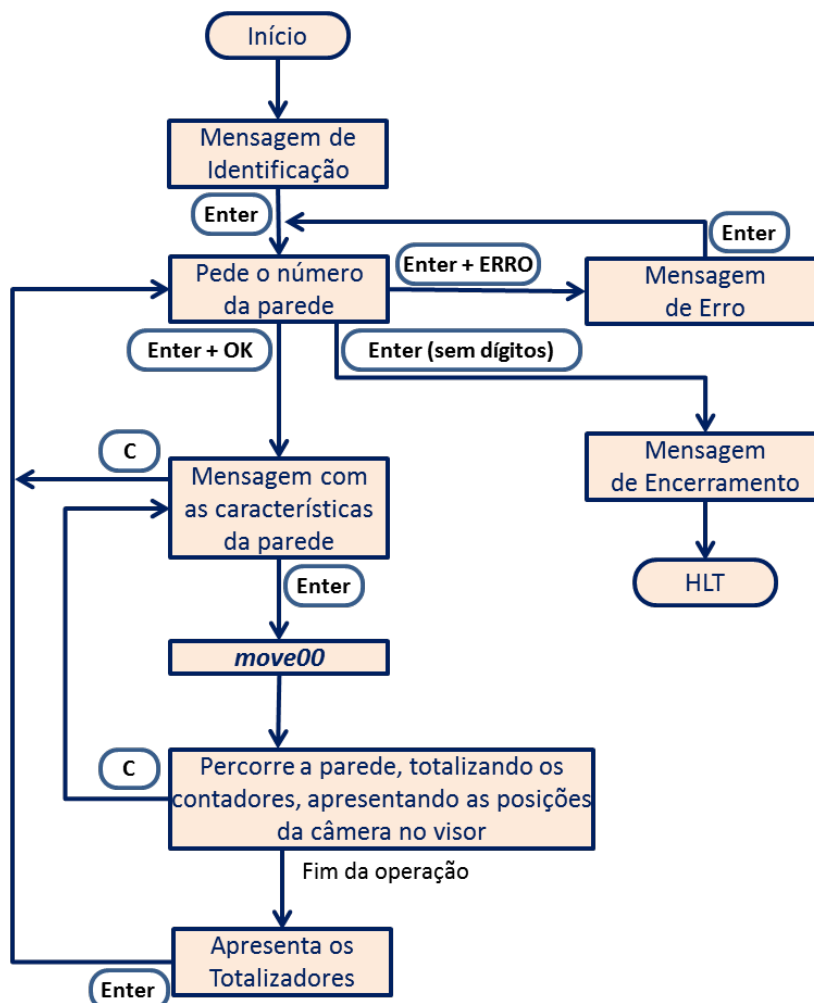


Figura 1. – Diagrama Geral da Implementação

Seu programa inicia em “Início” e a primeira ação a ser realizada é colocar no visor uma “Mensagem de Identificação”, semelhante a esta:

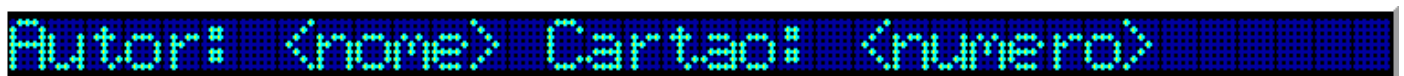


Figura 2. – Mensagem de Identificação

Depois disso, o programa deve aguardar que o usuário digite “Enter”, para passar à etapa onde o programa “Pede o número da parede”. Nessa etapa o programa deverá solicitar que o usuário informe o número de identificação da parede, que terá seus ladrilhos contados, conforme figura abaixo.

