	**Universidade Federal do Rio Grande do Norte** **Centro de Ciências Exatas e da Terra** **Departamento de Informática e Matemática Aplicada**

DIM0109.0 / DIM0109.1	**Circuitos Lógicos / Laboratório de Circuitos Lógicos**

3ª Unidade

Projeto Final

2017.2

Descrição:

Implementar, usando a linguagem VHDL, e prototipar, usando a placa de prototipagem DE2-115 da Altera, um caminho de dados simplificado, capaz de executar operações lógicas e aritméticas sobre registradores e memória. O caminho de dados a ser implementado consiste em uma versão simplificada de um processador genérico de propósito geral baseado na filosofia RISC, que será estudado na disciplina seguinte, de Organização e Arquitetura de Computadores.

1 - Definição do Caminho de Dados:

O caminho de dados será constituído de 7 (sete) módulos principais:

1. **Módulo de memória** (fornecido pelo professor).
2. **Módulo de acesso à memória** (lê/escreve na memória fornecida e nas entradas e saídas).
3. **Unidade lógica e aritmética.**
4. **Banco de registradores**, contendo 8 registradores.
5. **Decodificador hexadecimal/sete segmentos**.
6. **Módulo de entrada de operações.**
7. **Módulo de saída.**

1.1 - Módulo memória

Um arquivo ram.vhdl será fornecido. Nesse arquivo é possível inserir instruções e dados, ambos de 16 bits. As instruções e dados salvos nessa memória RAM serão lidos ou escritos pelo circuito a ser implementado neste trabalho. Para ler ou escrever, o circuito deve fornecer um endereço a ser acessado. Se a operação for uma escrita, além de fornecer o endereço, o circuito deve fornecer também o novo valor a ser escrito na memória. Um sinal do tipo std_logic determina se a operação é uma leitura ou escrita.

1.2 - Módulo de acesso à memória

O módulo de acesso à memória tem a função de fazer a comunicação entre o caminho de dados e o módulo de memória citado no Item 1.1. Além da comunicação com essa memória, o módulo de acesso à memória também é responsável por fazer a interface entre o caminho de dados e os módulos de entrada e saída que contém os *displays* de 7 segmentos, os *switches* e os LEDs.
O acesso à memória RAM pode ser para leitura ou escrita. Em algumas operações (detalhadas na Seção 3) um dos dados virá da memória. Nesses casos, o endereço especificado será o endereço da memória que deverá ser lido em busca do dado antes que a operação seja realizada. O módulo de acesso deve pegar este endereço, realizar a leitura da memória e receber o dado. Somente uma dessas operações (ver Tabela 1) requer uma escrita na memória. Neste caso, o endereço será de leitura e a operação também especificará qual dado deve ser escrito na memória. Assim a memória também deve suportar escritas e deve ter um registrador de dados para escrita.

1.3 – Unidade lógica e aritmética

Uma ULA (Unidade Lógica e Aritmética) deve ser implementada. As operações realizadas por esta ULA devem levar em conta a codificação dos dados em complemento de dois. Os operadores da ULA serão aqueles necessários e suficientes para a implementação das operações descritas na Tabela 1.

IMPORTANTE: a ULA deve ser de 16 bits e, sendo um circuito combinacional, não possui clock.

1.4 – Banco de registradores

Os dados a serem operados virão, na maioria dos casos, dos registradores. O banco de registradores deve ter 8 registradores de 16 bits cada um. Os registradores serão reconhecidos (na codificação) por seus endereços (do endereço zero [000] ao endereço sete [111]).

1.5 – Decodificador hexadecimal/sete segmentos

Para facilitar a depuração das operações realizadas e a demonstração do trabalho na apresentação, um módulo decodificador hexadecimal/sete segmentos deverá ser implementado. O decodificador deve controlar 4 displays de sete segmentos a fim de mostrar todos os valores possíveis de 16 bits (0000 à FFFF, com cada *display* mostrando um dígito).

1.6 - Módulo de entrada de operação

Este módulo consiste de 3 botões e de 16 chaves (*switches*). Um dos botões deve reiniciar (*resetar*) o circuito, fazendo com que eventuais máquinas de estados e registradores sejam setados para um valor predefinido. O segundo botão deve ser acionado quando um valor de 16 bits tiver sido colocado nos *switches*, fazendo com que o circuito saiba que esse valor é válido. A cada vez que o terceiro botão for acionado o circuito deverá realizar uma operação.

