Universidade Federal de Minas Gerais
Departamento de Computação
Disciplina: Organização de Computadores II
Professor: Omar Paranaíba Vilela
Monitor: Laysson Oliveira Luz
Data: 26/10/2017

Trabalho Prático III

No terceiro trabalho prático da disciplina os alunos deverão desenvolver dois novos módulos do caminho de dados do processador objetivado como trabalho final, são eles: **memória de instruções** e **unidade de controle**.

No trabalho prático II foi construída uma **unidade lógico-aritmética** que foi conectada a um **banco de registradores** e dessa forma, têm-se hoje um circuito capaz de realizar operações lógico-aritméticas, porém o circuito desenvolvido não é capaz de decodificar as instruções a serem realizadas e nem de armazenar informações.

Portanto, deverão ser construídos uma memória que deverá conter as instruções a serem executadas e uma unidade para controlar o funcionamento de cada unidade de hardware do caminho de dados.

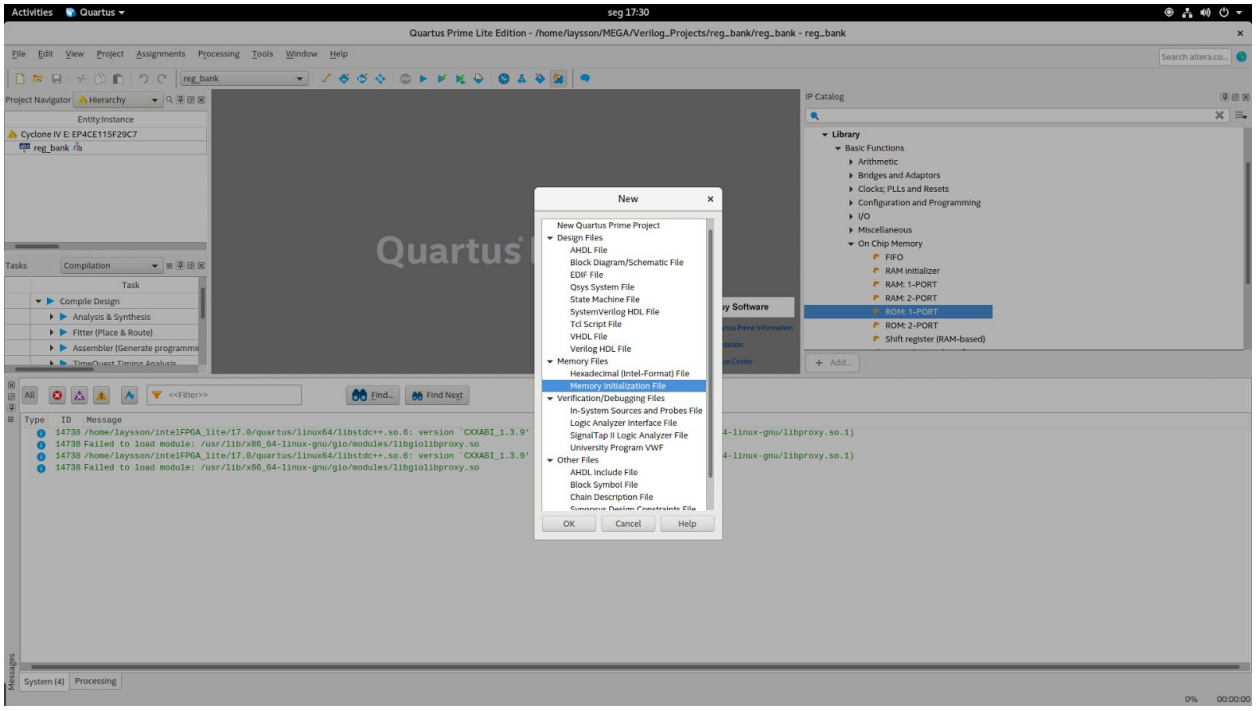
A seguir cada novo módulo é descrito com mais detalhes:

Memória de Instruções:

