Universidade Federal de Pernambuco - UFPE

Centro de Informática - CIn

Infra-Estrutura de Hardware - if674

Especificação de Projeto de um Subconjunto de Instruções do Processador MIPS

Recife - 2015.

Apresentação

Neste documento é apresentada a especificação de projeto da disciplina Infra-Estrutura de Hardware. A elaboração de tal documento tem como base a experiência adquirida durante os períodos que os escritores desempenharam suas atividades como monitor. Em tal especificação buscamos esclarecer tudo o que deve ser feito durante as etapas do projeto e deixar claras as dúvidas que mais frequentemente aparecem durante a implementação deste trabalho.

Este documento é composto por três capítulos. No primeiro é apresentada a especificação das duas primeiras etapas do projeto. No segundo é dada uma visão geral da CPU que será implementada. Neste capítulo também está presente o repertório de instruções que o processador desenvolvido deve conter e a descrição dos componentes fornecidos pela equipe da disciplina. No terceiro capítulo é apresentada a especificação do relatório de projeto. Ainda está presente um anexo que ensina como configurar e carregar a memória que irá compor o projeto.

Deve-se ressaltar que é dever de cada equipe ler e entender o que aqui está descrito. Caso existam dúvidas, os monitores deverão ser consultados, pois estes estão aptos e a disposição para esclarecer qualquer dúvida que possa existir durante a execução deste trabalho. É importante destacar que ao final de cada capítulo existem algumas observações que devem ser atentamente seguidas pelas equipes. Caso alguma dessas observações não seja seguida, a nota no projeto será coerente com a falha cometida.

A correção do projeto será totalmente baseada neste documento, portanto a equipe que desenvolver o projeto seguindo as recomendações e exigências aqui contidas e conseguir implementar corretamente todas as instruções do repertório do processador não terá problemas com relação à nota que será atribuída ao seu trabalho. Não serão aceitas reclamações que contrariem as regras estabelecidas neste documento.

É importante que o aluno tenha profissionalismo ao fazer seus trabalhos e respeite as regras que estão definidas. Ressalta-se ainda que os monitores não são autorizados a relevar qualquer descumprimento das regras em qualquer parte do trabalho e, caso isso venha a ocorrer, a nota do trabalho entregue refletirá tal descumprimento. É importante lembrar que as regras servem para facilitar o trabalho de alunos e corretores. Caso elas sejam cumpridas a risca não haverá problemas com relação à correção, portanto a monitoria recomenda que as equipes façam seu trabalho de forma séria, atentos aos detalhes e sempre buscando aprender cada vez mais, pois esse é o verdadeiro objetivo que os alunos devem buscar.

Bom Trabalho!

Índice

Capítulo 1 - Especificação Inicial

Neste capítulo são apresentados os procedimentos que devem ser seguidos para a construção das duas primeiras etapas do projeto do processador descrito no segundo capítulo desta especificação. Para essas duas etapas os grupos devem usar apenas cartolinas que, preferencialmente, terão a cor branca. Caso exista alguma dúvida as equipes deverão consultar os monitores.

1.1 - Primeira Etapa: Projetando o Processador

Nesta etapa os grupos deverão projetar na cartolina a arquitetura de seu processador. O desenho apresentado deverá conter todos os circuitos que comporão o projeto e suas ligações. Além disso, o grupo deverá saber o sentido de se usar cada um dos componentes.

1.1.1 - Descrição dos módulos

Os circuitos que irão compor o projeto devem ser desenhados na cartolina tomando como base a figura 1 do Capítulo 2 desta especificação. A quantidade e quais elementos serão desenhados irá depender da forma como cada grupo decidir implementar o conjunto de instruções que é pedido neste trabalho. Apesar dos grupos terem a liberdade de escolher quantas unidades irão compor o circuito do processador, a arquitetura desenvolvida deve dar sentido a cada um dos componentes que a forma. Além disso, é recomendável que o projeto dê ênfase ao melhor aproveitamento de todos os seus componentes, ou seja, não adianta fazer um circuito com várias unidades que façam pouca ou nenhuma atividade. Deve-se sempre pensar em economia de hardware e rapidez de execução das instruções.

1.1.2 - Ligações dos Módulos

Todas as ligações que forem relevantes ao entendimento de como cada instrução é executada deverão estar presentes no desenho da cartolina, sendo que todos os sinais que partem ou chegam à unidade de controle deverão obrigatoriamente ser apresentados. Devem estar presentes também os sinais que interligam as demais unidades.

Observação: não é necessário desenhar os sinais de clock e reset que estarão presentes na maioria das unidades, pois a presença destes sinais é intuitivamente percebida. Os sinais que saem da unidade de controle deverão estar nomeados, pois isso facilita no entendimento geral do circuito.

