"FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
UNIVEM

Bacharelado em Ciência da Computação

Disciplina

Organização e Arquitetura de Computadores

Capítulo III

Subsistemas de Uma UCP

prof. - Ildeberto de Genova Bugatti

bugatti.univem@gmail.com

Tema

Subsistemas Básicos de Uma UCP – Síntese, Projeto, Implementação e Validação.

Identificação dos pré-requisitos

São considerados pré-requisitos para o efetivo acompanhamento do tema os seguintes conceitos ou técnicas: bases numéricas, representação de números nas bases dois e hexadecimal, álgebra de boole, lógica digital; síntese, projeto, implementação e validação de circuitos lógicos digitais combinacionais e seqüenciais; elementos multiplexadores; elementos decodificadores; técnicas de associação de componentes eletrônicos em série e paralelo.

Objetivos

O principal objetivo do capítulo é o de capacitar o aluno a projetar, dimensionar, implementar e avaliar sistemas computacionais de acordo com as necessidades e características de uso dos mesmos.

Uma arquitetura de computador deve ser avaliada ou dimensionada de acordo com as características de sua aplicação. Ou seja, não podemos simplesmente afirmar que uma arquitetura é eficiente (boa) ou ineficiente (ruim) se não for considerada a aplicação na qual ela está sendo empregada.

8.1- Estruturas Básicas de Arquiteturas de Computadores

Os computadores são formados por vários subsistemas digitais, interligados através de vários conjuntos de fios denominados barramentos. O sistema formado pelos subsistemas digitais e os barramentos, aliado a forma como eles estão interligados para realizar a troca de informações é denominada de Arquitetura de um Computador. Entre as arquiteturas precursoras dos computadores digitais, destaca-se a Arquitetura de Von Neumann. Os subsistemas digitais básicos que constituíam o computador de Von Neumann foram denominados de: Memória; Unidade Lógica Aritmética (ULA), Unidade de Controle (UC) e Unidades de Entrada e Saída (UES), interligados por conjuntos de fios organizados tanto através das funções exercidas pelos mesmos quanto pelas características ou tipo das informações que trafegam através deles. A Figura 8.1 mostra essa arquitetura.

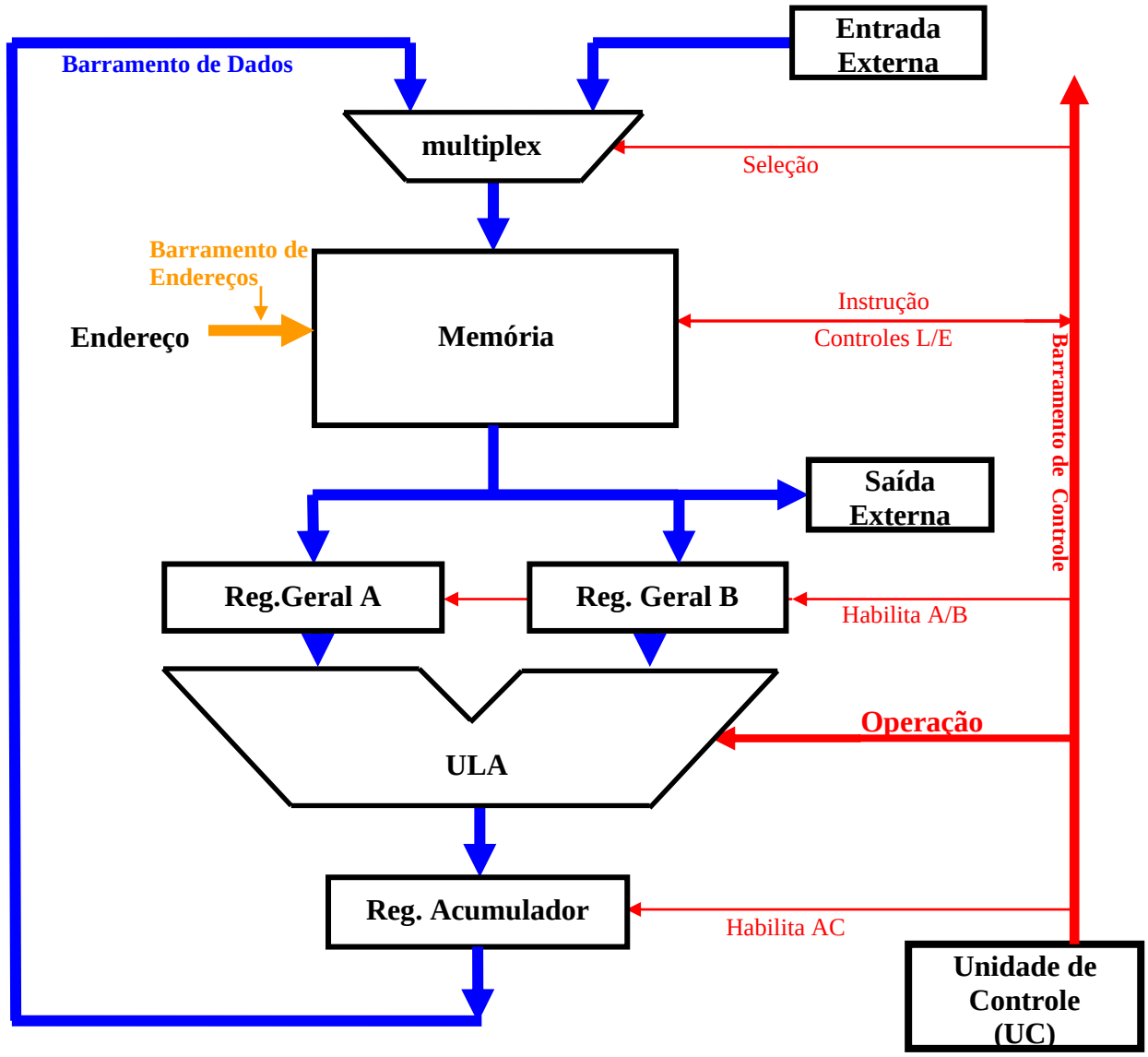


Figura 8.1- Arquitetura de Von Neumann

Essa denominação, “Von Neumann”, foi dada para homenagear o pesquisador que a idealizou, pois a forma de interligar os subsistemas da arquitetura incluindo subsistema de memória foi considerada, na época, inovadora e trouxe enormes contribuições para o avanço da área de computação tanto em hardware quanto de software possibilitando também a integração dessas duas áreas. A arquitetura de Von Neumann foi a primeira arquitetura que continha programas armazenados na memória, sendo considerada a principal inovação e contribuição dessa arquitetura para o desenvolvimento dos computadores digitais. Até hoje, essa arquitetura está presente na maioria dos processadores comerciais, que utilizam os mesmos subsistemas com formas alternativas ou diferenciadas de interligá-los, incluindo inovações que o desenvolvimento tecnológico na área de semicondutores e microeletrônica viabilizaram.

Uma Arquitetura de um Computador Digital convencional é formada por três subsistemas principais: **Memória, Unidades de Entrada e Saída (UES) e Unidade Central de Processamento (UCP).** A UCP, por sua vez, é formada pelas seguintes subsistemas e unidades: **Unidade Lógica Aritmética; Registradores de Uso Geral; Registradores Acumuladores e Unidade de Controle**. Interligando esses subsistemas existem três conjuntos de fios para realizar a comunicação entre eles, esses conjuntos de fios são organizados de acordo com a função dos mesmos e denominados de barramentos. A organização desse universo de fios em subconjuntos é realizada da seguinte maneira: subconjunto de fios que transportam endereços de memória e endereços de unidades de entrada e saída, denominado "**Barramento de Endereços**"; subconjunto de fios que transportam dados, denominado "**Barramento de Dados**" e subconjunto de fios que realizam a ordenação e a correta seqüência de atividades a serem realizadas na arquitetura para a obtenção de um resultado desejado, denominado "**Barramento de Controle**", oriundos da Unidade de Controle. A Figura 8.2 ilustra uma evolução da arquitetura de Von Neumann. Deve-se observar que na arquitetura proposta por Von Neumann a Unidade de Controle era um subsistema separado da UCP. Na arquitetura contida na Figura 8.2 a UCP já contém a Unidade de Controle. Cada um dos subsistemas contido nessa máquina possui ou executa funções específicas:

- o **Subsistema de Memória** tem a função de armazenar (conter) informações temporariamente. O subsistema de memória está subdividido em Memória Principal e Memória Secundária. Os conceitos, características e formas de organização dessas memórias estão contidos no item Subsistemas de Memória (cap.VII).

- a **Unidade de Entrada e Saída** (**UES**) tem a função de realizar a comunicação entre o Homem e a Máquina e também a comunicação entre equipamentos. A UES contém os subsistemas que controlam tanto, os equipamentos periféricos de um computador (mouse, teclado, monitor de vídeo, impressora, disco rígido, unidade de disquetes, unidade de CD/DVDs, subsistemas de multimídia, entre outros), quanto, subsistemas que possibilitam a comunicação entre equipamentos, tais como: placas de rede, placas de fax-modem e muitos outros. Os conceitos e formas de implementação de Unidades de Entrada e Saída serão apresentados no capítulo IX.

- a Unidade **Central de Processamento** (**UCP**) é formada pela Unidade Lógica e Aritmética (ULA), registradores de uso geral, registrador acumulador e Unidade de Controle (UC), conforme mostra a Figura 8.1. Na Unidade Lógica e Aritmética são executadas as

instruções contidas em um programa. Os registradores de uso geral armazenam temporariamente os operandos das instruções a serem executadas pela ULA. O registrador Acumulador armazena temporariamente os resultados das operações executadas na ULA. A Unidade de Controle gera os comandos (microordens) na seqüência adequada para executar as atividades inerentes a cada programa em execução e a comunicação com o exterior (Unidades de Entrada e Saída). Essa unidade coordena o instante exato de execução das atividades para obter o resultado esperado. Fazendo uma comparação grosseira com o corpo humano, a Unidade de Controle é equivalente ao sistema de coordenação motora. Os elementos básicos de uma UCP, com exceção da Unidade de Controle, serão apresentados de forma mais detalhada do item 8.2.1 ao item 8.2.6. As características, tipos e propostas de implementação de uma Unidade de Controle serão apresentadas no capítulo IX

Nesse capítulo serão descritas as diferentes formas de interligar, controlar e organizar esses subsistemas tanto a nível teórico quanto prático através da implementação um computador com arquitetura reduzida, onde os conceitos serão verificados e consolidados. Cada aluno irá propor, sintetizar, simular e implementar o seu computador. Após a implementação inicial, propostas de modificações serão inseridas na arquitetura original com o objetivo de realizar análises de desempenho e relações de custo-benefício das modificações inseridas na arquitetura original. Dessa forma, serão repassados aos alunos os conceitos de organização e arquitetura de computadores possibilitando comparações e a análise de desempenho das diversas alternativas e também de subsistemas contidos em uma arquitetura de computador convencional. As atividades práticas de implementação das máquinas propostas pelos alunos serão realizadas utilizando componentes programáveis denominados FPGAs. Serão utilizadas placas contendo FPGAs e o ambiente de desenvolvimento da mesma empresa contendo o software denominado “Project Manager”. Esse software contém ferramentas que possibilitam a edição e simulação de componentes e/ou subsistemas. A edição pode ser realizada nas formas: gráfica; diagrama de estados e linguagem VHDL. Inicialmente utilizaremos a edição gráfica para projetar os componentes de um computador com arquitetura de Von Neumann expandida.

Um dos objetivos do texto é capacitar cada leitor para implementar o seu próprio conjunto de subsistemas de um computador digital com o objetivo de gerar um processador com arquitetura equivalente a mostrada na Figura 8.1. Para tanto, os alunos devem aprender e dominar as principais características funcionais de cada um dos subsistemas utilizados. Os subsistemas e elementos auxiliares contidos na arquitetura da Figura 8.2 serão todos sintetizados, projetados, implementados e validados através de simulação. Após essa fase os

componentes implementados serão integrados para a obtenção de um computador básico com arquitetura de Von Neumann Expandida da Figura 8.2 e com as seguintes características principais: palavra de oito bits e memória RAM estática de 1024 palavras de 8 bits. Essa implementação pode ser verificada na Figura 8.17.

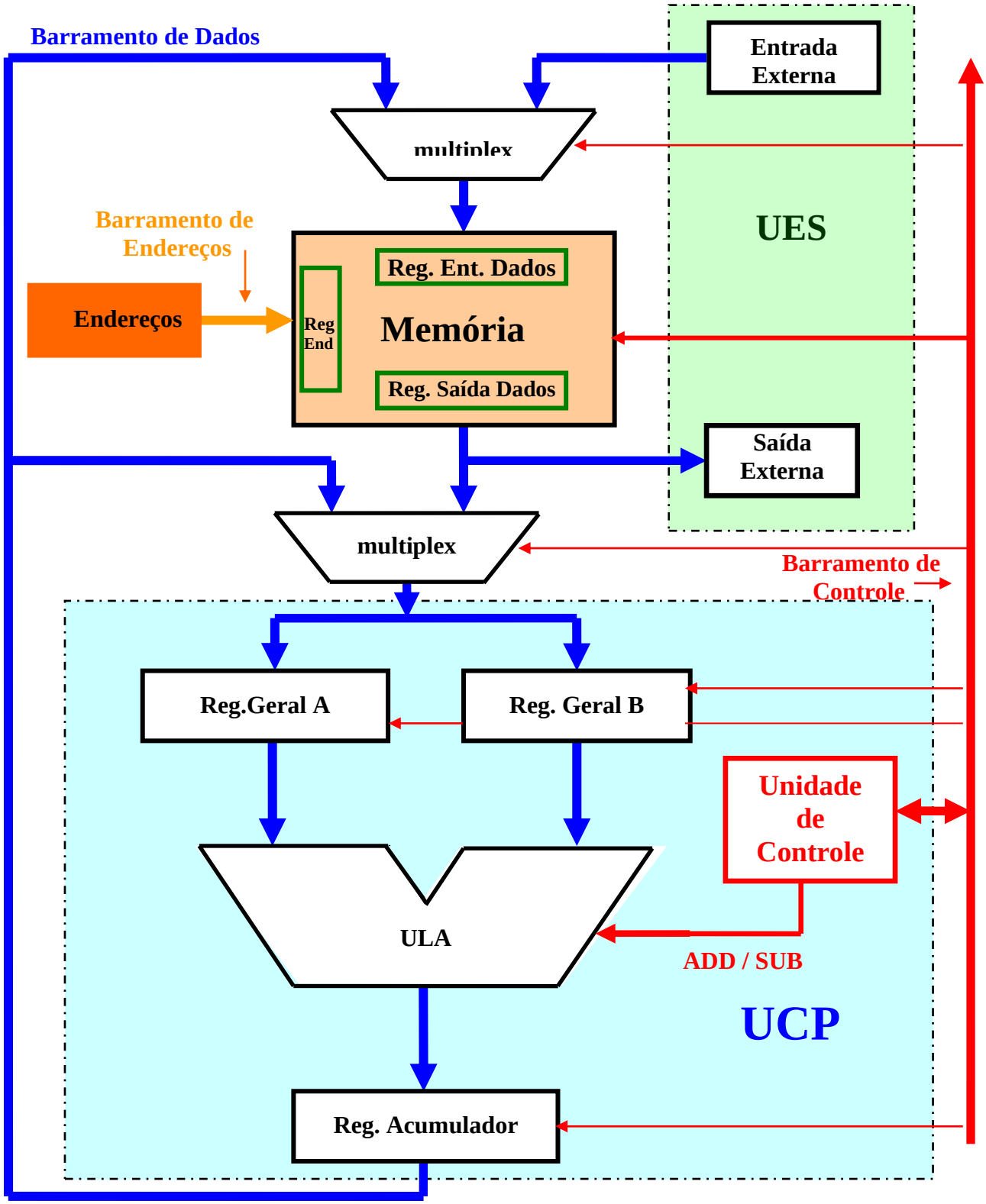


Figura 8.2 - Arquitetura de Von Neumann Expandida

8.2- Conceitos e descrição dos componentes básicos de uma UCP

Os conceitos e técnicas necessárias para utilizar, dimensionar, propor, organizar e implementar arquiteturas de computadores serão apresentados nas seções que seguem. A seqüência de apresentação, discussão, proposta, dimensionamento e implementação dos principais subsistemas de uma arquitetura será a seguinte: Registradores de Uso Geral; Registrador Acumulador (AC); Registradores Contadores; Elementos Multiplexadores; Unidade Lógica Aritmética (ULA), Unidade Central de Processamento (UCP); Subsistema de Memória e Unidade de Controle. Os subsistemas implementados serão apresentados da seguinte maneira: descrição; diagrama esquemático; implementação realizada no software Project Manager e tabela verdade descrevendo o funcionamento do elemento em foco.

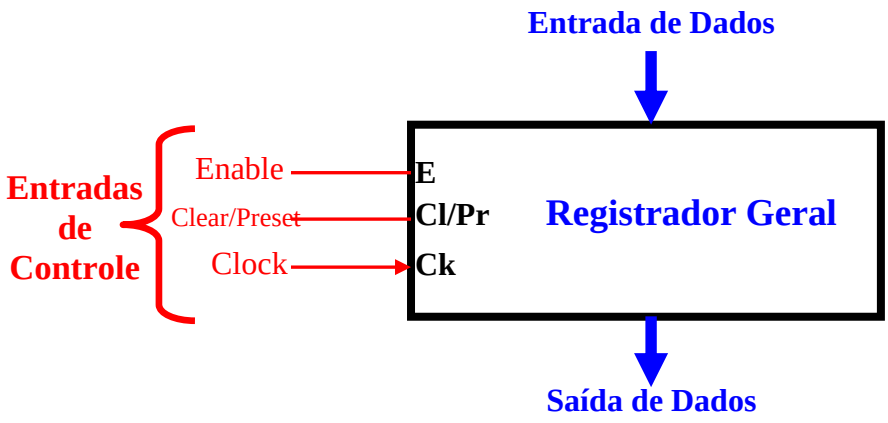
8.2.1- Registradores de Uso Geral

Um registrador de Uso Geral é um dos elementos que compõem a Unidade Central de Processamento (UCP). A função dos registradores gerais é conter momentaneamente os operandos das instruções que serão executadas pela Unidade Lógica Aritmética (ULA). Os registradores de Uso geral estão localizados nas entradas da ULA e pode receber dados tanto do Subsistema de Memória quanto do Registrador Acumulador.

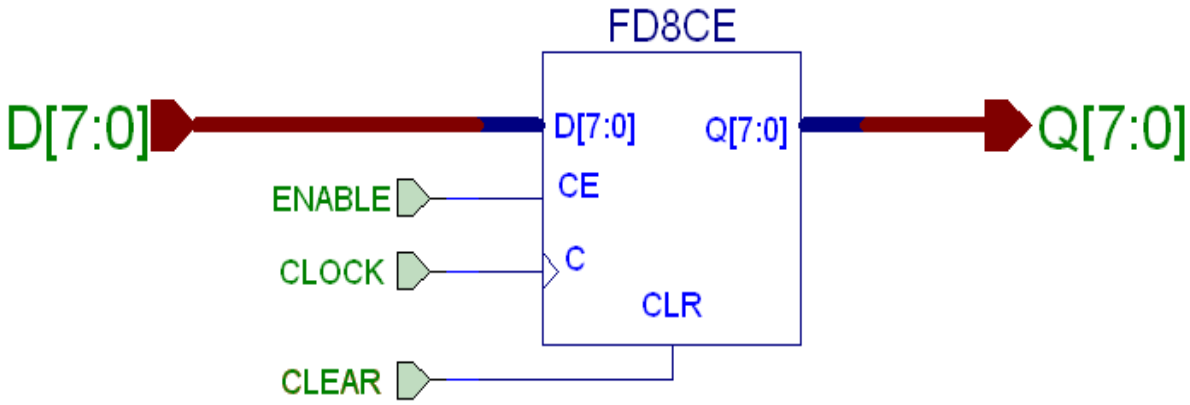
Os Registradores de Uso Geral é classificado como um registrador que possui Entradas Paralelas e Saídas Paralelas, quando se considera as características de suas entradas e saídas de dados. Assim, um Registrador de Uso Geral possui Entrada de Dados Paralela e Saída de Dados Paralela. Do ponto de vista lógico e temporal, um registrador de uso geral funciona na forma e seqüência descrita a seguir: o registro (dados) contido na sua Entrada de Dados é armazenado no Registrador quando a entrada de controle denominada “Enable” (habilitar) assumir o nível lógico “1” (um), e a entrada de controle denominada “Clear” (limpar) assumir o valor lógico “0” (zero) e no instante que acontecer uma borda de subida na entrada de controle de “Clock” (relógio). As Entradas “Clear" e “Preset” não podem ser ativadas simultaneamente. A entrada de controle “Clear” predomina sobre as outras entradas de controle. Quando ela assumir o valor lógico “1” (um) o conteúdo do Registrador Geral é zerado ou limpo, assim, todas as suas Saídas de Dados ficam iguais a zero, independente dos valores contidos em todas as outras entradas, tanto de controle quanto de dados. Quando a entrada de controle “Preset” estiver ativa (Preset=1), o conteúdo de todos os flip-flops

existentes no Registrador Geral assumirá o valor lógico “1”, ou sela todas as linhas da Saída de Dados assumirão o valor lógico “1”.

A Figura 8.3a mostra o diagrama de blocos de um Registrador de Uso Geral. A Figura 8.3b mostra a implementação do Registrador de Uso Geral no Software “Project Manager”. A Tabela 8.1 descreve a forma de funcionamento do Registrador da Figura 8.3b.



(a) Diagrama de Blocos de um Registrador de Uso Geral



(b)- Uma Implementação do Registrador de Uso Geral no software Project Manager

Figura 8.3- Registrador de Uso geral

Entrada de Dados	Entradas de Controle			Saída de Dados	Comentários
	Enable	Clock	Clear		
D_{7i} ... D_{0i}	0	0	0	D_{7i-1} ... D_{0i-1}	Mantém o estado anterior
D_{7i} ... D_{0i}	0	↑	0	D_{7i-1} ... D_{0i-1}	Mantém o estado anterior
D_{7i} ... D_{0i}	1	↑	0	D_{7i} ... D_{0i}	Armazena as entradas atuais
XX_H	X	X	1	00_H	Limpa (zera) as Saídas

Tabela 8.1- Tabela Verdade do Registrador Geral da Figura 8.3b

Exercício 8.1- Sintetizar e projetar um registrador de uso geral de 32 bits utilizando um componente registrador de apenas 16 bits.

Resolução-

Para realizar a síntese e implementação do registrador de uso geral de 32 bits solicitado é requerido o conhecimento dos seguintes itens:

a- funcionamento e tabela verdade do componente eletrônico disponível para ser utilizado no projeto

b- conceitos e técnicas de síntese e projeto de circuitos combinacionais e seqüenciais (associação série-paralelo de componentes eletrônicos).

Item a- *O registrador de 16 bits disponibilizado para o projeto é mostrado na figura 8.4 possui a tabela verdade equivalente ao registrador de 8 bits cujas características estão contidas na tabela 8.1 e figura 8.3.*

Item b- *Um registrador de 16 bits como o FD16CE é caracterizado como uma memória de 1 palavra de 16 bits com entradas paralelas e saídas paralelas. Para obter uma memória de uma palavra de 32 bits basta interligar dois registradores de 16 bits através de uma associação paralela. Dessa forma foi obtido o registrador de uso geral de 32 bits contido na figura 8.5 foi obtido através da associação paralela de dois componentes FD16CE.*

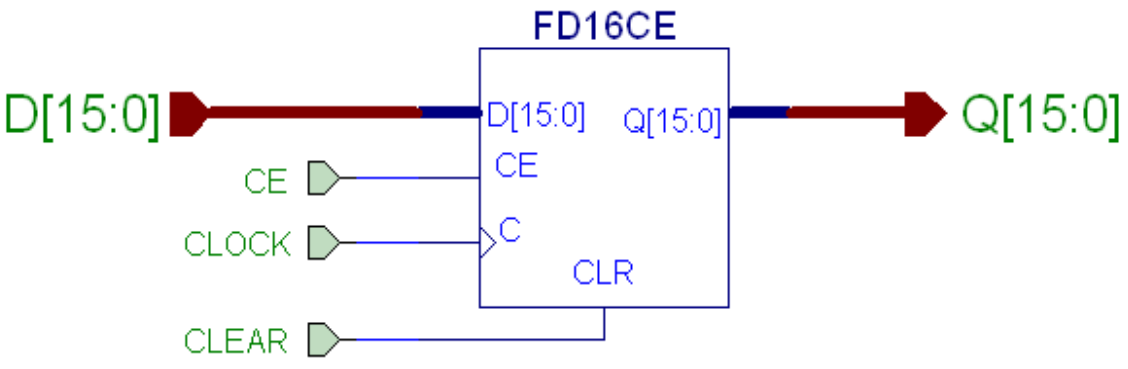


Figura 8.4- Componente Registrador FD16CE do software Project Manager

A figura 8.5 mostra o registrador de uso geral de 32 bits. Nela, verifica-se as seguintes características: -

- as entradas de controle: CE (Chip Enable), C (Clock) e CLR (Clear); dos dois componentes FD16CE estão interligas em uma mesma origem;

- o barramento de entrada de dados de 32 bits (D_RG[31:0] está subdividido em dois sub-barramentos de 16 bits; D_RG[31:16] e D_RG[15:0] respectivamente.