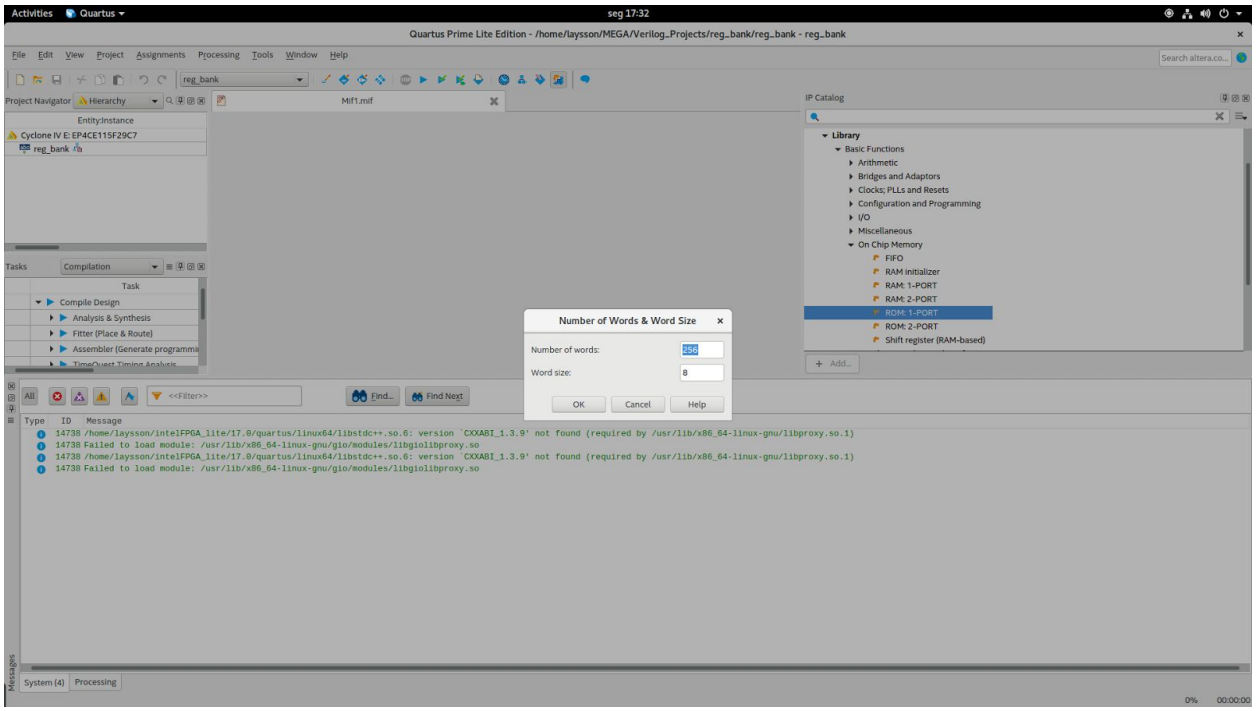
A memória de instruções deve ser construída utilizando um arquivo de inicialização da memória, que vai armazenar o conteúdo da memória, e um módulo de memória de propriedade intelectual da Altera que vai controlar as escritas e leituras nesse arquivo de inicialização da memória, seguindo os seguintes passos:

- Para criar o **Arquivo de Inicialização da Memória (MIF):**

1. Na janela inicial do Quartus, com seu projeto aberto, vá na aba “File” -> “New” e selecione **Memory Initialization File** como mostrado na imagem a seguir:



2. Na janela seguinte, determine a quantidade de palavras e o tamanho da palavra da memória:



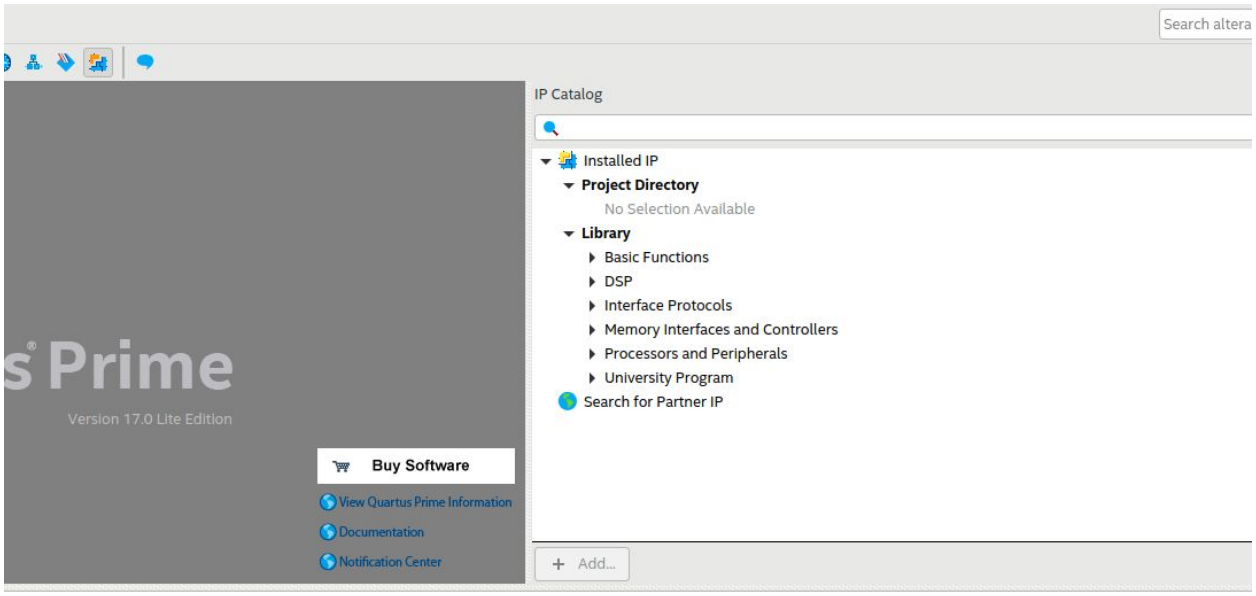
Para **tamanho da palavra** coloque 16 bits.

