Organização e Arquitetura de Computadores II

Trabalho II - Versão 5

Características introdutórias:

- Sistema computacional composto por dois conjuntos computacionais independentes contendo CPU, Memória e UART.
- A comunicação entre os conjuntos computacionais ocorre através de uma UART (*Universal Asynchronous Receiver/Transmitter*)
- Para cada conjunto computacional, a CPU é mestre do barramento (*single master*), com entrada/saída mapeada em memória.
- A CPU pode a qualquer momento acessar a memória ou a UART através de instruções *LW* e *SW* - leitura e escrita, respectivamente.
- O mecanismo de interrupção serve para a UART avisar que tem dados para enviar à CPU ou para solicitar dados da CPU.

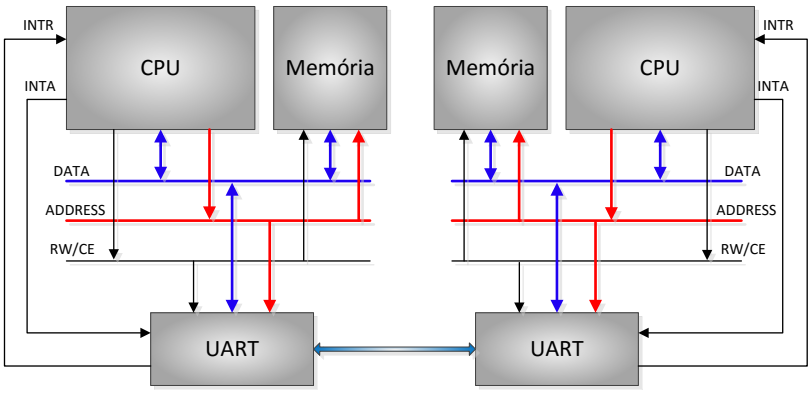

Figura 1 - Sistema computacional composto por dois conjuntos independentes de CPU, memória e UART, com comunicação através de um sistema assíncrono.

Operação da interrupção (hardware):

1) A UART requer interrupção à CPU via sinal de controle **INTR**.
2) A CPU, estando apta a atender à interrupção, responde através do sinal **INTA** e inicia o processamento da interrupção correspondente a UART.

Modificações na CPU (MIPS)

As modificações envolvem principalmente o bloco de controle e a inclusão dos sinais externos **INTR** e **INTA**. Uma nova instrução – **ERET** – deve ser incluída para a rotina de tratamento de interrupção informar ao hardware o término de sua execução.

Procedimento de tratamento da interrupção - software

Para ser possível trabalhar com interrupções, deve se definir o tratamento dados às interrupções e onde gravar o "**vetor de interrupções**". Abaixo temos um exemplo de código, supondo que:

- O programa principal (semelhante a um Sistema Operacional - SO) inicie na posição 0x00400000;
- O programa do usuário inicie na posição 0x004000FC;
- O trecho de programa que analisa a interrupção requerida pela UART (no caso interrupção de envio ou de recepção de dados) e desvia para o endereço de tratamento inicia na posição 0x00400004;
- O vetor de interrupções está armazenado nos endereços 0x00000060 e 0x00000064 (neste trabalho

temos apenas a UART com possibilidade de gerar interrupções referentes à transmissão ou recepção de dados). Assim, o endereço 0x00000060 é referente à transmissão de dados para a UART (CPU_TO_UART) e o endereço 0x00000064 é referente à recepção de dados da UART (UART_TO_CPU).