1.1.3 - Sentido do Circuito

A equipe deverá estar consciente de como cada uma das instruções pedidas na especificação é executada no processador, ou seja, cada integrante deverá saber quais as unidades estão envolvidas nos diferentes estágios da instrução e como cada uma delas se comporta.

Esta etapa do projeto deverá ser apresentada à professora durante uma aula de laboratório com data definida na página da disciplina. Todos os integrantes deverão estar presentes e serão questionados quanto a forma como o processador opera suas instruções.

1.2 - Segunda Etapa: Desenvolvendo a Máquina de Estados da Unidade de Controle

As equipes deverão apresentar o diagrama de estados da máquina de estados da unidade de controle do processador. Tal diagrama deverá ser desenhado em uma nova cartolina e terá que conter os estados que a unidade de controle enquanto o processador executa suas instruções. Cada estado deverá conter os valores e os nomes dos sinais de saída da unidade de controle. Não é necessário especificar todos os sinais da unidade em cada estado, porém é necessária a presença de todos os sinais que foram alterados em cada estado específico.

Esta etapa do projeto também deverá ser apresentada para a professora durante uma aula de laboratório com data definida na página da disciplina. Todos os integrantes deverão estar presentes e serão questionados quanto aos estados presentes na máquina de estados desenvolvida pela equipe e quais deles estão envolvidos quando alguma das instruções é executada.

Observações:

Não é dever da monitoria avaliar estas etapas do trabalho, pois a professora faz questão de ela mesma avaliar os projetos apresentadas pelos alunos.

Durante o desenvolvimento destes trabalhos os alunos terão algumas aulas de laboratório para o acompanhamento de seus projetos. Caso ainda existam dúvidas, os alunos poderão procurar os monitores para que sejam feitos os devidos esclarecimentos.

Após o término destas etapas do projeto os alunos deverão guardar as suas descrições da arquitetura, pois elas serão muito úteis no desenvolvimento do projeto em SystemVerilog.

Recomendamos que os alunos marquem previamente um horário para o esclarecimento de suas dúvidas mandando um e-mail para o monitor responsável por sua equipe.

Capítulo 2 - Especificação da CPU

2.1 - Descrição Preliminar

O projeto a ser entregue consiste na implementação de um processador, a qual poderá ser feita com base na implementação descrita no livro texto e na figura abaixo apresentada. O processador deverá incluir três blocos básicos: unidade de processamento, unidade de controle e memória. A seguir é dada uma descrição sucinta do processador a ser projetado.

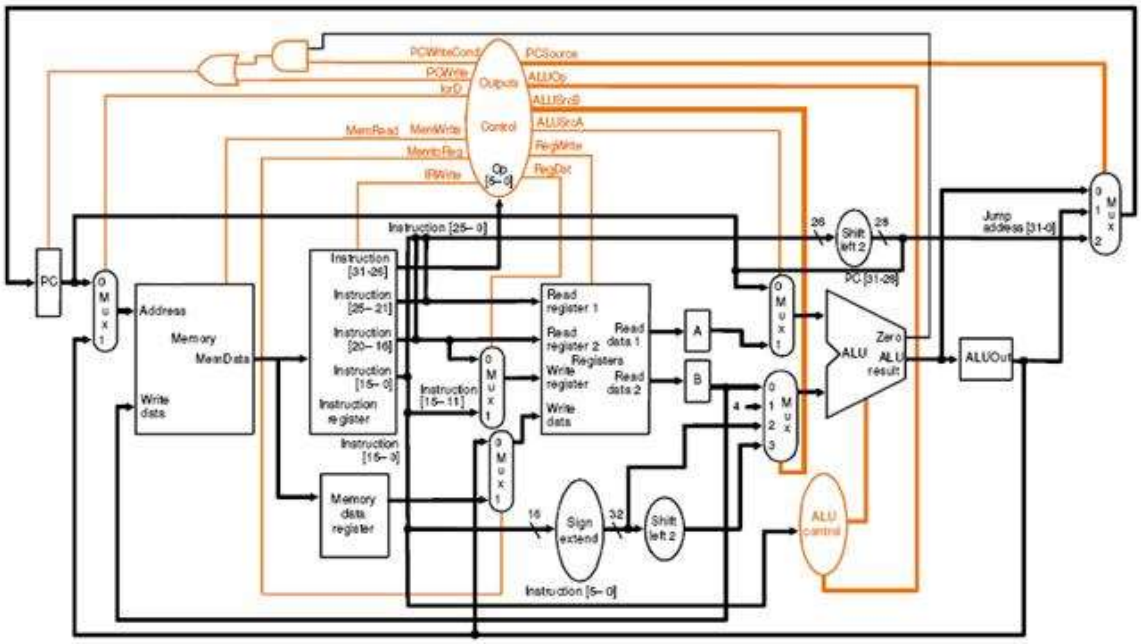


Figura 1 – Circuito base para a construção do projeto.

