Aula prática N.º 2

Objetivos

- Utilizar o *core timer* do MIPS para gerar atrasos programáveis.

Introdução

O *core* MIPS disponível no microcontrolador PIC32 implementa, no coprocessador 0, um contador crescente de 32 bits (designado por *core timer*) atualizado a cada dois ciclos de relógio do CPU. Na placa DETPIC32 o relógio do CPU está configurado a 40 MHz, pelo que o contador é incrementado a uma frequência de relógio de 20 MHz. Isto significa que o tempo necessário para incrementar o contador desde o valor 0 até 20.000.000 é 1 segundo.

A placa DETPIC32 disponibiliza dois *system calls* para interagir com esse contador: ler o valor atual do contador (**`readCoreTimer()`**) e reiniciar a zero o seu valor (**`resetCoreTimer()`**).

Trabalho a realizar

Parte I

1. O programa seguinte incrementa o valor de uma variável e, de cada vez que a variável é atualizada, o seu valor é apresentado no ecrã do PC.

```
int main(void)
{
   int counter = 0;
   while(1)
   {
      resetCoreTimer();
      while(readCoreTimer() < 200000);
      printInt(counter++, 10 | 4 << 16);    // Ver nota[1]
      putChar('\r');            // cursor regressa ao inicio da linha
   }
   return 0;
}
```

   a) Qual a frequência de incremento da variável **`counter`**?

   b) Traduza o código C fornecido para *assembly* do MIPS, e teste-o na placa.

```
        .equ    READ_CORE_TIMER,11
        .equ    RESET_CORE_TIMER,?
        .equ    PUT_CHAR,?
        .equ    PRINT_INT,?
        .data
        .text
        .globl  main
main:   li      $t0,0                   # counter=0
while:                                  # while (1) {
        li      $v0,RESET_CORE_TIMER    #
        syscall                         #   resetCoreTimer()
        ...

        jr      $ra                     #
```

[1] O *system call* **`printInt`** permite formatar o resultado da impressão, através da parametrização do número mínimo de dígitos com que o valor é impresso. Essa configuração é feita nos 16 bits mais significativos do registo usado para determinar a base da representação. Por exemplo, para a impressão em decimal com 4 dígitos, o valor a colocar no registo **`$a1`** é **`0x0004000A`**, ou, em C, (**`10 | 4 << 16`**).

c) Altere sucessivamente o código que escreveu de forma a que a variável seja incrementada com uma frequência de 10 Hz, 5 HZ e de 1Hz. Teste e verifique o código para cada um desses casos.

2. O objetivo da função **delay()**, apresentada a seguir, é gerar um atraso temporal programável múltiplo de 1ms.

```
void delay(unsigned int ms)
{
   resetCoreTimer();
   while(readCoreTimer() < K * ms);
}
```

a) Determine o valor da constante "**K**", de modo a que para "**ms**" igual a 1 o atraso gerado seja de 1ms (note que **K=20*10⁶*t**, em que "**t**" é o valor do atraso que se pretende gerar, em segundos).

b) Com o valor de "**K**" que obteve na alínea anterior, calcule o valor máximo de atraso que é possível gerar com a função **delay()**.

Traduza para *assembly* do MIPS a função **delay()** e teste-a com diferentes valores de entrada (para o teste utilize como base o código C fornecido no ponto 1).

Notas:

- Para as operações de divisão devem, obrigatoriamente, ser usadas as instruções virtuais: "**div $Rdst,$Rsrc1,$Rsrc2**" ou "**rem $Rdst,$Rsrc1,$Rsrc2**".
- As funções devem obrigatoriamente ser colocadas (no segmento de código), após a função **main()**.
- Para valores de atraso que não sejam múltiplos de 1ms, a função **delay()**, tal como implementada anteriormente, não pode ser usada.

Parte II

1. Usando como base a função **delay()**, escreva um programa em linguagem C que incremente, em ciclo infinito, 3 variáveis inteiras: a variável **cnt1** deve ser incrementada a uma frequência de 1Hz, a variável **cnt5** deve ser incrementada a uma frequência de 5Hz, e a variável **cnt10** deve ser incrementada a uma frequência de 10Hz.

   O valor das 3 variáveis deve ser mostrado no ecrã, sempre na mesma linha, formatado em base 10 com 5 dígitos. Exemplo de visualização:

   00020 00100 00200

   Nota: para imprimir sempre na mesma linha deve começar por imprimir o caracter '**\r**', usando o *system call* **putChar()**.

2. Traduza o programa, que escreveu no ponto anterior, para *assembly* do MIPS e teste-o na placa.

3. Altere o programa de modo a que quando for premida a tecla **'A'**, a frequência de incremento dos contadores passe para o dobro, i.e., 2Hz, 10Hz e 20Hz. Premindo a tecla **'N'**, a frequência de incremento dos contadores deve voltar ao valor normal. Para a leitura do caracter utilize o *system call* **inkey()**.

4. Altere o programa que resultou do ponto anterior de modo a que quando for premida a tecla **'S'**, a contagem dos contadores seja suspensa e quando for premida a tecla **'R'** a contagem seja retomada. Para a leitura do caracter utilize o *system call* **inkey()**.

Exercícios adicionais

1. Considere agora a função **`timeDone()`** que se apresenta a seguir. Esta função permite verificar se já decorreu um determinado tempo (múltiplo de 1ms) desde a última vez que foi efetuado o seu *reset*. Caso o tempo não se tenha esgotado a função devolve o valor **`0`** (zero). Caso contrário devolve o número de milissegundos que decorreram desde o último *reset* ao *core timer*.

   Nota: Este código não pode ser usado juntamente com a função **`delay()`**, uma vez que ambas as funções efetuam, ou podem efetuar, *reset* ao *core timer*.

```
unsigned int timeDone(int ms, unsigned char reset)
{
   unsigned int curCount;
   unsigned int retValue = 0;

   if (reset > 0)
   {
      resetCoreTimer();
   }
   else
   {
      curCount = readCoreTimer();
      if (curCount > (K * ms))
         retValue = curCount / K;
   }
   return retValue;
}
```

   NOTA: use o valor de "**`K`**" que determinou no exercício 2 da parte 1.

   a) Traduza para *assembly* do MIPS a função **`timeDone()`** e teste-a com diferentes valores de entrada (para o teste adapte o código C fornecido no ponto 1).

2. Retome o exercício 1 da parte 2 e reescreva o programa de modo a utilizar a função **`timeDone()`**. Por cada segundo decorrido o código deve enviar para o terminal o caracter '**`\n`**'.

PDF criado em 28/02/2023