- o barramento de saída de dados de 32bits, denominado "Q_RG[31:0]" é composto pelos dois sub-barramentos de 16 bits denominados "Q_RG[31:16]" e "Q_RG[15:0]", originários nos dois componentes de FD16CE respectivamente.

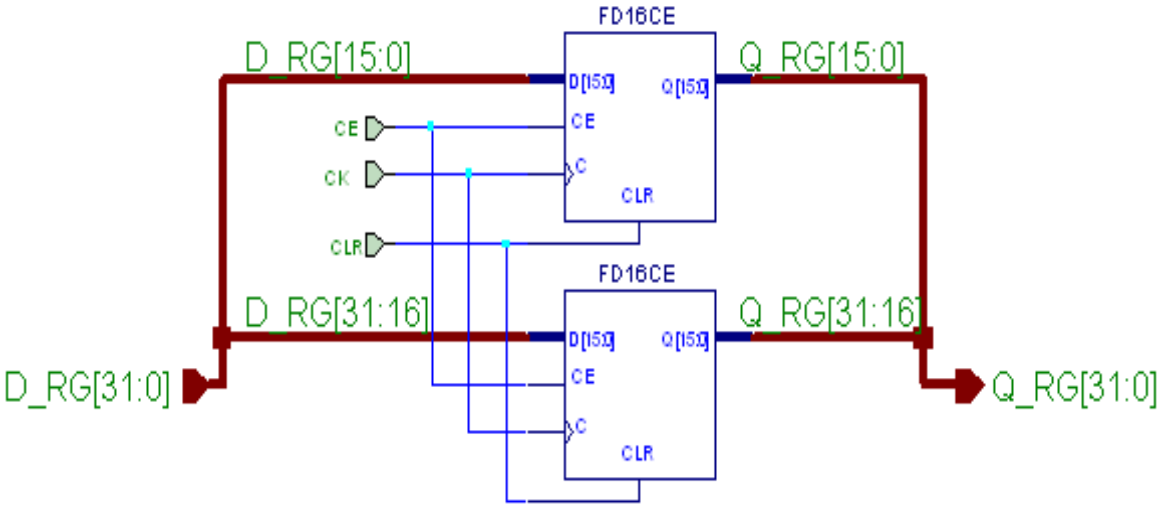


Figura 8.5- Registrador de Uso Geral de 32 bits

A tabela 8.2 mostra o funcionamento do Registrador de Uso Geral de 32 bits obtido. Ou seja, o registrador de uso geral armazena em seu interior o conteúdo disponibilizado no barramento de entrada de dados somente quando ocorrer a seguinte configuração nas suas entradas de controle: CLR =0, CE=1 e CK = ↑. Quando a entrada de controle CLR=1 o conteúdo do registrador é 00000000(H) e Quando CLR=0 e CE=0 o registrador mantém o conteúdo de seu estado anterior, independente dos valores de todas as outras entradas.

Entradas				***Saídas (Dados)***
Controle			***Dados***	
CLR	***CE***	***CK***	***D_RG[31:0]***	***Q_RG[31:0]***
1	*X*	*X*	$XXXXXXXX_{(H)}$	$00000000_{(H)}$
0	*0*	*X*	$XXXXXXXX_{(H)}$	*Mantém estado anterior*
0	*1*	↑	D_RG_{31i} ... D_RG_{0i}	D_RG_{31i} ... D_RG_{0i}

Tabela 8.2- Tabela Verdade do Registrador Geral de 32 bits da Figura 8.5

8.2.2- Registrador Acumulador (AC)

O registrador Acumulador (AC) é mais um elemento presente na Unidade Central de Processamento. Ele fica localizado na saída da ULA e sua função é armazenar temporariamente os resultados das operações executadas na ULA, para posterior utilização pela própria ULA, via realimentação da ULA através dos registradores gerais ou para serem armazenados na memória principal.

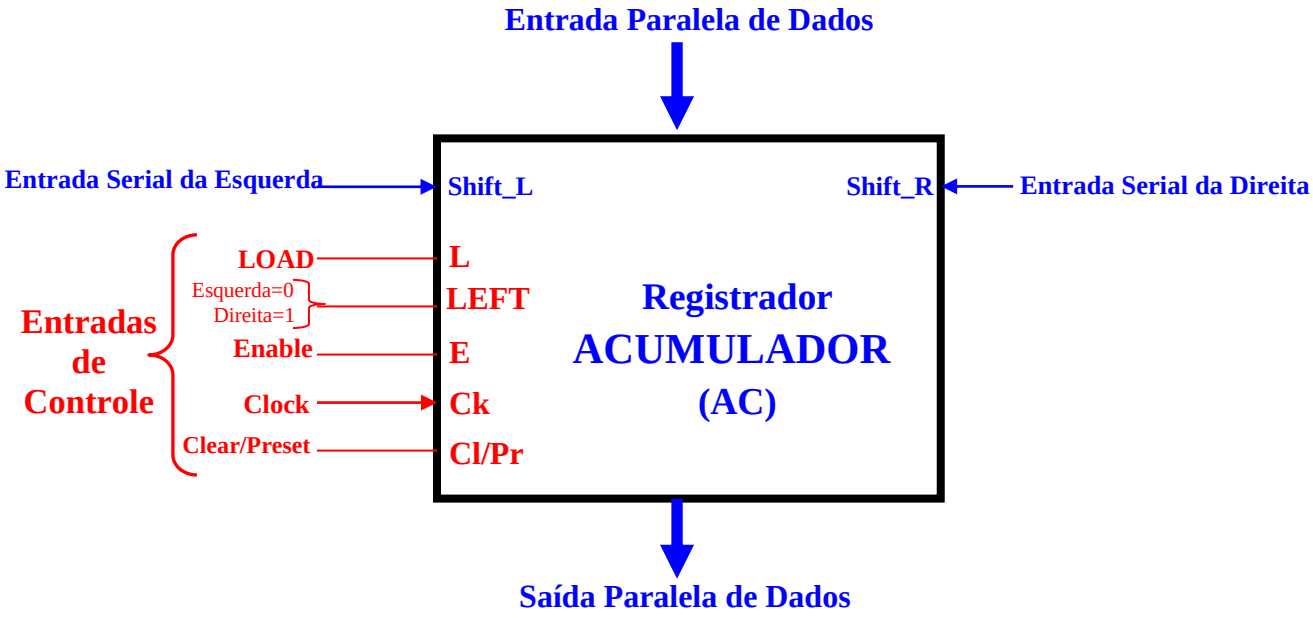
Geralmente, o Registrador Acumulador possui as seguintes características: Entrada Paralela; Entrada Serial a Esquerda com Deslocamento a Direita; Entrada Serial a Direita com Deslocamento a Esquerda; Saída Paralela; Saída Serial Direita; Saída Serial a Esquerda. A Figura 8.6a mostra um Registrador Acumulador em diagrama de blocos. A Figura 8.6b mostra uma implementação do Acumulador no software Project Manager. Para tanto foi utilizado um registrador de oito bits denominado SR8CLED. O registrador SR8CLED é um registrador com entradas paralelas e seriais, deslocamento bidirecional (à esquerda e a direita). A Tabela 8.3 descreve a forma de funcionamento do Acumulador da Figura 8.6b, mostrando as entradas e saídas de dados com os respectivos valores das entradas de controle para cada uma das funções exercidas pelo Acumulador. A entrada de controle “LOAD” tem a função de armazenar no Acumulador os dados contidos na Entradas de Dados Paralelo.

Para armazenar os valores contidos na Entrada Paralela de Dados em paralelo as entradas de controle devem assumir a seguinte configuração: ENABLE=1, LOAD=1, CLEAR=0 e gerar borda de subida na entrada de controle de relógio (CLOCK= ↑); os valores das demais entradas de controle são irrelevantes na função de armazenamento paralelo de dados.

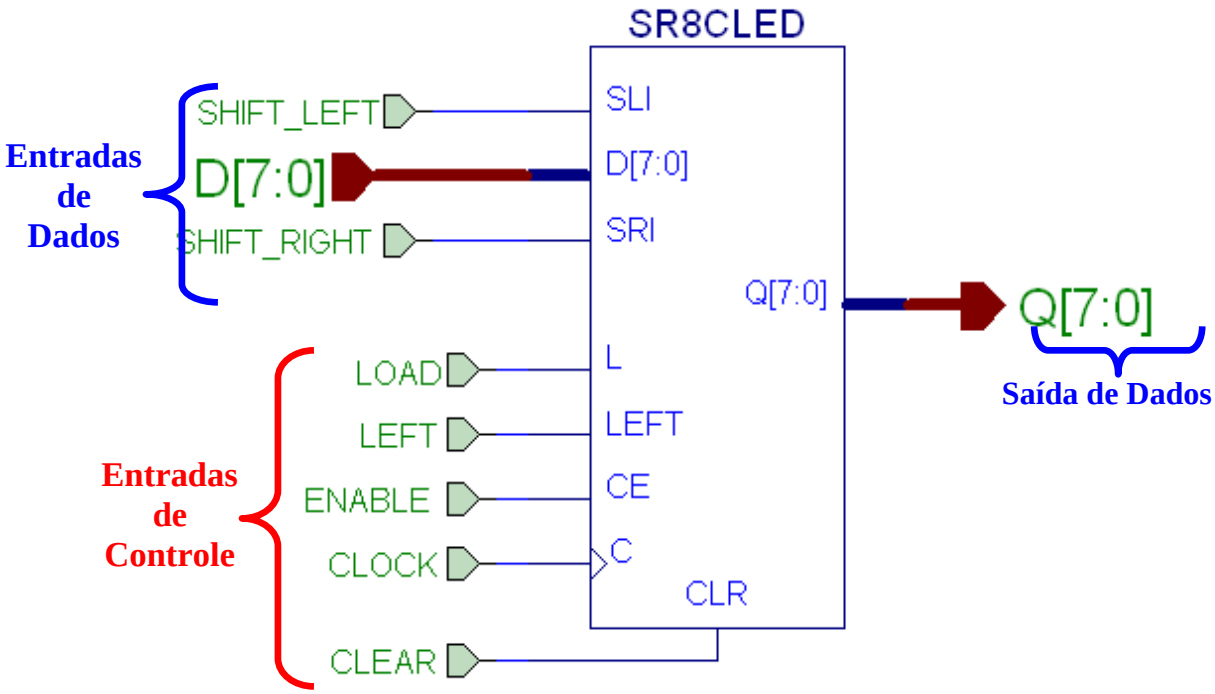
Para deslocar os dados contidos no registrador da esquerda para a direita e inserir o valor contido na entrada de dado serial (SHIFT_R) as entradas de controle devem assumir a seguinte configuração: ENABLE=1; LOAD=0; LEFT=0; CLEAR=0 e gerar borda de subida na entrada de controle de relógio (CLOCK= ↑).

Para deslocar os dados contidos no registrador da direita para a esquerda e inserir o valor contido na entrada de dado serial (SHIFT_L) as entradas de controle devem assumir a

seguinte configuração: ENABLE=1; LOAD=0; LEFT=1; CLEAR=0 e gerar borda de subida na entrada de controle de relógio (CLOCK= ↑).



(a) Diagrama de Blocos de um Registrador Acumulador



(b)- Uma Implementação do Registrador Acumulador no software Project Manager

Figura 8.6- Registrador Acumulador

A Tabela 8.3 contém as formas de funcionamento do registrador Acumulador da Figura 8.6b. Os valores das entradas de controle estão especificados para cada uma das funções exercidas pelo Acumulador.

Observe que o Acumulador permite entrada de dados nas formas paralela e serial. Como conseqüência da disponibilidade de entradas seriais o Acumulador possui a capacidade de deslocar o conteúdo armazenado em seus Flip-Flops em dois sentidos (deslocamento bidirecional), tanto da esquerda para a direita quanto da direita para a esquerda. O deslocamento para a direita é equivalente à operação matemática de divisão por 2, e o deslocamento a esquerda é equivalente à operação matemática de multiplicação por 2.

Entrada de Dados			Entradas de Controle					**Saída de Dados**	Comentários
Paralelo	Serial								
	Shift_L	Shift_R	Enable	Load	Left	Clock	Clear		
$D_{7i} \dots D_{0i}$	X	X	0	0	X	X	0	$D_{7i-1} \dots D_{0i-1}$	Mantém estado anterior
$D_{7i} \dots D_{0i}$	X	X	X	1	x	0	0	$D_{7i-1} \dots D_{0i-1}$	Mantém estado anterior
$D_{7i} \dots D_{0i}$	X	X	X	1	x	↑	0	$D_{7i} \dots D_{0i}$	Armazena entradas atuais
$D_{7i} \dots D_{0i}$	X	0	1	0	0	↑	0	$0, D_{6i} \dots D_{0i}$	Desloca direita, insere Shift_R (divide por 2)
$D_{7i} \dots D_{0i}$	X	1	1	0	0	↑	0	$1, D_{6i} \dots D_{0i}$	Desloca direita, insere Shift_R (divide e soma 1)
$D_{7i} \dots D_{0i}$	0	X	1	0	1	↑	0	$D_{7i} \dots D_{1i}, 0$	Desloca esquerda, insere Shift_L (multiplica por 2)
$D_{7i} \dots D_{0i}$	1	X	1	0	1	↑	0	$D_{7i} \dots D_{1i}, 1$	Desloca esquerda, insere Shift_L (multiplica,soma1)
XX_H	X	X	X	X	X	X	1	00_H	Limpa (zera) as Saídas

Tabela 8.3- Tabela Verdade do Registrador Acumulador Figura 8.6b

Exercício 8.2- Sintetizar e implementar um registrador acumulador de 32 bits através da utilização de um componente registrador acumulador de apenas 16 bits.

Resolução-

Para realizar a síntese e implementação do registrador acumulador de 32 bits solicitado é necessário dominar os seguintes itens:

a- funcionamento e tabela verdade do componente eletrônico disponível para ser utilizado no projeto

b- conceitos e técnicas de síntese e projeto de circuitos combinacionais e seqüenciais (associação série-paralelo de componentes eletrônicos).

***Item a**- O registrador acumulador de 16 bits disponibilizado para o projeto "SR16CLED" é mostrado na figura 8.7 possui a tabela verdade equivalente ao registrador de 8 bits cujas características estão contidas na tabela 8.3 e figura 8.6. O componente "SR16CLED" pode ser utilizado como registrador acumulador pois*

possui as características necessárias e intrínsecas a um registrador acumulador: entrada e saídas paralelas, entradas e saídas seriais e deslocamento bidirecional.

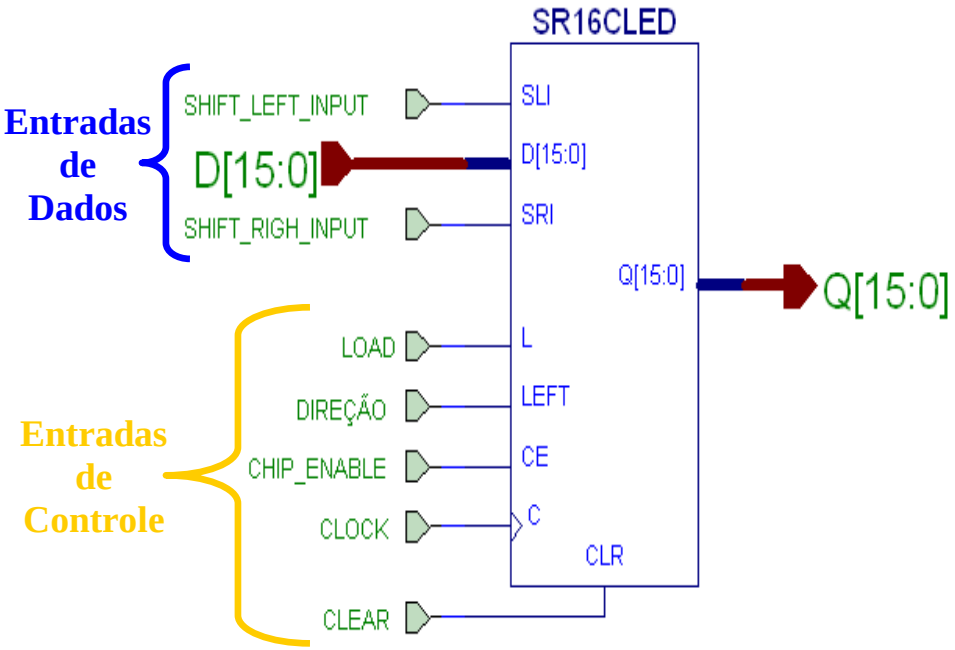


Figura 8.7- Componente Registrador SR16CLED do software Project Manager

Item b- *Um registrador acumulador de 16 bits como o SR16CLED também é caracterizado como uma memória de 1 palavra de 16 bits. No entanto esse componente possui a capacidade de modificar o seu conteúdo através de operações de controle que realiza o deslocamento bidirecional (direita ou esquerda) de seu conteúdo. Para obter uma memória de uma palavra de 32 bits utilizando dois registradores com entrada paralela e saída paralela basta interligar dois registradores de 16 bits através de uma associação paralela. Como o registrador, além de armazenara dados em paralelo também possui a capacidade de realizar entrada de dados de forma serial e em sentido bidirecional. Assim, além da associação paralela de dois componentes SR16CLED é necessário verificar a dependência de dados existente entre os dois registradores nas operações que exigem deslocamento bidirecional de seu conteúdo.Considerando todas as dependências de dados existentes, foi gerado o registrador acumulador de 32 bits contido na figura 8.8.*

Verifica-se na figura 8.8 que o bit 15 do componente SR16CLED, que armazena a parte menos significativa do conteúdo do registrador acumulador, está interligada à entrada serial de dados SRI (Shift_Right_Input) permitindo o deslocamento à direita de dados entre os dois componentes de 16 bits. De forma similar, a saída 16 do

componente SR16CLED que armazena a parte mais significativa do conteúdo do registrador acumulador está interligada à entrada serial de dados SLI (Shift_Left_Input) possibilitando o deslocamento `a esquerda de dados entre os componentes de 16 bits.

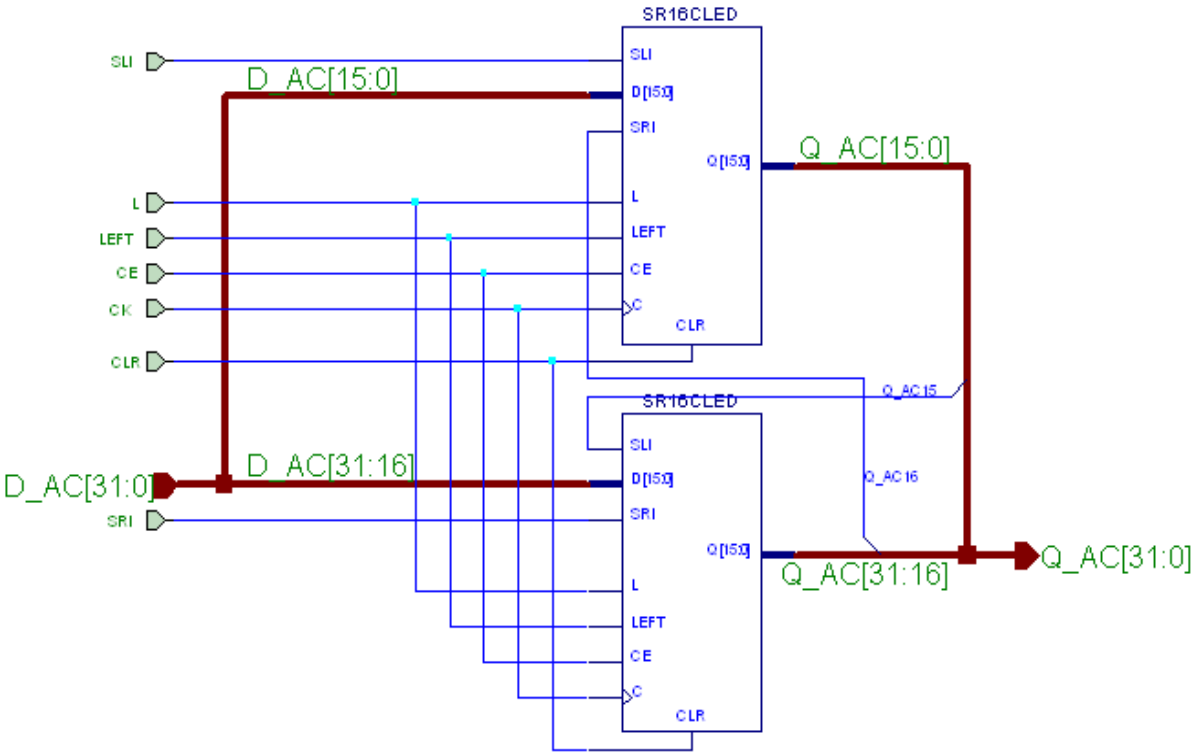


Figura 8.8- Implementação de um Registrador Acumulador de 32 bits

A tabela 8.4 descreve a forma de funcionamento do registrador acumulador de 32 bits da figura 8.8

Tabela 8.4- Tabela verdade do registrador Acumulador de 32 bits mostrado na figura 8.8

Entradas								***Saída de Dados***	***Comentários***
Controle					***Dados***				
CLR	***L***	***CE***	***LEFT***	***Ck***	***SLI***	***SRI***	***D_AC[31:0]***	***Q_AC[31:0]***	
1	*X*	*X*	*X*	*X*	*X*	*X*	$XXXXXXXX_{(H)}$	$00000000_{(H)}$	*Limpa Acumulador*
0	*1*	*X*	*X*	↑	*X*	*X*	D_AC_{31i} ... D_AD_{0i}	D_AC_{31i} ...D_AC_{0i}	*Entrada Paralela*
0	*0*	*1*	*1*	↑	*1*	*X*	$XXXXXXXX_{(H)}$	D_AC_{30i} ... D_AC_{0i},*0*	*Deslocamento a esquerda (SLI=1)*
0	*0*	*1*	*1*	↑	*0*	*X*	$XXXXXXXX_{(H)}$	D_AC_{30i} ... D_AC_{0i},*1*	*Deslocamento a esquerda (SLI=0)*
0	*0*	*1*	*0*	↑	*X*	*1*	$XXXXXXXX_{(H)}$	*1*, D_AC_{31i} ... D_AC_{0i}	*Deslocamento a Direita (SRI=1)*
0	*0*	*1*	*0*	↑	*X*	*0*	$XXXXXXXX_{(H)}$	*0*, D_AC_{31i} ... D_AC_{0i}	*Deslocamento a Direita (SRI=0)*

8.2.3- Unidade Lógica Aritmética (ULA)

As funções exercidas pela Unidade Lógica Aritmética (ULA) estão explicitadas em sua denominação. A ULA é responsável por executar as operações aritméticas e lógicas contidas na UCP. Além disso, a ULA é um circuito combinacional e, para executar as suas operações ela utilizada dados (operandos) armazenados temporariamente nos Registradores Gerais e os resultados obtidos são armazenados sempre no Registrador Acumulador.

As operações aritméticas serão compostas pelas 4 operações básicas: Adição, Subtração, Multiplicação e Divisão; envolvendo operandos Inteiros e Reais. Este item mostra a implementação de um circuito somador que executa somente a operação soma. No entanto, com a utilização de técnicas e algoritmos implementados em hardware serão implementadas as demais operações: Subtração, Multiplicação e Divisão. Essa forma de implementação irá também contribuir para o leitor dominar toda e qualquer sequencia de microordens necessárias para executar qualquer instrução na máquina que será implementada e validada pelos mesmos, incluindo a definição dos seguintes conjuntos de instruções: Instruções executadas pela UCP (ULA); Instruções de Movimentação de Dados; Instruções de Entrada e Saída e, Instruções de Controle.

A operação de subtração será transformada em uma operação de adição em complemento do maior dígito da base binária (Complemento de 1). Para inserir as operações de Multiplicação e Divisão serão desenvolvidos em implementados em hardware algoritmos que realizam essas operações através de combinações de operações de Adição, Subtração e Deslocamentos. Para tanto, serão utilizados subsistemas complementares. Tais como: Comparadores de Magnitude, descrito no item 8.2.6 e Contadores Bidirecionais descritos no item 8.2.4.

As operações lógicas incluem as operações Booleanas e operações relacionais. As operações Booleanas envolvem operandos e resultados lógicos (Falso, Verdade) e incluem as seguintes operações: AND, OR, NOT, NAND, NOR e XOR. As operações relacionais envolvem operandos de qualquer natureza e resultados sempre lógicos, incluindo as seguintes operações: Maior, Menor, Igualdade, Desigualdade, Maior ou Igual e Menor ou Igual. Essas operações serão implementadas por um circuito combinacional que está descrito no item 8.2.6. Posteriormente, esse circuito será também incluído no subsistema ULA.