Inicialmente, será projetada a unidade de processamento e a seguir a unidade de controle. A unidade de processamento será desenvolvida a partir da utilização dos seguintes componentes já descritos em VHDL/SystemVerilog e fornecidos através da página da disciplina:

1. **ALU**
2. **Registrador de Deslocamento**
3. **Registrador de 32 bits**
4. **Registrador de Instruções**
5. **Banco de Registradores**
6. **Memória**

Após o projeto da unidade de processamento com a definição de todos os seus componentes e suas respectivas portas de entrada e saída, deverá ser projetada a unidade de controle, que será implementada como uma máquina de estados finita (FSM).

O processador possui 32 registradores de propósito geral, cada um de 32 bits. Um subconjunto das instruções do MIPS deverá ser implementado, de acordo com esta especificação. As instruções estão descritas neste capítulo e agrupadas por formato. Um resumo dos formatos existentes pode ser visto na Tabela 1.

Formato	[31..26]	[25..21]	[20..16]	[15..11]	[10..6]	[5..0]
R	opcode	rs	rt	rd	shamt	funct
I	opcode	rs	rt	address/immediate		
J	opcode	offset				

Tabela 1 – Formatos de instruções do MIPS

assembly		opcode	rs	rt	rd	shamt	funct	Comportamento
add	rd, rs, rt	0x0	rs	rt	rd	0x0	0x20	rd ← rs + rt
addu	rd, rs, rt	0x0	rs	rt	rd	0x0	0x21	rd ← rs + rt *
and	rd, rs, rt	0x0	rs	rt	rd	0x0	0x24	rd ← rs & rt
jr	RS	0x0	rs	-	-	0x0	0x8	PC ← rs
sll	rd, rt, shamt	0x0	-	rt	rd	shamt	0x0	rd ← rt << shamt
sllv	rd, rt, rs	0x0	rs	rt	rd	0x0	0x4	rd ← rt << rs
slt	rd, rs, rt	0x0	rs	rt	rd	0x0	0x2a	rd ← (rs < rt) ? 1 : 0
sra	rd, rt, shamt	0x0	-	rt	rd	shamt	0x3	rd ← rt >> shamt **
srav	rd, rt, rs	0x0	rs	rt	rd	0x0	0x7	rd ← rt >> rs **
srl	rd, rt, shamt	0x0	-	rt	rd	shamt	0x2	rd ← rt >> shamt
sub	rd, rs, rt	0x0	rs	rt	rd	0x0	0x22	rd ← rs - rt
subu	rd, rs, rt	0x0	rs	rt	rd	0x0	0x23	rd ← rs – rt *
xor	rd, rs, rt	0x0	rs	rt	rd	0x0	0x26	rd ← rs ^ rt
break		0x0	-	-	-	0x0	0xd	para a execução do programa
nop		0x0	-	-	-	0x0	0x0	no operation
rte		0x10	-	-	-	0x0	0x10	PC ← EPC (retorno de exceção)

Tabela 2 – Instruções tipo R

* não geram exceção por overflow, caso o resultado tenha mais que 32 bits
** essas instruções devem estender o sinal (deslocamento aritmético)

assembly		opcode	rs	rt	address/immediate	comportamento
addi	rt, rs, imm	0x8	rs	rt	imm	rt ← rs + imm ***
addiu	rt, rs, imm	0x9	rs	rt	imm	rt ← rs + imm ***
andi	rt, rs, imm	0xc	rs	rt	imm	rt ← rs & imm ***
beq	rs, rt, offset	0x4	rs	rt	offset	(ver 'desvios')
bne	rs, rt, offset	0x5	rs	rt	offset	(ver 'desvios')
lbu	rt, address (rs)	0x24	rs	rt	address	rt ← byte [address (rs)] ****
lhu	rt, address (rs)	0x25	rs	rt	address	rt ← halfword [address (rs)] ****
lui	rt, imm	0xf	-	rt	imm	rt ← imm << 16
lw	rt, address (rs)	0x23	rs	rt	address	rt ← word [address (rs)] ****
sb	rt, address (rs)	0x28	rs	rt	address	address (rs) ← byte [rt] ****
sh	rt, address (rs)	0x29	rs	rt	address	address (rs) ← halfword [rt] ****
slti	rt, rs, imm	0xa	rs	rt	imm	rt ← (rs < imm) ? 1 : 0 ***
sw	rt, address (rs)	0x2b	rs	rt	address	address (rs) ← word [rt] ****
xori	rt, rs, imm	0xe	rs	rt	imm	rt ← rs ^ imm ***

