Universidade Federal de Pernambuco - UFPE

Centro de Informática - CIn

Infra-Estrutura de Hardware - if674cc

Especificação de Projeto

Recife - 2019.2

Infra-Estrutura de Hardware – if674cc
Especificação de Projeto

Documento desenvolvido pela monitoria da disciplina Infra-Estrutura de Hardware sob orientação da professora Drª. Edna Barros. Modificado pelo professor Dr. Adriano Sarmento

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 ao professor 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 o professor 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:

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 cartolinas, pois elas serão muito úteis no desenvolvimento do projeto em Verilog.

Recomendamos que os alunos marquem previamente um horário para o esclarecimento de suas dúvidas mandando um email 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.

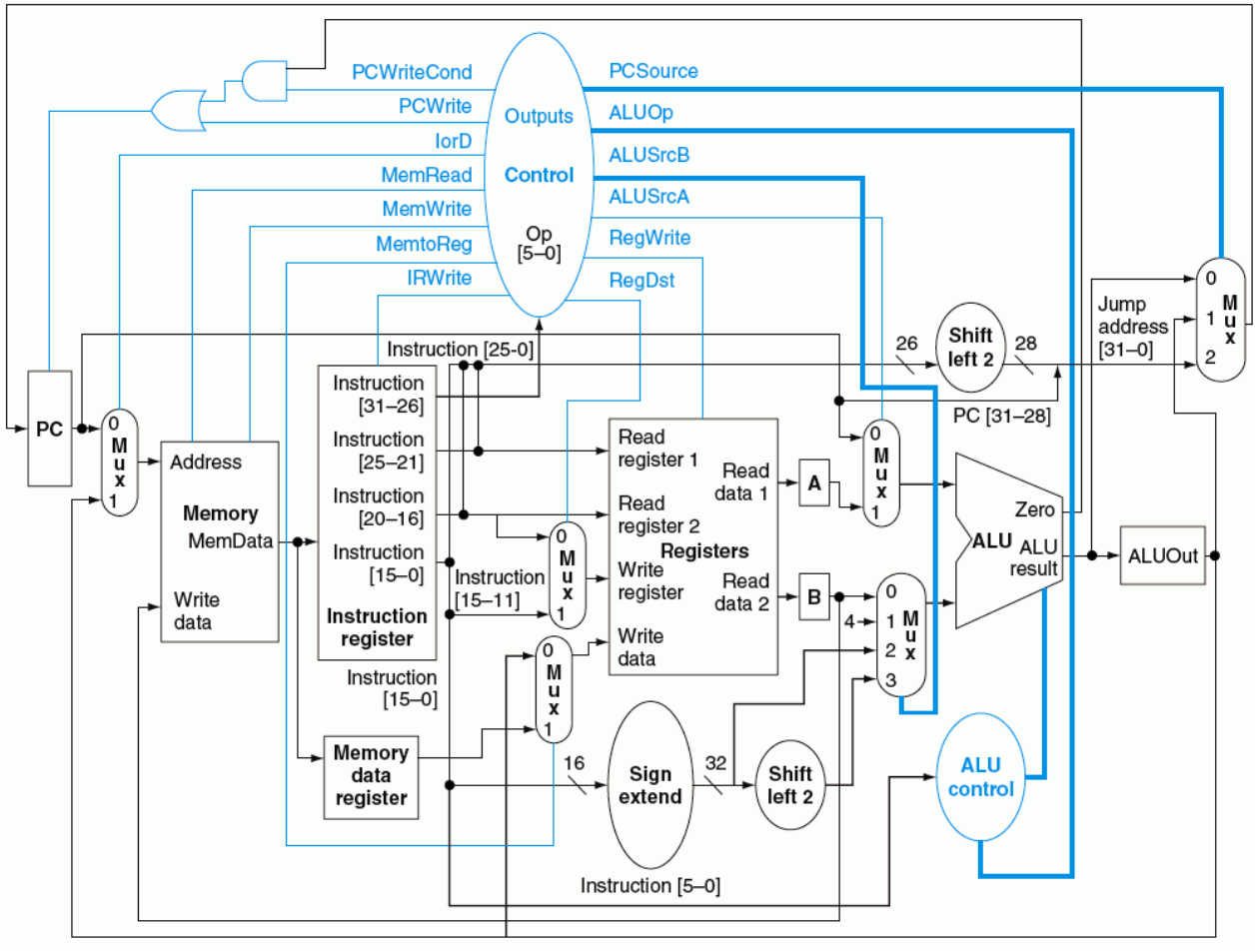
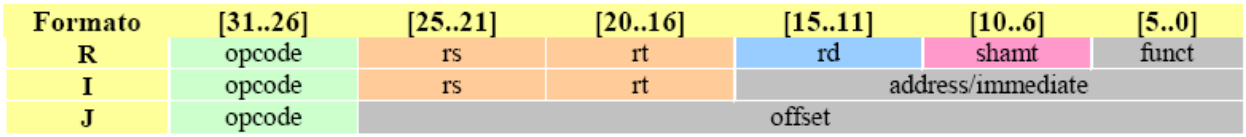