Para exemplificar e mostrar o benefício de transformar uma operação de Subtração em uma “Soma em Complemento de 1”, a figura 8.9 mostra um circuito combinacional que realiza as operações de adição e subtração de duas palavras de 4 bits denominado Somador

Subtrator de 4 bits. O circuito é constituído apenas de quatro circuitos somadores completos e, um circuito inversor realiza o complemento de uma palavra também de 4 bits utilizando portas lógicas XOR. A Figura 8.10 mostra o circuito AD_SUB_16, contido no software Project Manager, que contém um Somador e Subtrator de 16 bits. Através da associação série/paralelo de dois circuitos AD_SUB_16 gera-se um circuito somador/subtrator de 32 bits, mostrado na figura 8.11.

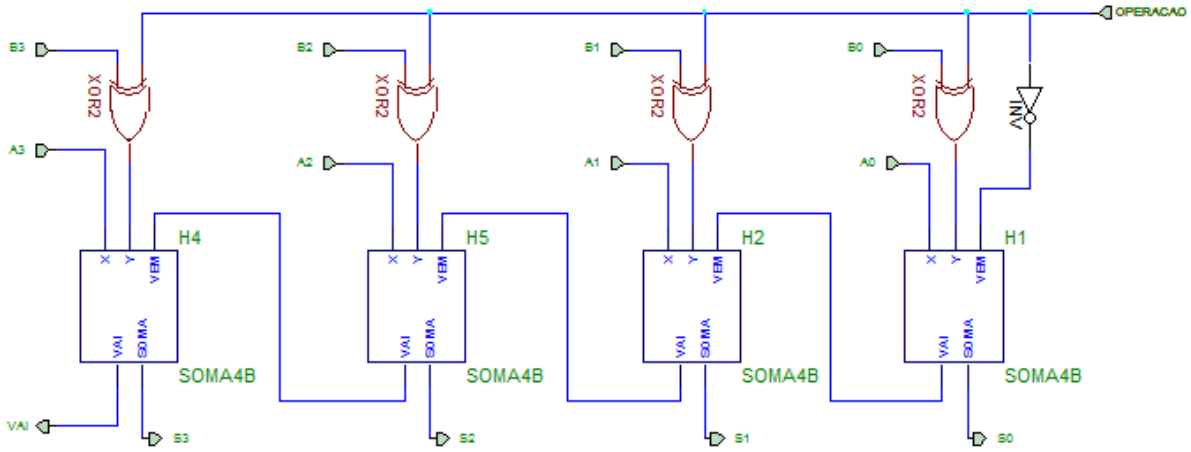


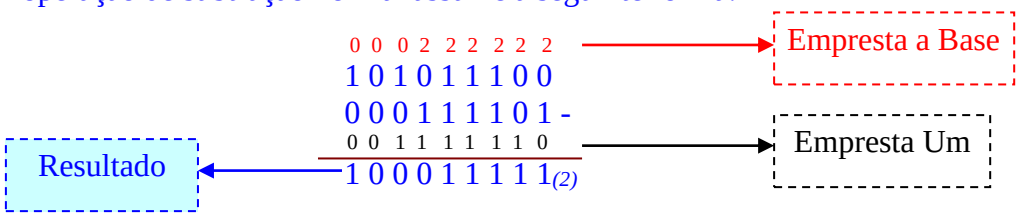
Fig. 8.9- Circuito Somador Subtrator de 4 bits

O exemplo que segue mostra a transformação da operação de subtração em uma soma em complemento de 1 de dois operandos binários.

Exemplo 1- Transformar a operação **$101011100_{(2)}$ - $111101_{(2)}$ em uma operação de Soma em Complemento de 1**

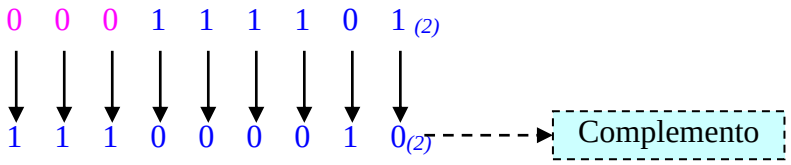
Resolução:

A operação de subtração normal assume a seguinte forma:



A transformação da operação de subtração em uma soma em complemento é realizada da seguinte forma:

Passo 1- complementa o diminuendo ($111101_{(2)}$), verificando o comprimento do minuendo;



Passo 2- Somar o complemento obtido com o minuendo e corrigir o resultado:

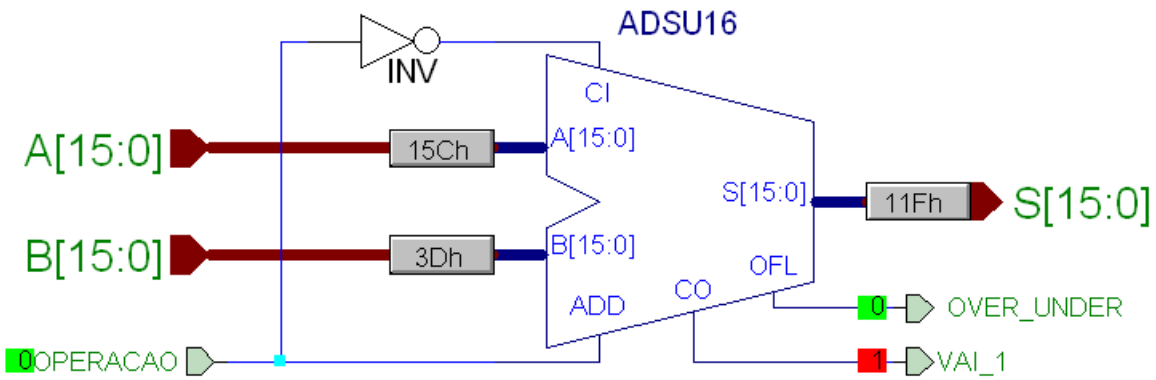
No circuito a linha de controle ADD_SUB define a operação que será realizada. O valor “1” define a operação de Adição e o valor “0” inserido na linha ADD_SUB define a operação de Subtração. Além disso, quando ADD_SUB = 0 são realizadas as seguintes atividades:

a- O operando B (B3, B2, B1 e B0) é complementado nas sápidas das portas lógicas Ou-Exclusivo (XOR).

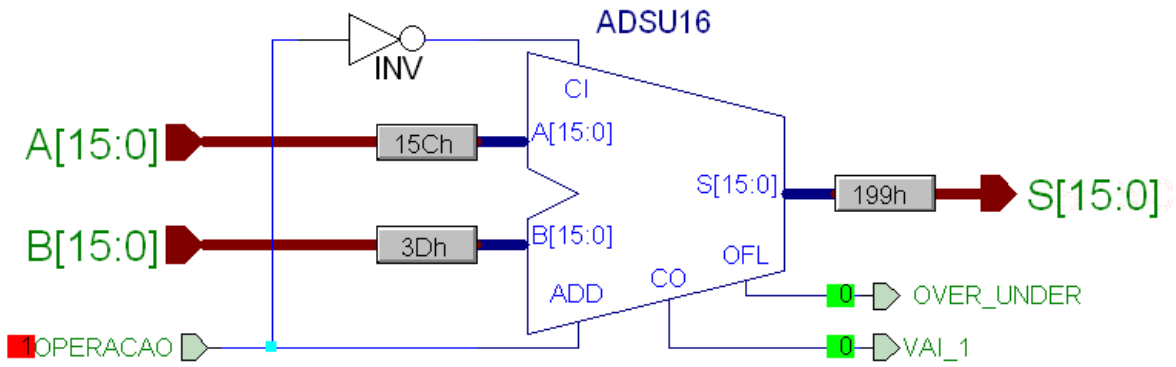
b- A linha ADD_SUB alimenta também a entrada VEM do circuito Somador que realiza a soma dos bits menos significativos das palavras e, dessa forma, realiza a correção do resultado parcial obtendo o resultado final, pois na operação Subtração ADD_SUB=0 e segue complementado (1) para as entradas das portas XOR .

A Figura 8.10 mostra a implementação de um circuito somador subtrator de 16 bits utilizando o componente AD_SU16 existente no software Project Manager. Os valores mostrados na figura são resultados da simulação do circuito através da utilização dos valores contidos na operação descrita no exemplo 1.

A saída VAI_1 também é utilizada para associar dois componentes ADSU16 em Série, gerando circuitos somadores subtratores que realizam operações com palavras de comprimento múltiplo de 16 bits (32, 48, 64, ..., 128 bits).



(a) simulação da operação Subtração com valores do exemplo 1



(b) simulação da operação Adição com valores do exemplo 1

Figura 8.10- Componente ADSU16 do software Project Manager

A figura 8.11 mostra um circuito somador subtrator de 32 bits implementado através da associação de dois componentes ADSU16 contido no software Project Manager. Portanto os barramentos possuem 32 fios. Conforme descrito nos itens 8.2.1 e 8.2.2 os barramentos devem ser muito bem organizados e a convenção de utilizar o índice "0" para o bit menos significativo contribui para evitar problemas nas implementações de todos os subsistemas da máquina que será gerada pelos leitores. Essa padronização evita erros ou falhas de implementação no instante em que os mesmos serão interligados para constituir a máquina desejada.

Na figura 8.11, a Saída UNDER representa a ocorrência de um underflow na operação de Subtração e a Saída denominada OVER representa um Overflow (estouro da capacidade de representação do resultado de uma operação de Adição).

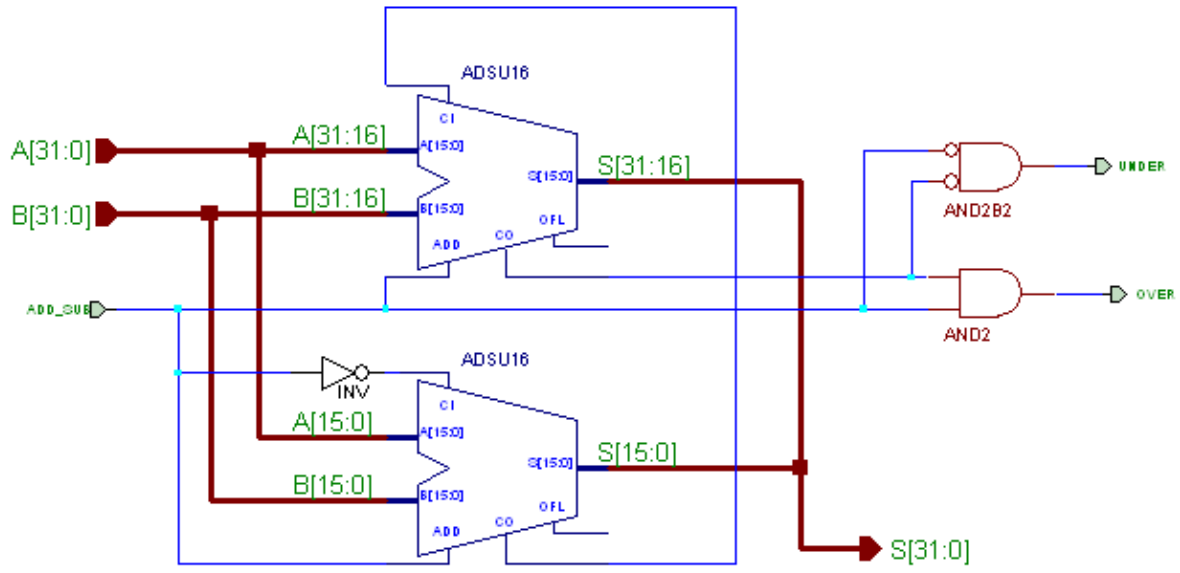


Figura 8.11- Circuito Somador_Subtrator de 32 bits (ADD_SUB32)

8.2.4- Registradores Contadores Bidirecionais (UP/DOWN)

Os circuitos contadores estão presentes em vários subsistemas de uma arquitetura. Com ênfase na Unidade de Controle e Unidade Lógica Aritmética, ambas unidades contidas na Unidade Central de Processamento.

O registrador contador que está presente na Figura 8.12 é um contador bidirecional, gera contagem tanto na forma crescente quanto na forma decrescente (contador UP-DOWN). Além disso, o contador permite que a contagem possa iniciar em qualquer valor que ele possa conter. A entrada de controle denominada LOAD viabiliza essa função através do armazenamento de um valor registro (dado/valor) qualquer nos flip-flops do contador. Trata-se de um contador de 8 bits que permite expansão e construção de contadores de maior comprimento através da associação serial de mais de dois ou mais elementos (unidades), formando contadores com quantidade de bits (comprimento) sendo múltiplo de oito (8). Para tanto deve ser utilizada, de forma apropriada, a saída de controle denominada Enable_Out. Para implementar o contador contido na figura 8.9b no software Project Manager foi utilizado o elemento CB8CLED.

A Tabela 8.5 descreve a forma de funcionamento do Registrador UP/DOWN da Figura 8.9b, mostrando as entradas e saídas de dados com os respectivos valores das entradas de controle para cada uma das funções exercidas pelo Contador. A entrada de controle "LOAD" tem a função de armazenar no Contador, o valor dos dados contido na Entrada Paralela de Dados.

Entrada de Dados	Entradas de Controle					**Saída de Dados**	Comentários
	Enable	Load	Direção	Clock	Clear		
D_{7i} ... D_{0i}	X	0	X	X	0	$Q_{7i-1}...Q_{0i-1}$	Mantém estado anterior
XX_H	X	1	X	↑	0	D_{7i} ... D_{0i}	Armazena entradas de dados
XX_H	1	0	0	↑	0	$(Q_{7i-1}...Q_{0i-1})-1$	Conta para baixo
01_H	1	0	0	↑	0	00_H	Enable_Out=1; Detecta_00/FF=1
XX_H	1	0	1	↑	0	$(Q_{7i-1}...Q_{0i-1})+1$	Conta para Cima
FE_H	1	0	1	↑	0	FF_H	Enable_Out=1; Detecta_00/FF=1
XX_H	X	X	X	X	1	00_H	Limpa o conteúdo (insere zero)

Tabela 8.5- Tabela Verdade do Registrador Contador UP/DOWN da Figura 8.12.

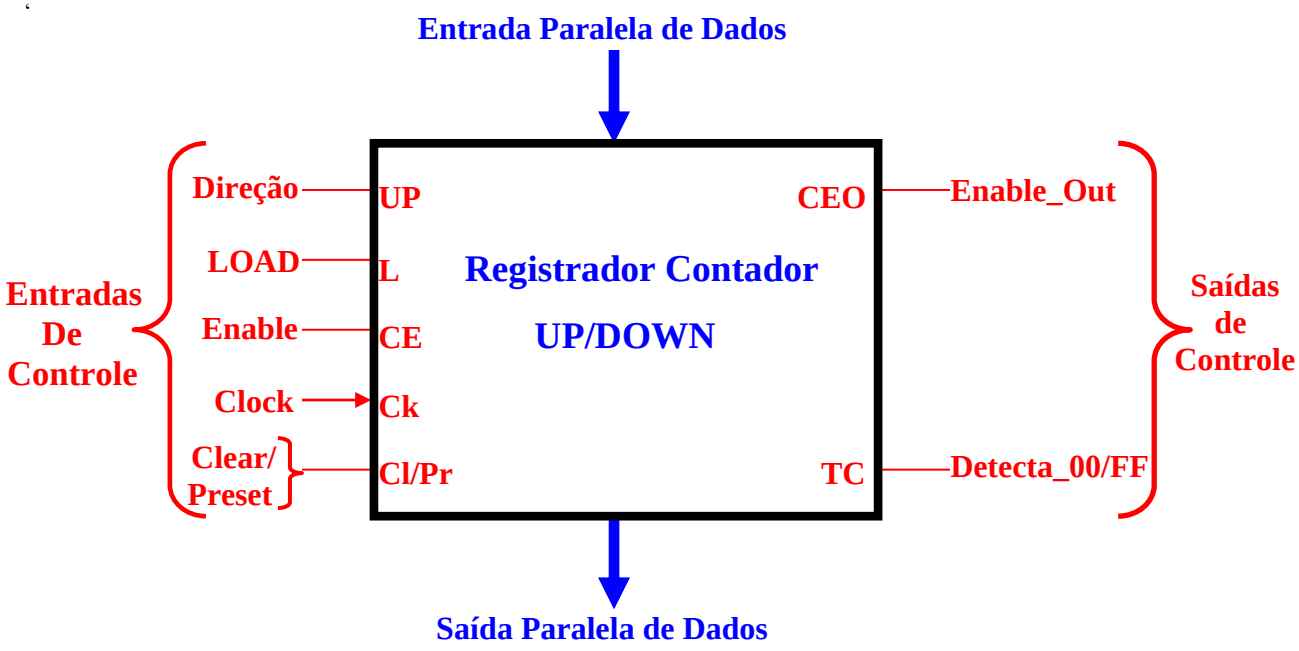
Para armazenar os valores contidos na Entrada Paralela de Dados, as entradas de controle devem assumir a seguinte configuração: LOAD=1; CLEAR=0 e gerar borda de subida na entrada de controle de relógio (CLOCK= ↑); os valores das demais entradas de controle são irrelevantes na função de armazenamento paralelo de dados.

Para que o contador gere um seqüência de contagem decrescente, as entradas de controle devem assumir a seguinte configuração: ENABLE=1; LOAD=0; Direção=0; CLEAR=0 e gerar borda de subida na entrada de controle de relógio (CLOCK= ↑).

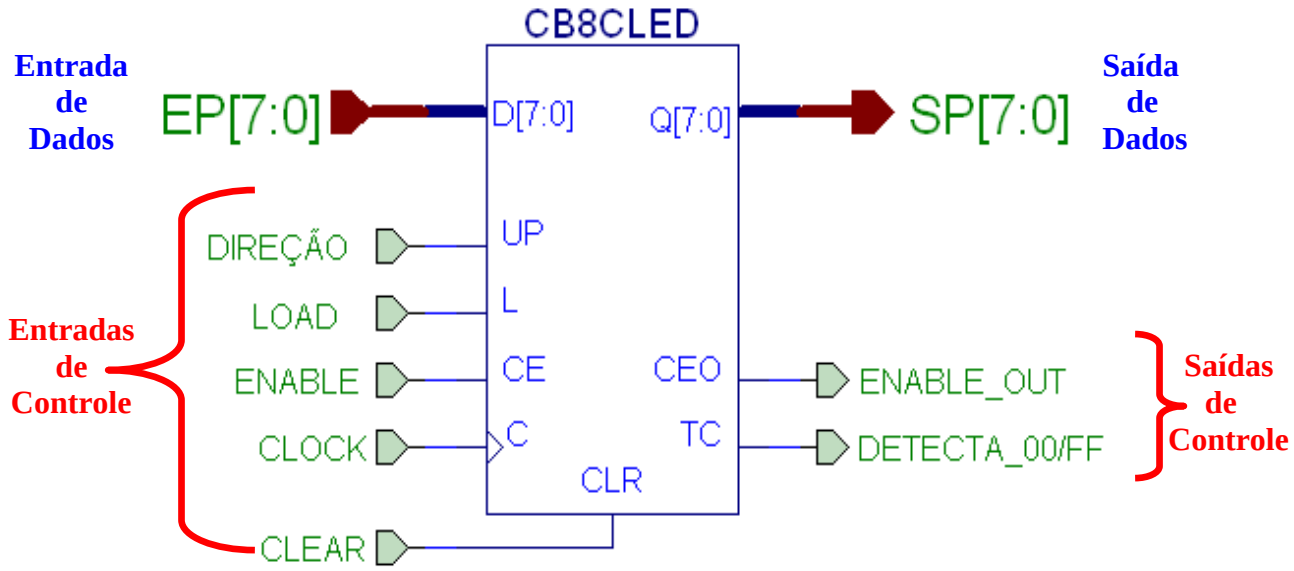
Para que o contador gere um seqüência de contagem crescente as entradas de controle devem assumir a seguinte configuração: ENABLE=1; LOAD=0; Direção=1; CLEAR=0 e gerar borda de subida na entrada de controle de relógio (CLOCK= ↑).

A saída de controle Enable_Out, pode ser utilizada para projetar contadores com maior quantidade de bits através da associação em série de dois ou mais elementos contadores "CB8CLED". Para tanto, a saída de controle Enable_Out do elemento "CB8CLED" que contém os bits menos significativos, do contador projetado, deve ser ligado à entrada de controle Enable do elemento "CB8CLED", que contém os bits mais significativos do contador. Essa configuração é válida tanto para a contagem na forma crescente quanto para a contagem na ordem decrescente. Na contagem crescente, a saída Enable_Out, assume o valor lógico "1" quando as saídas de dados do contador atingir o valor hexadecimal "FF". O mesmo acontece na forma de contagem decrescente, quando as saídas de dados do contador atingir a contagem "00_H".

A saída Detecta_00/FF tem a função de detectar os limites inferiores e superiores de contagem do registrador contador. Na forma decrescente ele detecta o valor Hexadecimal "00" w na forma crescente ele detecta o valor Hexadecimal "FF".



(a) Diagrama de Blocos de Registrador Contador UP/DOWN



(b)- Uma Implementação de Registrador Contador UP/DOWN no software Project Manager

Figura 8.12- Registrador Contador Bidirecional

A Figura 8.12a mostra o diagrama lógico de um Registrador Contador bidirecional genérico de 8 bits. A Figura 8.12b mostra a implementação de um registrador Contador Bidirecional de oito bits, utilizando um elemento contador UP/DOWN de oito bits contido na biblioteca do software Project Manager.

Exercício 8.3- Sintetizar e projetar um registrador contador bidirecional de 32 bits utilizando um componente contador UP-DOWN de apenas 16 bits.

Resolução- *Para realizar a síntese implementação do contador solicitado é requerido o conhecimento dos seguintes itens:*

a- funcionamento e tabela verdade do componente eletrônico disponível para ser utilizado no projeto

b- técnicas de associação série-paralelo de componentes eletrônicos e;

c- conceitos e técnicas de síntese e projeto de circuitos combinacionais e seqüenciais.

Item a*- O contador UP-DOWN de 16 bits disponibilizado para o projeto é mostrado na figura 8.13 e possui a tabela verdade equivalente ao registrador contador UP-DOWN de 8 bits, cujas características estão contidas na figura 8.12 e descritas na tabela 8.5. É facilmente verificado que a diferença no comprimento da palavra entre os dois componentes (16 e 8 bits respectivamente) gera como consequencia as seguintes modificações nas funções das saídas denominadas "CEO" e "TC" no componente de 16 bits:*

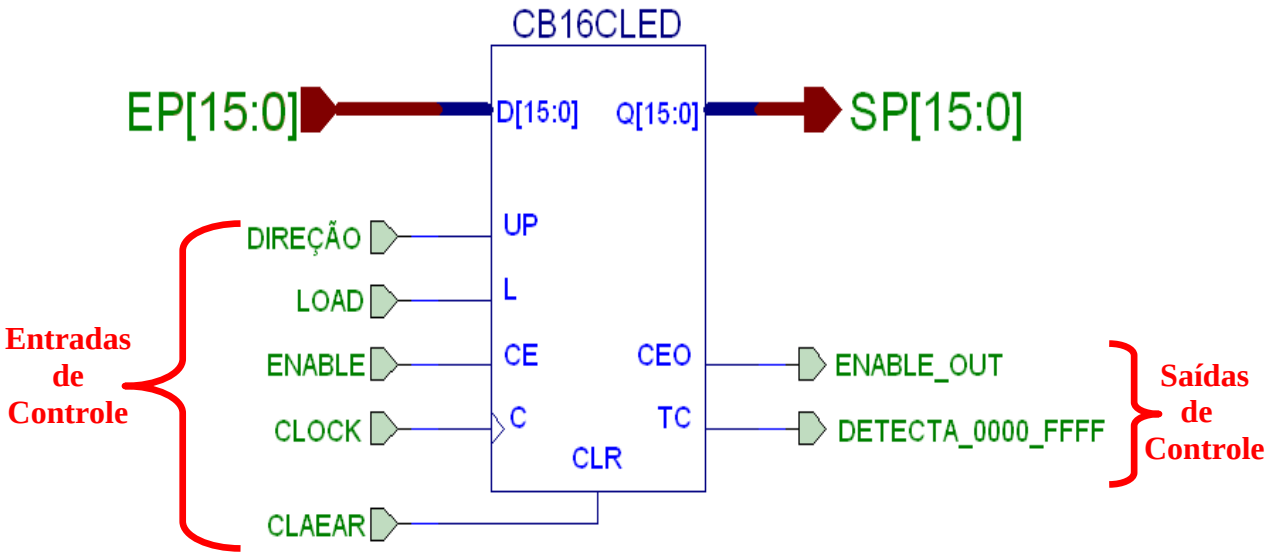


Figura 8.13- Registrador contador UP-DOWN de 16 bits (Project Manager)

- As saídas "CEO-ENABLE_OUT" será ativada (nível lógico =1) quando o contador atingir o valor máximo ($FFFF_{(H)}$) e/ou o valo mínimo ($0000_{(H)}$) quando a entrada de controle "UP-Direção" estiver definindo o sentido de contagem crescente (UP=1) e/ou decrescente (UP=0) respectivamente. As funções das entradas de controle não sofrem modificações.

