

## Trabalho de Programação 2 – PARTE 2

### Processador CESAR16i

#### 1. Descrição Geral

Sua missão nesta segunda etapa do trabalho é implementar o “programa principal” que simulará um cronômetro, usando o sistema de interrupções do Cesar16i e a ISR desenvolvida na parte 1 para tratar as interrupções do timer e do teclado. O programa inicializará o cronômetro com uma hora fornecida pelo usuário e depois irá medir intervalos de tempo e exibir informações no visor. O programa principal valerá 60% da nota do trabalho.

Para o desenvolvimento desta parte do trabalho será colocado à disposição um arquivo APP\_REF.CED, que deverá ser usado como base para o desenvolvimento da sua solução, a exemplo do que foi feito na parte 1.

O programa principal usará as funções implementadas na ISR (desviando para `_RESET` para inicializar o sistema de interrupções e chamando as funções `_SET_TIME` e `_GET_TIME`, através de `_ROTINAS`). É importante notar que o programa principal não deverá acessar diretamente o `TIMER` nem o `TECLADO`. Ele apenas se comunicará com as rotinas de atendimentos de interrupções através das variáveis que armazenam a hora do dia (atualizada pela ISR a cada segundo) e o código ASCII da tecla digitada (`TECLA`). Para escrever nos LEDs do visor (ou ler o valor que está armazenado neles, se for o caso), ele acessará diretamente os endereços 65500 a 65535.

#### 2. Implementação

A função do **programa principal** é solicitar ao usuário uma hora inicial e, a partir daí, exibir a hora no visor e aguardar que o usuário pressione teclas que comandarão o cronômetro, para medir intervalos de tempo decorridos, e exibir as informações de início, fim e duração do intervalo medido no visor (medirá intervalos de até 999 segundos).

Ao iniciar, após a inicialização do sistema de interrupções (com um `JMP _RESET`), o programa deverá exibir mensagens de identificação do autor (nome e número do cartão) e com instruções de uso. Após exibir cada mensagem, deverá aguardar que seja digitado um `ENTER` antes de continuar.

A seguir, para solicitar a hora inicial, o programa deverá exibir uma mensagem como a mostrada a seguir:



Uma vez exibida a mensagem, devem ser lidas do teclado as informações de hora e minuto (2 dígitos para cada uma, sem espaços entre eles). Para isto, o programa principal deverá escrever no visor uma mensagem solicitando esta informação. Por exemplo:



Na leitura da hora inicial deve ser usado um cursor (caractere ‘\_’) para indicar a posição onde será exibida a próxima informação digitada e também deverá ser processado corretamente o uso da tecla `BACKSPACE`, permitindo que sejam digitados apenas 4 dígitos, a partir do LED 29 (no exemplo acima). O programa principal deverá ignorar qualquer caractere digitado que não seja um dígito decimal (0 a 9), `BACKSPACE` ou `ENTER`.

A digitação da hora inicial será encerrada por um `ENTER` e o programa deverá verificar se a hora é válida ( $00 \leq HH \leq 23$  e  $00 \leq MM \leq 59$ ). Se a hora informada não for válida, o programa deverá exibir uma mensagem de erro, identificando se o erro está na hora (HH) ou no minuto (MM) informado, esperar que no usuário digite um `ENTER` e voltar a exibir a mensagem de solicitação da hora inicial.

Após receber uma hora inicial correta, o programa deverá chamar a sub-rotina `_SET_TIME`, que faz parte do sistema desenvolvido na parte 1 do trabalho, para armazenar a hora inicial fornecida nas variáveis `HORA` e `MINUTO` e colocar 0 na variável `SEGUNDO`. A partir daí, a ISR irá atualizar a hora armazenada nestas variáveis a cada segundo.

Em seguida, o programa principal deverá exibir a hora do dia no visor no seguinte formato:



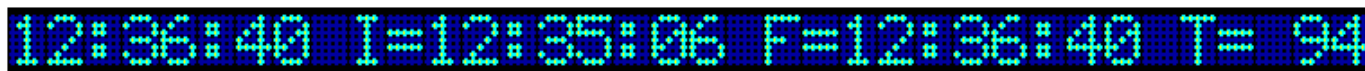
Os campos indicados por “I=”, “F=” e “T=” ficarão inicialmente em branco e serão usados durante a cronometragem para indicar a hora de início da cronometragem a hora de fim e o tempo cronometrado, respectivamente.