Tabela 3 - Instruções tipo I

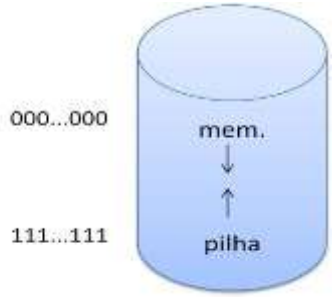
*** o valor de 'imm' deve ser estendido para 32 bits, estendendo seu sinal (bit mais significativo da constante).
**** o valor de 'address' deverá ser somado ao valor de 'rs' (registrador base) para o cálculo do endereço a ser lido da memória ou do endereço onde os dados devem ser escritos.

assembly		opcode	offset	comportamento
j	offset	0x2	offset	(ver 'desvios')
jal	offset	0x3	offset	(ver 'desvios')

Tabela 4 - Instruções do tipo J

2.2 - A Pilha

Para permitir a chamada de rotinas reentrantes e recursivas, será possível o armazenamento do registrador 31 numa estrutura do tipo pilha a qual será implementada numa parte da memória como mostrado na figura abaixo:



O endereço do topo da pilha será guardado no registrador 29 (vinte e nove) do banco de registradores, que funciona como o SP (Stack Pointer). Quando o reset é ativado este registrador deverá apontar para o endereço onde começa a pilha (nesta versão: 227). O salvamento e recuperação do registrador 31 na pilha é sempre feito por software (compilador).

2.3 - Desvios

Os desvios do MIPS podem ser classificados como desvios condicionais ou incondicionais. Os desvios condicionais realizam algum teste ou verificação para, somente então executar o desvio. Os incondicionais sempre executam o desvio. O cálculo do endereço de destino é realizado diferentemente nos dois tipos de desvio:

Desvios condicionais: O valor de 'offset' é multiplicado por 4 e somado ao PC atual. O resultado desta operação é o endereço de destino do salto (desvio), caso o resultado do teste condicional seja verdadeiro.

Exemplo: ***beq r3, r4, 0xf*** *- Supondo que r3 e r4 sejam, ambos, iguais a 0x3a e que o PC atual seja 0x1c, a próxima instrução a ser executada é a que ocupa a posição (0xf * 4) + 0x1c, que é a 0x58.*

Instruções		
Beq	rs, rt, offset	Realiza o desvio se rs for igual a rt (rs == rt).
Bne	rs, rt, offset	Realiza o desvio se rs for diferente de rt (rs != rt).

Tabela 6 - Desvios condicionais.

Desvios incondicionais: No caso de instruções que utilizem offset, este valor deve ser multiplicado por 4, resultado em um endereço de 28 bits. Como todo endereço deve possuir 32 bits, os 4 bits restantes vêm do PC atual, representando os 4 bits mais significativos do endereço de destino. No caso de instruções que utilizam registrador, o conteúdo do registrador já é o próprio endereço de destino.

Exemplos:

***jal 0x1a2b** - Supondo que o PC atual seja igual a 0xba12fa00, o endereço de destino será igual a [(PC & 0xf0000000) | (0x1a2b * 0x4)], que é igual a 0xb00068AC. Note que os 4 bits mais significativos do PC (1011) foram conservados.*

***jr 6** - Supondo que o r6 seja igual a 0x1a2b, o endereço de destino será igual ao próprio valor de r6, neste caso, 0x1a2b.*

Instruções		
j	offset	Desvia, incondicionalmente, para o endereço calculado.
jal	offset	Desvia para endereço calculado, porém salva o endereço da instrução subsequente ao jal (endereço de retorno) no registrador r31 (SEMPRE).
jr	rs	Desvio incondicional para o endereço apontado por rs.

Tabela 7 - Desvios incondicionais.

Instruções Adicionais

Operações de Rotação: abaixo temos uma tabela onde podemos ver as quatro instruções de rotação que serão executadas nesta versão do projeto. As operações de rotação também deverão ser efetuadas com o auxílio do registrador de deslocamento oferecido na página da disciplina.

assembly	opcode	rs	rt	rd	shamt	funct	Comportamento
rla rd, rt, shamt	0x0	-	rt	rd	shamt	0x0F	rd ← rt <<< shamt
rlv rd, rt, rs	0x0	rs	rt	rd	0x0	0x1F	rd ← rt <<< rs
rra rd, rt, shamt	0x0	-	rt	rd	shamt	0x2F	rd ← rt >>> shamt
rrv rd, rt, rs	0x0	rs	rt	rd	0x0	0x3F	rd ← rt >>> rs

Operações de Desvio Condicional: abaixo temos uma tabela onde podemos ver as operações de desvio condicional adicionais que deverão ser implementadas no semestre. As instruções possuem o formato imediato.

