

Descrição dos Trabalhos da Disciplina

Instruções:

1. Grupos de NO MÁXIMO 2 alunos – DUPLA.
2. NÃO são permitidos dois projetos iguais, porém é permitido mesmo projeto de descrição de hardware com linguagens diferentes. Ex.: SystemC, VHDL. E alguns projetos em VHDL ou Assembly, desde que aprovado pela professora da disciplina.
3. A entrega do trabalho será considerada completa com a entrega dos seguintes itens: Implementação, Relatório Final e Apresentação do trabalho.
4. Os integrantes do grupo serão avaliados individualmente, e devem saber responder as perguntas relativas a todo o trabalho.
5. O relatório final deve seguir as orientações do documento “Regras para avaliação”.
6. As regras para avaliação estão no documento “Regras para avaliação”.

7. Prazos:

	Data
Definição dos grupos e trabalhos	26/04/2015
Apresentação do projeto final	08/06/2015

Trabalho 1: Implementação de um processador pipeline de 32 bits com banco de registradores que executa somente operações lógicas e aritméticas.

Descrição: Implementar a parte operativa e o controle de um processador que executa instruções em pipeline. O processador executa apenas operações lógicas e aritméticas e lê e escreve apenas no banco de registradores.

As instruções são semelhantes às instruções do MIPS e as operações são ADD, SUB, MUL, AND, OR, XOR.

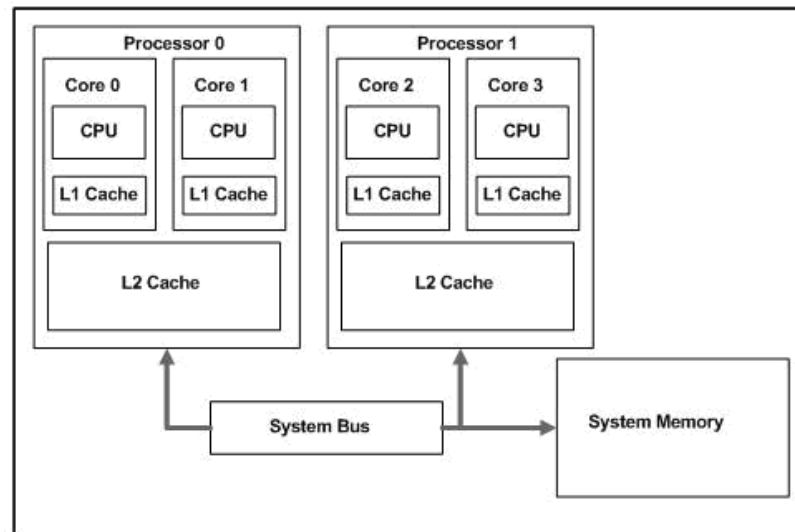
Deve ter:

1. Parte Operativa: ULA, multiplexadores
2. Parte de Controle: sinais para controlar os componentes em máquina de estados
3. Registradores: banco de registradores, contador de programa (PC)

Linguagem: VHDL

Trabalho 2: Simulador de hierarquia de memória em Multicore

Descrição: o sistema de memória terá dois níveis de cache (L1 e L2) e a memória principal. Cada core terá uma cache L1. Uma cache L2 será compartilhada entre dois cores. A memória principal será compartilhada por todos os cores. Observar figura abaixo.



No simulador, o usuário informará quantos cores o sistema terá (somente quantidades múltiplas de 2 são permitidas). O sistema automaticamente cria as caches e a memória principal. Um arquivo, lido pelo simulador, carrega os dados na memória principal. O usuário informa o endereço da memória principal para leitura e o core que irá utilizar aquele dado. O sistema automaticamente carrega o dado na cache L2 e na cache L1 do respectivo core. O sistema também deve ser capaz de atualizar o dado, caso o core modifique esse valor, e atualizar todos os níveis da hierarquia imediatamente (*write-through*).

