Kernel

Este documento descreve as estruturas a serem gerenciadas pelo Kernel assim como as funções da API de acesso ao Kernel, que deverão ser implementadas no trabalho com o Computador CESAR16i.

Gerência do Visor

Seu kernel vai receber caracteres através das funções "_putchar" e "_putmsg", e terá como responsabilidade coloca-los, corretamente, na posição do visor indicada pelo cursor. Após cada caractere colocado no visor, seu kernel deve fazer o cursor deve "andar" uma posição para a direita.

Caso a posição escrita seja a última posição mais a direita do visor, o visor deverá ser "limpo" e o cursor deverá ser posicionado na posição mais à esquerda do visor (primeira posição do visor).

Além dos caracteres visíveis (números, letras e alguns símbolos), seu kernel também deverá ser capaz de tratar (realizar as tarefas características) alguns caracteres de controle. Esses caracteres são o "CR", "LF" e "BS" (ver a funções "_putchar" e "_putmsg" a seguir).

Gerência do Teclado

Seu kernel deverá ser capaz de informar para a aplicação da eventual existência de teclas digitadas. Para isso, seu kernel deve implementar as funções "_kbhit" e "_getchar". (ver detalhes a seguir). Notar que a função "_getchar" não coloca os caracteres digitados no visor. A única funcionalidade do "_getchar" é devolver para a aplicação os caracteres digitados.

Gerência do Relógio

O kernel desenvolvido deverá dar suporte para um relógio. Esse relógio deverá ser formado por HORA, MINUTO e SEGUNDO. Seu kernel deverá fornecer duas funções para gerenciar o relógio: a função "_getclock", através da qual o kernel entrega a informação do relógio para a aplicação; e "_setclock", através da qual a aplicação solicita ao kernel que o relógio seja alterado.

Gerência do Alarme

Além das funções de leitura e escrita no relógio, o kernel também deverá fornecer uma função que permita a programação de um ALARME. Para isso, o kernel deve implementar a função "_setalarm", através da qual a aplicação vai informar o horário no qual o kernel deve informar a aplicação e o endereço da rotina de "callback".

Quando o relógio do kernel atingir o horário programado pela "_setalarm", o kernel deverá realizar o "callback" para o endereço programado pela aplicação. Para isso, o kernel deverá realizar uma chamada de subrotina para o endereço programado pela "_setalarm", da seguinte forma:

JSR R7,<endereço-programado>

Muito importante:

- O kernel deve realizar um único "callback" para cada chamada da função "_setalarm". Para que o kernel faça um novo "callback" a aplicação deve chamar, novamente, a função "_setalarm".
- Notar, também, que a própria função chamada no "callback" pode realizar qualquer chamada de função da API, incluindo a própria "_setalarm".

Funções do Kernel

A seguir são descritas as funções da API e a forma como devem ser chamadas pelos programas de aplicação, incluindo os parâmetros de entrada e valores resultantes de saída.

Essas funções permitem que o programa de aplicação possa utilizar os periféricos disponíveis no CESAR16i (teclado, visor e *timer*), sem a necessidade de conhecer o funcionamento do hardware e das portas de acesso a esses periféricos.

As funções a serem implementadas deverão ser colocadas na área de memória reservada para o kernel.

Iniciando no endereço H0100, você deve definir a Tabela de Vetores do Kernel. Cada vetor (ponteiro, com 2 bytes) dessa tabela deve conter o endereço, no kernel, onde inicia a implementação da função correspondente ao vetor.

Observar que a ordem desses vetores deve ser, rigorosamente, obedecida. Os vetores e suas funções correspondentes estão indicados abaixo:

Vetor	Função
[0]	_kbhit
[1]	_getchar
[2]	_putchar
[3]	_putmsg
[4]	_getclock
[5]	_setclock
[6]	_setalarm

Abaixo você encontra a descrição das funções a serem implementadas.

0. Função: "_kbhit"

Função através da qual a aplicação solicita ao kernel a informação da existência de alguma tecla digitada. A função deve retornar com a informação da existência de tecla. Essa função retorna imediatamente, sem aguardar pela digitação de qualquer tecla.

- *Parâmetros de entrada*: nenhum.
- *Parâmetro de saída*: registrador R0, com a informação da existência de tecla.

A função retorna no registrador R0 a informação se existe tecla ou não.

- Se há tecla, o valor em R0 será zero;
- Se não há tecla, o valor em R0 será um valor qualquer diferente de zero.

1. Função: "_getchar"

Função através da qual a aplicação solicita ao kernel que informe a tecla digitada ou, se não houver tecla digitada, que aguarde pela digitação de uma. A função deve retornar o código ASCII da tecla digitada. Portanto, se não houver tecla digitada no teclado, a função deve aguardar pela digitação de uma tecla.

- *Parâmetros de entrada*: nenhum.
- *Parâmetro de saída*: registrador R0, com a tecla digitada.