***Item b**- É natural que serão necessários dois registradores contadores de 16 bits para obter um contador de 32 bits. Além disso é possível também concluir que os dois contadores de 16 bits deverão ser associados (interligados) em paralelo para obter o contador de 32 bits, pois a associação em paralelo de dois componentes de memória é utilizada para aumentar ao comprimento da palavra.*

- No entanto, existe também dependência de sinais de controle entre os dois componentes. Essa interdependência gera uma associação lógica serial entre os dois componentes que serão necessários para obter os 32 bits solicitados. O circuito obtido a partir dessas considerações é mostrado na figura 8.14. Na figura observa-se que a entrada de controle "CE-Chip Enable" do contador que conterá os bits mais significativos do conteúdo está ligada à saída "CEO-Enable_Out" do contador que conterá os bits menos significativos do valor de contagem.

***Item c**- Com a configuração mostrada na figura 8.14 o registrador irá gerar as seguintes as seqüências de contagem mostrada na tabela 8.6. Ou seja, quando o contador está configurado para gerar a seqüência de contagem na ordem crescente (UP=1, LOAD=0, CE=1) e com a contagem inicial igual a $00000000_{(H)}$, as saídas irão mostrar uma sequencia crescente de contagem culminando com o valor*

FFFFFFFF$_{(H)}$. De forma similar, quando o contador estiver configurado para gerar a contagem na ordem decrescente (UP=0, LOAD=0, CE=1) as saídas assumirão as configurações: FFFFFFFF$_{(H)}$, FFFFFFFE$_{(H)}$, ... , 00000000$_{(H)}$.

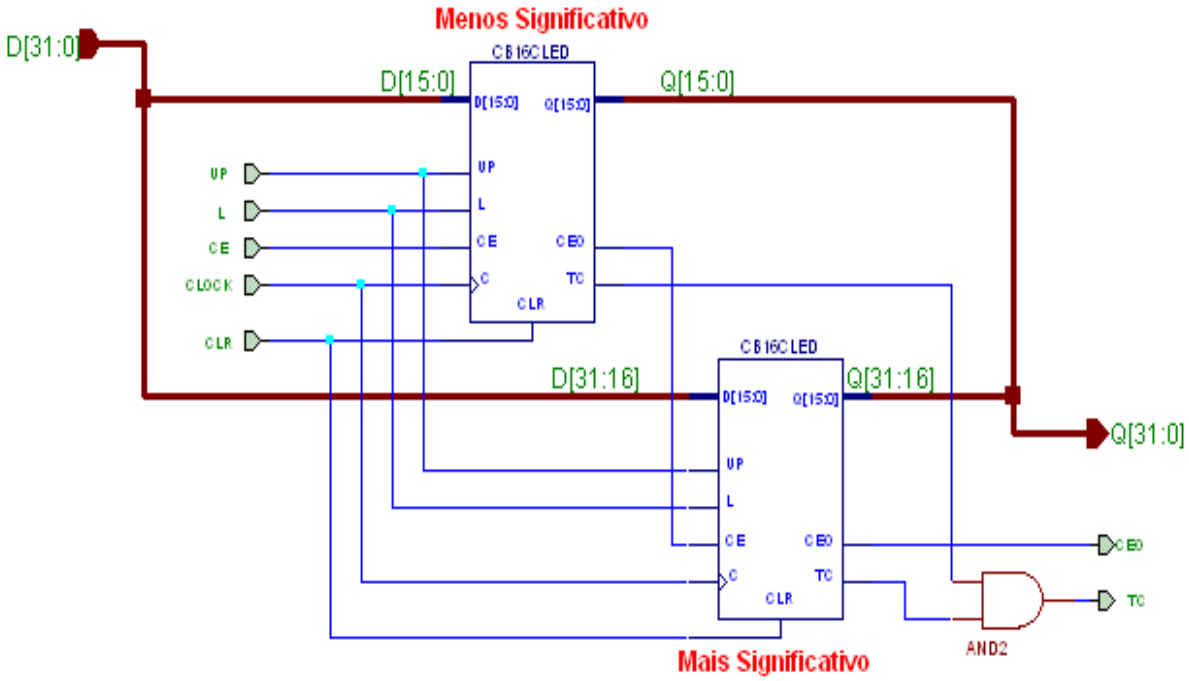


Figura 8.14- Registrador Contador Bidirecional de 32 bits

Ordem crescente					
	Clock	**UP**	**Q[31:16]**	**Q[15:0]**	**CEO (sup)**
inicial		1	0000	0000	0
	↑	1	0000	0001	0
	…	1	0000	…	0
	↑	1	0000	FFFE	0
	↑	1	0000	FFFF	1
	↑	1	0001	0000	0
	↑	1	0001	0001	…
	…	1	…	…	…
	↑	1	FFFE	FFFF	1
Final	↑	1	FFFF	FFFF	0
	↑	1	0000	0000	
Ordem Decrescente					
	Clock	**UP**	**Q[31:16]**	**Q[15:0]**	**CEO (inf)**
inicial		0	FFFF	FFFF	0
	↑	0	FFFF	FFFE	0
	…	0	FFFF	…	0
	↑	0	FFFF	0001	0
	↑	0	FFFF	0000	1
	↑	0	FFFE	FFFF	0
	…	0	…	…	…
final	↑	0	0000	0000	1

Tabela 8.6- Seqüência de saídas do contador de 32 bits

8.2.5- Elementos Multiplexadores

Um multiplex pode ser utilizado para implementar várias funções, desde a capacidade de implementar circuitos comutadores de sinais em centrais de comutação telefônica e até mesmo na a implementação de toda e qualquer função lógica (expressões booleanas). Na arquitetura proposta na Figura 8.2 estão sendo utilizados dois elementos multiplex com a função de viabilizar o acesso a recursos que devem ser compartilhados por vários subsistemas.

Um elemento multiplex transfere para a sua saída de dados apenas um único valor contido em uma de suas entradas de dados, de acordo com o valor contido nas suas entradas de controle de seleção. A quantidade de entradas de seleção define e limita a quantidade de entradas de dados que pode e deve existir em um elemento multiplex, obedecendo sempre a seguinte relação: a quantidade de entradas de dados é dada por dois (2) elevado ao número de entradas de seleção.

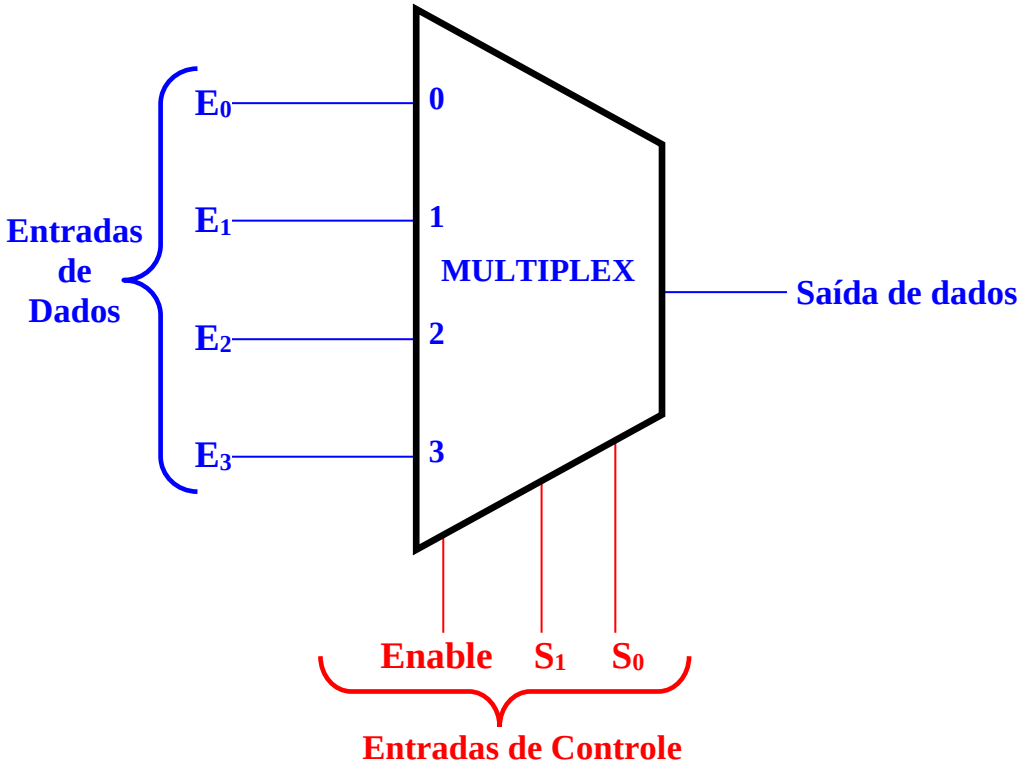
$$\textbf{Quantidade de Entradas} = 2^{\textbf{entradas de seleção}}$$

A Figura 8.15b1 mostra um elemento multiplex existente na biblioteca do software Project Manager denominado "M2_1" contendo uma única (1) entrada de seleção e duas entradas de dados ($2=2^1$). A Figura 8.15.b2 mostra um elemento multiplex denominado "M4_1E", esse elemento contém duas (2) entradas de seleção e quatro entradas de dados ($4=2^2$) e a Figura 8.15.b3 mostra um elemento multiplex denominado "M8_1E", contendo 3 entradas de seleção e conseqüentemente oito ($8=2^3$) entradas de dados. A entrada de controle Enable (habilitar), está presente nos elementos denominados "M2_1" e "M4_1E". A entrada Enable, permite o funcionamento normal do multiplex quando ela assume o valor 1. Quando ela assume o valor 0 (zero) a Saída de Dados será sempre igual a zero, independente dos valores das Entradas de Dados E_i.

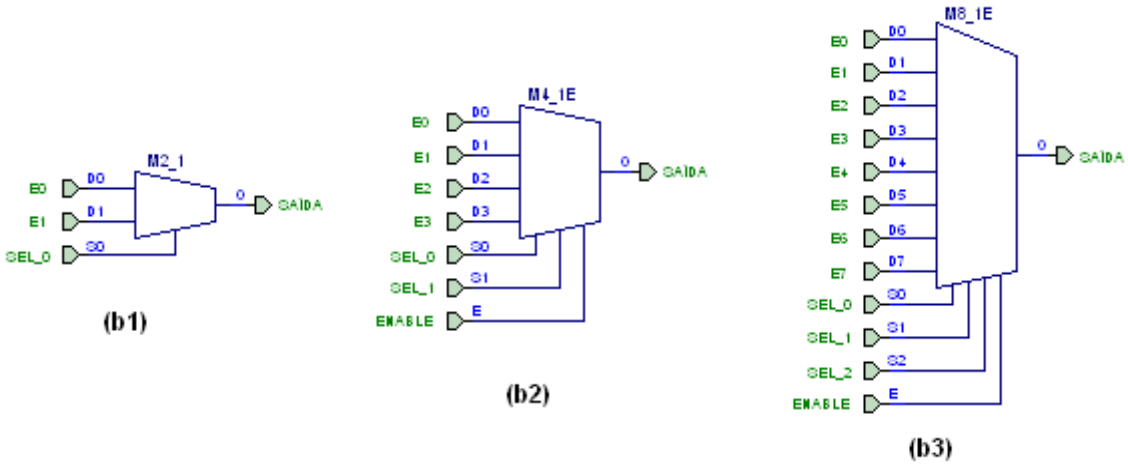
Entradas de Dados	**Entradas de Controle**			**Saída**
	Enable	Sel_1	SEL_0	
E_3 ... E_0	**1**	**0**	**0**	**E_0**
E_3 ... E_0	**1**	**0**	**1**	**E_1**
E_3 ... E_0	**1**	**1**	**0**	**E_2**
E_3 ... E_0	**1**	**1**	**1**	**E_3**
E_3 ... E_0	**0**	**X**	**X**	**0**

Tabela 8.7- Tabela Verdade do Multiplex M4_1E do item b2 da Figura 8.15.

A tabela 8.7 mostra a forma de um elemento multiplex de duas (2) entradas de controle de seleção e quatro ($4=2^2$) entradas de dados e uma entrada de controle Enable, equivalente ao multiplex mostrado na figura 8.11b2. Um multiplex com essas características seleciona uma entre 4 diferentes entradas para ser transferida para sua saída e é também denominado multiples 4x1.



(a) Diagrama de Blocos de um Elemento Multiplex



(b)- Elementos Multiplex do software Project Manager

Figura 8.15 Elementos Multiplexadores

Exercício 8.4- Sintetizar e Implementar um multiplex que transfira para a sua saída um entre 4 bytesutilizando elementos multiplex 4x1.

A resolução do exercício 8.4 requer o conhecimento da tabela verdade e do funcionamento de circuitos multiplexadores, além do domínio técnicas de síntese e projeto de circuitos combinacionais (associação série-paralelo de elementos multiplexadores).

Resolução:

O exercício solicita a implementação de um multiplex que transfira para sua saída um byte que deve ser selecionado entre diferentes 4 bytes de entrada. Como cada byte é formado por um conjunto de 8 bits o elemento multiplex solicitado terá 32 entradas e 8 saídas, ou seja, um multiplex 32x8.

Foi também disponibilizado para a síntese e implementação do multiplex 32x8, circuitos mutiplexadores 4x1. A descrição do funcionamento e a forma gráfica do multiplex 4x1, estão explicitadas na figura 8.16 e tabela 8.8 respectivamente.

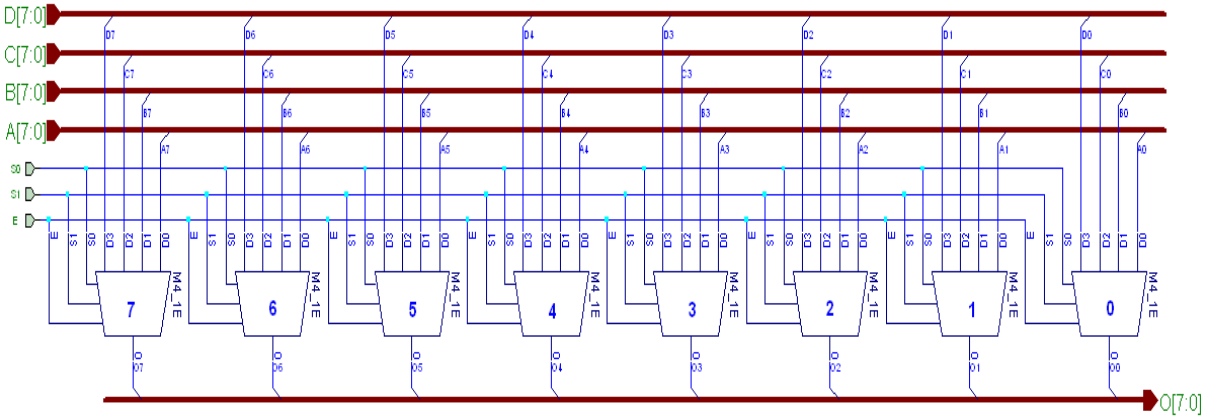


Figura 8.16- Circuito Multiplex 32x8 implementado com 8 ememento multiplex 4x1

Para obter um multiplex 32x8 foram utilizados 8 elementos multiplex 4x1 para gerar um multiplex de 32 entradas e 8 saídas (multiplex 32x8). Os 8 elementos multiplex 4x1 serão interligados em paralelo. Cada elemento multiplex 4x1, irá transferir para a sua saída um dos bits do byte escolhido para ser transferido para a saída do multiplex 32x8. A figura 8.16 mostra a disposição dos 8 multiplex 4x1 interligados de forma paralela. Cada um dos multiplex seleciona um dos bits do byte escolhido (determinado) pelas entradas de seleção para a sua saída, compondo um barramento de saída de 8 bits denominado O[7:0]. Verifica-se também na figura a existência de 4 barramentos de 8 bits denominados: A[7:0], B[7:0], C[7:0] e D[7:0]. Cada uma das

entradas de seleção de todos os 8 multilexadores 8x1 estão interligadas a uma única origem, essa configuração permite a melhor organização gráfica do circuito.

As entradas de seleção foram denominadas S1 e S0 e a tabela 8.8 mostra o funcionamento do multiplex 32x8 da figura 8.16.

Entradas							**Saída**
Dados				**Controle**			
D3	**D2**	**D1**	**D0**	**E**	**S1**	**S0**	**O[7:0]**
D[7:0]	**C[7:0]**	**B[7:0]**	**A[7:0]**	**1**	**0**	**0**	**A[7:0]**
D[7:0]	**C[7:0]**	**B[7:0]**	**A[7:0]**	**1**	**0**	**1**	**B[7:0]**
D[7:0]	**C[7:0]**	**B[7:0]**	**A[7:0]**	**1**	**1**	**0**	**C[7:0]**
D[7:0]	**C[7:0]**	**B[7:0]**	**A[7:0]**	**1**	**1**	**1**	**D[7:0]**
D[7:0]	**C[7:0]**	**B[7:0]**	**A[7:0]**	**0**	**X**	**X**	**$00_{(H)}$**

Tabela 8.8- Tabela Verdade do circuito Multiplex 32x8

Exercício 8.5- Nas máquinas de 32 bits é comum o compartilhamento de um barramento de 32 bits por dois outros barramentos também de 32 bits. Para possibilitar essa aplicação, sintetize e implemente circuito multiplex que transfira para sua saída um único barramento de 32 bits escolhidos entre dois diferentes barramentos contidos em suas entradas.

Resolução:-

A resolução do exercício 8.5 requer o conhecimento da tabela verdade e do funcionamento de circuitos miltiplexadores, além do domínio de técnicas associação paralela de elementos multiplexadores.

O exercício requer a síntese e construção de um circuito multiplex que selecione um entre dois barramentos de 32 bits contidos em suas entradas para a sua saída. Podemos interpretar o problema utilizando 32 elementos mutiplexadores 2x1 igual ao mostrado no item b1 da figura 8.15 para implementar a função solicitada no exercício. O multiplex 64x32 obtido é mostrado na figura 8.17 e o seu funcionamento descrito na tabela 8.9.

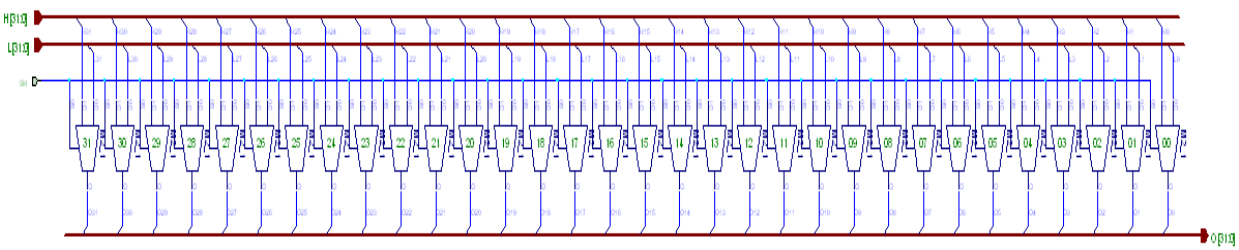


Figura 8.17- Circuito Multiplex 64x32 construído com 32 elementos mutiplex 2x1

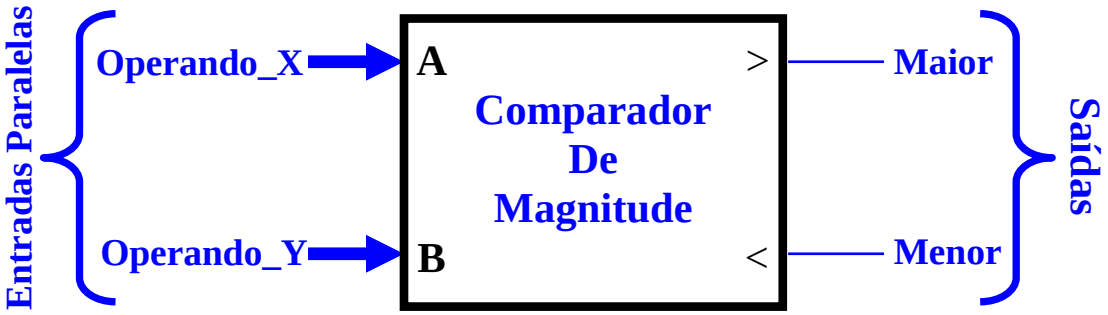
Entradas				Saída
Dados		Controle		
D1	D0	E	S0	O[31:0]
B[31:0]	A[31:0]	1	0	A[7:0]
B[31:0]	A[31:0]	1	1	B[7:0]
B[31:0]	A[31:0]	0	X	$00000000_{(H)}$

Tabela 8.9- Tabela verdade do multiplex 64x32 da figura 8.17

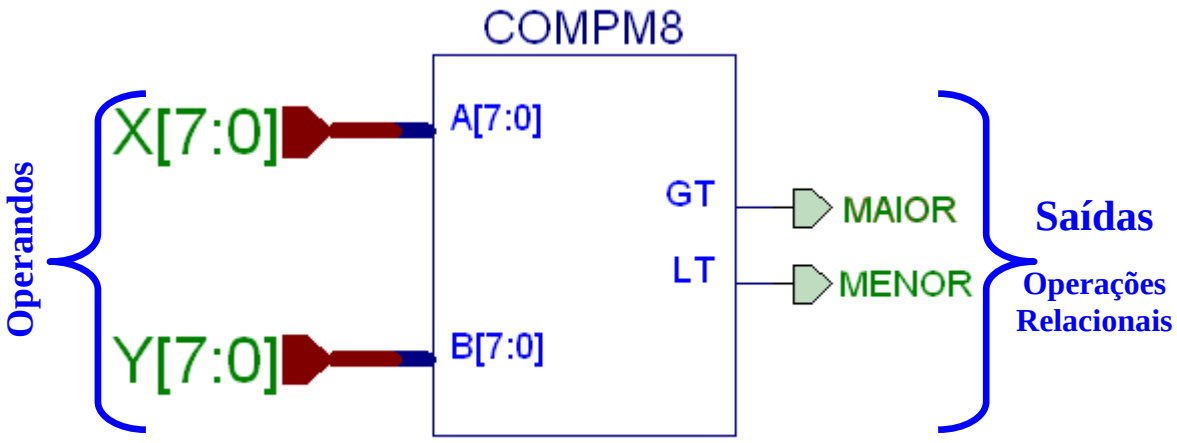
8.2.6- Comparadores de Magnitude

Um elemento comparador de magnitude realiza operações relacionais entre dois números. A capacidade de comparar grandezas em uma arquitetura viabiliza a construção de comandos repetitivos e operações lógicas. Portanto, esses elementos estão presentes tanto na Unidade Lógica Aritmética (ULA) quanto na Unidade de Controle (UC). As operações relacionais: Maior (>), Menor (<), Igualdade (=), Maior ou Igual (<=) e Menor ou Igual (<=) permitem a construção de estruturas tanto repetitivas, quanto condicionais, e como conseqüência viabilizam a implementação de operações lógicas e matemáticas complexas e não existentes na ULA, através da utilização de combinações ou repetições de suas operações básicas.

A Figura 8.18a mostra o diagrama de blocos de um Comparador de Magnitude Genérico e a Figura 8.18b mostra a implementação de um comparador de magnitude de oito bits no software Project Manager, utilizando o elemento “COMPM8”.



(a) Diagrama de Blocos de um Comparador de Magnitude



(b)- Uma Implementação de um Comparador de Magnitude no software Project Manager

Figura 8.18- Comparador de Magnitude

A Tabela 8.10 descreve a forma de funcionamento do Comparador de Magnitude da Figura 8.18b. Observa-se na referida tabela que um circuito comprador de magnitude é um circuito combinacional que não necessita de entradas de controle, as saídas do mesmo são geradas a partir da disponibilização dos valores dos dois operandos em suas entradas paralelas de dados. A combinação dos valores lógicos das suas duas saídas, denominadas "Maior" e "Menor" possuem o seguinte significado: Maior=1 e Menor=0 o valor presente na entrada X e Maior que o valor presente na entrada Y (X>Y); Maior=0 e Menor=1 quando Y>X; e Maior=Menor=0 quando X=Y.

Operandos	Saídas		Comentários
	Maior	Menor	
X > Y	1	0	**Operando_X Maior que Operando_Y**
X < Y	0	1	**Operando_X Menor que Operando_Y**
X = Y	0	0	**Operando_X Igual a Operando_Y**

Tabela 8.10- Tabela Verdade do Comparador de Magnitude da Figura 8.18b.

