UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Departamento de Sistemas de Computação - SSC

SSC-110 Elementos de lógica digital I
SSC-111 Laboratório de elementos de lógica digital I

2º Semestre

Projeto CPU

O projeto consiste na implementação de uma CPU que lê um programa codificado de forma binária (conjunto de instruções e dados) na memória, e executa essas instruções. O conteúdo dos registradores deve ser apresentado no display após a execução de cada instrução.

Cada palavra da memória deve conter **pelo menos** 4 bits que codificam a instrução a ser executada e 6 bits de dados (utilizados nas operações aritméticas), na forma "iiiidddddd". O registradores devem ter no mínimo 4 bits. A CPU deve ser capaz de executar, **no mínimo**, as seguintes instruções:

Instruções básicas:

ADD - Soma o valor do dado ao valor de um registrador (ADD dddddd)
SUB - Subtrai o valor do dado ao valor de um registrador (SUB dddddd)
MUL2 - Multiplica o valor de um registrador por 2
DIV2 - Divide o valor de um registrador por 2
CLR - Zera o conteúdo de um registrador continua a execução do programa
RST - Zera o conteúdo de todos os registradores e reinicia a execução da primeira instrução na memória
MOV – Copia o conteúdo de um registrador em outro registrador (MOV regA,regB)
JMP – Altera a seqüência de execução das instruções definindo o endereço da próxima instrução a ser executada (JMP dddddd)

Instruções avançadas:

JZ - JMP condicional (JMP if Zero) salta se o conteúdo de um determinado registrador for zero (JZ dddddd)
CMP – Compara o conteúdo de 2 registradores (CMP regA, regB)
JE - JMP condicional (JMP if Equal) salta se o resultado do comando CMP executado anteriormente indicar que o conteúdo dos registradores forem iguais (JE dddddd)

O projeto deve ser elaborado utilizando-se **apenas** os seguintes componentes do quartus: **flip-flops, portas lógicas e memórias.**

A avaliação do projeto levará em conta:

a) o funcionamento correto do circuito
b) a originalidade no projeto do circuito
c) a **criatividade na extensão do projeto** através da ampliação do circuito e da criação de novas instruções
d) o desenvolvimento do projeto nas semanas que antecedem a apresentação
e) o número de integrantes no grupo (**no máximo 3**)

A avaliação será **individual** durante a apresentação do projeto.