1.7 – Módulo de saída

Além dos *displays* de 7 segmentos, o módulo de saída será constituído de 17 LEDs (*light-emitting diode*) da placa DE2-115. O 17º LED será aceso quando a execução de uma operação terminar. Depois que esse LED acende, o caminho de dados estará pronto para iniciar a entrada de uma nova operação. Os demais 16 LEDs são controlados via software e podem ser usados, por exemplo, para indicar a finalização de cada algoritmo implementado ou então indicar que o programa está pedindo que o usuário entre com algum dado (para os algoritmos, ver Item 4).

2 - Formato do conjunto de operações a serem implementadas

As operações realizadas pelo caminho de dados serão codificadas em 16 bits. Essas operações deverão ser armazenadas no arquivo de memória `ram.vhdl` que será fornecido. O formato das operações são definidos pelos campos mostrados na Figura 1:

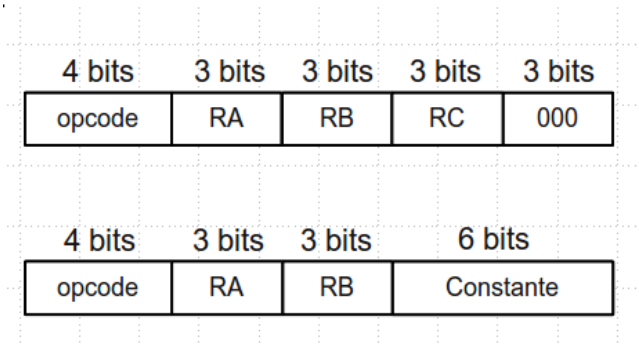

FIGURA 1 – Formatos das instruções

- **opcode**: define qual é a operação (soma, subtração, etc)
- **RC**: registrador destino, onde é salvo o resultado da operação
- **RA**: registrador que contém o 1º operando da operação
- **RB**: registrador que contém o 2º operando da operação
- **Constante**: constante em complemento de dois que, em algumas operações, funciona como 2º operando ao invés de RB. Nesses casos, onde RB não é operando, o registrador de destino será o RB ao invés de RC.

3 – Operações a serem implementadas

A Tabela 1 descreve as operações a serem implementadas.

Operação	Sintaxe	Significado	OPCODE
add	add RC, RA, RB	RC = RA + RB	0000
sub	sub RC, RA, RB	RC = RA - RB	0001
and	and RC, RA, RB	RC = RA and RB	0010
nor	nor RC, RA, RB	RC = RA nor RB	0011
or	or RC, RA, RB	RC = RA or RB	0100
sll	sll RC, RA	RC = RA << 1	0101
slr	slr RC, RA	RC = RA >> 1	0110
lw	lw RB, RA, const	RB = m[RA+const]	1000
sw	sw RB, RA, const	m[RA+const] = RB	1001
addi	addi RB, RA, const	RB = RA + const	1110

TABELA 1 - Conjunto de instruções a serem implementadas

Para compreender melhor a Tabela 1 seguem algumas explanações e exemplos:

- RC=RA+RB, por exemplo, significa que os registradores RA e RB devem ser lidos do banco de registradores, em seguida ser feito uma operação de adição com os valores lidos e então salvar esse resultado no registrador RC. A mesma lógica se aplica as demais operações que seguem esse formato.
- m[RA+Const] representa o valor do registrador RA somado com a constante. Esse valor (RA+Const) funciona como o endereço a ser acessado na memória, seja para leitura ou escrita.
- RB=RA+Const significa que deve ser lido o valor contido no registrador RA e somar esse valor com a constante. (O valor da constante está presente nos próprios bits da operação). O resultado deve ser armazenado no registrador RB.
- LW é uma operação que lê um dado da memória e salva em um registrador. Já a operação SW é o inverso: lê o conteúdo do registrador e salva na memória. Em ambos os casos, o resultado de RA+Const é quem fornece a posição a ser acessada na memória.

Na Figura 2, são mostrados exemplos de como as operações da Tabela 1 são codificadas em binário.