Para a **quantidade de palavras** na memória determine 4096 palavras.

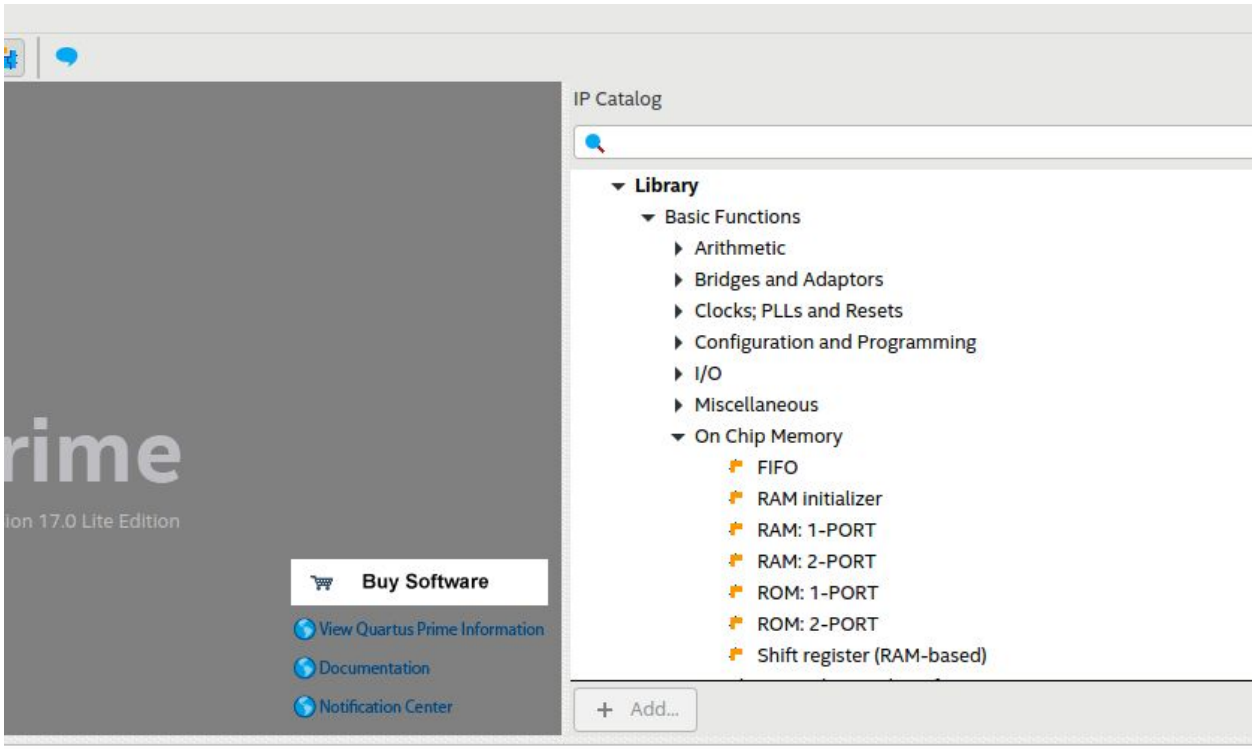
3. Logo em seguida o Quartus já abrirá o MIF criado, porém o arquivo ainda não foi salvo e recebe um nome padrão. Portanto, vá em File -> Save As e salve o arquivo com um nome à sua escolha.