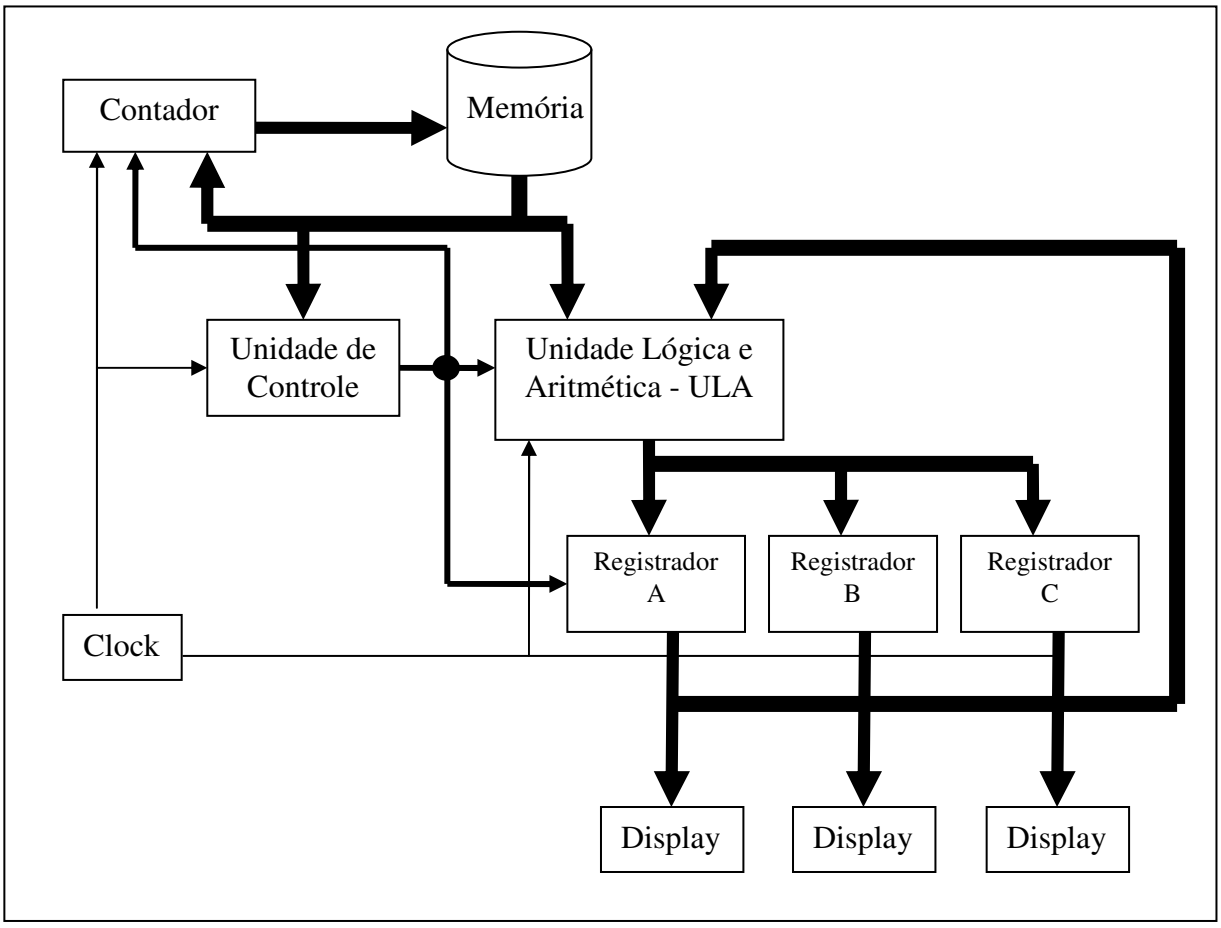


Diagrama do circuito de uma CPU com 3 registradores

Sugestões:

- Faça um projeto inicial da CPU, do fluxo de dados entre os componentes e do formato das instruções. **Inicie com um projeto simples,** e depois incremente as funcionalidades se houver tempo**.**

- Adicione cada componente do projeto de forma incremental, testando cuidadosamente sempre que um novo componente for adicionado.

- Ordem sugerida para o desenvolvimento do projeto:

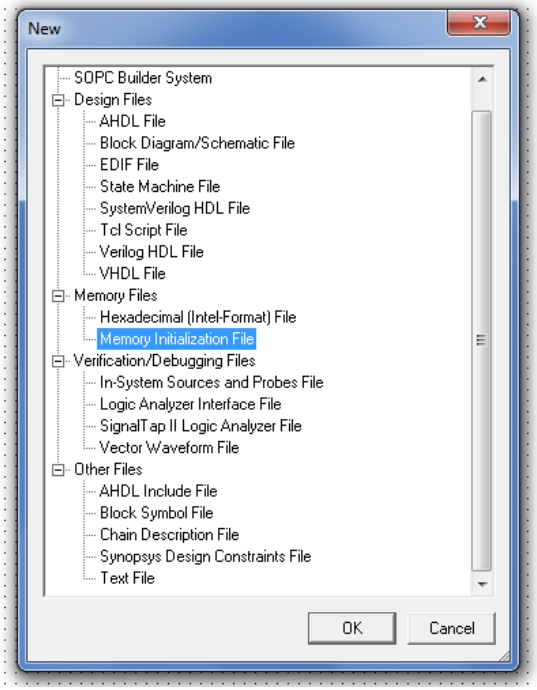
1) **Memória** – inserir os dados e testar usando chaves na entrada de endereço e display na saída de dados

2) **Contador** – Conectar um contador assíncrono na entrada de endereço da memória e verificar a saída no display

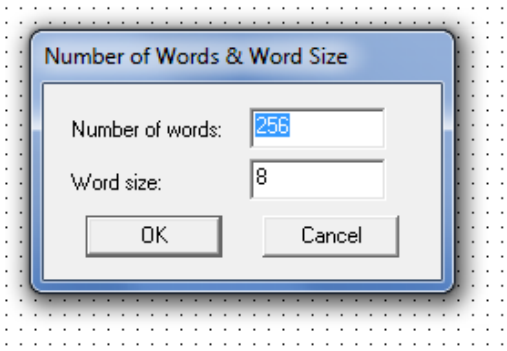
3) **ULA** – Conectar a saída de dados da memória em uma das entradas da ULA. A outra entrada pode ter um valor fixo. Verificar a saída da ULA no display.