Exercício 8.6- A Unidade Lógica Aritmética de uma Unidade Central de Processamento é que executa as operações lógicas, aritméticas e relacionais existentes em nível de hardware. As operações relacionais executadas em uma UCP, além de ser utilizada nos cálculos existentes nos programas de usuários, é também utilizada para implementar comandos iterativos (repetitivos) na máquina. Assim, a existência de operações relacionais, implementadas como instruções em "hardware", aumentam a eficiência de uma arquitetura. Dada a importância das operações relacionais, sintetize e implemente um

circuito que gere todas as operações relacionais com operandos de 32 bits utilizando dois comparadores de magnitude de 16 bits com funcionamento equivalente ao comparador de magnitude de 8 bits contido na figura 8.18 e descrito na tabela 8.10.

Resolução: *Para implementar as operações relacionais em circuito lógico é necessário realizar as seguintes atividades:*

a- Primeiramente, sintetizar a tabela verdade de todas as operações relacionais: Maior, Menor, Igualdade, Diferença, Maior ou Igual e Menor ou Igual;

b- Identificar a dependência de informações geradas quando for realizada a associação paralela dos dois componentes comparadores disponibilizados para o projeto. Essa dependência de informações e influi na síntese da tabela verdade do circuito combinacional a ser gerado;

c- Associar, em paralelo, dois comparadores de magnitude de 16 bits que implementam apenas as operações "Maior que" e "Menor que";

Item a*:- O comparador de magnitude de 16 existente na biblioteca do software Project Manager e que satisfaz os requisitos do projeto é o componente "COMPMC16", cujo funcionamento é equivalente ao componente comparador de magnitude de 8 bits mostrado na figura 8.18 e tabela 8.10.*

O componente "COMPMC16" compara duas entradas disponibilizadas em barramentos de 32 bits e gera o resultado da comparação entre os valores contidos nos dois barramentos nas saídas lógicas "GT" (Maior que) e "LT" (Menor que). Quando é realizada a associação paralela dos dois comparadores de 16 bits, mostrada na figura 8.19, verifica-se a dependência entre as saídas "GT" e "LT' dos dois componentes. Sendo, que o componente que contem os 16 bits mais significativos dos dados de 32 bits, contidos nos seus barramentos de entrada, foi denominado de "componente 1" e, consequentemente, suas saídas lógicas "Maior que" e "Menor que" são denominadas respectivamente de "GT1" e "LT1". De forma similar e complementar, o componente que contem os 16 bits menos significativos dos dados comparados foi denominado de "componente 0" e suas saídas identificadas por "GT0" e "LT0".

A tabela verdade que descreve todas as operações relacionais a serem implementadas foi sintetizada e está contida na tabela 8.11.

Tabela Verdade das Operações Relacionais de 32 bits utilizando dois COMPMC16										
	Entradas				Saídas					
Decimal	GT_1	LT_1	GT_0	LT_0	GT	LT	Eq	GE	LE	DIF
0	0	0	0	0	0	0	1	1	1	0
1	0	0	0	1	0	1	0	0	1	1
2	0	0	1	0	1	0	0	1	0	1
3	0	0	1	1	X	X	X	X	X	X
4	0	1	0	0	0	1	0	0	0	1
5	0	1	0	1	0	1	0	0	0	1
6	0	1	1	0	0	1	0	0	0	1
7	0	1	1	1	X	X	X	X	X	X
8	1	0	0	0	1	0	0	1	1	1
9	1	0	0	1	1	0	0	1	1	1
10	1	0	1	0	1	0	0	1	1	1
11	1	0	1	1	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X

Tabela 8.11- Tabela Verdade das operações relacionais de 32 bits implementadas com dois componentes "COMPMC16'

Item b- *Para sintetizar os circuitos lógicas que geram todas as seis operações relacionais foi gerado seis mapas de karnaugh de 4 bits. Nesses mapas as entradas das funções lógicas sás as saídas "GT1", LT1", "GT0" e "LT0" dos dois componentes "COMPMC16" utilizados.*

Os mapas de Karnaugh auxiliaram na síntese obtenção das expressões mínimas das operações relacionais e a conseqüente implementação de seus circuitos lógicos.

Para efeito de implementação do circuito final e documentação adequada ao software "Project Manager" as operações relacionais foram batizadas na forma que segue: "Maior que = GT", "Menor que = LT", "Igualdade = EQ", "Maior ou Igual = GE", "Menor ou Igual = LE" e "Diferença = DIF".

As respectivas expressões mínimas obtidas, foram:

$GT = LT_1 + LT_1' . GT_0$

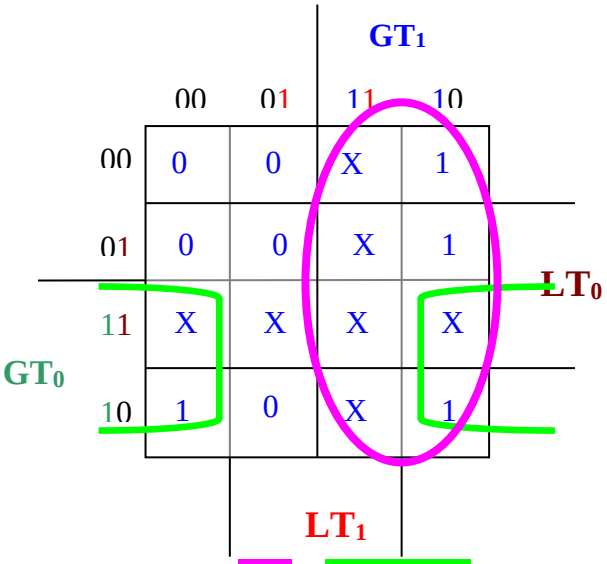
$LT = LT_1 + GT_1' . LT_0$

$EQ = GT_1' . LT_1' . GT_0' . LT_0'$

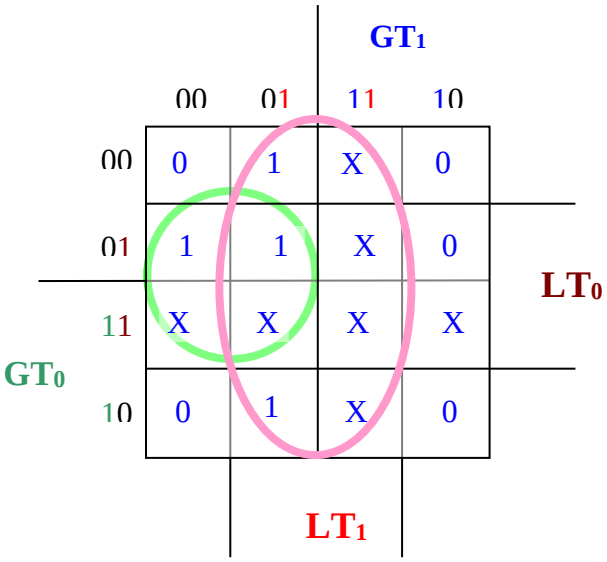
$GE = GT_1 + LT_1' . LT_0'$

$LE = LT_1 + GT_1' . GT_0'$

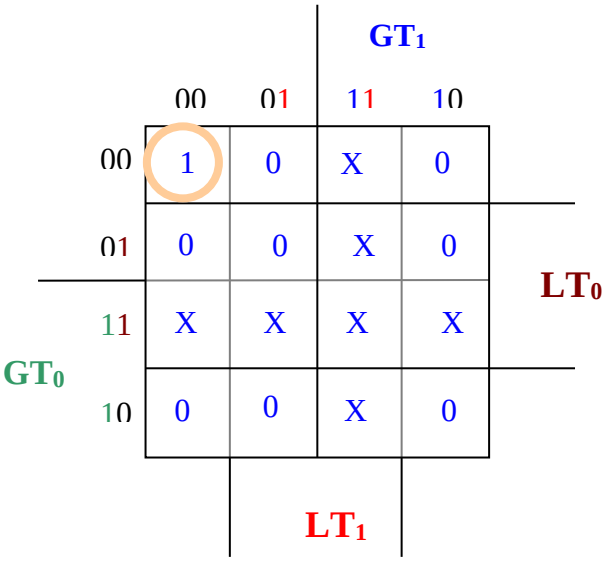
$DIF = GT_1 + LT_1 + GT_0 + LT_0$



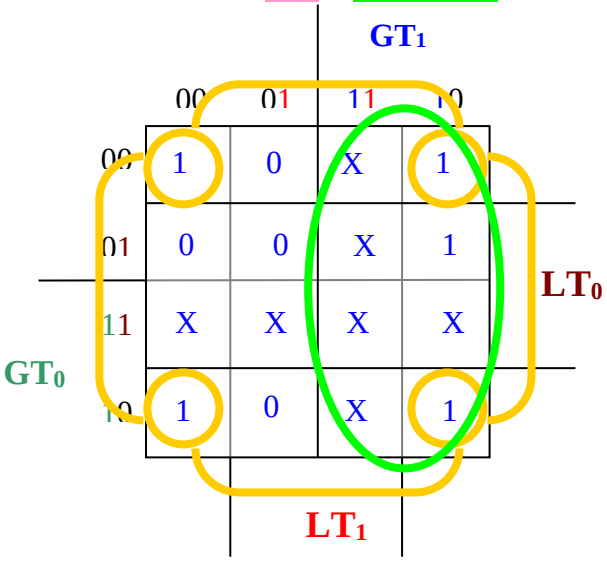
GT = LT_1 + LT_1' . GT_0



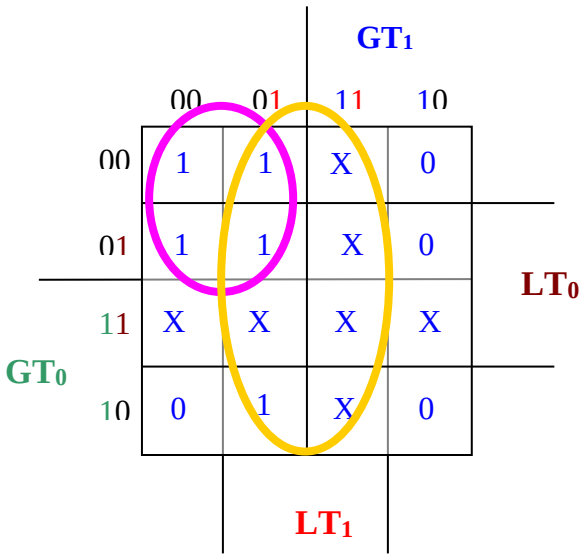
LT = LT_1 + GT_1' . LT_0



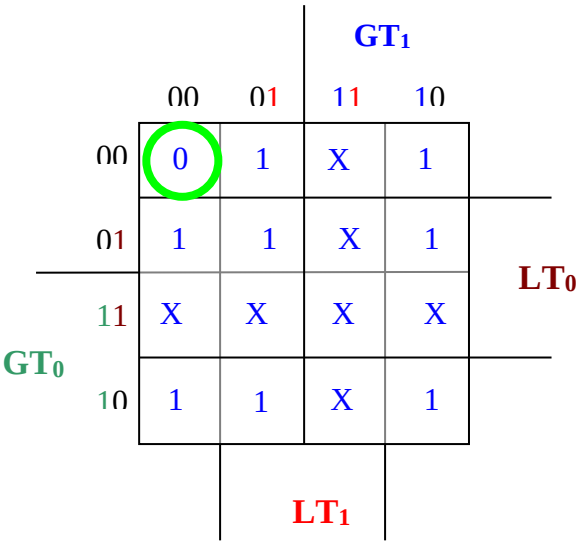
Eq = GT_1' . LT_1' . GT_0' . LT_0'



GE = GT_1 + LT_1' . LT_0'



LE = LT_1 + GT_1' . GT_0'



DIF = GT_1 + LT_1 + GT_0 + LT_0

O circuito final obtido é mostrado na figura 8.19. O circuito contém dois comparadores de magnitude de 16 bits "COMPMC16" associados em paralelo e os circuitos lógicos implementados a partir das expressões lógicas obtidas através do auxílio dos respectivos mapas de Karnaugh.

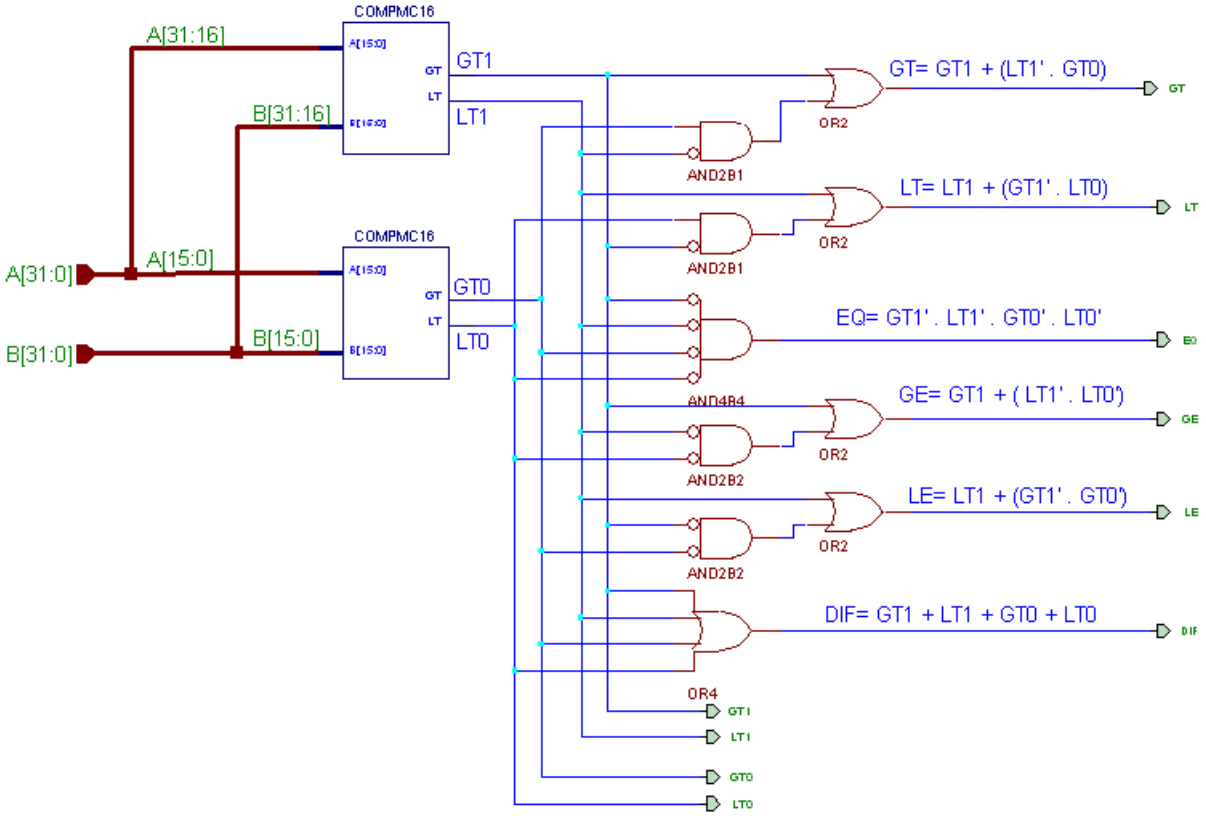


Figura 8.19- Circuito que implementa as Operações Relacionais de 32 bits, com dois componentes COMPMC16

A tabela 8.x descreve a foram de funcionamento do circuito implementado que executa todas as operações relacionais em "hardware".

Entradas		Relação	Saídas					
			GT	LT	EQ	GE	LE	DIF
A[31:0]	B[31:0]	A > B	1	0	0	1	0	1
A[31:0]	B[31:0]	A < B	0	1	0	0	1	1
A[31:0]	B[31:0]	A = B	0	0	1	1	1	0
A[31:0]	B[31:0]	A ≥ B	X	0	X	1	X	X
A[31:0]	B[31:0]	A ≤ B	0	X	X	X	1	X
A[31:0]	B[31:0]	A # B	X	X	0	X	X	1

Tabela 8.12- Tabela Verdade do Comparador de Magnitude de 32 bits da figura 8.19

8.2.7- Unidade Central de Processamento (UCP)

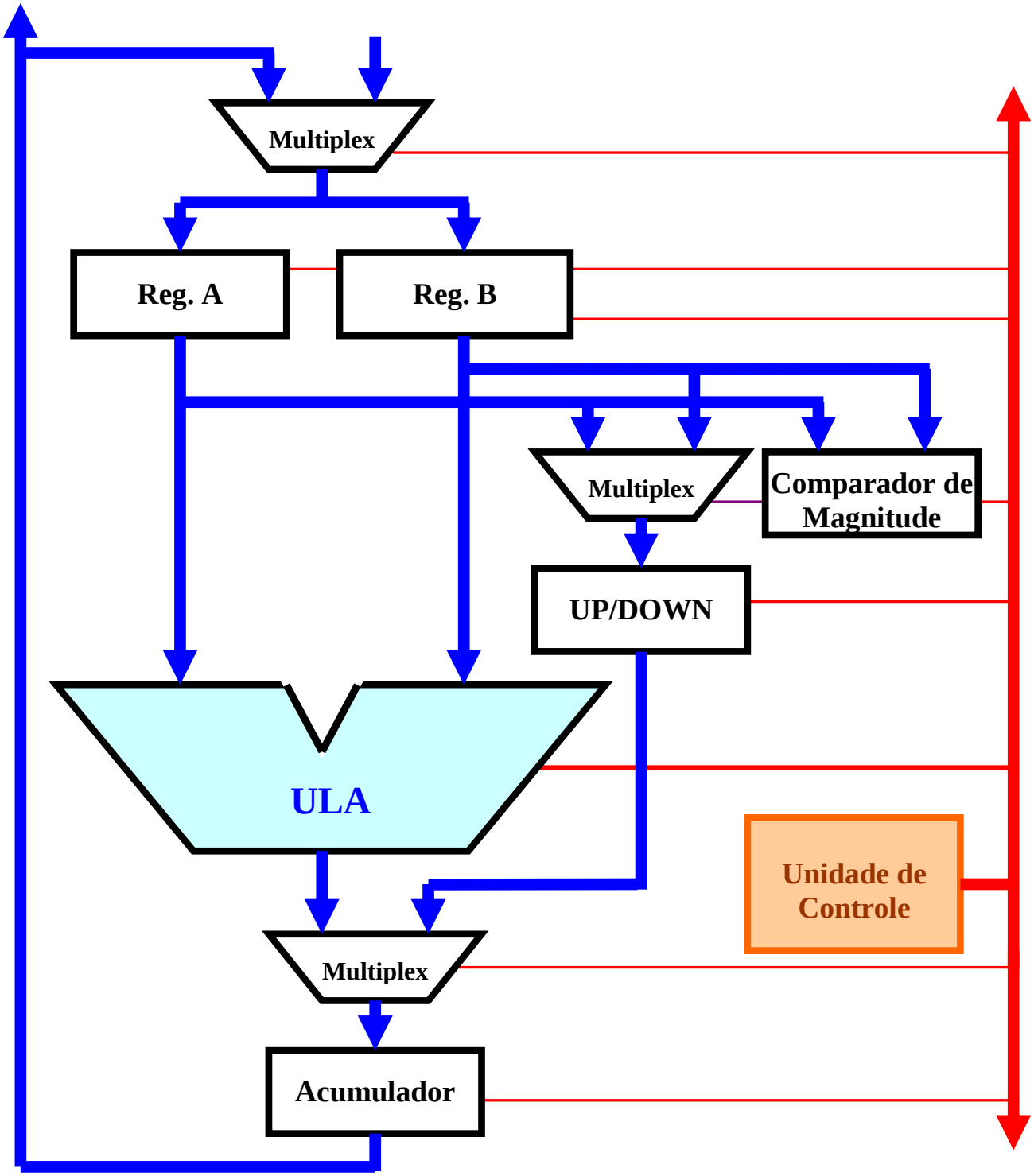
A Unidade Lógica Aritmética (ULA) é o principal componente da Unidade Central de Processamento. A ULA é responsável pela execução das instruções contidas em um programa, cada instrução de um programa é constituída por várias operações existentes na ULA. Assim, quanto maior a quantidade de operações existentes na ULA maior será a eficiência do computador. No entanto, se uma ULA executar as operações matemáticas básicas; soma, subtração, multiplicação e divisão; as operações lógicas básicas; And, Or e Not; e as operações relacionais; Maior, Menor, Igualdade e Desigualdade; é possível construir um conjunto enorme de operações mais complexas através da elaboração de microprogramas que implementam tais operações e que farão parte das operações fornecidas pela ULA em conjunto com os demais elementos contidos na CPU: Registradores de uso Geral, Acumulador, Comparadores de Magnitudes e a Unidade de Controle que irá conter os microprogramas que irão contribuir para a execução de operações mais elaboradas e por sua vez executar todas as instruções contidas em um programa.

A tabela 8.13 mostra as operações executadas na CPU proposta. A coluna denominada código de operação possui 4 bits, assim a CPU possui dezesseis (16= 2^4) diferentes operações. Sendo: sete (7) operações matemáticas (soma, subtração, multiplicação, divisão, MOD, Complemento e Inverso); cinco (5) operações relacionais (Maior, Menor, Igualdade, Maior ou Igual e Menor ou Igual) e quatro (4) operações lógicas (AND, OR, NOT e OU-Exclusivo).

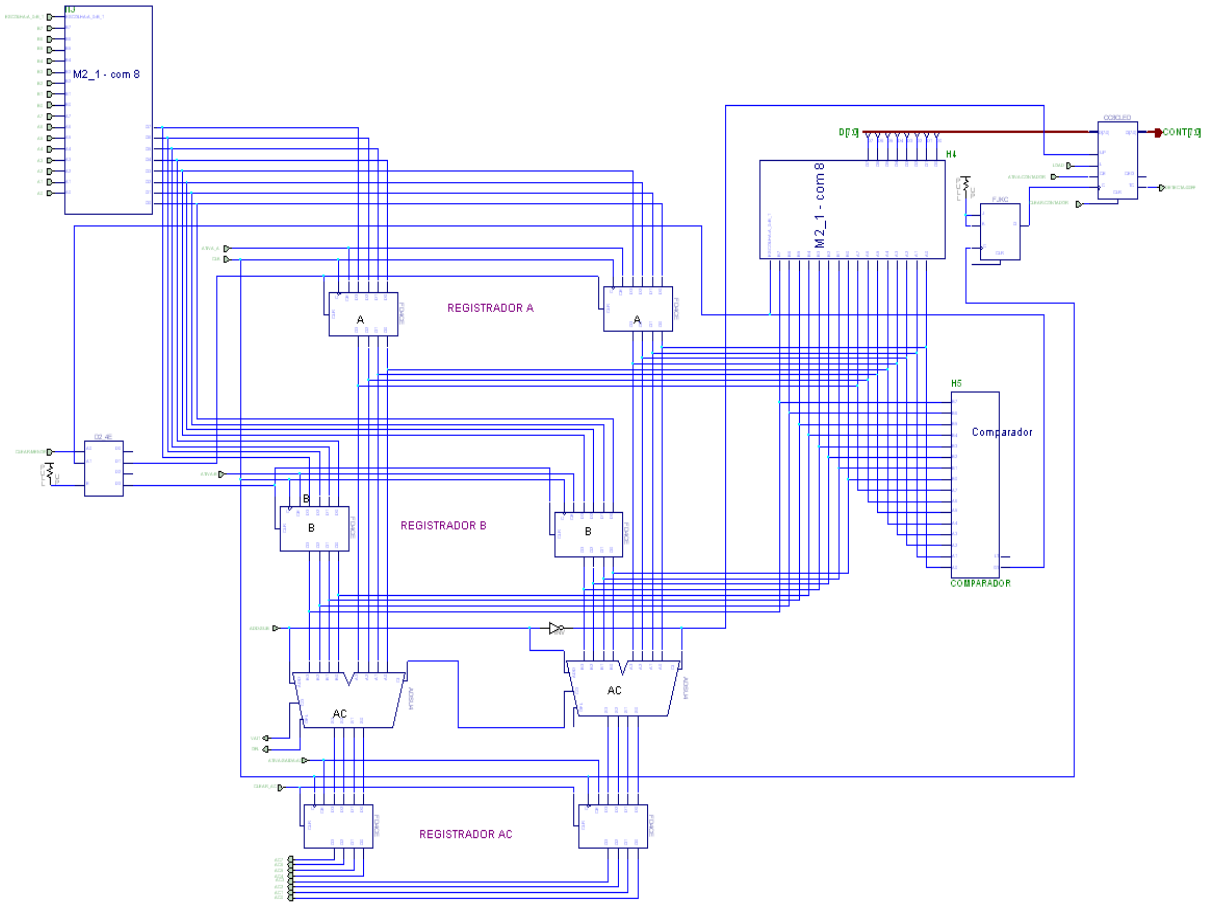
Operandos		Código da Operação				Operação	Descrição	Tipo
A	B	0	0	0	0	No Operation	---	Matemáica
A	B	0	0	0	1	Soma	AC = A + B	
A	B	0	0	1	0	Subtração	AC = A – B	
A	B	0	0	1	1	Multiplicação	AC = A * B	
A	B	0	1	0	0	Divisão	AC = A DIV B	
A	B	0	1	0	1	Mod	AC = A MOD B	
A		0	1	1	0	Complemento de X	AC=X' = FF_H – A	
A		0	1	1	1	Soma com Zero	AC = A + 00_H	
A	B	1	0	0	0	Maior	AC = (A > B)	Relacional
A	B	1	0	0	1	Menor	AC = (A < B)	
A	B	1	0	1	0	Igualdade	AC = (A = B)	
A	B	1	0	1	1	Diferente	AC = (A # B)	
A	B	1	1	0	0	AND	AC = A AND B	Lógica
A	B	1	1	0	1	OR	AC = A OR B	
A	B	1	1	1	0	NOT X	AC = NOT A	
A	B	1	1	1	1	X XOR Y	AC = A ⊕ B	

Tabela 8.13- Tabela de operações da UCP da Figura 8.16.