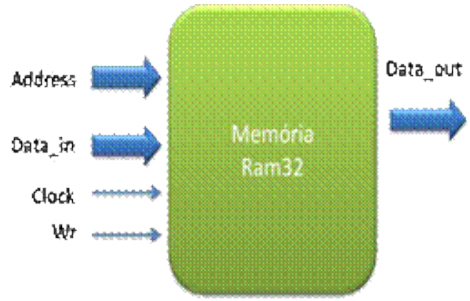
assembly	opcode	rs	rt	Offset	Comportamento
BGEZ rs, offset	0x01	-	0x01	offset	if (RS >= 0) then PC = PC + signextend(offset)*4
BGEZAL rs, offset	0x01	-	0x11	offset	if (RS >= 0) then r31= PC; PC = PC + signextend(offset)||00
BGTZ rs, offset	0x07	-	0x00	offset	if (RS > 0) then PC = PC + signextend(offset)*4
BLEZ rs, offset	0x06	-	0x00	offset	if (RS<= 0) then PC = PC + signextend(offset)*4
BLTZ rs, offset	0x01	-	0x00	offset	if (RS < 0) then PC = PC + signextend(offset)*4
BLTALL rs, offset	0x01	-	0x12	offset	if (RS < 0) then r31=PC; PC = PC + signextend(offset)*4

2.5 - Componentes Fornecidos

Nesta seção são descritos os seis componentes do projeto que são fornecidos pela equipe da disciplina. Tais componentes poderão ser baixados a partir da página da disciplina.

2.5.1 - Memória

A memória usada no projeto possuirá palavras de 32 bits com endereçamento por byte. Apesar de o endereço possuir 32 bits, a memória só possui 256 bytes. As entradas e saídas da memória são:



1. **add : entrada de 32 bits**
   O endereço de entrada da memória (32 bits).
2. **Data_in : entrada de 32 bits**
   A entrada de Dados da memória (32 bits).
3. **clokc : entrada de 1 bit**
   Bit de clock comum a todo o "computador".
4. **wr : entrada de 1 bit**
   O bit que diz se vai ler ou escrever.
5. **Data_out : saída de 32 bits**
   A saída de Dados da memória (32 bits).

Peculiaridades da Memória:

1. Enquanto o bit "wr" estiver com o valor 0 (zero) ele estará lendo, quando estiver com o valor 1 (um) ele estará escrevendo.
2. A memória está trigada na subida do clock.
3. Ao se fazer uma requisição de leitura, o valor pedido só estará disponível no próximo pulso de clock após o clock onde foi feito a requisição.
4. A escrita leva apenas um ciclo.
5. Instruções e dados serão armazenados na memória usando a estratégia *big-endian*.

2.5.2 - Registrador de 32 bits

Para armazenar instruções e dados, bem como o endereço de instruções serão utilizados registradores de 32 bits, conforme ilustrado abaixo.

2.5.3 - Banco de Registradores

O banco de registradores é composto por 32 registradores de 32 bits cada. Dois registradores podem ser visíveis simultaneamente.

A leitura dos registradores é combinacional, isto é se os valores nas entradas **ReadRegister1** ou **ReadRegister 2** forem alteradas, os valores nas saídas **ReadData1** ou **ReadData2** podem ser alterados. O registrador a ser escrito é selecionado pela entrada **WriteRegister** e quando a entrada **RegWrite** é ativada (igual a 1) o registrador selecionado recebe o conteúdo da entrada **WriteData**. O sinal de **reset** limpa todos os registradores e é assíncrono.

2.5.4 - Registrador de Deslocamento

O registrador de deslocamento deve ser capaz de deslocar um número inteiro de 32 bits para esquerda e para a direita. No deslocamento a direita o sinal pode ser preservado ou não.

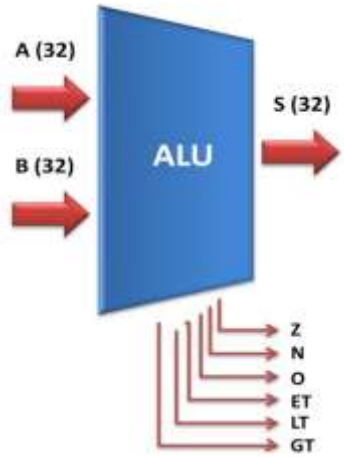
O número de deslocamentos pode variar entre 0 e 7 e é especificado na entrada **n (3 bits)** do registrador de deslocamento. A funcionalidade desejada do registrador de deslocamento é especificada na entrada **shift** (3 bits menos significativos) conforme figura abaixo. As atividades discriminadas na entrada shift são síncronas com o **clock (ck)** e o **reset** é assíncrono.

shift	Descrição
000	Nada a fazer.
001	Load no registrador.
010	Shift a esquerda **n** vezes.
011	Shift a direita lógico **n** vezes.
100	Shift a direita aritimético **n** vezes.
101	Rotação a direita **n** vezes.
110	Rotação a esquerda **n** vezes.

