# Arquitetura de Computadores II

# Trabalho II - Versão 13

## Características introdutórias:

* Sistema computacional composto por dois conjuntos computacionais independentes denominados de SisA e SisB, cada um contendo CPU, Memória e um sistema de comunicação (UART);
* A comunicação SisA e SisB ocorre através da UART implementada no Trabalho I;
* A CPU é mestre do barramento, com entrada/saída mapeada em memória; a qualquer a CPU momento acessar a memória ou a UART através de instruções *LW* e *SW* - leitura e escrita, respectivamente;
* Os pinos irTX e irRX são usados como mecanismos de interrupção, 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.

## Operação da interrupção (hardware):

1. Toda a vez que a UART finalizar a recepção de um dado, ou iniciar a transmissão de um dado, ela deve informar à CPU, através de um pedido de interrupção;
2. A UART requer interrupção à CPU via sinal de controle **irTX** ou **irRX**, dependendo se for uma interrupção de transmissão ou recepção, respectivamente;
3. A CPU, estando apta a atender à interrupção, responde através do sinal **ackTX** ou **ackRX** e inicia o processamento da interrupção correspondente.

## Modificações na CPU (MIPS)

As modificações no código básico do MIPS entregue ao aluno envolvem principalmente o bloco de controle e a inclusão dos sinais externos **irTX, irRX, ackTX** e **ackRX**. Note que foi incluída uma nova instrução – **ERET** –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, devem ser definidos os tratamentos das 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) inicia na posição 0x00400000;
* O programa do usuário inicia na posição apontada por MyMain – ver assembly ao final deste documento;
* O vetor de interrupções está armazenado nos endereços 0x00400004 e 0x00400008, que contêm desvios incondicionais para as rotinas de tratamento de interrupção de transmissão (CPU\_to\_UART, associada a irTX) e recepção (UART\_to\_CPU, associada a irRX) respectivamente;
* A UART está mapeada em memória na posição 0xFFE00000 (referenciada pelo rótulo EnderecoUART no assembly);

**Procedimento de tratamento da interrupção**

Processador deve realizar as seguintes atividades:

* Salvar o **PC** (em um registrador interno ao processador) e desviar a execução do programa para a posição de memória correspondente ao pedido de interrupção - 0x00400004 ou 0x00400008 (**PC 🡨 0x00400004 ou PC 🡨 0x00400008**);
* Setar o flip-flop interno **EM\_INT** para informar que está em atendimento de interrupção. Este flip-flop deve permanecer em um enquanto estiver na rotina de interrupção e somente voltar a zero quando a interrupção terminar. Ele serve como uma flag para que não possa ser possível entrar duas interrupções simultâneas (sistema não aceita aninhamento de interrupção);
* Ativar sinal **ackTX** ou **ackRX**, dependendo de qual interrupção foi atendida, para sinalizar para a UART que o pedido foi atendido;
* Voltar para o ciclo de busca de instrução, para a instrução contida no vetor de interrupção;

A execução da instrução **ERET** significa que terminou o tratamento da interrupção corrente, o **PC** deve voltar ao antigo valor e o flip-flop **EM\_INT** deve ser resetado. Note que os registradores devem voltar ao estado anterior à interrupção (recuperação de contexto). Isto possivelmente deve ter sido feito em software.

## Projeto do Sistema

A operação completa do sistema implica em fazer no alto nível a transferência de uma mensagem com o conteúdo “**Isto é um teste**” de SisA para SisB, e obter como resposta a mensagem “**Teste compreendido**” (de SisB para SisA). Desta forma, embora os sistemas suportem comunicação full-duplex, apenas uma comunicação half-duplex será realizada. Ao término da execução, as memórias dos conjuntos computacionais destino deverão ter todas as mensagens transmitidas pelos conjuntos computacionais 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. Adaptar a UART feita no T1 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. Dois testes adicionais que podem ser fora da execução básica do programa:
   1. Simular uma interrupção aninhada e provar que esta não ocorre devido ao sinal **EM\_INT**;
   2. Simular uma interrupção de entrada, enquanto uma de saída está sendo tratada e mostrar que esta não é perdida 🡺 é enfileirada;
7. Relatório do projeto.

## Passos para iniciar o trabalho

Passos: (i) Salvar os **arquivos** do projeto em uma área de trabalho; (ii) Abrir o simulador VHDL de sua preferência; (iii) Iniciar um novo projeto; (iv) Adicionar os arquivos vhd e txt que estão no zip ao projeto; (v) 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

# VETOR DE INTERRUPÇÃO

########################################################

VetorDeInterrupcoes:

j CPU\_to\_UART

j UART\_to\_CPU

# ROTINAS PARA TRATAR AS INTERRUPÇÕES

#####################################

CPU\_to\_UART:

subu $sp, $sp, 8

sw $t0, 0($sp)

sw $t1, 4($sp)

# ...

lw $t1, 4($sp)

lw $t0, 0($sp)

addiu $sp, $sp, 8

eret

#############################################

UART\_to\_CPU:

subu $sp, $sp, 8

sw $t0, 0($sp)

sw $t1, 4($sp)

# ...

lw $t1, 4($sp)

lw $t0, 0($sp)

addiu $sp, $sp, 8

eret

# INÍCIO DO PROGRAMA DO USUÁRIO

#############################################

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

.data

# Mensagem transmitida do 'SisA' pro 'SisB' na primeira Interrupcao

####################

mensagemTransmitida: .asciiz "Isto é um teste!"

# Mensagem recebida pelo 'SisA' do 'SisB' na Segunda Interrupcao

####################

mensagemRecebida: .asciiz " "

# ENDEREÇO DA UART

####################

EnderecoUART: .word 0xFFE00000