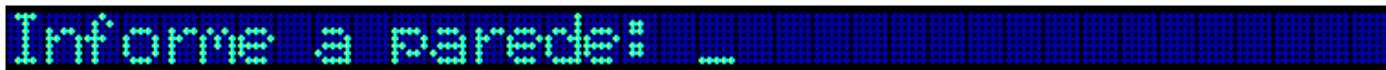


Figura 3. – Mensagem para pedir o número de identificação da parede

Após digitar o número de identificação da parede, o usuário deve digitar “Enter”. Se for digitado apenas “Enter”, sem nenhum número de parede, o programa deverá terminar a execução, exibindo uma mensagem de encerramento e parar. Caso o número da parede seja inválido, o programa deve apresentar uma “Mensagem de Erro”, e aguardar que o usuário digite “Enter” para retornar à etapa onde será pedido o número da parede. Abaixo está um exemplo dessa mensagem de erro:



Figura 4. – Mensagem de erro

Se o número da parede for válido, o programa deve seguir para a “Mensagem com as características da parede”. As informações a serem apresentadas no visor, nessa etapa, são o número de identificação da parede e as dimensões da mesma: largura e altura. Portanto, todas as janelas são retangulares. Abaixo está um exemplo de mensagem a ser colocada no visor:

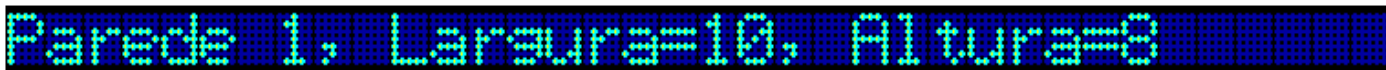


Figura 5. – Mensagem com as características da parede

Então, depois de colocar as informações da parede no visor, o programa aguarda o usuário digitar algo. Caso seja digitado “C” ou “c” (cancelar), o programa deve retornar à etapa que pede o número da parede. Se for digitado “Enter”, o programa segue para a etapa “move00”. Qualquer outra tecla é ignorada.

Na etapa de “move00”, seu programa deve comandar a câmera para que seja posicionada nas coordenadas (0,0) da parede. E exibir as coordenadas e o estado do ladrilho que ficou no centro da câmera, como mostrado na figura abaixo:



Figura 6. – Mensagem com o estado do ladrilho no canto superior esquerdo da parede

A partir daí, seu programa deve “Percorrer a parede, totalizando os contadores, e apresentando as posições da câmera no visor”. Em cada posição que a câmera parar, o programa deve apresentar no visor as informações das coordenadas onde parou e o estado do ladrilho nessas coordenadas (ladrilho no centro da câmera), conforme exemplos nas figuras abaixo:



Figura 7. – Mensagens com o estado do ladrilho no centro da câmera

Durante essa etapa, se o usuário digitar “C” ou “c” (cancelar), a operação de percorrer a parede deve ser encerrada e o programa deve seguir para a etapa onde é pedido um número de parede. Quando a parede for completamente percorrida, chegando ao “Fim da operação”, o programa deve seguir para a atividade onde ele “Apresenta os Totalizadores”, exibindo uma mensagem como a mostrada a seguir:

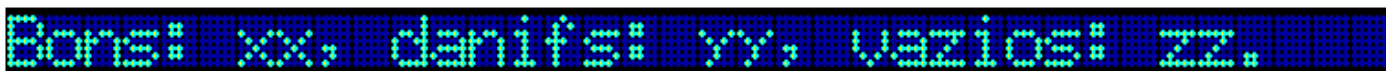


Figura 8. – Mensagem com os totalizadores

O programa permanece apresentando os totalizadores até que o usuário digite “Enter”, quando vai seguir para a etapa em que o programa pede um (novo) número de parede.

2. Implementação.

Para implementar o programa descrito na sessão anterior, você deverá utilizar um conjunto de funções que possibilitarão ler teclado, colocar mensagens e caracteres no visor, controlar a movimentação da câmera e ler as informações de estado dos ladrilhos.

As funções que você terá disponíveis para implementar o seu programa estão descritas na Tabela de Funções abaixo.