2.5.5 - Unidade Lógica e Aritmética (ALU)

A Unidade Lógica e Aritmética (ALU) é um circuito combinacional que permite a operação com números de 32 bits na notação complemento a dois. A funcionalidade é especificada pela entrada F conforme descrito na tabela abaixo.

Função	Operação	Descrição	Flags
000	S = A	Carrega A	Z, N
001	S = A + B	Soma	Z, N, O
010	S = A – B	Subtração	Z, N, O
011	S = A and B	And lógico	Z
100	S = A + 1	Incremento de A	Z, N, O
101	S = not A	Negação de A	Z
110	S = A xor B	OU exclusivo	Z
111	S = A comp B	Comparação	EG, GT, LT



2.6 - Tratamento de Exceções

O processador deve incluir tratamento de dois tipos de exceções: opcode inexistente e **overflow**. Na ocorrência de uma exceção, o endereço da instrução que causou a exceção deverá ser salvo no registrador **EPC** a ser inserido na unidade de processamento e o **PC** será carregado, posteriormente, com o valor do endereço da rotina de tratamento. O byte menos significativo do endereço da rotina de tratamento está armazenado nos seguintes endereços de memória: **254** (opcode inexistente) e **255** (overflow).

A rotina de tratamento estará armazenada na memória a partir do endereço **228** (decimal) até o endereço **251** (decimal). Esta rotina armazenará o valor 1 no registrador 30 e parar a execução de um programa no caso de um **opcode inexistente**. No caso de um **overflow**, o valor 2 deverá ser armazenado no registrador 30 e a execução do programa deve continuar.

Basicamente os passos que devem ser implementados após ocorrer uma exceção são os seguintes:

1 - Ao ser detectada uma exceção o endereço da instrução que a causou deve ser salvo no EPC.

2 - O byte localizado na posição de memória 254 ou 255 (dependendo da exceção) deverá ser lido e estendido para 32 bits. Essa extensão é feita apenas com zeros.

3 - O número de 32 bits resultante do passo anterior deverá ser o novo valor de PC. Após completar tais passos a rotina de tratamento de exceção que está armazenada na memória será executada, fazendo assim por software a tarefa de armazenar o valor adequado no registrador 30.

Observação: veja que os passos que acabam de ser descritos não são referentes à instrução rte, pois esta instrução deve apenas armazenar em PC o valor contido atualmente em EPC.

Observação:

As instruções addi, addiu, lui, lw, lhu e lbu são de extrema importância para o projeto, pois estas instruções sevem para carregar dados da memória ou da própria instrução para que operações sejam realizadas e valores sejam armazenados nos registradores da CPU. Sem tais instruções não é possível avaliar as outras, pois não será possível carregar valores e operá-los. Portando, caso algum projeto apresente nenhuma das instruções anteriores funcionando, o mesmo terá consideradas erradas todas as instruções que foram implementadas.

Capítulo 3 - Especificação do Relatório

Neste capítulo são apresentados os componentes que devem estar presentes nos relatórios apresentados pelas equipes. As regras aqui descritas devem ser obrigatoriamente seguidas. Caso exista alguma dúvida, a equipe deverá entrar em contato com o monitor responsável.

3.1 - Formato do Relatório

O relatório deverá conter as seguintes partes:
a) Capa
b) Índice
c) Introdução
d) Descrição dos Módulos
e) Descrição das Operações
f) Descrição dos Estados do Controle
g) Conjunto de Simulações
h) Conclusão

É obrigatória a presença de todas as partes citadas acima e uma descrição mais detalhada do conteúdo de cada uma delas é apresentada na próxima seção.

3.2 - Descrição das Partes do Relatório

3.2.1 - Capa

A capa deve conter um cabeçalho com o nome da universidade e do centro onde a disciplina é ensinada. Deve também apresentar o título: **Relatório de Projeto** (título único que deve estar presente em todos os relatórios). O nome dos alunos que compõem a equipe deve vir logo em seguida, juntamente com seus logins, e por último deve vir a data da entrega do projeto.

Não será necessária nenhuma imagem na capa que tenha como simples objetivo embelezar o trabalho. A capa padrão é definida no molde já descrito, portanto não faça além do que foi pedido.

3.2.2 - Índice

Um índice simples deve compor o relatório para que a correção seja facilitada.

3.2.3 - Introdução

A introdução deve conter os objetivos pretendidos na construção do relatório e deve dar uma visão geral do que será apresentado no relatório.

3.2.4 - Descrição das Módulos

Cada dos módulos que forem implementados* pelas equipes deve ser descrito em detalhes. O objetivo almejado ao construir tais módulos deve estar claramente apresentado, bem como sua funcionalidade dentro do escopo do projeto. Os sinais de entrada e saída devem ser especificados assim como o conjunto de atividades executadas pelo algoritmo interno de cada um para prover os resultados esperados.