Figure 1 Circuito base para construção do projeto

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 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, sendo, 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 Figure 2.



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				

Figure 2 Formatos de Instruções

Assembly	Opcode	rs	rt	rd	shamt	funct	Comportamento
add rd, rs, rt	0x0	rs	rt	rd	0x0	0x20	rd ← rs + rt
and rd, rs, rt	0x0	rs	rt	rd	0x0	0x24	rd ← rs & rt
div rs, rt	0x0	rs	rt	-	0x0	0x1a	**(Ver divisão)**
mult rs,rt	0x0	rs	rt	-	0x0	0x18	**(Ver multiplicação)**
jr rs	0x0	rs	-	-	0x0	0x8	PC ← rs
mfhi rd	0x0	-	-	rd	0x0	0x10	rd ← hi
mflo rd	0x0	-	-	rd	0x0	0x12	rd ← lo
sll rd, rt, shamt	0x0	-	rt	rd	shamt	0x0	rd ← rt << shamt
sllv rd, rs, rt	0x0	rs	rt	rd	0x0	0x4	rd ← rs << rt
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, rs, rt	0x0	rs	rt	rd	0x0	0x7	rd ← rs >> rt*
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
break	0x0	-	-	-	0x0	0xd	PC não se altera
Rte	0x0	-	-	-	0x0	0x13	PC ← EPC
xchg rs,rt	0x0	rs	rt	-	0x0	0x5	rs ↔ rt, troca o conteúdo dos 2 registradores

Table 1 Instruções Formato R

* instruções devem estender o sinal (deslocamento aritmético)

Assembly	Opcode	rs	rt	End/Imediato	Comportamento
addi rt, rs imediato	0x8	rs	rt	imediato	rt ← rs + imediato**
addiu rt, rs imediato	0x9	rs	Rt	imediato	rt ← rs + imediato**
beq rs,rt, offset	0x4	rs	rt	offset	Desvia se rs == rt (ver desvios)
bne rs,rt, offset	0x5	rs	rt	offset	Desvia se rs != rt (ver desvios)
ble rs,rt,offset	0x6	rs	rt	offset	Desvia se rs <= rt (ver desvios)
bgt rs,rt,offset	0x7	rs	rt	offset	Desvia se rs > rt (ver desvios)
blm rs, rt, offset	0x1	rs	rt	offset	Desvia se Mem[rs] < rt (ver desvios)

lb rt, offset(rs)	0x20	rs	rt	offset	rt ← byte Mem[offset + rs]
lh rt, offset(rs)	0x21	rs	rt	offset	rt ← halfword Mem[offset + rs]
lui rt, imediato	0xf	-	rt	imediato	rt ← imediato << 16
lw rt, offset(rs)	0x23	rs	rt	end	rt ← Mem[offset + rs]
sb rt, offset(rs)	0x28	rs	rt	offset	Mem[offset +rs] ← byte[rt] (ver stores)
sh rt, offset(rs)	0x29	rs	rt	offset	Mem[offset +rs] ← halfword[rt] (ver stores)
slti rt, rs, imediato	0xa	rs	rt	imediato	rt ← (rs < imediato) ?1 :0
sw rt, offset(rs)	0x2b	rs	rt	offset	Mem[offset +rs] ← rt