❖ Para criar o módulo de gerenciamento da memória:

1. Identifique essa janela, no lado direito, no seu Quartus:

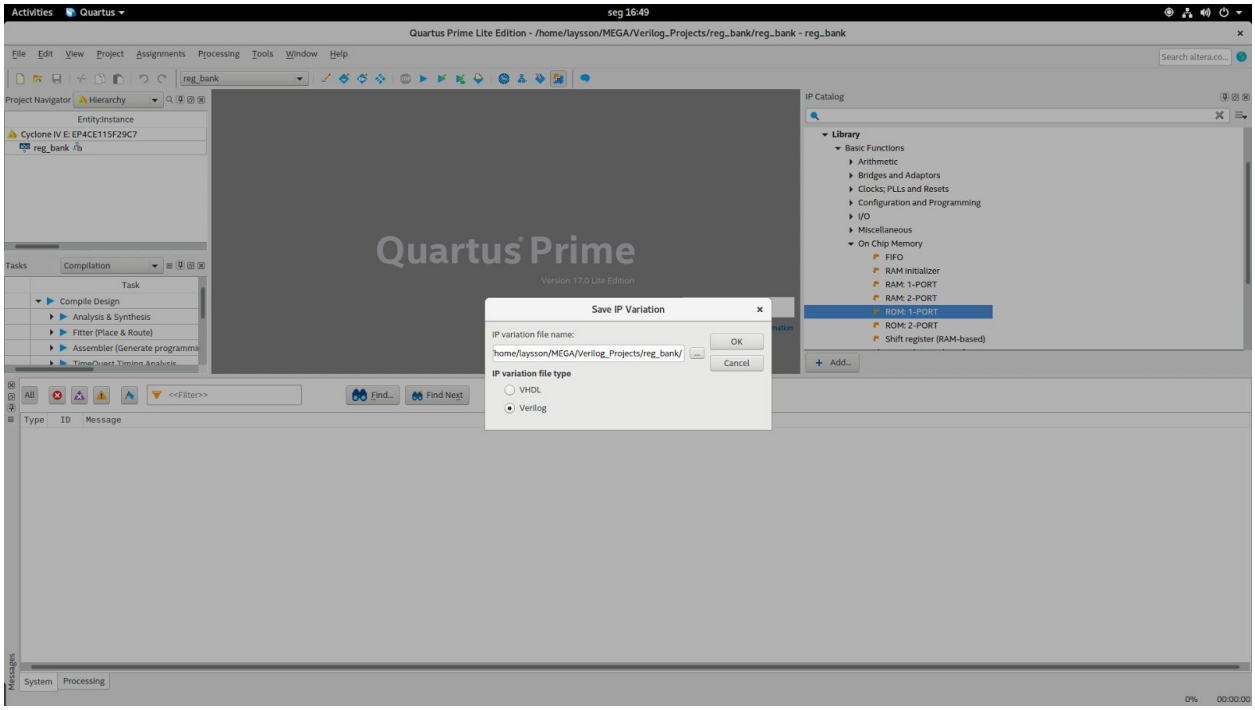


2. Clique no item “Basic Functions” e em seguida no item “On Chip Memory”:



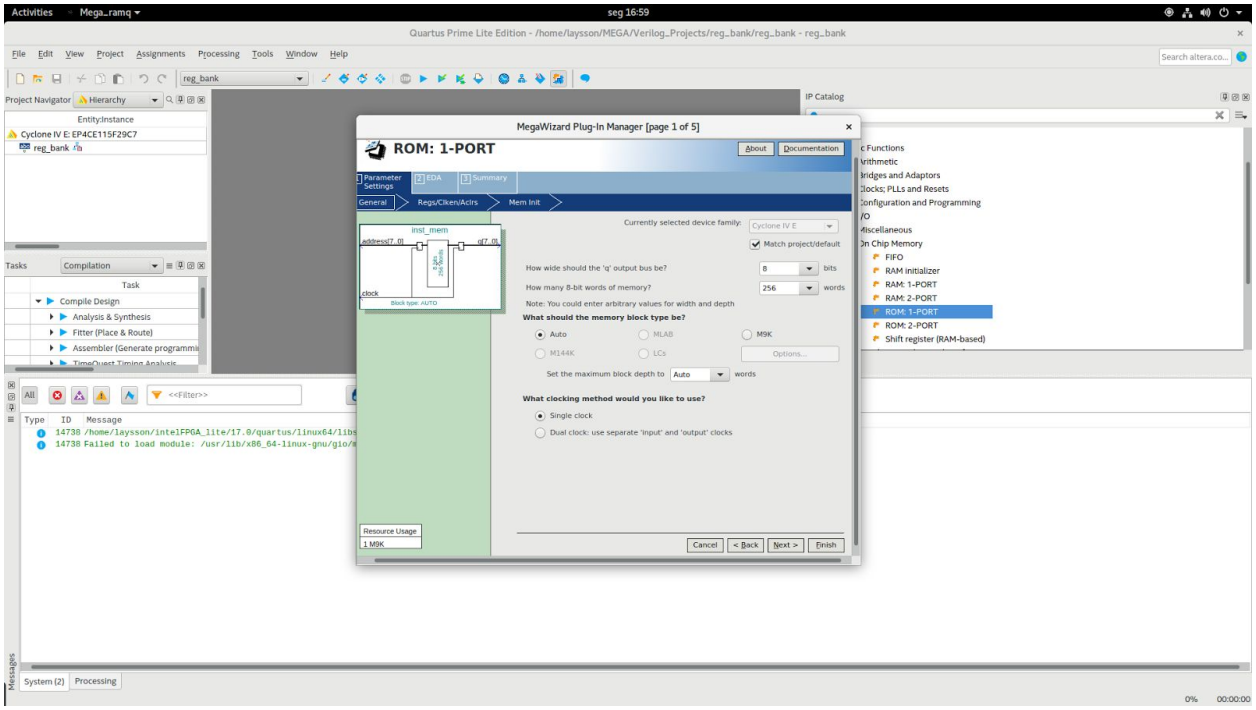
Escolha a opção ROM: 1-PORT.

3. Irá abrir uma janela semelhante a essa:



Nesta tela você deve digitar um nome para o seu módulo de memória e marcar a opção **IP variation file type** como “Verilog”.

4. Na janela seguinte você deverá especificar o tamanho da palavra de memória e quantas palavras seu bloco possui:



Para o tamanho do registrador de saída determine 16 bits.

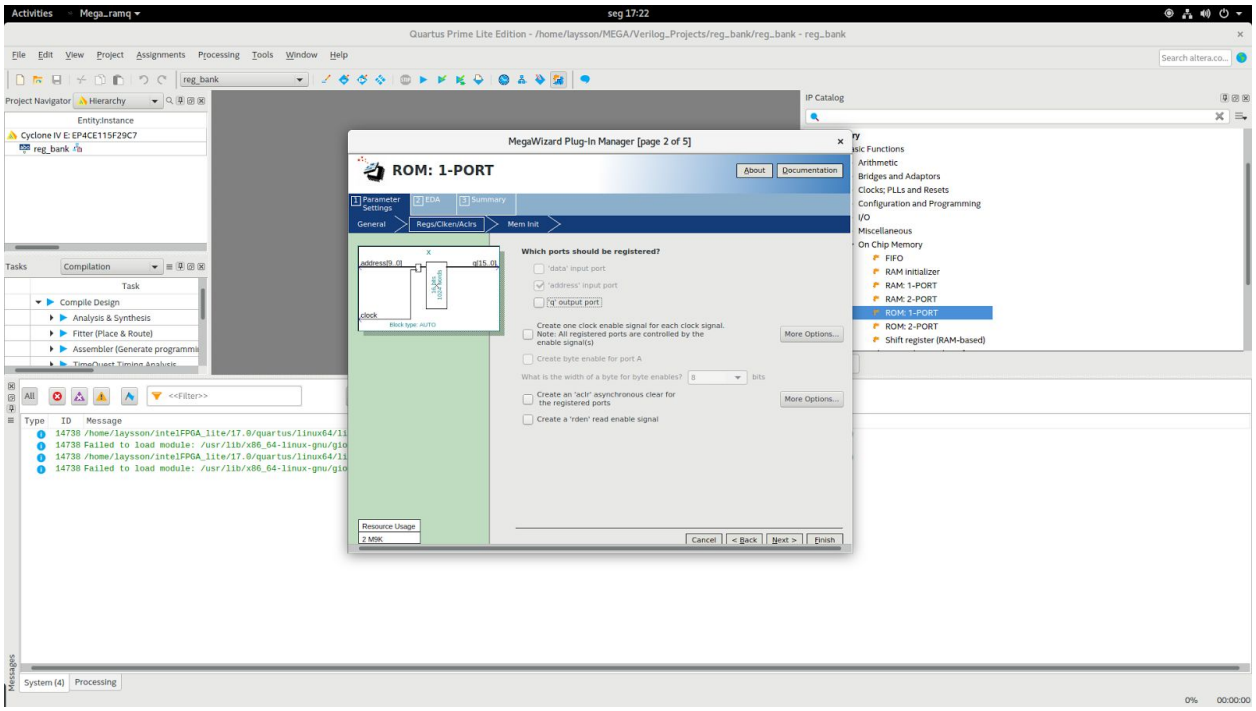
Para a quantidade de palavras na memória determine 4096 palavras.