End. Mem.	Instrução	Comentários	
0x00400000	J 0x004000FC	Desvia para o endereço do início do programa do usuário	Primeira instrução do programa
0x00400004	SUBU $SP, $SP, 4		Trecho de programa onde consta o código básico para desviar para as rotinas de tratamento de interrupção
0x00400008	SW $RA, 0($SP)	Salva $RA na pilha	
0x0040000C	LA $K0, 0xFFE00000	$K0 ← endereço da UART	
0x00400010	LW $K0, 0($K0)	Lê a UART	
0x00400014	SLL $K0, $K0, 0X02	$K0 ← $K0 << 2 pois o endereçamento é composto por 4 bytes	
0x00400018	LA $K1, 0x00400060	$K1 ← endereço de início do tratamento de interrupção	
0x0040001C	ADDU $K0, $K0, K1		
0x00400020	JALR $K0		
0x00400024	ADDIU $SP, $SP, 4		
0x00400028	RFE		
0x0040002C	JR $RA		
0x00400060	*End. rot. Escrita de dados na UART*		Endereços das rotinas de tratamento de interrupção (limitado em 16 interrupções)
0x00400064	*End. rot. Leitura de dados na UART*		
...	...	...	

O código acima deve ser carregado junto com o programa do usuário (ver um exemplo completo ao final deste documento).

Procedimento de tratamento da interrupção

Processador deve realizar as seguintes atividades:

- Salvar o PC (em um registrador interno ao processador) e desviar a execução para a posição de memória 0x00400004 (**PC ← 0x00400004**), exatamente como em uma subrotina;
- Ativar sinal **INTA**, sinalizando para a UART que o pedido será atendido;
- Voltar para o ciclo de busca de instrução, para a instrução contida no endereço 0x00400004.

Ao encontrar a instrução **ERET** significa que terminou o tratamento da interrupção corrente e os registradores devem voltar ao estado anterior a interrupção (recuperação de contexto).

Projeto da UART

A UART deve operar com um modelo de comunicação bidirecional full-duplex assíncrono. O dado deve ser transmitido por uma serial conforme protocolo P82 (paridade par, 8 bits de dados e dois stop bits além do start bit).

A comunicação full-duplex acontece através de dois canais paralelos e simétricos. Um de entrada de dados (RX) e outro de saída (TX).

Devido à limitação de apenas um par de sinais de controle de interrupção (INTR e INTA) para dois canais de comunicação, o pedido de interrupção dos canais da UART para a CPU é feito de forma multiplexada. A CPU deve endereçar o endereço base da UART (EnderecoUART = 0xFFE00000) e verificar se a interrupção é de entrada ou saída. Fica definido que, caso o conteúdo do EnderecoUART for 0, a interrupção é de saída (UART transmitiu um dado), sendo necessário produzir mais dados, ou seja, chamar a CPU_TO_UART. Caso o conteúdo do EnderecoUART for 1, a interrupção é de entrada (UART recebeu um dado), sendo necessário

consumir o dado recebido, ou seja, chamar a interrupção UART_TO_CPU.

Para a transmissão de dados, a UART tem um registrador de escrita (RW) e um registrador de deslocamento de saída (RDO). O RW é um registrador de 8 bits, enquanto que o RDO é de 12 bits, de forma a suportar o protocolo P82. O RW é acessado pela CPU através do endereço EnderecoUART + 4. A cada escrita em RW, a UART tenta inicialmente transferir o dado para o RDO. Caso o RDO esteja ocupado, a UART espera terminar a transmissão de todos os bits serialmente, para então copiar RW para RDO. A transferência de um dado de RW para RDO gerar um pedido de interrupção para a CPU. Nesta interrupção (i.e., CPU_TO_UART), a CPU pode inserir mais um dado no RW. Note que é importante que associado ao dado do registrador, deve ter um bit informando que o conteúdo é válido para não transmitir um dado qualquer. A conversão do dado de 8 bits para o protocolo (12 bits) é feita internamente a UART e de forma transparente para a CPU.

Para a recepção de dados, a UART tem um funcionamento análogo da transmissão. Esta, dispõe de um registrador de leitura (RR) e um de deslocamento de entrada (RDI). RR é um registrador de 8 bits, enquanto que o RDI é de 12 bits, de forma a suportar o protocolo P82. O RR é acessado pela CPU através do endereço EnderecoUART + 8. O RR o dado do RDI sempre que o RDI completar a recepção de um byte de dados, e neste evento, a UART deve gerar um pedido de interrupção para a CPU (i.e., UART_TO_CPU). Nesta interrupção, a CPU deve ler o dado do RR. Note que, caso a CPU não leia o dado a tempo, então pode ocorrer uma sobre escrita de RR. Este problema não é tratado neste trabalho. A conversão do protocolo de 12 bits para o dado de 8 bits é feita internamente a UART de forma transparente para a CPU.