Table 2 Instruções Formato I

** o valor de 'imediato' deve ser estendido para 32 bits, estendendo seu sinal (bit mais significativo da constante).

Assembly	Opcode	Offset/Endereco	Comportamento
j offset	0x2	offset	**(ver desvios)**
jal offset	0x3	offset	**(ver desvios)**

Table 3 Instruções Formato J

2.2 - A Pilha e o Reset

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:

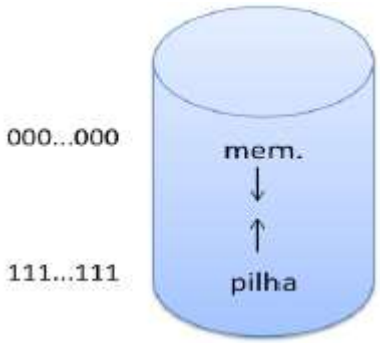


Figure 3 Pilha

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 - Divisão

A divisão deve ser implementada como um componente à parte, o **Divisor**, não sendo recomendada a reutilização da ALU principal do processador para efetuar qualquer operação referente a esta função. Este componente deve realizar corretamente a divisão de dois números de 32 bits, com sinal, resultando em um número de 32 bits, também com sinal.

Além das entradas A e B (valores de 32 bits a serem divididos), esse componente deve ter um sinal de **clock** e **reset**, que serão ligados ao sinal de clock e reset globais. O reset deve limpar TODOS os registradores internos do Divisor, inclusive registradores temporários e o(s) registrador(es) de resultado. É importante que haja uma comunicação bidirecional entre a unidade de controle e o divisor.

O Divisor deve ter também um sinal de saída que indica se houve divisão por zero. Este sinal deve servir para lançar a exceção de divisão por zero (ver exceções).

Dica: Ler o capítulo do livro texto que trata sobre operações aritméticas em computadores

Instrução	Descrição
div rs, rt	Realiza a divisão entre rs e rt, armazenando internamente (em dois registradores próprios do divisor) o resultado.
mfhi rd	Copia a parte alta do resultado da divisão - o resto (32 bits mais significativos) para o registrador rd.
mflo rd	Copia a parte baixa do resultado da divisão - quociente (32 bits menos significativos) para o registrador rd.

Importante: Note que o resultado da divisão inteira é armazenado em dois registradores. O registrador Hi deve conter o resto e o registrador Lo o quociente.

2.4 Multiplicação

A multiplicação deve ser implementada como um componente à parte, o **Multiplicador**, não sendo recomendada a reutilização da ALU principal do processador para efetuar qualquer operação referente a esta função. Este componente deve implementar o **algorítmo da multiplicação binária sinalizada de Booth**, realizando corretamente a multiplicação de dois números de 32 bits, com sinal, resultando em um número de 64 bits, também com sinal. O número mínimo de ciclos necessários para a conclusão da multiplicação, segundo o algoritmo de Booth, é de **32 ciclos**. Uma vez que este é o melhor resultado possível, será dada uma tolerância, de forma que **multiplicadores de até 40 ciclos também serão aceitos.**

Além das entradas A e B (valores de 32 bits a serem multiplicados), esse componente deve ter um sinal de **clock** e **reset**, que serão ligados ao sinal de clock e reset globais. O reset deve limpar TODOS os registradores internos do Multiplicador, inclusive registradores temporários e o(s) registrador(es) de resultado. É importante que haja uma comunicação bidirecional entre a unidade de controle e o multiplicador.

Instrução		Descrição
mult	rs, rt	Realiza a multiplicação entre rs e rt, armazenando internamente (em dois registradores próprios do multiplicador) o resultado.
mfhi	rd	Copia a parte alta do resultado da multiplicação (32 bits mais significativos) para o registrador rd.
mflo	rd	Copia a parte baixa do resultado da multiplicação (32 bits menos significativos) para o registrador rd.

Importante: Note que o resultado da multiplicação inteira é armazenado em dois registradores. O registrador Hi deve conter os 32 bits mais significativos e o registrador Lo os bits menos significativos.