Observe que neste passo você está determinando exatamente os mesmos parâmetros configurados para o MIF criado anteriormente.

Clique em “Next”.

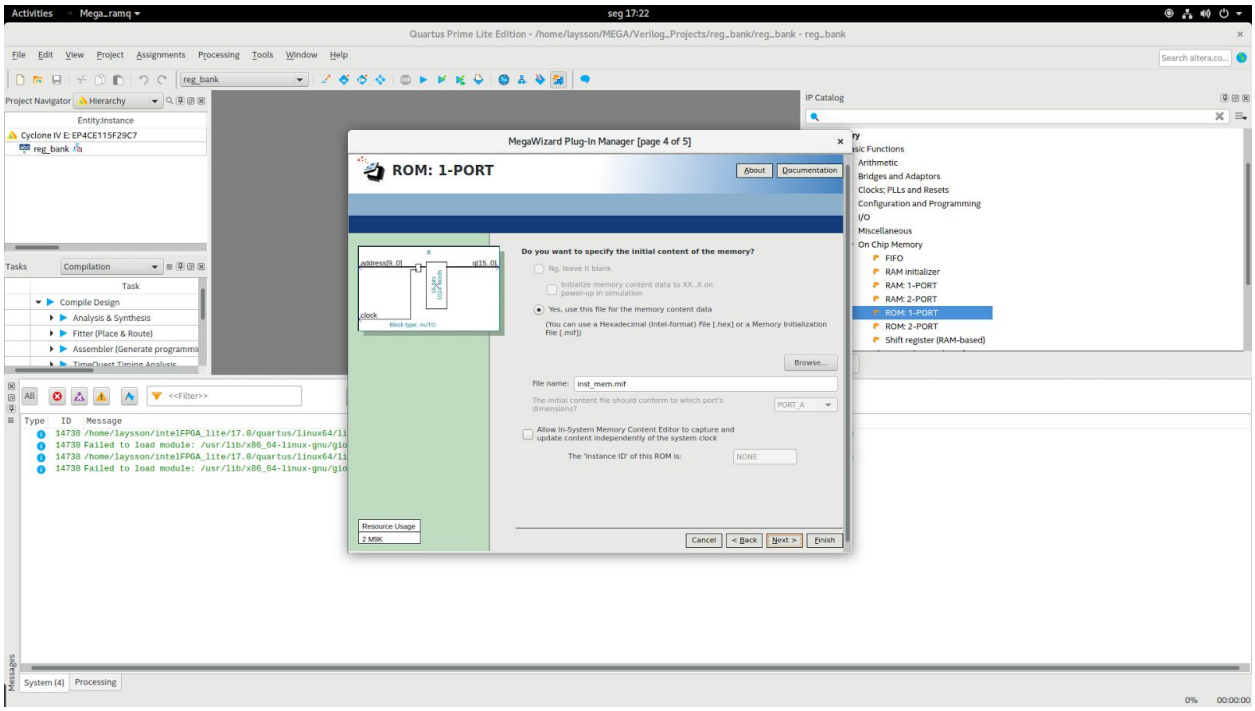
Os demais itens cujas seções são determinadas pelos textos em negrito não devem ser alterados.

5. Na janela seguinte desmarque a opção ‘q output port’:



A criação de um registrador de saída da memória é desnecessário para os objetivos do trabalho, pois a *altsyncram* da altera já possui um registrador interno para tal saída.