A Figura 8.20a contém os principais elementos presentes em uma UCP, todos os elementos presentes na referida figura foram discutidos e propostas implementações de cada um deles, exceto a Unidade de Controle (UC) que ainda não foi proposta uma implementação. Essa Unidade será implementada após a apresentação dos conceitos e propostas de Implementação do subsistema de memória presente na Figura 8.2. Pois, a UC irá também controlar a forma e os instantes de operar/utilizar o subsistema de memória.



(a) Diagrama de Blocos de uma Unidade Central de Processamento (ULA)



(b)- Implementação de uma UCP no software Project Manager

Figura 8.20- Unidade Central de Processamento

A UC coordena o instante exato de execução das atividades para obter o resultado esperado em uma instrução contida em um programa; e dentre as atividades inerentes à execução de qualquer instrução está à busca de operandos e o armazenamento do resultado. Essas operações são realizadas no Subsistema de Memória e são denominadas respectivamente de operação de leitura e operação de escrita na memória. Assim, primeiramente serão apresentados os conceitos relacionados ao subsistema de memória e propostas de implementação do mesmo, para, após isso, propor a implementação de uma Unidade de Controle para a arquitetura proposta na Figura 8.2. Essa implementação deve ser discutida em sala de aula e implementada em conjunto e com propostas elaboradas e sugeridas pelo corpo discente.

8.3- Implementação da Arquitetura de Von Neumann Expandida

A arquitetura de Von Neumann expandida proposta na Figura 8.2 foi, projetada, implementada, simulada e validada utilizando o software Project Manager da XILINX e está apresentada na Figura 8.21.

Para implementar uma máquina com a arquitetura proposta é necessário definir as características que influenciam diretamente na implementação do projeto. Tais como, comprimento de palavra; dimensões da Memória Principal.

O comprimento da palavra define a quantidade de linhas do Barramento de Dados, a quantidade de bits dos seguintes registradores: Registradores Gerais, Registrador Acumulador. Registrador de Dados de Entrada da Memória; Registrador de Saída de Dados da Memória e de todos os registradores da Memória Principal.

As dimensões da memória definem a quantidade mínima de linhas do Barramento de Endereço e a quantidade mínima do Registrador de Endereço da Memória.

A palavra de nossa máquina terá 8 bits e a memória será organizada em 4 páginas de 32 bytes, gerando uma memória de 128 bytes. Salienta-se que a modificação dessas características não influenciará na elaboração da Unidade de Controle. Assim, para aumentar as dimensões da memória deve-se dimensionar com folga o Barramento de Endereço; e para aumentar a quantidade de bits da palavra, basta realizar a associação em paralelo dos elementos de memória.

Nessa instância do projeto não estão sendo implementados os recursos inerentes à Unidade de Controle, assunto tratado no item denominado Unidade de Controle (Cap. 9).

Todos os conceitos, componentes e técnicas necessários para projetar, implementar, simular e validar o projeto foram discutidos anteriormente. Nessa seção, os elementos e subsistemas já propostos, simulados e validados nas seções anteriores serão integrados em um única FPGA gerando uma máquina de 8 bis e com memória principal de 128 bytes.

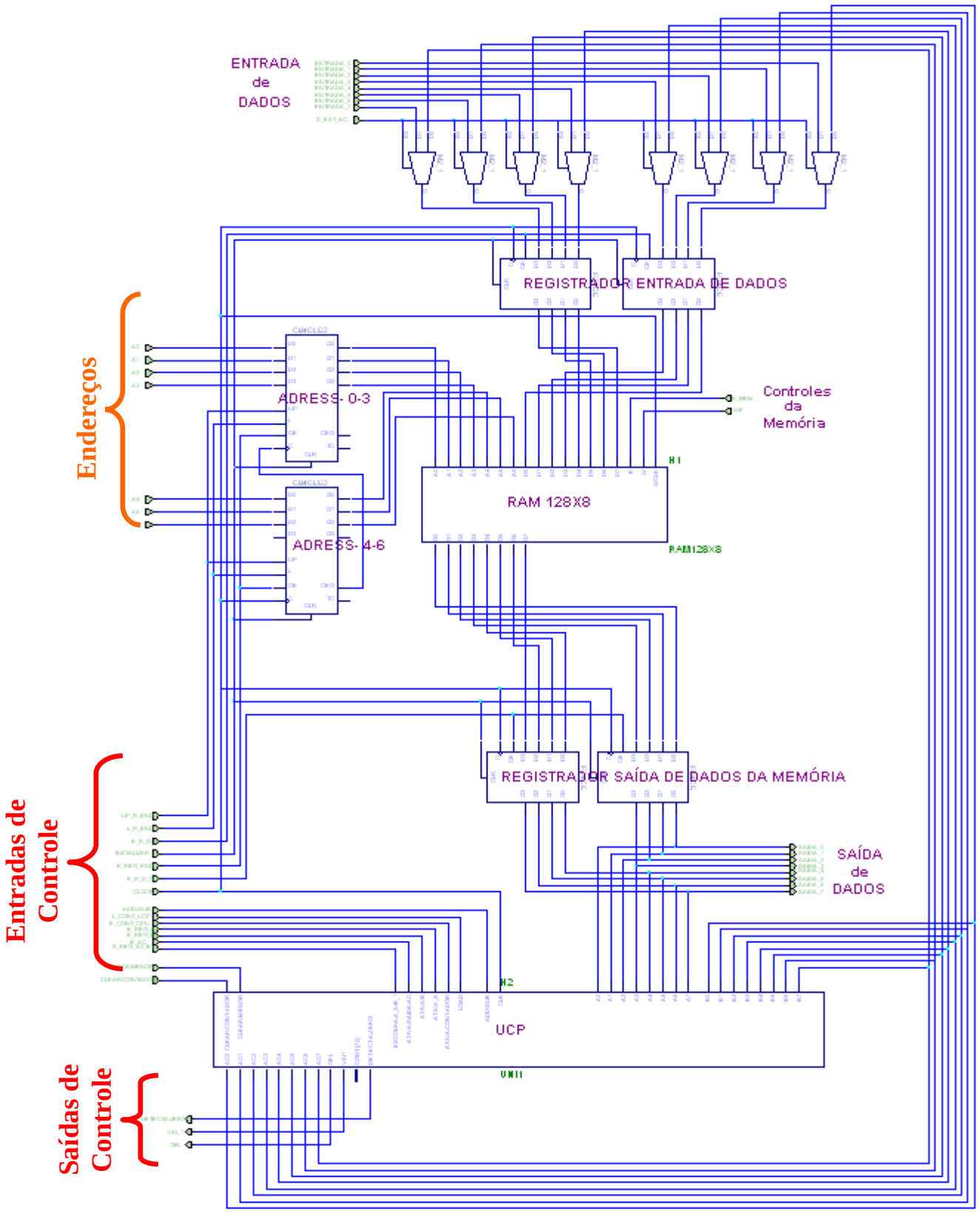


Figura 8.21- Implementação da Arquitetura de Von Neumann Expandida no Software Projet Manager

Note, que as características da máquina não estão atreladas aos conceitos adquiridos para realizar essa implementação. Os conceitos são válidos e suficientes para implementar uma máquina com quaisquer características que envolvam dimensões de memória ou comprimento de palavra. Assim essa implementação tem o objetivo principal de consolidar os conceitos e técnicas inerentes a Organização e Arquitetura de Computadores através da proposta, síntese, implementação, simulação e validação de um exemplo real.

A máquina obtida será uma ferramenta que apoiará a disseminação de conceitos mais elaborados sobre organização arquitetura de computadores, tais como: Unidade de Controle Fixo, Unidade de Controle Microprogramada, características da palavra de instrução, inteiros com sinal, linguagem de montagem, paralelismo, concorrência, instruções de ponto flutuante, e inserção de Unidades básicas de Entrada e Saída, entre outros.

Para tanto e quando necessário, serão introduzidas modificações na máquina da Figura 8.21 Tais como, novas operações na UCP, simulações sobre atividades concorrentes na arquitetura implementada, aumento da quantidade de Registradores Gerais, aumento da memória principal e inserção de Unidade de Entrada e Saída.

Esses conceitos serão detalhados no item Unidade de Controle (cap. IX).

8.3.1- Instruções e Seqüência de Microordens .

A máquina implementada executa possui as instruções de máquina relacionadas na Tabela 8.13. Nessa seção serão geradas seqüências de microordens necessárias para executar de forma correta algumas das instruções disponíveis. Dentre elas destacamos as operações matemáticas: soma, subtração, multiplicação, divisão e MOD (resto da divisão inteira) e as operações relacionais Maior, Menor, Igualdade, Maior ou Igual e Menor ou Igual. As seqüências geradas serão úteis para a implementação da Unidade de Controle.

A inserção do elemento Comparador de Magnitude na Unidade Lógica Aritmética disponibilizou tanto a execução das operações relacionais quanto a implementação de cláusulas Condicionais e comandos repetitivos (iterações).

Toda instrução para ser executada necessita ser realizada em três fases distintas; fase de busca dos operandos; fase de execução da operação e fase de armazenamento do resultado.

A seqüência de microordens para realizar a busca de um operando foi proposta na primeiramente na tabela 8.14 e como uma operação de leitura (LER) da memória e reutilizada nas tabelas 8.15 e 8.16. A mesma seqüência de mocroordens será utilizada para a busca dos operandos, no final da operação de leitura do operando ele será armazenado em um dos registradores gerais. O primeiro operando será sempre armazenado no "**Registrador Geral A"** e o segundo operando, quando existir, será armazenado no **"Registrador Geral B".** Essa ordem de escolha dos registradores gerais só poderá ser modificada se for exigido pela operação em execução. A escolha dos registradores gerais será determinada pelo campo de operação da instrução.

A seqüência de microordens para realizar o armazenamento do resultado da operação executada sobre os operandos foi proposta na Tabela 8.14 como uma operação de escrita (ARMAZENA) na memória e reutilizada nas tabelas 8.15 e 8.16. A seqüência de microordens será utilizada na operação de armazenamento do resultado de uma operação. A origem do dado no salvamento do resultado na memória (armazenamento) será sempre o conteúdo do **Registrador Acumulador.**

A fase de execução da operação será semelhante entre algumas instruções. Por exemplo, existe apenas uma diferença entre a realização de uma operação soma e a realização de uma operação subtração. A diferença é encontrada no passo **"1"** da fase de execução da operação e está relacionada à definição do código da operação a ser executada na ULA. Em todas as outras fases, a seqüência de microordens é idêntica, mudando, evidentemente, os endereços e os valores dos operandos quando necessário.

As diferenças nas operações relacionais resumem-se, também, ao passo que define o código da operação em execução na ULA.

A fase de busca de operandos e armazenamento dos resultados são sempre muito semelhantes em todas as instruções da UCP.

As instruções de "**Divisão**" e "**MOD"** são executadas simultaneamente na UCP. A diferença não está somente no código da operação a ser executada na ULA, mas também, na origem do resultado da operação executada. Na operação "**MOD**", o resultado da operação está contido no **Registrador Acumulador,** e o resultado da operação "**Divisão**" está contido no contador "**UP/DOWN"** da UCP. Assim, as escolhas dos caminhos a serem percorridos pelos resultados das duas operações dentro da UCP são diferentes.

Observa-se que para assimilar e gerar as seqüências de microordens é indispensável dominar a arquitetura proposta ou a arquitetura implementada pela equipe do aluno. Para tanto é necessário conhecer cada um dos elementos contidos na implementação.

8.3.1.1- Seqüência de Microordens da Operação Soma

A seqüência de microordens e atividades que devem ser respeitadas para realizar uma operação de Soma na máquina da Figura 8.21 é mostrada na tabela 8.14. Como exemplo, será executada a instrução a abaixo com as seguintes condições iniciais:

W:= X + Y; - o endereço da variável **X** é **AB_H**; e o seu valor é **59_H**;

- o endereço da variável **Y** é **AC_H**; e o seu valor é **$4F_H$**;

- o endereço da variável **W** é **AD_H**.

O algoritmo que executa a operação soma nas condições acima assume a seguinte forma na arquitetura implementada:

"Ler(X);

Armazenar o X no endereço AB_H;

Salvar X no REG_A;

Ler(Y);

Armazenar o Y no endereço AC_H;

Salvar Y no REG_B;

AC := REG_A + REG_B

Armazenar AC no Endereço AD_H (W);

Mostre AD_H".

Seqüência de Controle da Operação SOMA			
Fase	**Passo**	**Comentário / Atividade**	**Microordem**
L E R X	1	- Selecionar caminho do Dado; - Disponibilizar valor de X na Entrada Externa;	- S_EXT_AC = 1; - X = **59_H**;
	2	- Disponibilizar Endereço de X; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = **AB_H**; - E_R_END = 1; - L_R_END= 1;
	3	- Ativar Enable do Registrador de Entrada de Dados;	- E_R_D_I = 1;
	4	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
Armazenar x	5	- Ativar Enable da Memória; - Definir operação de Escrita na Memória;	- E_MEM = 1; - WE =1;
	6	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑

Fase	Passo	Comentário / Atividade	Microordem
BUSCA de X	1	- Disponibilizar Endereço de **X**; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = **AB**$_H$; - E_R_END = 1; - L_R_END= 1;
	2	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	3	- Ativar Enable da Memória; - Definir operação de Leitura na Memória; - Ativar Enable do Reg. de Saída de Dados da Mem.;	- E_MEM =1; - WE = 0; - E_R_D_O = 1;
	4	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	5	- Selecionar caminho de X até o Reg. Geral A; - Ativar Enable do Reg. Geral A;	- S_REG_A_B = 0; - S_MULT_UCP= 0; - E_REG_A =1;
	6	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
LER Y **Armazenar Y em 7B$_H$**	1	- Selecionar caminho do Dado; - Disponibilizar valor de Y na Entrada Externa;	- S_EXT_AC = 1; - Y = **4F**$_H$;
	2	- Disponibilizar Endereço de Y; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = **AC**$_H$**;** - E_R_END = 1; - L_R_END= 1;
	3	- Ativar Enable do Registrador de Entrada de Dados;	- E_R_D_I = 1;
	4	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
	5	- Ativar Enable da Memória; - Definir operação de Escrita na Memória;	- E_MEM = 1; - WE =1;
	6	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
BUSCA de Y	1	- Disponibilizar Endereço de **Y**; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = **AC**$_H$**;** - E_R_END = 1; - L_R_END= 1;
	2	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	3	- Ativar Enable da Memória; - Definir operação de Leitura na Memória; - Ativar Enable do Reg. de Saída de Dados da Mem.;	- E_MEM =1; - WE = 0; - E_R_D_O = 1;
	4	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	5	- Selecionar caminho de Y até o Reg. Geral B; - Ativar Enable do Reg. Geral B;	- S_REG_A_B = 1; - S_MULT_UCP= 0; - E_REG_B =1;
	6	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
Execução	1	- Disponibilizar código da Operação SOMA;	- ADD_SUB = 1;
	2	- Ativar Enable do Registrador Acumulador;	- E_AC = 1;
	3	- Gerar/ aguardar Borda de subida no relógio (clock);	- Clock = ↑;
ARMAZENA W	1	- Selecionar caminho do Dado;	- S_EXT_AC = 0;
	2	- Disponibilizar Endereço de W; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = **AD**$_H$; - E_R_END = 1; - L_R_END= 1;
	3	- Ativar Enable do Registrador de Entrada de Dados;	- E_R_D_I = 1;
	4	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
	5	- Ativar Enable da Memória; - Definir operação de Escrita na Memória;	- E_MEM = 1; - WE =1;
	6	-Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑

Fase	Passo	Comentário / Atividade	Microordem
BUSCA de W	1	- Disponibilizar Endereço de **W**; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = **AD_H;** - E_R_END = 1; - L_R_END= 1;
	2	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	3	- Ativar Enable da Memória; - Definir operação de Leitura na Memória; - Ativar Enable do Reg. de Saída de Dados da Mem.;	- E_MEM =1; - WE = 0; - E_R_D_O = 1;
	4	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	5	- Selecionar caminho de Y até o Reg. Geral B; - Ativar Enable do Reg. Geral B;	- S_REG_A_B = 1; - S_MULT_UCP= 0; - E_REG_B =1;
	6	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;

Tabela 8.14- Operação SOMA – Seqüência de Atividades

8.4.1.2- Seqüência de Microordens das Operações Relacionais.

A seqüência de microordens e atividades, que deve ser respeitada para realizar as operações Relacionais Maior, Menor, Igualdade, Menor ou Igual e Maior ou Igual, na máquina da Figura 8.21 é mostrada na Tabela 8.15. Note que a diferença entre a seqüência de microordens para cada uma das operações acontece apenas na definição do código da operação para a ULA. As operações relacionais possibilitam a implementação de cláusulas condicionais e comandos repetitivos.

Para exemplificar será implementado um algoritmo que escolha entre duas variáveis, aquela de maior valor. Para exemplificar considere as seguintes condições iniciais:

- o endereço da variável **X** é **$7A_H$**; e o seu valor é **$F2_H$**;
- o endereço da variável **Y** é **$7B_H$**; e o seu valor é **45_H**;
- o endereço da variável **W** é **AA_H**.

O algoritmo assume a seguinte forma na arquitetura implementada:

“Ler(X);
Armazenar o X no endereço $7A_H$;
REG_A := X;
Ler(Y);
Armazenar Y no Endereço $7B_H$;
Armazenar Y no Endereço AA_H;
REG_B := Y;
Se REG_A > REG_B
Então
{ AC := REG_A;
Armazenar AC=X no Endereço AA_H};
Mostre AA_H”.

Seqüência de Controle das Operações RELACIONAIS			
Fase	**Passo**	**Comentário / Atividade**	**Microordem**
L E R X **Armazenar X**	1	- Selecionar caminho do Dado; - Disponibilizar valor de X na Entrada Externa;	- S_EXT_AC = 1; - X = **F2**$_H$;
	2	- Disponibilizar Endereço de X; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = **7A**$_H$; - E_R_END = 1; - L_R_END= 1;
	3	- Ativar Enable do Registrador de Entrada de Dados;	- E_R_D_I = 1;
	4	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
	5	- Ativar Enable da Memória; - Definir operação de Escrita na Memória;	- E_MEM = 1; - WE =1;
	6	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
B U S C A de X **REG_A:=X**	1	- Disponibilizar Endereço de **X**; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = **7A**$_H$; - E_R_END = 1; - L_R_END= 1;
	2	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	3	- Ativar Enable da Memória; - Definir operação de Leitura na Memória; - Ativar Enable do Reg. de Saída de Dados da Mem.;	- E_MEM =1; - WE = 0; - E_R_D_O = 1;
	4	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	5	- Selecionar caminho de X até o Reg. Geral A; - Ativar Enable do Reg. Geral A;	- S_REG_A_B = 0; - S_MULT_UCP= 0; - E_REG_A =1;
	6	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
L E R Y **Armazenar Y em 7B**$_H$	1	- Selecionar caminho do Dado; - Disponibilizar valor de Y na Entrada Externa;	- S_EXT_AC = 1; - Y = **45**$_H$;
	2	- Disponibilizar Endereço de Y; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = **7B**$_H$**;** - E_R_END = 1; - L_R_END= 1;
	3	- Ativar Enable do Registrador de Entrada de Dados;	- E_R_D_I = 1;
	4	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
	5	- Ativar Enable da Memória; - Definir operação de Escrita na Memória;	- E_MEM = 1; - WE =1;
	6	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
Armazenar Y em AA$_H$	1	- Disponibilizar Endereço de Y; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = **AA**$_H$**;** - E_R_END = 1; - L_R_END= 1;
	2	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
	3	- Ativar Enable da Memória; - Definir operação de Escrita na Memória;	- E_MEM = 1; - WE =1;
	4	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑

Fase	Passo	Comentário / Atividade	Microordem
BUSCA de Y **REG_B:=Y**	1	- Disponibilizar Endereço de **Y**; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = **$7B_H$;** - E_R_END = 1; - L_R_END= 1;
	2	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	3	- Ativar Enable da Memória; - Definir operação de Leitura na Memória; - Ativar Enable do Reg. de Saída de Dados da Mem.;	- E_MEM =1; - WE = 0; - E_R_D_O = 1;
	4	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	5	- Selecionar caminho de Y até o Reg. Geral B; - Ativar Enable do Reg. Geral B;	- S_REG_A_B = 1; - S_MULT_UCP= 0; - E_REG_B =1;
	6	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
Execução	1	- Se X > Y	- GT = 1;
	2	- Limpar REG_B (REG B := 0)	- CL_REG_B
	3	- Disponibilizar código da Operação SOMA;	- ADD_SUB = 1;
	4	- Ativar Enable do Registrador Acumulador;	- E_AC = 1;
	5	- Gerar/ aguardar Borda de subida no relógio (clock);	- Clock = ↑; (AC:= X)
ARMAZENA Maior	1	- Selecionar caminho do Dado;	- S_EXT_AC = 0;
	2	- Disponibilizar Endereço; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = AA_H; - E_R_END = 1; - L_R_END= 1;
	3	- Ativar Enable do Registrador de Entrada de Dados;	- E_R_D_I = 1;
	4	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
	5	- Ativar Enable da Memória; - Definir operação de Escrita na Memória;	- E_MEM = 1; - WE =1;
	6	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
BUSCA de W	1	- Disponibilizar Endereço de **W**; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = **AA_H;** - E_R_END = 1; - L_R_END= 1;
	2	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	3	- Ativar Enable da Memória; - Definir operação de Leitura na Memória; - Ativar Enable do Reg. de Saída de Dados da Mem.;	- E_MEM =1; - WE = 0; - E_R_D_O = 1;
	4	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	5	- Selecionar caminho de Y até o Reg. Geral B; - Ativar Enable do Reg. Geral B;	- S_REG_A_B = 1; - S_MULT_UCP= 0; - E_REG_B =1;
	6	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;

Tabela 8.15- Operações RELACIONAIS – Seqüência de Atividades

8.4.1.3- Seqüência de Microordens da Operação Multiplicação

A seqüência de microordens e atividades que deve ser respeitada para realizar uma operação de Multiplicação de dois inteiros sem sinal na máquina da Figura 8.21 é mostrada na tabela 8.16. A operação de multiplicação será implementada primeiramente através da utilização do algoritmo de multiplicação utilizando Somas Sucessivas. Serão desenvolvidos, oportunamente, algoritmos mais eficientes, que implementam a operação de multiplicação dois inteiros com sinal e, também a operação de multiplicação em ponto flutuante.

O algoritmo de multiplicação utilizando somas sucessivas é adequado para validar a máquina, pois utiliza todos os recursos disponíveis na arquitetura implementada, incluindo a execução de comandos repetitivos e cláusulas condicionais.

Para exemplificar será implementado o algoritmo de Somas Sucessivas que realiza execução da seguinte expressão:

"W := X * Y"

O algoritmo assume a seguinte forma na arquitetura implementada.

"Ler(X);
Armazenar X no endereço $7A_H$;
Salvar X no REG_A;
Ler(Y);
Armazenar Y no endereço $7B_H$;
Salvar Y no REG_B;
Se Y > X
Então
{ Salvar o valor de X no Registrador Geral A;
Salvar o valor de Y no contador UP/DOWN;
Zerar Registrador Geral B };
Enquanto Contador UP/DOWN > 0
Faça
{ AC := AC + REG_B;
REG_B := AC };
Armazenar AC no Endereço W;
Mostre W".

O detalhamento da seqüência de microordens está apresentado na tabela 8.16. Para tanto, estão sendo consideradas as seguintes condições iniciais:

- o endereço da variável **X** é **$7A_H$**; e o seu valor é **59_H**;
- o endereço da variável **Y** é **$7B_H$**; e o seu valor é **$4F_H$**;
- o endereço da variável **W** é **AD_H**.