A função só retorna (só termina) se houver uma tecla já digitada ou quando o usuário digitar alguma tecla. O código ASCII da tecla digitada deve ser retornado no registrador R0.

Sempre que a função "_getchar" for chamada e estiver bloqueada aguardando por uma tecla, a posição do cursor (ver detalhes na função "_putchar" abaixo) deve ser apresentado no visor através do símbolo "_" (*underscore*). Esse símbolo deve ser alternado com o caractere que estiver sendo apresentado nessa posição. Essa alternância deve ter uma periodicidade tal que cada símbolo permaneça no visor por 250ms.

2. Função: "_putchar"

A aplicação usa essa função para solicitar que seja colocado no visor um caractere. Esse caractere pode ser um caractere visível (que tenha representação simbólica) ou um caractere de controle.

- *Parâmetros de entrada*: registrador R5, com o caractere a ser colocado no visor, codificado em ASCII.
- *Parâmetro de saída*: registrador R0, com o código de erro de retorno.

O código no registrador R5 pode representar caracteres visíveis ou caracteres de controle.

Os **caracteres visíveis** são aqueles cujos códigos iniciam em H20 (SPACE) e vão até H7A ("z"). Ao receber um desses caracteres, o kernel deve coloca-lo no visor, na posição indicada pelo cursor.

O cursor, cujo controle é do kernel, indica a posição onde deve ser escrito o caractere recebido.

Após escrever um caractere na posição indicada pelo cursor, o kernel deverá incrementar a posição do cursor, preparando-o para a próxima escrita no visor. Caso a escrita tenha ocorrido na última posição do visor, o kernel deverá limpar o cursor (colocar espaços em branco em todo o cursor) e posicionar o cursor na primeira posição mais à esquerda do visor.

A tabela ASCII tem 32 **caracteres de controle**. Entretanto, nessa implementação do kernel serão usados apena três deles. Os códigos dos caracteres de controle restantes devem ser ignorados. Os caracteres de controle a serem processados são os seguintes:

- CR (H0D) – *Carriage Return*: move o cursor para a primeira posição mais a esquerda do visor.
- LF (H0A) – Line Feed: limpa o visor (coloca espaços em branco em todo o visor) e mantém o cursor na posição que estava.
- BS (H08) – *Back Space*: move o cursor uma posição para a esquerda. Caso o cursor esteja no início do visor (posição mais a esquerda), esse caractere de controle deve ser ignorado.

Na inicialização do kernel, o visor deve ser limpo (apagado) e o cursor deve ser posicionado no início do visor.

A função retorna no registrador R0 um código de erro.

- Se não houve erro, o valor em R0 será zero;
- Se houve algum erro na execução da função ou informação inválida nos parâmetros de entrada, o valor em R0 será um valor qualquer diferente de zero.

3. Função: "_putmsg"

A aplicação usa essa função para solicitar que seja colocado no visor um string de caracteres (bytes) terminado por um byte H00 (o mesmo delimitador usado em string "C"). Os caracteres ASCII (visíveis e de controle) que formam o string devem ser tratados conforme definido na função "_putchar".

- *Parâmetros de entrada*: registrador R5, com o endereço de memória onde inicia o string.
- *Parâmetro de saída*: registrador R0, com o código de erro de retorno.

A função retorna no registrador R0 um código de erro.

- Se não houve erro, o valor em R0 será zero;
- Se houve algum erro na execução da função ou informação inválida nos parâmetros de entrada, o valor em R0 será um valor qualquer diferente de zero.

4. Função: "_getclock"

A aplicação vai chamar essa função sempre que desejar que o kernel forneça a informação do relógio.

- *Parâmetros de entrada*: nenhum.
- *Parâmetro de saída*: valor atual do relógio. R2 com HORA, R1 com MINUTO e R0 com SEGUNDO.

5. Função: "_setclock"

Essa função vai permitir que a aplicação altere (ajuste) o valor atual do relógio.

- *Parâmetros de entrada*: valor a ser escrito no relógio: R5 com HORA, R4 com MINUTO e R3 com SEGUNDO.
- *Parâmetro de saída*: registrador R0, com o código de erro de retorno.

A função retorna no registrador R0 um código de erro.

- Se não houve erro, o valor em R0 será zero;
- Se houve algum erro na execução da função ou informação inválida nos parâmetros de entrada, o valor em R0 será um valor qualquer diferente de zero.

6. Função: "_setalarm"

Essa função vai permitir que a aplicação altere (ajuste) o valor do ALARME.

- *Parâmetros de entrada*:
  - endereço de callback: R5, com o endereço da rotina de callback.
  - valor a ser escrito no relógio: R4, com HORA; R3, com MINUTO; e R2, com SEGUNDO.
- *Parâmetro de saída*: registrador R0, com o código de erro de retorno.

A função retorna no registrador R0 um código de erro.

- Se não houve erro, o valor em R0 será zero;
- Se houve algum erro na execução da função ou informação inválida nos parâmetros de entrada, o valor em R0 será um valor qualquer diferente de zero.