Atenção:

- Assumir que os dados são números inteiros
- O sistema deverá dar opção de ler ou alterar o dado, indicando qual core fará a operação
 - Em caso de leitura: o dado será carregado nas caches (se ainda não estiver)
 - Em caso de escrita: o dado será alterado e atualizado em todos os níveis da hierarquia

Deve ter:

- Pode ser modo texto ou interface gráfica, contanto que o conteúdo de todas as memórias sejam exibidos
- Em caso de cache cheia, o algoritmo de substituição será o LRU (menos recentemente utilizado)

Linguagem: Qualquer linguagem

Trabalho 3: Simulador de sistema de predição de salto

Descrição: Implementar um simulador de predição de salto que recebe como entrada um arquivo contendo os saltos e tem como saída 1) a entrada; 2) o resultado da predição de cada salto; 3) a percentagem de acerto em cada predição e 4) qual o melhor algoritmo de predição para aquela entrada. O simulador deve conter quatro sistemas de predição de salto: 1 bit, 2 bits e dois propostos pela equipe.

Exemplo de arquivo de entrada: T N T T N N N T N T T T

Deve ter:

- Obrigatório ter interface gráfica
- Os algoritmos de predição de salto propostos não podem ser do tipo n-bits

Linguagem: Qualquer linguagem

Trabalho 4: Implementação de um processador VLIW

Descrição: Modificar a implementação do processador para executar duas instruções de ULA em paralelo. O processador é o desenvolvido na disciplina de circuitos lógicos e, se necessário, será disponibilizada uma versão funcional dele.

Para funcionar como VLIW, deve ser necessário alterar o formato da instrução para conter duas operações, ao invés de uma; controle e parte operativa.

Deve ter:

1. Parte Operativa: ULA; multiplexadores
2. Parte de Controle: sinais de controle dos componentes em máquina de estados
3. Registradores
4. Modificação no conjunto de instruções

Linguagem: VHDL

Trabalho 5: Implementação do controle de uma máquina de café.

Descrição: A máquina trabalha com quatro tipos de café de preços diferentes e aceita moedas de R\$ 0,10; R\$ 0,50 e R\$ 1,00. Um sinal de entrada indica a entrada de moedas, enquanto a outra entrada indica a solicitação do tipo de café, o qual deve ser previamente escolhido através da entrada de seu código. A máquina está preparada para devolução do troco, caso o valor da(s) moeda(s) exceda o valor do café escolhido, ou caso valor seja inferior ao seu preço. Para isso foram implementados os sinais de saída: “libera troco”, “troco” e “libera café”.

Tipos de café e preços:

- Café espresso – R\$ 2,00
- Mocaccino – R\$ 3,00
- Cappuccino – R\$ 3,50
- Chocolate quente – R\$ 2,50

Deve ter:

1. Fluxograma do funcionamento do sistema
2. Parte Operativa: ULA, comparador, multiplexadores
3. Parte de Controle: sinais de controle dos componentes em máquina de estados
4. Registradores: dados de entrada, dados de saída

Linguagem: VHDL ou Assembly

Trabalho 6: Implementar interrupção com salvamento de contexto no processador

Descrição: Modificar a implementação do processador para atender interrupções. O processador é o desenvolvido na disciplina de circuitos lógicos e, se necessário, será disponibilizada uma versão funcional dele.

Para atender interrupções é necessário acrescentar uma instrução de interrupção ao conjunto de instruções, adicionar a etapa de salvar e restaurar contexto e criar uma rotina de interrupção. A rotina de interrupção pode ser apenas uma instrução de soma.

Deve ter:

1. Parte Operativa: ULA; multiplexadores
2. Parte de Controle: sinais de controle dos componentes em máquina de estados
3. Registradores
4. Modificação no conjunto de instruções

Linguagem: VHDL

Trabalho 7: Implementar um jogo no Assembly ARM usando o simulador ARMSim#

Descrição: Implementar um dos jogos: Snake, Campo Minado, Resta 1, em Assembly ARM usando o simulador ARMSim#

Deve ter:

- Pode ter ou não interface gráfica
- Download do Simulador:
 - <http://armsim.cs.uvic.ca/DownloadARMSimSharp.html>