Note que devido ao sistema ser full-duplex, a UART deve tratar a possibilidade de requisição de dois pedidos de interrupção simultâneos. Ou seja, dos canais RX e TX. Desta forma, ela deve ter um mecanismo para enfileirar os pedidos de interrupção, de forma a não perder os mesmos.

Projeto do Sistema

A operação completa do sistema implica em fazer no alto nível a transferência de uma mensagem de 128 bytes de um conjunto computacional para outro. Desta forma, embora os sistemas suportem comunicação full-duplex, apenas uma comunicação unidirecional deverá ser alcançada. Ao término da execução, a memória do conjunto computacional destino deverá ter toda a mensagem transmitida pelo conjunto computacional de origem. Assim, os alunos devem implementar em software, tanto um programa de transmissão, quanto um programa de recepção.

A Fazer e Entregar

(1) Modificar a arquitetura do MIPS (e.g., MR2) e validar a modificação por simulação;

(2) Implementar a UART e validar por simulação;

(3) Implementar no MARS o software do sistema e verificar se o mesmo está adequado;

(4) Implementar o sistema completo e validar por simulação;

(5) Simular um programa com atendimento de interrupção. Mostrar a execução do programa, o momento que o mesmo é interrompido, e a volta ao programa principal;

(6) Relatório do projeto.

Passos para iniciar o trabalho

- Passos:
  1. Salvar os arquivos do projeto em uma área de trabalho;
  2. Abrir o simulador VHDL de sua preferência;
  3. Iniciar um novo projeto;
  4. Adicionar os arquivos vhd e txt que estão no zip ao projeto;
  5. Iniciar as atividades ...

Anexo: Exemplo de código para tratamento de interrupção

```
.text
.globl main

# SALTA PARA O INÍCIO DO CÓDIGO DO USUÁRIO
#####################################################
main:
        j MyMain

# ROTINA PARA TRATAMENTO DAS CHAMADAS DE INTERRUPÇÃO
#####################################################
DesvioParaTratamentoDeInterrupcoes:
        subu $sp, $sp, 4
        sw $ra, 0($sp)
        la $k0, EnderecoUART
        lw $k0, 0($k0)
        sll $k0, $k0, 0x02
        la $k1, TabelaDeInterrupcoes
        addu $k0, $k0, $k1
        jalr $k0
        lw $ra, 0($sp)
        addiu $sp, $sp, 4
        eret
        jr $ra

# ENDEREÇO DA TABELA DE ENDEREÇAMENTO DE INTERRUPÇÕES
#########################################################
TabelaDeInterrupcoes:
        j CPU_TO_UART
        j UART_TO_CPU

# ROTINAS PARA TRATAR AS INTERRUPÇÕES
#####################################
CPU_TO_UART:
        subu $sp, $sp, 20
        sw $t0, 0($sp)
        sw $t3, 4($sp)
        sw $t4, 8($sp)
        sw $t6, 12($sp)
        sw $ra, 16($sp)

...

        lw $ra, 16($sp)
        lw $t6, 12($sp)
        lw $t4, 8($sp)
        lw $t3, 4($sp)
        lw $t0, 0($sp)
        addiu $sp, $sp, 20
        jr $ra

############################################
UART_TO_CPU:

# INÍCIO DO PROGRAMA DO USUÁRIO NO ENDEREÇO
############################################
MyMain:
        li $t0, 0
        li $t1, 0
        li $t2, 0
SaltoMyMain:
        addiu $t0, $t0, 1
        addu $t1, $t1, $t0
        addu $t2, $t1, $t0
        j SaltoMyMain

# ENDEREÇO DA UART
##################
.data  0xFFE00000
EnderecoUART:
```