Seqüência de Controle da operação MULTIPLICAÇÃO			
Fase	**Passo**	**Comentário / Atividade**	**Microordem**
LER X **Armazenar X em $7A_H$**	1	- Selecionar caminho do Dado; - Disponibilizar valor de X na Entrada Externa;	- S_EXT_AC = 1; - X = **59_H**
	2	- Disponibilizar Endereço de X; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = **$7A_H$**; - E_R_END = 1; - L_R_END= 1;
	3	- Ativar Enable do Registrador de Entrada de Dados;	- E_R_D_I = 1;
	4	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
	5	- Ativar Enable da Memória; - Definir operação de Escrita na Memória;	- E_MEM = 1; - WE =1;
	6	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
BUSCA de X **Salvar X em REG_A**	1	- Disponibilizar Endereço de **X**; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = **$7A_H$**; - E_R_END = 1; - L_R_END= 1;
	2	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	3	- Ativar Enable da Memória; - Definir operação de Leitura na Memória; - Ativar Enable do Reg. de Saída de Dados da Mem.;	- E_MEM =1; - WE = 0; - E_R_D_O = 1;
	4	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	5	- Selecionar caminho de X até o Reg. Geral A; - Ativar Enable do Reg. Geral A;	- S_REG_A_B = 0; - S_MULT_UCP= 0; - E_REG_A =1;
	6	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
LER Y **Armazenar Y em $7B_H$**	1	- Selecionar caminho do Dado; - Disponibilizar valor de Y na Entrada Externa;	- S_EXT_AC = 1; - Y = **$4F_H$**;
	2	- Disponibilizar Endereço de Y; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = **$7B_H$;** - E_R_END = 1; - L_R_END= 1;
	3	- Ativar Enable do Registrador de Entrada de Dados;	- E_R_D_I = 1;
	4	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
	5	- Ativar Enable da Memória; - Definir operação de Escrita na Memória;	- E_MEM = 1; - WE =1;
	6	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
BUSCA de Y **Salvar Y em REG_B**	1	- Disponibilizar Endereço de **Y**; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = 7B**H;** - E_R_END = 1; - L_R_END= 1;
	2	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	3	- Ativar Enable da Memória; - Definir operação de Leitura na Memória; - Ativar Enable do Reg. de Saída de Dados da Mem.;	- E_MEM =1; - WE = 0; - E_R_D_O = 1;
	4	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	5	- Selecionar caminho de Y até o Reg. Geral B; - Ativar Enable do Reg. Geral B;	- S_REG_A_B = 1; - S_MULT_UCP= 0; - E_REG_B =1;
	6	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;

Fase	Passo	Comentário / Atividade	Microordem
Execução	1	- Disponibilizar código da Op. MULTIPLICAÇÃO;	- S_MULT_UCP = 1;
Salvar menor no CONT_UCP	2	- Ativar LOAD do Contador UP/DOWN da UCP;	- L_CONT_UCP = 1;
	3	- Aguardar Borda de subida no relógio (clock);	- Clock = ↑;
	4	- Limpa REG_A ou REG_B (operando menor)	- Clear_Menor = 1;
Iterações	5 (1)	- EnquantoDetecta_Zero = 0	- Detecta_Zero = 0;
	6(2)	- Disponibilizar código da operação SOMA	- ADD_SUB =1;
	7(3)	- Ativar Enable do Registrador Acumulador;	- E_AC = 1;
	8(4)	- Gerar clock; (AC:= REG_A + REG_B);	- Clock = ↑;
	9(5)	- Ativar ENABLE do contador UP/DOWN da UCP;	- E_CON_UCP = 1;
	10(6)	- Gerar clock; (REG_menor:=AC;CONT_UCP:= CONT_UCP-1);	- Clock = ↑;
ARMAZENA W	1	- Selecionar caminho do Dado;	- S_EXT_AC = 0;
	2	- Disponibilizar Endereço de W; - Ativar Enable do Registrador de Endereço; - Ativar LOAD do Registrador de Endereço;	- END = AD_H; - E_R_END = 1; - L_R_END= 1;
	3	- Ativar Enable do Registrador de Entrada de Dados;	- E_R_D_I = 1;
	4	- Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑
	5	- Ativar Enable da Memória; - Definir operação de Escrita na Memória;	- E_MEM = 1; - WE =1;
	6	-Gerar/ aguardar Borda de subida no relógio (clock)	- Clock = ↑

Tabela 8.16- Operação MULTIPLICAÇÃO – Seqüência de Atividades

Anexo VIII.1

Projetos Propostos

Projeto 8.1

ULA de 32 bits

Sintetizar, projetar, implementar e validar através de simulação, uma Unidade Lógica Aritmética capaz de realizar as operações de soma e subtração sobre palavras de 32 bits. O circuito deverá também conter dois registradores gerais para conter os operandos de 32 bits e um registrador acumulador de 32 bits para conter o resultado das operações realizadas pelo circuito somador/subtrator.

A figura P81-1 mostra em diagrama de blocos uma sugestão da estrutura do projeto solicitado.

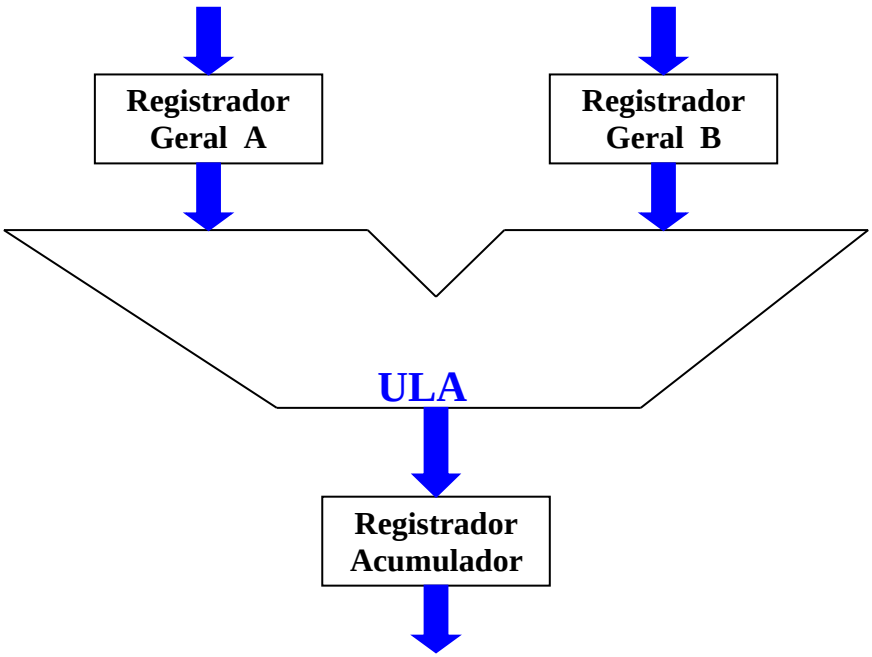


Figura P81-1- Diagrama de blocos de uma Unidade Aritmética de 32 bits

Observações:

a- Para realizar a operação de subtração devem ser utilizados meios somadores completos. Assim, a subtração deverá ser transformada através em uma operação de soma em complemento do maior dígito da base (complemento de 1).

b- O relatório deverá seguir o formato sugerido no documento "Roteiro para a Geração de Relatórios de Projetos" (Anexo VII-2).

Projeto 8.2

UCP de 32 Bits – Estruturas Iterativas e Condicionais (Operação de Multiplicação)

Utilizando circuito da ULA de 32 bits desenvolvida e implementada no projeto 8.1, sintetize, projete, implemente, simule e valide uma UCP de 32 bits contendo os subsistemas necessários para permitir a inclusão de operações relacionais, comandos (estruturas) condicionais e comandos iterativos.

Para tanto, devem ser inseridos no circuito da ULA de 32 bits, componentes que permitam a execução dos conjuntos de comandos e instruções solicitados.

A figura P82-1 mostra em diagrama de blocos a proposta de uma arquitetura para o projeto solicitado.

Relacione, descreva a forma de funcionamento e comente a função exercida por cada componente inserido para a ULA de 32 bits realizar os comandos e operações solicitadas.

Para validar o circuito desenvolvido, desenvolver um algoritmo para inserir na UCP a operação de multiplicação de dois números inteiros. Para tanto, deve ser utilizado o algoritmo que realiza a operação multiplicação através de somas sucessivas. Para auxiliar a validação do circuito, realize a simulação da seguinte instrução:

$$W := A7B4C02F_{(H)} * 5_{(H)}$$

Relacione e comente a seqüência de micro-ordens necessárias para executar a operação de multiplicação no circuito validado.

Observações:

a- o algoritmo deve otimizar a quantidade de somas necessárias para realizar a multiplicação através do algoritmo de somas sucessivas. Por exemplo, a instrução:

$$W := A7B4C02F_{(H)} * 5_{(H)}$$

Deve, ser implementada através da realização de 5 somas do valor $A7B4C02F_{(H}$.

Ou seja:

$$W:= A7B4C02F_{(H} + A7B4C02F_{(H} + A7B4C02F_{(H} + A7B4C02F_{(H} + A7B4C02F_{(H}$$

b- O relatório deverá seguir o formato sugerido no documento “Roteiro para a Geração de Relatórios de Projetos” (Anexo VII-2).

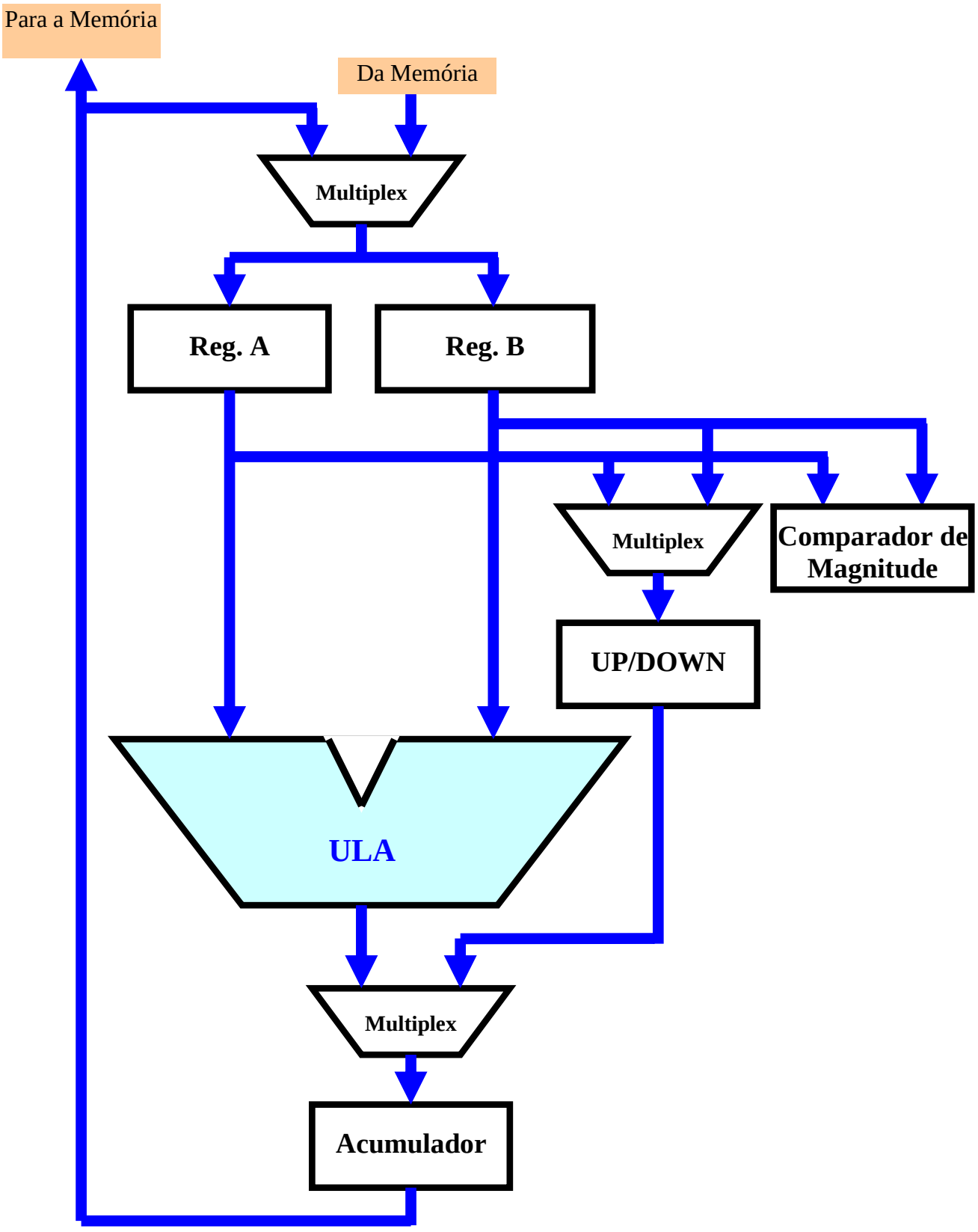


Figura P82-1- Representação em diagrama de blocos de proposta do circuito solicitado.

Projeto 8.3

UCP de 32 Bits – Estruturas Iterativas e Condicionais (Operação de Divisão)

Utilizando circuito da UCP de 32 bits desenvolvida e implementada no projeto 8.2, gerar a seqüência de microordens (microprograma) necessárias para dotar a UCP com a operação divisão de dois números inteiros sem sinal de 32 bits.

Relacione e descreva a forma de funcionamento e, comente a função exercida por cada componente inserido para a ULA de 32 bits, para realizar a operação de divisão

Para validar o circuito desenvolvido, executar o microprograma (seqüência de microordens) para realizar a seguinte operação de divisão:

W := AAAAAAAA DIV 33333333

Observações:

a- O projeto da operação de divisão inserido na UCP deverá prever a divisão por zero;

b- O relatório deverá seguir o formato sugerido no documento "Roteiro para a Geração de Relatórios de Projetos" (Anexo VII-2).

Projeto 8.4

Máquina de 32 bits com Arquitetura de Von-Neumann

O principal objetivo desse texto é capacitar o leitor a projetar, dimensionar, implementar e avaliar sistemas computacionais de acordo com as necessidades e características de uso dos mesmos.

Uma arquitetura de computador deve ser avaliada ou dimensionada de acordo com as características de aplicação ou uso das mesmas. Ou seja, não podemos simplesmente afirmar que uma arquitetura eficiente (boa) ou ineficiente (ruim) se não for considerada a aplicação na qual ela está sendo empregada.

O projeto em tela será utilizado tanto para consolidar os conceitos e técnicas de organização e arquitetura de computadores já repassados aos alunos; quanto para projetar, implementar, validar o hardware de uma máquina com todos os subsistemas existentes em arquitetura de Von Neumann e a conseqüente avaliação de desempenho, medida em MIPS, da mesma. A máquina será de utilidade para aplicar os conceitos contidos no item Unidade de Controle (cap. IX).

Para tanto, os subsistemas já desenvolvidos e validados separadamente (UCP-32 e RAM-2048x32) podem ser reutilizados através da organização e integração dos mesmos em um único sistema que constituirá a máquina solicitada.

A máquina a ser projetada deverá conter as seguintes características básicas:

- UCP de 32 bits contendo: dois registradores gerais; um registrador acumulador, uma ULA de 32 bits com capacidade para executar instruções aritméticas e relacionais; subsistemas para auxiliar a execução de instruções condicionais e iterativas;
- Memória RAM estática de 2048 palavras de 32 bits.

A Figura P84.1 mostra, em diagrama de blocos, uma sugestão de arquitetura para a máquina solicitada.

O trecho de programa que segue, escrito em uma linguagem de alto nível genérica, deverá ser utilizado para simular e validar a máquina implementada. Para tanto, deve ser gerada e comentada a seqüência de microordens necessárias para executar o programa na máquina obtida.

```
“ READ(X);
  READ(Y);
  R := X + Y;
  S := X – Y;
  WRITE(R);
  WRITE(S)”
```

Observações:

a- Os endereços das variáveis contidas no trecho de programa devem obedecer aos critérios contidos na tabela abaixo:

Variável	Endereço	Valor
X	Banco zero da memória	31E7CD39 (H)
Y	Banco zero da memória	1EF05248 (H)
R	Banco um da memória	calculado
S	Banco dois da memória	calculado

b- O relatório deverá seguir o formato sugerido no documento "Roteiro para a Geração de Relatórios de Projetos" (Anexo VII-2).

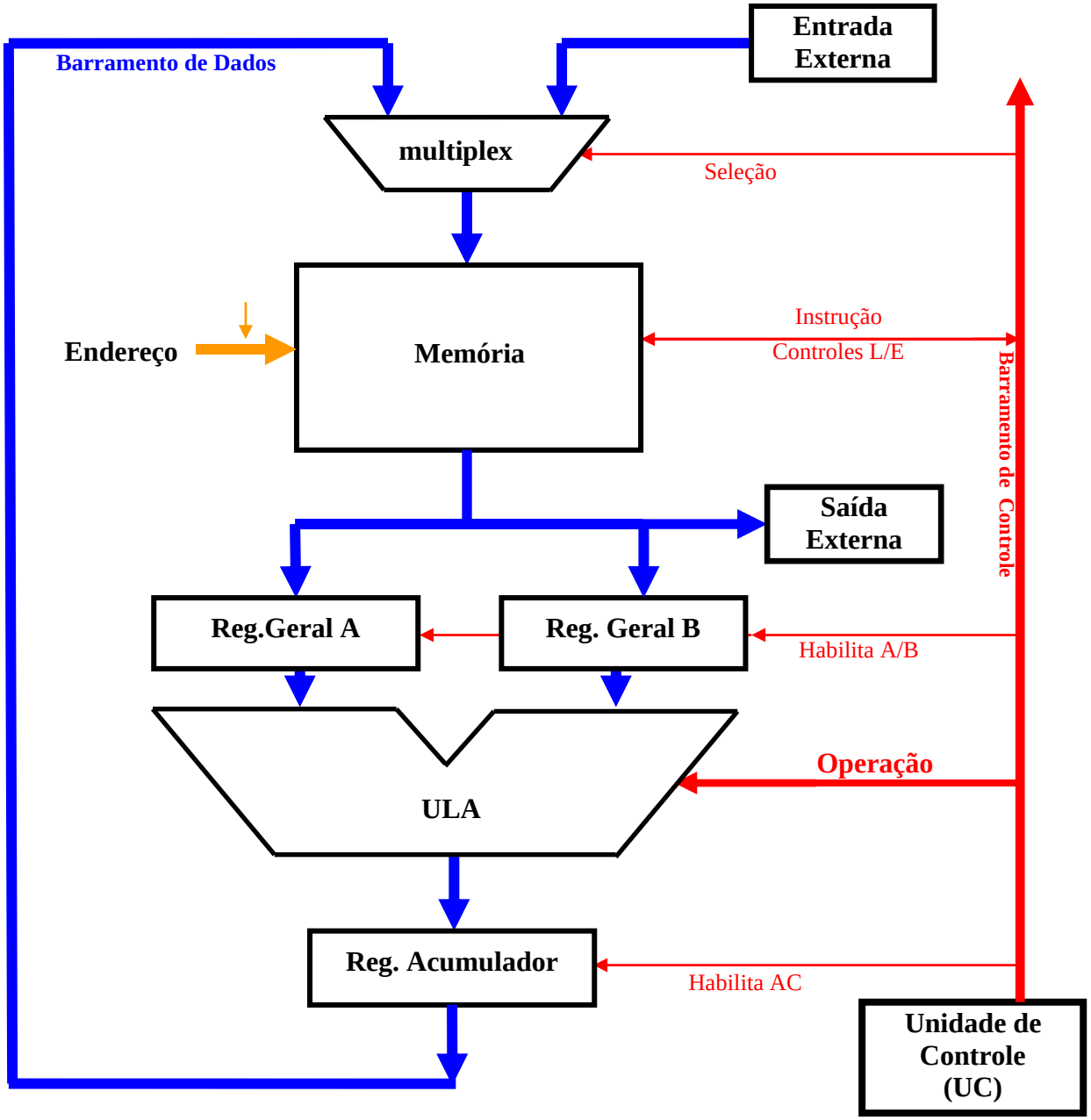


Figura P84.1– Diagrama e blocos de uma arquitetura de Von-Neumann genérica.

Projeto 8.5

Máquina de 32 bits com Arquitetura de Von-Neumann

(Operações Iterativas)

O principal objetivo do texto é o de capacitar o leitor a projetar, dimensionar, implementar e avaliar sistemas computacionais de acordo com as necessidades e características de uso dos mesmos.

Uma arquitetura de computador deve ser avaliada ou dimensionada de acordo com as características de sua aplicação. Ou seja, não podemos simplesmente afirmar que uma arquitetura é eficiente (boa) ou ineficiente (ruim) se não for considerada a aplicação na qual ela está sendo empregada.

O projeto em tela será utilizado tanto, para consolidar os conceitos e técnicas de organização e arquitetura de computadores já repassados aos alunos, quanto, para projetar, implementar, validar o hardware de uma máquina com todos os subsistemas existentes em uma arquitetura de Von Neumann clássica. Além, da conseqüente avaliação de desempenho, medida em MIPS, da mesma. A máquina será de utilidade para aplicar os conceitos contidos no capítulo IX (Unidade de Controle).

Para tanto, os subsistemas já desenvolvidos e validados separadamente nos projetos UCP-32 (projetos: 8.1; 8.2; 8.3 e 8.4) e RAM-1024x32 (projeto 7.1, contido no capítulo VII) devem ser reutilizados para implementar a arquitetura da máquina solicitada. A Figura P85.1 mostra, de forma sintética, os principais subsistemas existentes na arquitetura da máquina proposta. A Figura P85.2 mostra a UCP solicitada no projeto 8.2 (UCP de 32 bits - Estruturas Iterativas e Condicionais) em diagrama de blocos. A Figura P85.3, mostra, também em diagrama de blocos, a memória RAM solicitada no projeto 7.1 (RAM estática 2Kx32).

A máquina a ser projetada deverá conter as seguintes características básicas:

- UCP de 32 bits contendo: dois registradores gerais; um registrador acumulador, uma ULA de 32 bits com capacidade para executar instruções aritméticas e relacionais; subsistemas para auxiliar a execução de instruções condicionais e iterativas;
- Memória RAM estática de 2048 palavras de 32 bits.