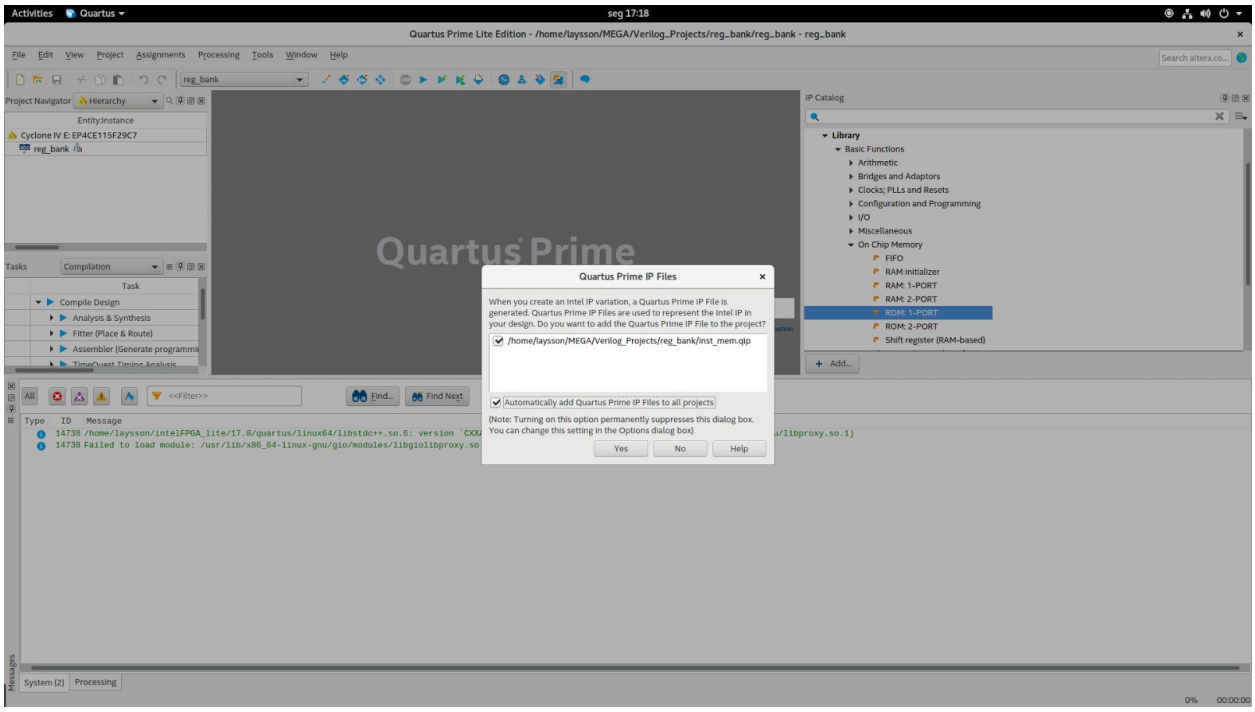
6. No passo seguinte selecione o arquivo de inicialização da memória a ser utilizado como conteúdo do módulo que está sendo criado.



No exemplo dado, o nome do arquivo de inicialização da memória é “inst_mem”.

Aqui você deve selecionar o MIF criado na seção anterior.

7. Por fim, você seleciona para adicionar o *IP variation file* ao projeto para que possa acessar seus arquivos de maneira prática.



Seu módulo de memória está criado e irá sempre utilizar como conteúdo da memória o MIF selecionado no passo anterior.

Unidade de Controle:

A unidade de controle do processador em desenvolvimento é fundamentalmente uma máquina de estados baseada no código de operação da instrução que envia os sinais de controle para cada componente do caminho de dados a fim de realizar uma operação.

Toda instrução, quando lida da memória, deve ser salva em um registrador, chamado **registrador de instruções.** Esse registrador deve enviar para a unidade de controle o **código de operação** da instrução, a partir do qual a mesma determina o valor de cada sinal de controle.