Feito isto, o programa começará a operar no modo cronômetro, no qual o usuário informará, através do teclado, o que fazer:

Se for digitado 'I' ou 'i', iniciar a cronometragem de um intervalo, exibindo no visor a hora de início:



Se for digitado 'F' ou 'f', encerrar a cronometragem de um intervalo e exibir a hora final e o tempo decorrido, em segundos:



O campo reservado para exibição do tempo decorrido em segundos tem apenas 3 dígitos, o que permite representar valores até 999 segundos. Caso o tempo decorrido seja maior do que este limite, o programa deverá exibir somente os três dígitos menos significativos do tempo decorrido e substituir o sinal "=" após a letra "T" por um "\*" (que indicará estouro):

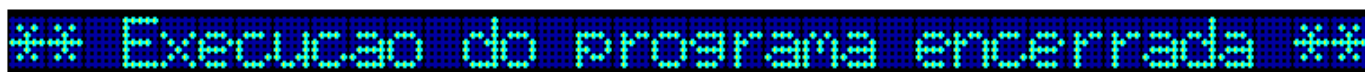


Durante a cronometragem, além das teclas 'I/i' e 'F/f', o programa deverá aceitar:

- 'Z/z' para limpar o visor (voltando a exibir somente a hora atualizada e os identificadores dos campos) e aguardar nova cronometragem:



- ESCAPE (código ASCII 27) para terminar a execução do programa, limpando o visor e exibindo uma mensagem de encerramento; por exemplo:



O programa deverá verificar as seguintes condições:

- aceitar 'I/i' somente se estiver no estado "cronômetro parado", ou seja, logo após ter lido e exibido a hora inicial ou logo após ter recebido um 'Z/z'. Opcionalmente, poderá aceitar 'I/i' após terminar a cronometragem sem ter recebido um 'Z/z' antes. Mas, neste caso, lembrar-se de voltar a exibir os campos 'F' e 'T' em branco.
- aceitar 'F/f' somente se estiver no estado "cronometrando", ou seja, após ter recebido um 'I/i'.

Qualquer outra tecla digitada deve ser ignorada pelo programa principal. Após processar uma tecla lida, seja qual for, colocar imediatamente um valor negativo na variável TECLA.

### 3. Arquivo base (APP\_REF.CED) e entrega do trabalho

Para a realização do trabalho você receberá um arquivo chamado "APP\_REF.CED", que deverá utilizar como base para o seu desenvolvimento. Nesse arquivo estão declaradas todas as áreas de memória, os endereços de acesso aos periféricos e as variáveis usadas na comunicação entre os tratadores de interrupção e o programa principal.

Nesta segunda parte do trabalho deverá ser entregue **apenas** o arquivo fonte do programa principal (aplicação). A correção será feita através da carga do sistema de interrupção do professor, seguida da carga parcial do programa principal (aplicação), cujo fonte vocês devem entregar.

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

Assim como na parte 1 do trabalho, o espaço de endereçamento do CESAR estará dividido da seguinte forma:

| Nome       | Faixa de Endereços em decimal (e hexadecimal)            | Descrição  |
|------------|--|--|
| <b>APP</b> | 0 a 32767 (0000 <sub>16</sub> a 7FFF <sub>16</sub> )     | Área para colocar o PROGRAMA PRINCIPAL (PP). Aqui devem ser colocadas as instruções e os dados locais do programa principal.                               |
| <b>AVR</b> | 32768 a 33023 (8000 <sub>16</sub> a 80FF <sub>16</sub> ) | Área para colocar as VARIÁVEIS DE COMUNICAÇÃO entre o programa principal e os tratadores de interrupção.   |
| <b>ARI</b> | 33024 a 33279 (8100 <sub>16</sub> a 81FF <sub>16</sub> ) | Área para colocar a rotina de inicialização do sistema de interrupções. Essa rotina deve iniciar no endereço 33024 (8100 <sub>16</sub> ). Essa rotina deve |

|            |   |   |
|------------|---|---|
|            |   | ser chamada no programa principal, ANTES DE HABILITAR AS INTERRUPÇÕES.  |
| <b>ATI</b> | 33792 a 65407<br>(8400 <sub>16</sub> a FF7F <sub>16</sub> ) | Área para colocar os TRATADORES DE INTERRUPÇÃO de tempo e de teclado. Aqui devem ser colocadas as rotinas e variáveis usadas pelos tratadores de interrupção. |
| <b>APR</b> | 65408 a 65535<br>(FF80 <sub>16</sub> a FFFF <sub>16</sub> ) | Área reservada e área de mapeamento dos periféricos.  |

A definição dessas áreas está pronta no arquivo “APP\_REF.CED” fornecido. **O programa principal não deverá ler ou escrever diretamente na região da memória entre os endereços 32768 e 65499**, pois o acesso a estes endereços será feito pela ISR e suas funções auxiliares. A exceção é apenas a variável TECLA.