O trecho de programa que segue, escrito em uma linguagem de alto nível genérica, deverá ser utilizado para simular e validar a máquina implementada. Para tanto, deve ser gerada e comentada a seqüência de microordens necessárias para executar o programa na máquina obtida.

```
“ READ(X);
  READ(Y);
  Produto := X * Y;
  READ(X);
  READ(Z);
  Quociente := X DIV Z;
   Resto := X MOD Z;
  WRITE(Produto);
  WRITE(Quociente);
  Write(Resto)”
```

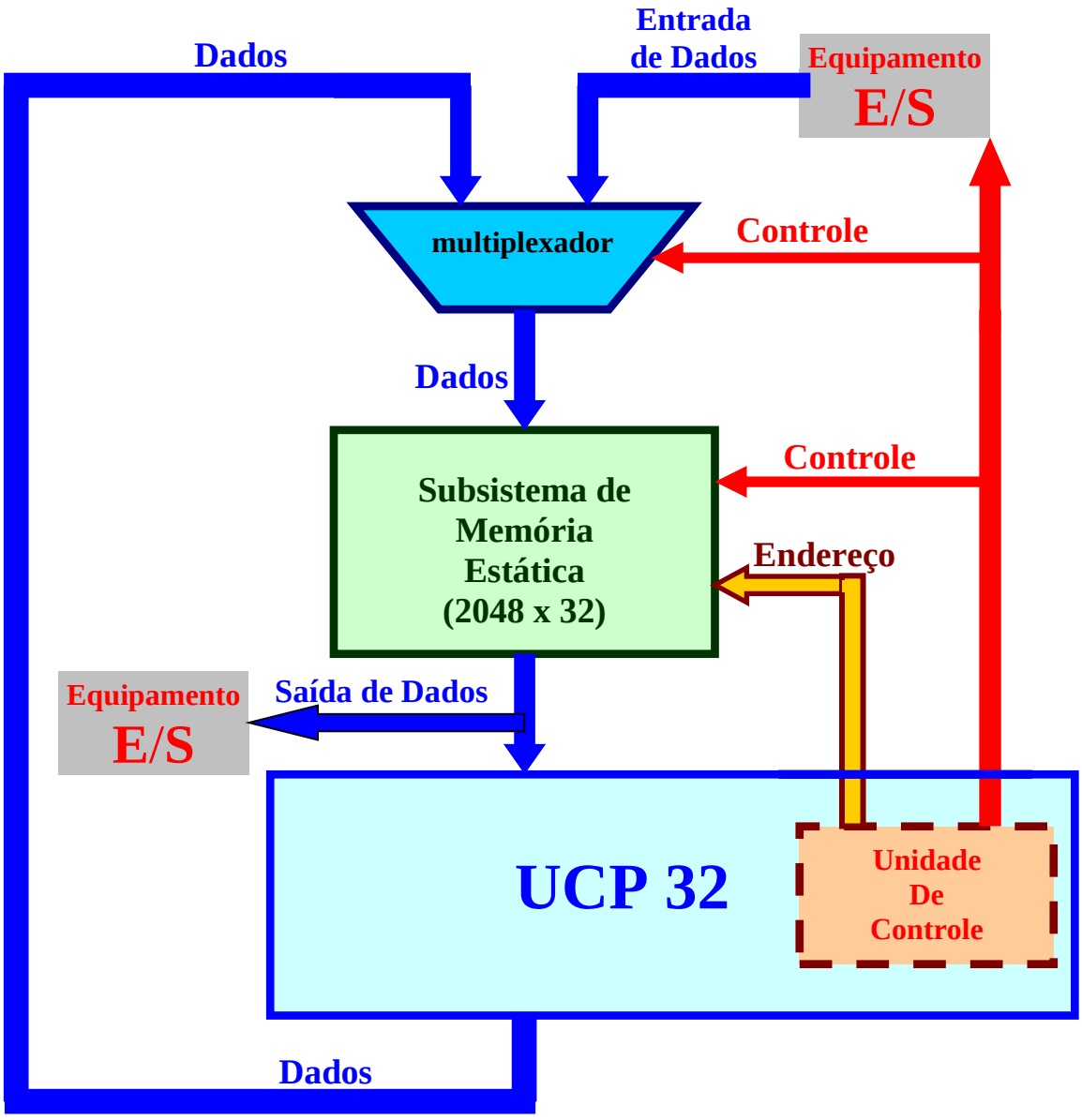


Figura P85.1 – Diagrama de Blocos da Arquitetura Proposta

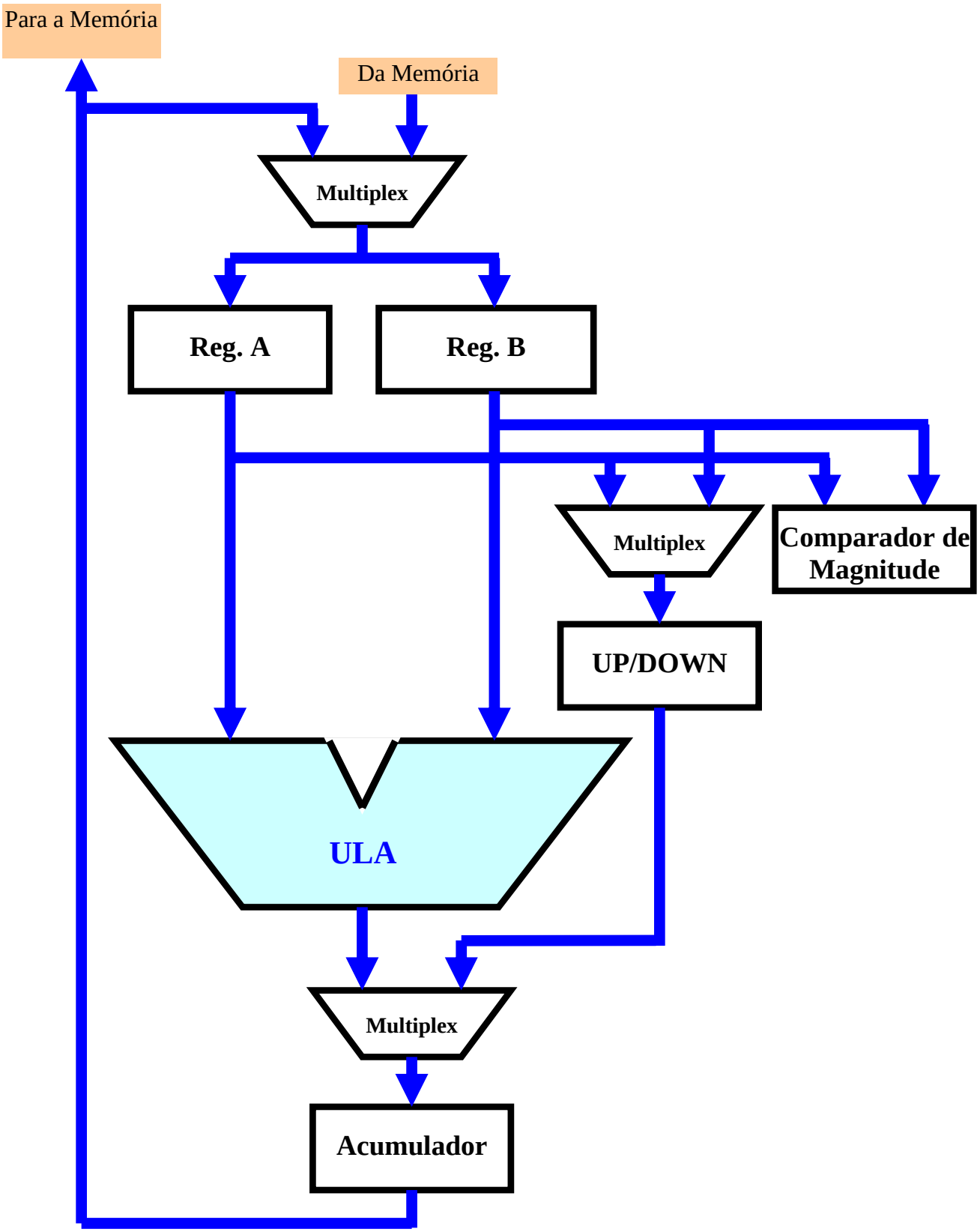


Figura P85.2- Diagrama de blocos da UCP-32, solicitada no Projeto 8.2

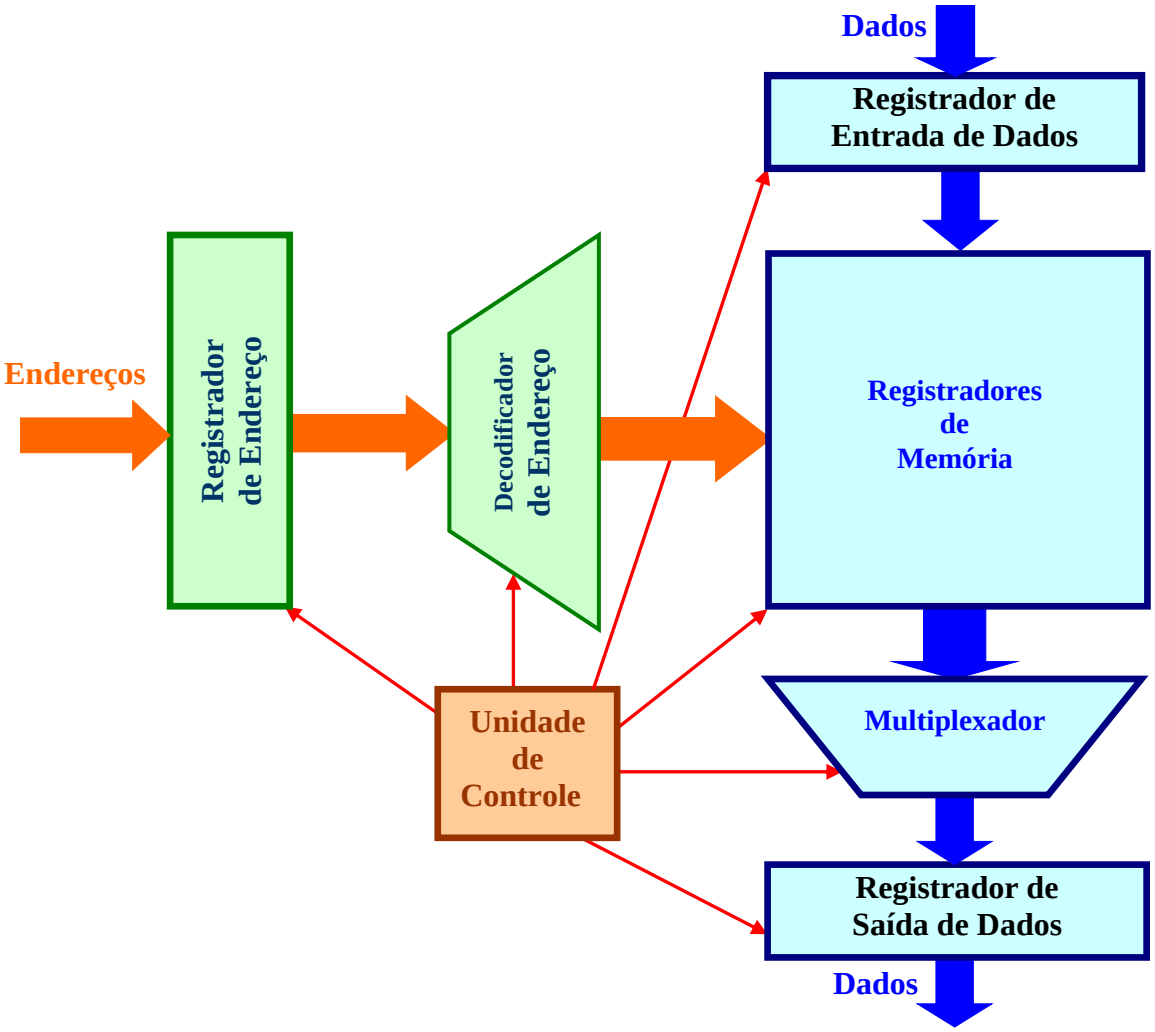


Figura P85.3- Diagrama de blocos da memória solicitada no Projeto 7.1

Observações:

a- As variáveis são todas do tipo Inteiro;

b- Os endereços das variáveis contidas no trecho de programa devem obedecer aos critérios contidos na tabela abaixo:

Variável	Endereço	Valor
X	Banco zero da memória	2DE7 (H)
Y	Banco zero da memória	4 (H)
Z	Banco zero da memória	0BFC (H)
Produto	Banco um da memória	calculado
Quociente	Banco dois da memória	calculado
Resto	Banco três da memória	calculado

b- O relatório deverá seguir o formato sugerido no documento "Roteiro para a Geração de Relatórios de Projetos" (Anexo VII-2).

Anexo VIII.2
Questões de Múltipla Escolha

1- A figura que segue, mostra em diagrama de blocos uma Unidade Central de Processamento. Calcule o tempo de execução de uma operação soma (W:=X+Y) e também a capacidade de execução de operações **soma** em MIPS (milhões de instruções por segundo) na arquitetura. Para tanto utilize as seguintes condições iniciais: tempo de leitura na memória (busca de operando) = 5ns (nanosegundos); tempo de escrita na memória = 9ns; tempo de atraso dos multiplex = 1ns, tempo de atraso dos registradores = 2ns; tempo de atraso da ULA = 5ns; tempos de atraso do Comparador de Magnitude e Contador UP/DOWN = 3ns. (Obs. 1ns = 10^{-9} segundos).

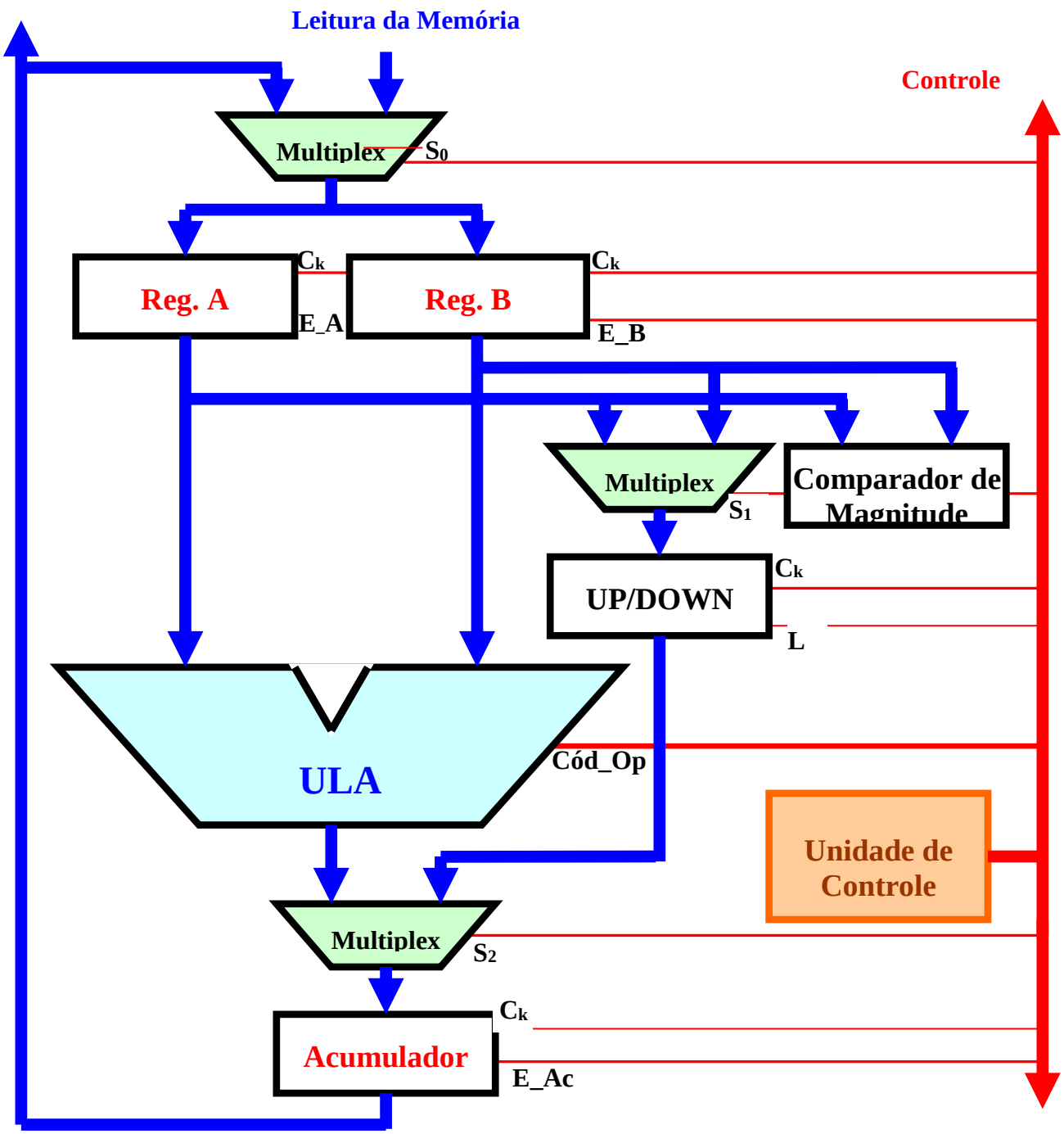


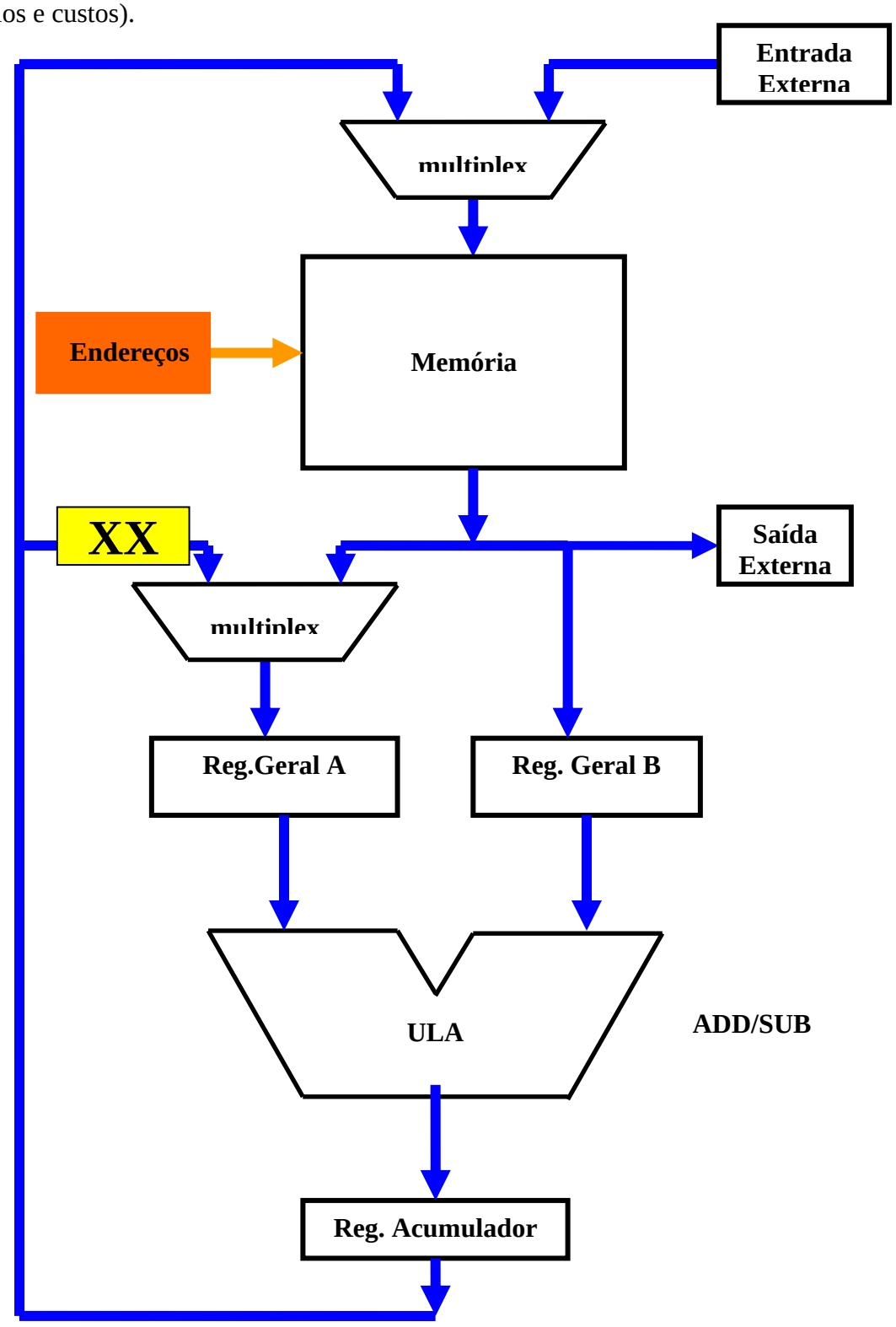
Diagrama de Blocos de uma Unidade Central de Processamento

(A) 25ns; 41,66 MIPS
(B) 32ns; 31,25 MIPS
(C) 33ns; 30,3 MIPS
(D) 25ns; 40 MIPS
(E) Nenhuma das anteriores

Respostas	
Questão	**Alternativa Correta**
1	**D**

Anexo VIII.3
Exercícios Propostos

1- Para responder a essa questão, considere a modificação enfatizada na cor amarela na figura que segue mostra a máquina com arquitetura de Von Neumann desenvolvida no projeto 8.2.
- Retirar a realimentação na entrada do registrador geral A, proveniente da saída do registrador Acumulador. Comentar os efeitos dessa mudança sobre a arquitetura original (benefícios e custos).



Arquitetura de Von Neumann em Diagrama de Blocos

2- A figura que segue mostra em diagrama de blocos a Unidade Central de Processamento (UCP), desenvolvida em sala de aula. Utilizando as informações contidas na figura, calcule a capacidade de execução de operações soma em MIPS na arquitetura. Para tanto utilize as seguintes condições iniciais: tempo de leitura na memória = 7ns; tempo de escrita na memória = 8ns; tempo de atraso dos multiplex = 1ns, tempo de atraso dos registradores=2ns; tempo de atraso da ULA=2ns; tempos de atraso do Comparador de Magnitude e Contador UP/DOWN = 2ns. Considere também que os valores dos operandos (variáveis) X e Y já estão contidos na memória.

W:=X +Y (valor de X=AF_H e Y:=$3C_H$, endereços X é $3A_H$, Y é $3B_H$ e W é $3C_H$)

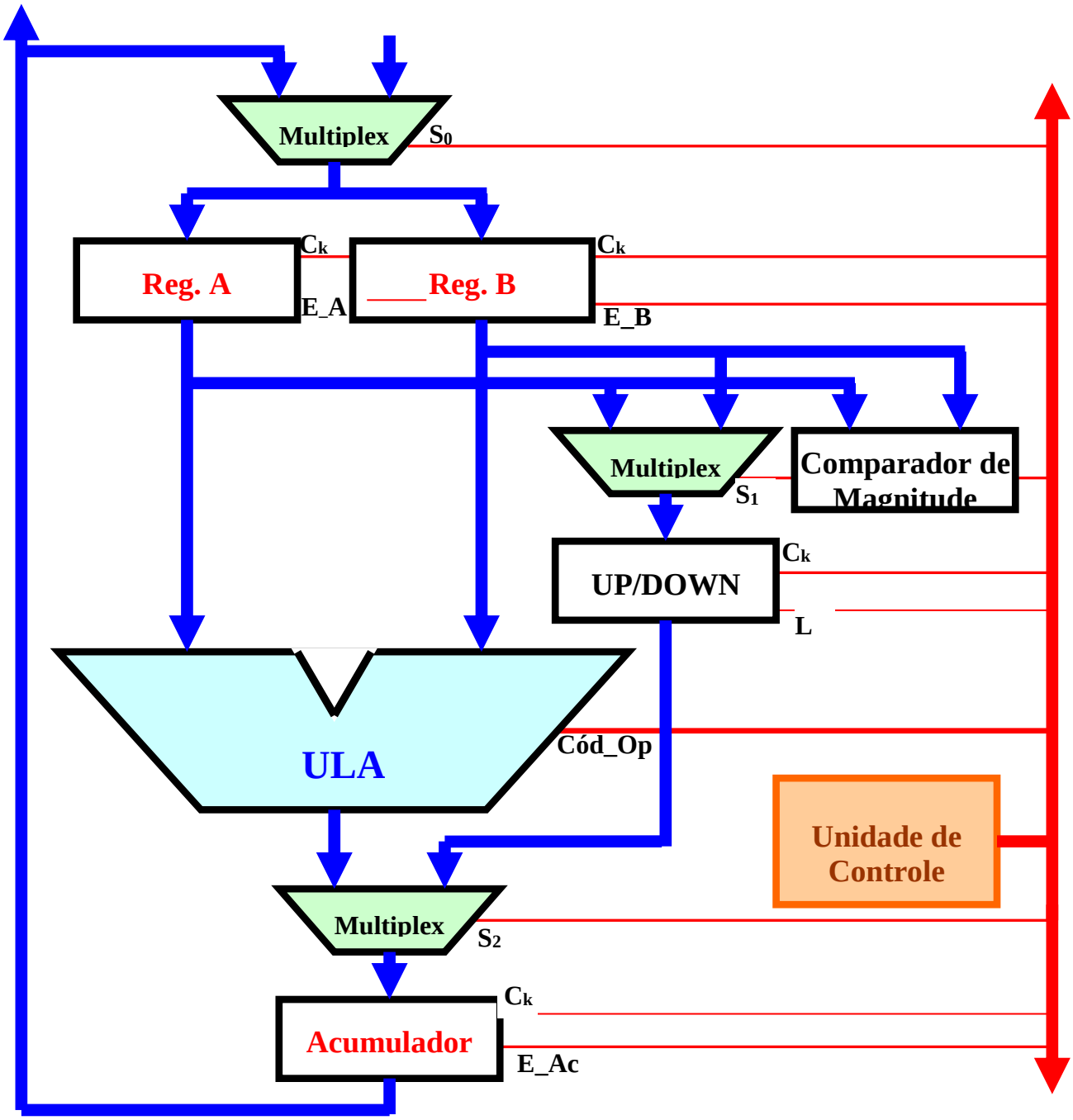


Diagrama de Blocos de uma Unidade Central de Processamento (UCP)

3- No Projeto 8.1, foi solicitado e previsto a implementação e simulação de um circuito combinacional, para realizar a soma e subtração de duas palavras de 32 bits (ULA de 32 bits), utilizando somente elementos Somadores Completos. Comente as seguintes questões:

- **a-** Descreva e comente a técnica de obtenção da operação de subtração utilizando apenas circuitos somadores completos;
- **b-** Utilizando um diagrama de blocos, descreva e comente, a solução encontrada na definição do controle, utilizado para realizar a escolha entre a execução de uma operação Soma e a execução de uma operação Subtração, implementadas no projeto;
- **c-** Utilizando o mesmo diagrama de blocos gerado no item "c", descreva e comente a solução utilizada na implementação da saída de overflow e/ou underflow do circuito projetado.

4- Sugerir ao menos uma nova operação que pode ser inserida na UCP desenvolvida. Descrever e comentar forma de execução ou o algoritmo (seqüência de microordens) da nova operação.
Obs. A operação sugerida deve ser diferente das contidas na tabela 8.13 do item 8.2.6 do desse texto.

5- Introduzir na Arquitetura de Von Neumann simplificada mostrada na figura 8.1 a seguinte modificação: Trocar a ULA que realiza somente a operação soma de dois operandos por uma ULA que realize as operações de soma e subtração (ADD/SUB). Comentar os efeitos dessa mudança sobre a arquitetura original (benefícios e custos).

'A imaginação é muito mais importante que o conhecimento'.

(Albert Einstein)