4) **Controle de operação lógica**: utilizar um dos bits de instrução para selecionar a operação lógica executada pela ULA
5) **Registrador** – Adicione um registrador na saída da ULA e realimente a entrada da ULA com a saída do registrador. É necessário usar o click do circuito para sincronizar adequadamente o registrador.
6) **Contador** – Altere o contador para que ele possa executar instruções de "salto" e de "reset".
7) **Registrador** – Aumente o número de registradores e crie barramentos de dados entre eles.

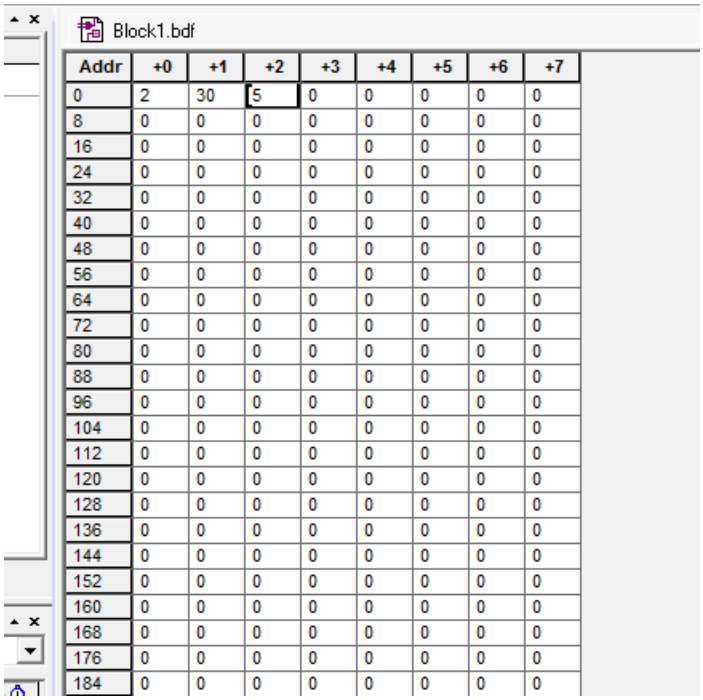
Utilização de memória ROM (somente leitura) no Quartus:
- Primeiro passo: criara um arquivo com o conteúdo (dados) da memória. O arquivo deve ser do tipo MIF (Memory Initialization File).



- Definir o tamanho da memória (número de palavras e tamanho das palavras). Ex: 256 palavras (8 bits de endereçamento) e tamanho de cada palavra de 8 bits.

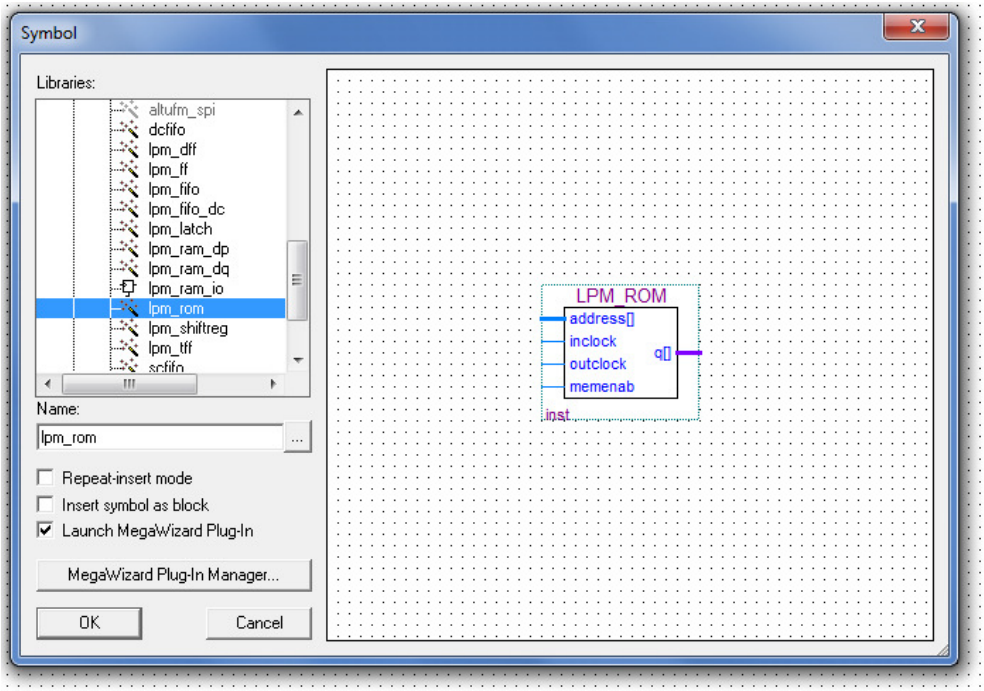


- Definir o conteúdo da memória e salvar o arquivo. Cada célula corresponde a uma palavra de dados.