Exemplo: Usamos como exemplo a descrição do registrador de 32 bits que é fornecido juntamente à especificação do projeto.

Módulo: Registrador de 32 bits.

Entradas:

Clk (1 bit): representa o clock do sistema.
Reset (1 bit): sinal que, quando ativado zera o conteúdo do registrador.
Load (1 bit): sinal que ativa o carregamento de um valor de entrada no registrador.
Entrada (32 bits): vetor de bits que será armazenado no registrador.
Saida (32 bits): vetor de bits que contém o valor armazenado no registrador.

Objetivo:

Módulo criado para armazenamento temporário de valores, que na implementação multiciclo devem ser preservados durante à execução de cada estágio das instruções.

Algoritmo:

Quando valor de Load está ativado e ocorre a ativação do sinal Clk o vetor de bits Entrada é disponibilizado como o sinal de Saida até que o sinal de load esteja novamente ativado na subida de Clk. Caso o sinal Clear esteja ativado o valor da saída passa a ser o vetor zero.

*Multiplexadores não devem ser descritos, pois se3u funcionamento é trivial

3.2.5 - Descrição das Operações

Nesta parte deverão ficar todos os passos que compõem a execução de cada uma das instruções exigidas no projeto destacando todas os módulos envolvidas na sua execução. Cada instrução deverá ser descrita separadamente.

Exemplo:

Instrução add reg1, reg2, reg3

Após identificar a instrução add no estágio de decodificação os valores dos registradores r2 e r3 são carregados a partir do banco de registradores e a operação de soma é realizada na ULA de acordo com o sinal de controle que vem da unidade de controle, posteriormente o resultado é gravado no registrador de destino r1 e uma nova instrução é buscada na memória.

3.2.6 - Descrição dos Estados do Controle

O objetivo aqui é que as equipes apresentem a descrição de cada um dos estados do controle do processador desenvolvido. Não será necessário que o aluno especifique o valor que cada sinal deve receber dentro do estado, pois isso já estará claro no código do projeto, porém é imprescindível que seja descrito o que cada estado deve fazer, ou seja, o que a equipe objetivava ao construir cada um. Deverá ser apresentado também o diagrama de estados do controle em forma de figura que obrigatoriamente deve estar dentro desta seção do relatório (Atenção: não será aceita a visualização do diagrama de estados gerada pelo software Quartus. Cada equipe deve se dar ao trabalho de gerar uma figura que **represente de forma clara** o que foi pedido).

Exemplo:

Estado: Decodifica

Neste estado o opcode e o function (quando necessário) são analisados. Isso define qual o novo estado que a máquina deve assumir para continuar o processamento.

3.2.7 - Conjunto de Simulações

É dever da equipe apresentar nesta seção ao menos uma simulação de cada uma das instruções que tiveram sua implementação exigida na especificação do projeto. Essa apresentação deverá consistir de uma imagem do relatório de simulação gerado pelo software Quartus e sua explicação detalhada. Nesta explicação deverá ficar claro o que cada sinal de entrada e saída envolvido na simulação representa, seus valores e o resultado esperado ao executar a operação.

Em cada simulação de instrução é importante que estejam presentes os valores dos registradores **PC, EPC, MDR, IR e Registrador 29**. Além desses, os **registradores envolvidos na operação** que está sendo testada deverão ser apresentados na simulação. Ainda é exigida a presença dos sinais de **clock** e **reset** e do **sinal que indica em quais estados a máquina de estados da unidade de controle passou durante a simulação da instrução em questão.**

Atenção:

a) Descrição dos sinais: Todos os sinais que importem na observação dos valores de entrada o dos resultados de uma operação específica devem estar descritos literalmente. A figura que prove a execução correta ou não da instrução deve ser disponibilizada logo após a descrição dos sinais.

b) As equipes devem nomear os sinais presentes na simulação de forma que fique claro na imagem o que cada um representa.

c) Deverá haver uma descrição dos sinais e uma figura para cada instrução testada.

Deverá está presente também a simulação (figura e descrição dos sinais envolvidos) do funcionamento de cada um dos módulos que forem **implementadas pelos alunos**, ou seja, cada unidade projetada pelos alunos deve ser simulada separadamente.