Função	Código	Descrição
Funções para leitura do teclado		
<u>kbhit</u>	0	Informa se foi digitada alguma tecla. Não bloqueia. Retorna R0=0, se nada foi digitado. Retorna R0<>0, se algo foi digitado.
<u>getchar</u>	1	Bloqueia, esperando que seja digitada uma tecla. Não coloca o caractere no visor. Retorna R0=caractere ASCII da tecla digitada.
Funções para colocar caracteres no visor		
<u>putchar</u>	2	Coloca um caractere no visor. Entra R0=caractere ASCII a ser colocado no visor O caractere será colocado na posição atual do cursor, o qual será incrementado. Ao receber um caractere BS – <i>Back Space</i> (8), o cursor é movido uma posição para a esquerda.
<u>putmsg</u>	3	Coloca um string no visor. Entra R0=ponteiro para o string. O string será colocado a partir da posição atual do cursor. O string é formado por bytes ASCII, terminado por “\0” (0).
<u>setcursor</u>	4	Move o cursor para a posição desejada. Entra R0=nova posição para o cursor. Esse valor deve estar entre 0 e 35. Outros valores farão com que o cursos desapareça do visor.
Função para obter informações da parede e dos ladrilhos		
<u>setwall</u>	5	Seleciona a parede de trabalho Entra R0=número da parede. Retorna R0=largura da parede (abscissas – coordenadas “X”) e R1=altura da parede (ordenadas – coordenadas “Y”). Se o número fornecido for inválido, retorna -1 (FFFF ₁₆) em R0 e R1.
<u>read</u>	6	Solicita informação sobre o estado de um ladrilho. Entra em R0 (coordenada “X”) e R1 (coordenada “Y”) as coordenadas do ladrilho, em relação à posição atual da câmera. Esses valores podem ser +1, -1 ou 0. Retorna em R0 o estado do ladrilho. Esses valores podem ser “0” (ladrilho não usado), “1” (ladrilho em boas condições) ou “2” (ladrilho danificado). Caso seja solicitada uma coordenada que esteja fora da parede, será retornado “3”.
Funções para controlar o movimento da câmera		
<u>move00</u>	7	Move a câmera para as coordenadas (0,0) da parede.
<u>move</u>	8	Move a câmera. Uma vez chamada a função, deve-se aguardar o encerramento do comando antes de enviar novo comando de movimentação. Entra em R0 (coordenada “X”) e R1 (coordenada “Y”) a distância, em ladrilhos, a ser deslocada a câmera. Esses valores podem ser +2, -2, +1, -1 ou 0. Outros valores resultarão no encerramento da função, com erro (R0 = -1). O deslocamento da câmera ocorre na horizontal ou na vertical (nunca na diagonal). O deslocamento da câmera demora $T = 100ms + (NL-1) \times 50ms$, onde NL é o número de ladrilhos total do movimento comandado.
<u>moving</u>	9	Informa se a câmera está em movimento. Retorna R0 = 0, se a câmera está parada. Retorna R0<>0, se a câmera está em movimento.
<u>getcamera</u>	10	Informa as coordenadas atuais da câmera. Retorna R0=coordenada “X” e R1=coordenada “Y” da câmera. Essa informação só é válida se a câmera está parada. Se a câmera estiver em movimento, será retornado “-1” em R0 e R1.

Para chamar essas funções, você deverá utilizar o seguinte trecho de código:

```
MOV      #CódigoDaFuncao, R5
JSR      R7, _SISTEMA
```

A primeira instrução visa colocar no registrador R5 o código da função a ser chamada. O código de cada uma das funções disponíveis está descrito na coluna “Código” da Tabela de Funções.

Na segunda instrução está a chamada de uma subrotina, que deve ser chamada com o registrador R7. O endereço representado pelo símbolo `_SISTEMA` é definido pela implementação do kernel, e será o valor `H0080` (`008016` ou `12810`).

Além do registrador R5, devem ser passados parâmetros específicos para cada função. Portanto, esses parâmetros serão passados através de outros registradores do CESAR16i, conforme descrição das mesmas.

3. Local na memória para o programa

Seu programa deve iniciar a partir do endereço `010016` (256) e pode estender-se até o endereço `3FFF16` (16.383).

Você não necessita programar interrupções ou o stack pointer. Essas operações já foram realizadas pelo kernel fornecido pelo professor.

Para testar seu programa, você deve carregar o kernel em primeiro lugar e, depois, carregar seu programa usando uma carga parcial do arquivo, para os endereços `010016` (256) até `3FFF16` (16.383).

4. Entregáveis

Deve ser entregue um arquivo fonte (arquivo `.CED`) com a implementação de sua 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.

5. 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.