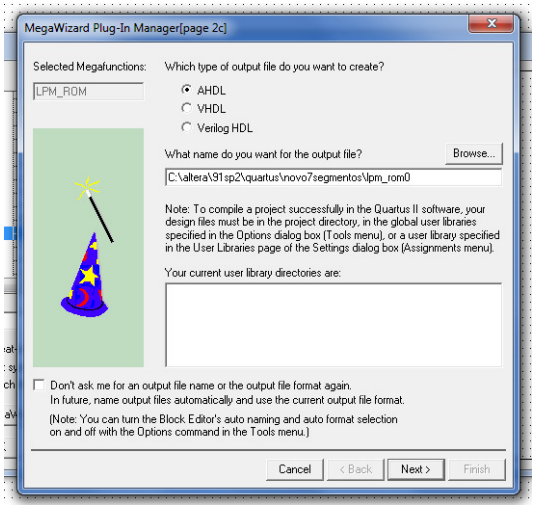


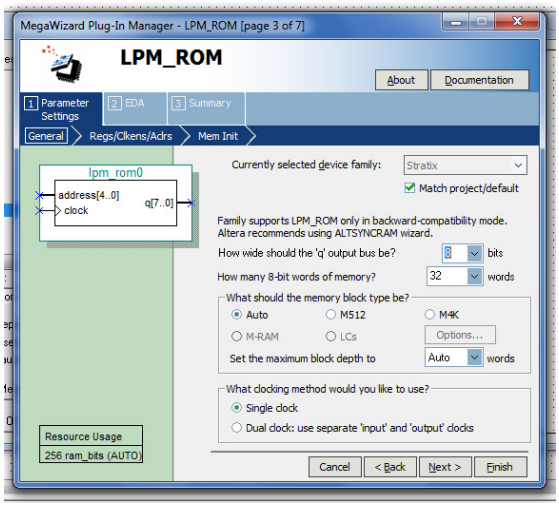
Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	2	30	5	0	0	0	0	0
8	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0
64	0	0	0	0	0	0	0	0
72	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0
88	0	0	0	0	0	0	0	0
96	0	0	0	0	0	0	0	0
104	0	0	0	0	0	0	0	0
112	0	0	0	0	0	0	0	0
120	0	0	0	0	0	0	0	0
128	0	0	0	0	0	0	0	0
136	0	0	0	0	0	0	0	0
144	0	0	0	0	0	0	0	0
152	0	0	0	0	0	0	0	0
160	0	0	0	0	0	0	0	0
168	0	0	0	0	0	0	0	0
176	0	0	0	0	0	0	0	0
184	0	0	0	0	0	0	0	0

- Criar a memória. O componente correspondente a memória ROM é: lpm_rom

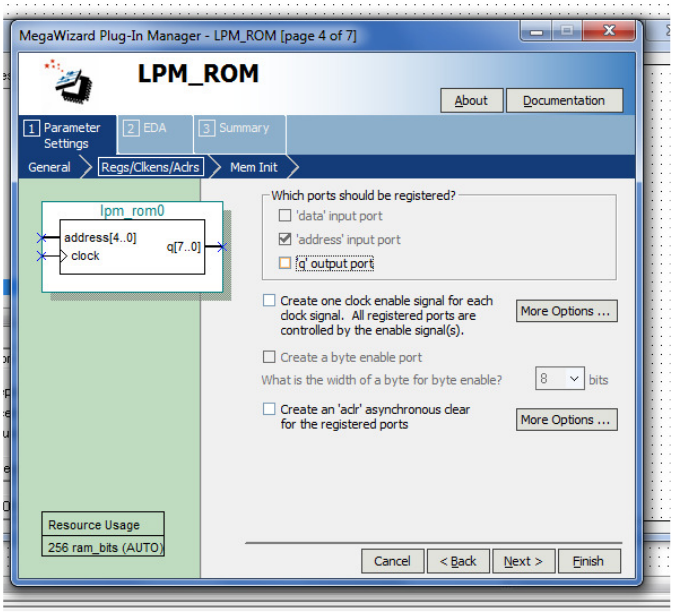


- Configurar a memória (single clock). O número de palavras e o tamanho das palavras devem ser os mesmos especificados no arquivo de conteúdo, criado anteriormente.