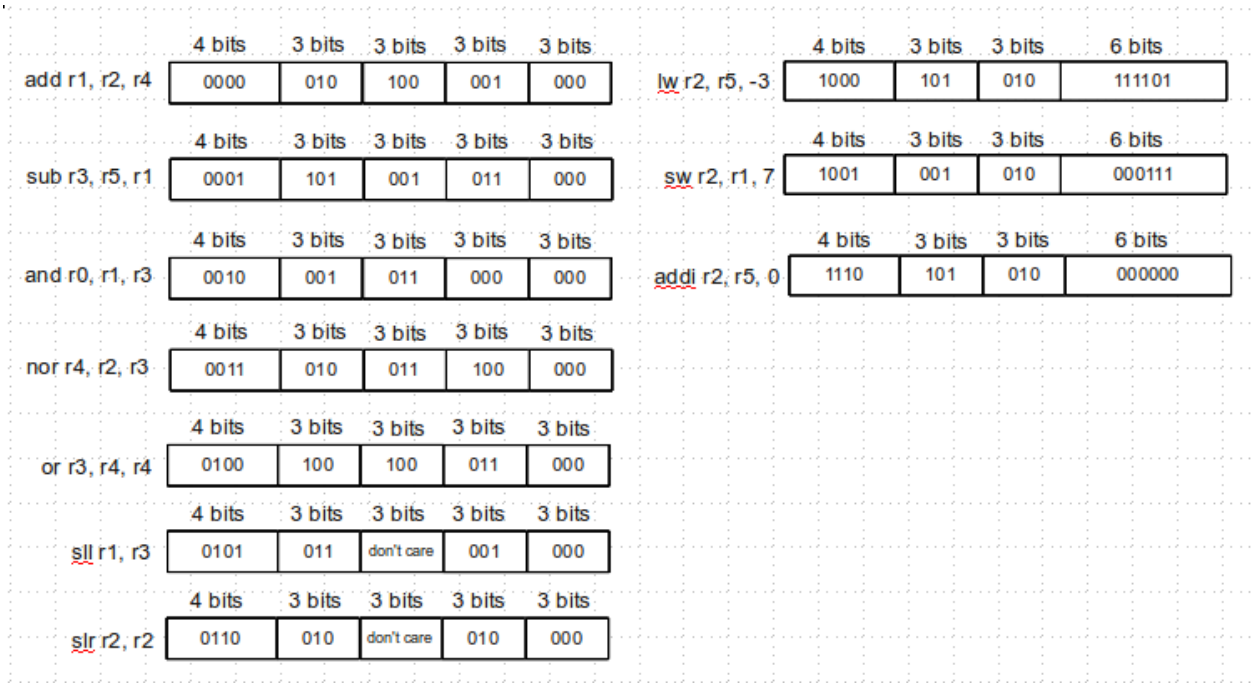

FIGURA 2 - Exemplo de codificação das instruções a serem implementadas

4 – Implementando algoritmos suportados pela arquitetura proposta

Esta seção descreve os passos necessários para a execução de algoritmos a serem implementados. Esses algoritmos DEVEM utilizar APENAS as operações descritas na Tabela 1. A construção dos algoritmos é parte da avaliação do trabalho e eles devem ser executados da seguinte forma:

1. Implementar cada algoritmo (toda a sequência de operações), de forma que execute operação por operação, da primeira até a última.
2. Mostrar o resultado da operação do algoritmo no módulo de saída (display de 7 segmentos e LEDs).
3. Procurar otimizar o algoritmo, minimizando a quantidade de memória e registradores utilizados.

4.1 – Algoritmo 1

1. Ler 2 valores A e B, um de cada vez, inseridos através dos *switches*.
2. Dividir A por 4;
3. Multiplicar B por 10;
4. Adicionar os resultados da divisão e da multiplicação (passos 2 e 3).
5. Mostrar o resultado nos *displays* de 7 segmentos.

4.2 – Algoritmo 2

1. Ler 3 valores: A, B e C;
2. Fazer: A = A + 1; B = B * 2 e C = C / 2;
3. Mostrar no *display* A **+** B **-** C.

4.3 – Algoritmo 3

1. Ler 2 valores A e B;
2. Realizar a operação (A xor B) and ((not A) or B);
3. Mostrar o resultado no *display* de 7 segmentos e nos 16 LEDs controlados via software.