Exemplo:

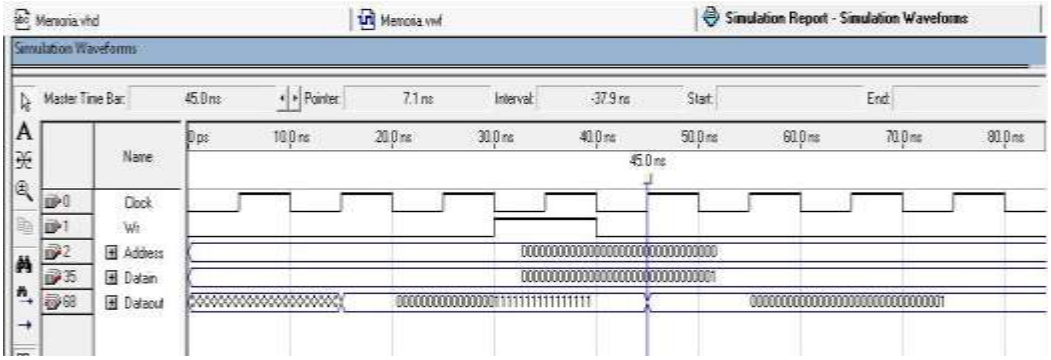
Módulo: Memoria

Descrição das Portas:

Clock: *representa o clock do sistema.*

Wr: *sinal que indica se a memória irá efetuar a leitura de dados (quando possui valor zero) ou a escrita (quando possuir valor um).*

Address: *vetor que indica o valor do endereço a ser lido ou escrito.*
Datain: *vetor que contém a palavra a ser escrita na memória.*
Dataout: *vetor que contém a palavra lida da memória.*



Descrição da Simulação:

Nos primeiros três ciclos é executada a leitura da palavra armazenada no endereço zero da memória. Vale destacar que a palavra presente em tal posição é 00000000000000001111111111111111. No quarto ciclo de clock o valor do vetor Datain é escrito no mesmo endereço que estava sendo lido. No quinto ciclo de clock o valor da nova palavra armazenada no endereço zero é lida.

Atenção:

a) Não se faz necessária a simulação dos módulos que forem fornecidos pela equipe da disciplina.

b) Não se faz necessária a presença das simulações dos multiplexadores implementados pelas equipes, pois seu funcionamento trivial descarta qualquer necessidade de prova de sua validade no projeto.

3.2.8 - Conclusão

Deve apresentar de forma simples os resultados obtidos e trazer uma análise que mostre se os objetivos apresentados na introdução foram alcançados.

Observações:

As regras descritas nesse capítulo devem ser seguidas a risca na elaboração do relatório, sendo que reclamações que não estejam de acordo com o que foi exigido nessa especificação não serão consideradas válidas.

Os monitores serão extremamente rígidos com relação a tais regras, portanto os alunos devem ler atentamente a especificação, entender o que foi pedido para, posteriormente, fazer o relatório.

O relatório deverá ser entregue impresso e encadernado na data especificada na página da disciplina. Os relatórios que não forem pontualmente entregues não serão em hipótese alguma aceitos e o mesmo se aplica aos relatórios que não estiverem impressos, ou seja, não adianta tentar entregar o arquivo em formato digital para posteriormente ser entregue o relatório impresso, pois isso não será aceito.

CRONOGRAMA DE AVALIAÇÃO.

- Avaliação do Projeto da Unidade de Processamento – 16/10/15

- Avaliação do Projeto da Unidade de Controle – 23/10/15

- Entrega e Apresentação do projeto: 13/11/15

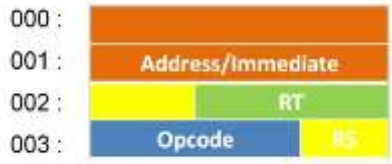
Observação: todos os componentes da equipe devem estar presentes nas avaliações do projeto

Anexo: Instruções para a Configuração e Carregamento da Memória.

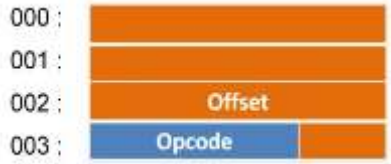
A.1 - Configurando o arquivo .mif

Para o teste do projeto o arquivo .mif deverá ser configurado com um conjunto de instruções específico. Isso é feito com a inserção das instruções nas posições de memória que estão marcadas na margem esquerda de cada linha do arquivo .mif. As figuras abaixo mostram como dispor as instruções de cada tipo nas posições de memória.

Instrução do tipo R



Instrução do tipo I



Instrução do tipo J

O arquivo .mif que é fornecido juntamente aos componentes do projeto deve possuir configuradas as rotinas de tratamento de exceção e seus endereços. As outras posições encontram-se apenas com instruções nop. Cabe a cada grupo configurar a memória com as instruções para o teste de seu projeto.

A.2 - Configurando a memória

Na posição do arquivo Memoria.vhd destacada abaixo insira o nome do arquivo .mif para ele passar a ser parte do seu projeto.

```
package ram_constants is
      constant DATA_WIDTH : INTEGER := 8;
      constant ADDR_WIDTH : INTEGER := 8;
      constant INIT_FILE : STRING := "nomeArquivo.mif";
end ram_constants;
```

Faça a sintese de seu projeto e os testes. Caso seja necessário mudar o arquivo .mif, uma nova Netlist deverá ser gerada.