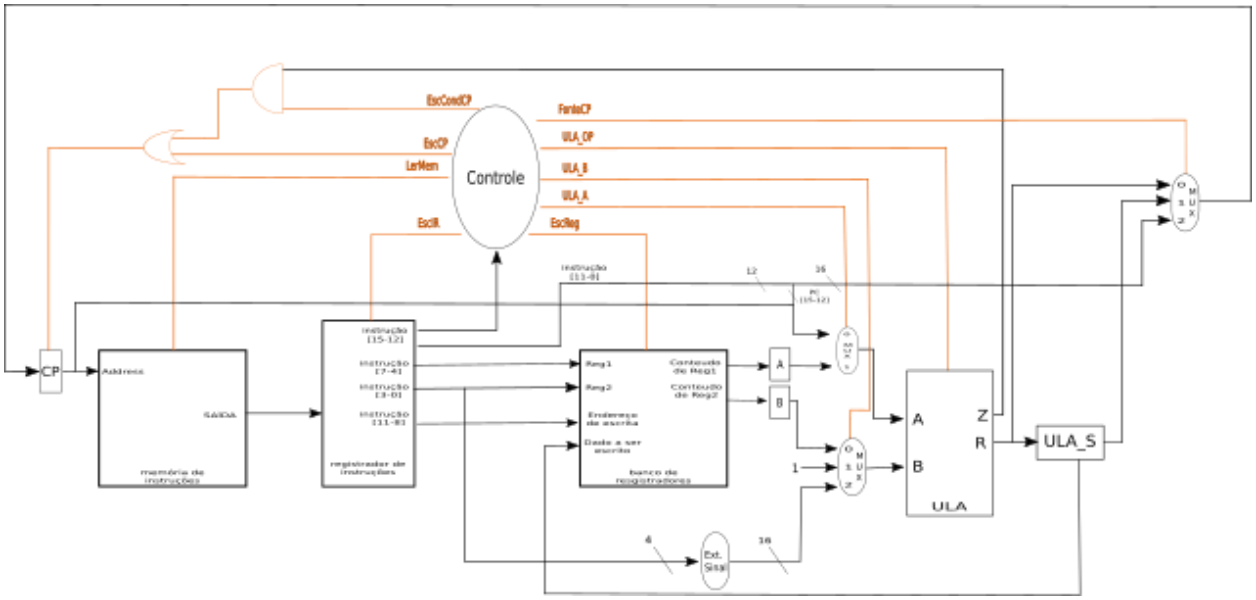
O registrador de instruções envia também os dois operandos a serem lidos do banco de registradores. E se houver imediato, o mesmo deve ser passado para o multiplexador da porta de entrada B da ULA.

Para o correto funcionamento da memória é necessário um registrador para manter o controle de qual linha da memória está sendo lida, isto é, que instrução está sendo executada. Esse registrador é chamado CP, **contador de programa**.

Com o controle da ordem de execução algumas instruções devem ser implementadas para permitir saltos entre diferentes rotinas, sejam eles *condicionais* ou *incondicionais*. Para isso, devem ser implementadas as instruções **Jump** e **Branch**. Tais instruções devem ter o seguinte comportamento:

CODOP	Instrução	Mnemônico	Operação
11	Jump	j Imm, Imm (12 bits)	$CP = Imm
12	Branch	bez - , $s3, $s2	Se $s3 = 0, CP = $s2

Para melhor visualização do caminho de dados utilizem a imagem abaixo:



Atentem para os sinais de controle:

- Por meio do caminho de dados pode-se perceber que os sinais *EscCondCP* e *EscCP* são sinais de controle para determinar se um salto condicional (*branch*) será tomado ou não.

- O sinal de 16 bits que é conectado na entrada 2 do multiplexador mais à direita na imagem, é o valor do endereço de salto das instruções *jump.* Esse é o

resultado da concatenação dos 4 bits mais significativos do registrador CP com os 12 bits de imediato da instrução.

- O valor imediato, de uma instrução, deve ser estendido de 4 bits para 16 bits, antes de ser utilizado na ULA. Excetuando instruções *jump.*

Sinal de Controle	Função
EscCondCP	‘1’ quando for instrução de branch quando não for permanece em ‘0’
EscCP	‘1’ quando for estágio de escrita no registrador CP
ULA_OP	Determina que operação deve ser realizada na ULA. Composto por 4 bits
ULA_A	Determina se o operador A da ULA é CP ou RegA.
ULA_B	Determina se o operador B da ULA é 1, imediato estendido ou RegB.
EscIR	‘1’ quando for estágio de leitura da memória e escrita da instrução em RI (registrador de instrução).
FonteCP	Determina qual o valor a ser escrito em CP: (CP + 1) [0], ULA_S[1] ou Endereço de salto [2].
EscReg	‘1’ quando for estágio de salvar no banco de registradores.

VALIDAÇÃO:

Como forma de validar a implementação feita, o aluno deve gravar na memória um programa para colocar em ordem decrescente um vetor de 5 inteiros. Os quais deverão ser gravados nos registradores de 11 a 15 do banco de registradores, sendo o registrador 11 o de maior valor, e consequentemente, o registrador 15 o de menor valor.

Não será cobrada execução em FPGA, apenas simulação (ModelSim). Dessa forma, o grupo deve gerar um *script* de simulação (arquivo .**do**) com o programa que o mesmo simulou.

ENTREGA:

O trabalho deve ser **entregue** por meio do moodle até o dia **06/11/2017**. No ato da entrega o aluno deve compactar tanto os arquivos de código fonte quanto o relatório dos resultados obtidos.

No relatório o aluno deve explicar seus detalhes de implementação e comentar os resultados obtidos nas simulações.