4.4 – Algoritmo 4

1. Ler um valor A;
2. Mostrar o resultado de A * 5 usando o mínimo de operações.

5 – Etapas para desenvolvimento do projeto

As seguintes etapas são sugeridas, mas não impostas, para o desenvolvimento do projeto:

5.1. Dividir tarefas: Considerando o tamanho das equipes, **2** alunos, e a quantidade de tarefas a serem executadas, é necessário que cada membro da equipe tenha tarefa específica. No caso de quem não for trabalhar em dupla, as tarefas devem ser bem escalonadas para que consiga desenvolvê-las no tempo necessário.

5.2. Definição das interfaces: Cada módulo desenvolvido deve estar pronto para ser "conectado" aos demais módulos, de modo a formar o caminho de dados completo. Sendo assim, as interfaces (*entity*) entre os módulos devem estar perfeitamente combinadas antes que o desenvolvimento inicie.

5.3. Reunião inicial, com o professor das disciplinas: Deve ser realizada no sentido de apresentar as estratégias de desenvolvimento do projeto. Nesta reunião serão dadas as últimas orientações antes do início do desenvolvimento. Data prevista: **sex24nov2017**

5.4. Desenvolvimento e validação dos módulos individuais: Cada módulo deve ser individualmente desenvolvido, considerando a interface previamente combinada, e testado (simulado) também individualmente.

5.5. Apresentação dos módulos individuais desenvolvidos e validados: Antes da integração, as equipes podem mostrar ao professor, os módulos individuais completamente desenvolvidos e validados.

5.6. Integração dos módulos individuais: Nesta fase os módulos serão instanciados na entidade principal (*top level*) para formar o caminho de dados completo.

5.7. Validação do caminho de dados: A validação será feita através da execução de um ou mais códigos desenvolvidos (incluindo os algoritmos descritos na Seção 4). Na demonstração final o professor fornecerá um roteiro de apresentação e um conjunto de testes que deverá ser utilizado.

6 – Sugestão de prioridades para o desenvolvimento dos módulos

Sugere-se que os módulos sejam desenvolvidos e validados na seguinte ordem:

6.1. Unidade lógica e aritmética e integração com registradores de entrada e saída.

6.2. Módulo de entrada integrado com os botões e *switches* da placa DE2-115.

6.3. Módulo de acesso a memória integrado com a memória RAM.

6.4. Desenvolvimento do banco de registradores e integração dos barramentos de leitura e escrita e seus respectivos decodificadores. Deste barramento deve partir uma interface para o decodificador hexadecimal/sete_segmentos.

6.5. Desenvolvimento do controle global e integração com o módulo de saída. Os módulos individuais, eventualmente, podem ser dotados de controles individuais (discutir com o professor). Neste caso, o controle global coordenaria as ações dos controles individuais.

6.6. Em paralelo aos passos anteriores, recomenda-se a implementação (em papel) dos algoritmos presentes na Seção 4. Após a validação desses passos, testar o algoritmo na arquitetura desenvolvida.

Última observação:

- Cada módulo pode ser desenvolvido de forma comportamental, mas a integração dos módulos deve ser feita de forma estrutural. Por exemplo, na implementação da ULA, isso pode ser feito usando os operadores do VHDL, mas o módulo da ULA será integrado ao projeto de forma estrutural. Em outras palavras, cada módulo desenvolvido (de forma comportamental, *dataflow* ou estrutural) será um componente da entidade principal, ou seja, a entidade *top-level* será descrita de forma estrutural.

7 - Etapas, datas e prazos

7.1. Reunião inicial com o professor:
sexta, **24nov2017**, às 10h50.

7.2. Envio da Atividade 3.1 (diagrama de blocos com integração dos módulos) pelo SIGAA:
ATÉ segunda, **27nov2017,** às 10h50.

7.3. Reunião com o professor antes da integração do projeto:
segunda, **27nov2017**, às 10h50.

7.4. Apresentação final do projeto:
sexta, **1ºdez2017**, às 10h50.

Natal, 17 de novembro de 2017