## 5. Variáveis de Comunicação

A área de memória **AVR** é reservada para as variáveis que serão usadas na troca de informações entre o Programa Principal e os Tratadores de Interrupção. As variáveis e estruturas definidas para essa comunicação estão declaradas no arquivo APP\_REF.CED que será fornecido e estão descritas a seguir.

**HORA** – uma palavra de 16 bits a partir do endereço 32768 (8000<sub>16</sub>), que armazena o número de horas da hora do dia.

**MINUTO** – uma palavra de 16 bits a partir do endereço 32770 (8002<sub>16</sub>), que armazena o número de minutos da hora do dia.

**SEGUNDO** – uma palavra de 16 bits a partir do endereço 32772 (8004<sub>16</sub>), que armazena o número de segundos da hora do dia.

As três variáveis que compõem a hora do dia serão inicializadas pelo programa principal chamando a sub-rotina \_SET\_TIME (para escrever) e \_GET\_TIME (para ler). As variáveis serão incrementadas a cada segundo pelo tratador de interrupções do timer, usando o formato de 24 horas. **O programa principal NÃO DEVE ler ou escrever diretamente nestas três variáveis.**

**TECLA** – uma palavra de 16 bits a partir do endereço 32774 (8006<sub>16</sub>). O tratador de interrupções do teclado recebe o código ASCII das teclas digitadas e coloca o código nessa variável somente quando ela contiver um valor negativo. O programa principal receberá os comandos do usuário através dessa variável. Se a variável TECLA contiver um valor  $\geq 0$ , o tratador de interrupções do teclado não colocará um novo valor na mesma e a última tecla digitada será perdida. **Importante: assim que o programa principal processar o valor armazenado na variável TECLA, ele deve colocar na mesma um valor negativo, para permitir que novas teclas sejam lidas pela ISR.**

### 5.1. Uso das variáveis HORA, MINUTO e SEGUNDO

Estas variáveis têm por função armazenar a hora do dia.

O **tratador de interrupções** do timer tem por tarefa incrementar essa variável a cada segundo. Isso foi feito na primeira parte do trabalho.

### 5.2. Uso da variável “TECLA”

A variável TECLA tem por função permitir a passagem dos códigos ASCII das teclas digitadas do tratador de interrupções de teclado para o programa principal. Sempre que essa tecla tiver um valor negativo, significa que ela não está armazenando nenhum código de tecla válido; sempre que esse valor for zero ou positivo, ela conterà o código ASCII da última tecla digitada.

O **tratador de interrupções** do teclado é acionado sempre que o usuário digitar algo. Então, se o valor de TECLA for negativo, o código da nova tecla digitada deve ser armazenado em TECLA. A restrição de só passar um novo código se TECLA for negativo garante que o programa principal tenha tempo de processar a última tecla digitada antes de receber uma nova tecla.

**O programa principal** saberá que nada foi digitado enquanto a variável TECLA contiver um valor negativo. Quando o valor dessa tecla for positivo, significa que um caractere foi digitado e deve ser processado. Após ter processado o valor da variável TECLA, o programa principal deverá armazenar um valor negativo nessa variável, para indicar ao tratador de interrupções que a tecla já foi lida.

## 6. Correção e Entregáveis

A correção da primeira parte do trabalho será feita através de um programa principal especialmente desenvolvido para esta finalidade. De forma semelhante, a correção da segunda parte (programa principal) será feita através de um tratador de interrupções padrão desenvolvido conforme especificado no enunciado da parte 1. Portanto, para que a correção seja possível, a leitura e escrita das variáveis de comunicação devem ser realizadas **exatamente** conforme especificado.

A primeira parte do trabalho terá **peso 4 sobre 10** enquanto que a segunda parte (programa principal) terá **peso 6 sobre 10**.

Cada parte do trabalho deverá ser entregue na entrada adequada do Moodle da turma. Deve ser entregue um arquivo fonte (arquivo .CED) com a solução correspondente, escrito em *assembly* do CESAR16i usando o montador Daedalus. Além disso, esse programa fonte deverá conter comentários descritivos da implementação. Sugere-se usar comandos da linguagem “C”.

Cada 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.

## 7. 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.