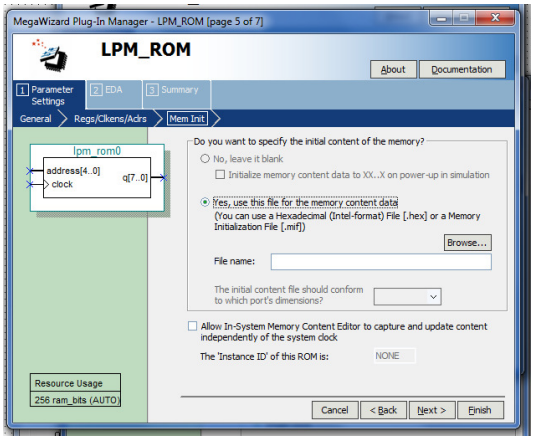


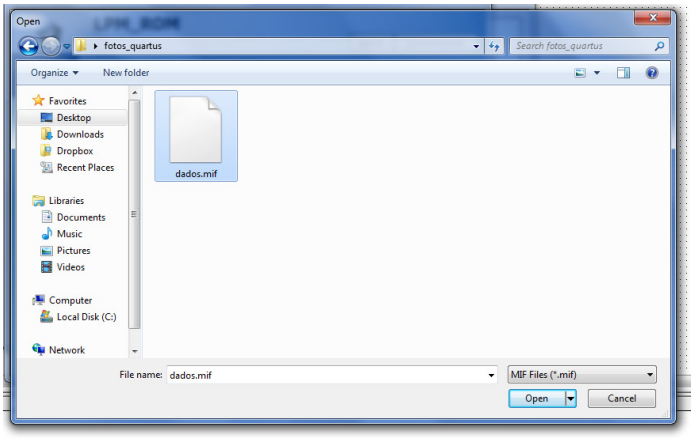


- Continuar a configuração da memória. Deve ser desabilitada a opção que registra a saída ‘q’, conforme a figura abaixo.

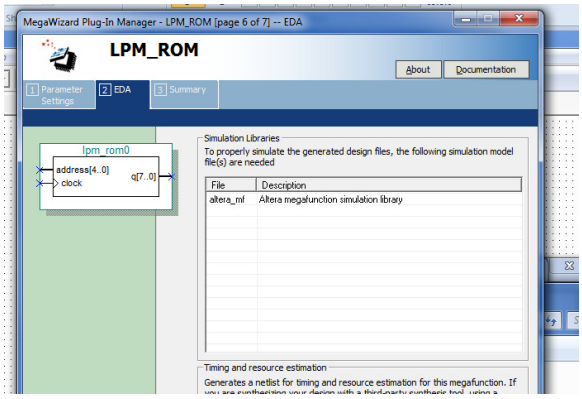


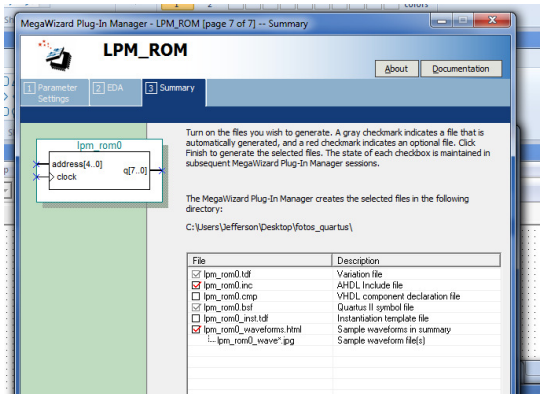
- Carregar o arquivo de inicialização (dados) da memória criado anteriormente, **tipo MIF.**





- Concluir a configuração da memória:





- A entrada de endereços e a saída de dados da memória são barramentos (vias de dados em paralelo). Para utilizá-las como vias simples, basta atribuir um nome ao barramento, com o número de vias de dados entre colchetes. Ex com oito vias de dados (de 7 a 0, onde 0 é o bit menos significativo): memdados[7..0]. Os fios simples devem ser conectados ao barramento, ter o mesmo nome e a indicação do bit correspondente entre colchetes, conforme figura abaixo.