2.5 - 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 + 4. 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 + 4 seja 0x1c, a próxima instrução a ser executada é a que ocupa a posição (0xf * 4) + 0x1c, que é a 0x58.*

Note que no caso da instrução ***blm rs,rt, offset***, o registrador ***rs*** armazena um endereço de memória do valor que deve ser comparado com o valor que está em ***rt***. Se o valor contido na memória for menor que o valor contido no registrador, desvia-se.

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.

2.6 - Stores

Nas instruções ***sb*** (***store byte***) e ***sh*** (***store halfword***) deve-se ter cuidado para não sobrescrever a palavra toda (4 bytes) na memória. A memória tem como entradas um endereço de 4 bytes e uma palavra para ser armazenada. Deve-se apenas salvar um byte (no caso de um sb) no endereço especificado e não substituir a palavra toda que está armazenada na memória pelo byte seguido de zeros.

2.7 - 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.

2.7.1 - Memória

A memória usada no projeto possuirá palavras de 32 bits com endereçamento por byte. Apesar do 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. **clock : 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).

Figure 4 Memória

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 *little-endian.*

2.7.2 - Registrador de 32 bits

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

Figure 5 Registrador de 32 bits

2.7.3 - 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.

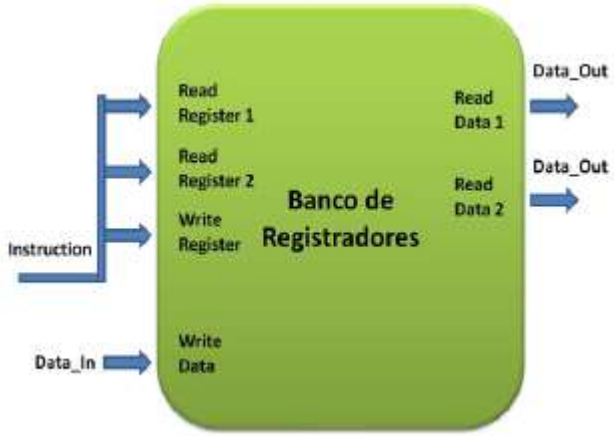


Figure 6 Banco de registradores de 32 bits

2.7.4 - 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 31 e é especificado na entrada **n (5 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.

Figure 7 Registrador de Deslocamento

2.7.5 - Unidade Lógica e Aritimética (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

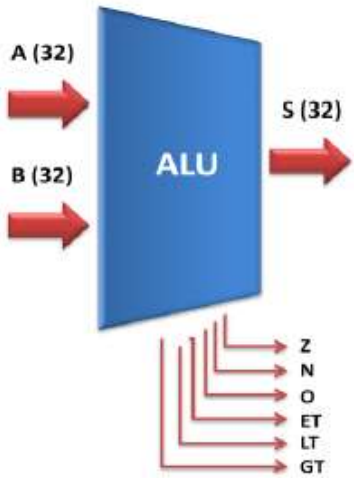


Figure 8 ALU

2.8 - Tratamento de Exceções

O processador deve incluir tratamento de três tipos de exceções: **opcode inexistente, overflow e divisão por zero**. 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: **253** (opcode inexistente) e **254** (overflow) e **255** (divisão por zero).

Basicamente, os passos devem ser implementados no projeto após ocorrer uma exceção são os seguintes:

1. O endereço da instrução que causou a exceção deve ser salvo no EPC.

2. O byte localizado na posição de memória 253, 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.

Observação: veja que os passos descritos acima 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, lh e lb 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.

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) Unidade de Processamento e) Descrição das Entidades f) Descrição das Operações g) Descrição dos Estados do Controle h) Conjunto de Simulações i) 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 - Unidade de Processamento

Esta seção deve conter o diagrama de blocos da Unidade de Processamento. Caso seja difícil colocar o diagrama no relatório, ou o diagrama fique ilegível, deve-se entregar a cartolina que contém o diagrama de blocos final da unidade de processamento.

3.2.5 - Descrição das Entidades

Cada uma das entidades que forem implementadas* pelas equipes deve ser descrita em detalhes. O objetivo almejado ao construir tais entidades 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 uma para prover os resultados esperados.

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

Unidade: Registrador de 32 bits.

Entradas:

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

Objetivo: Unidade criada 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 seu funcionamento é trivial

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). Caso seja difícil colocar o diagrama no relatório, ou o diagrama fique ilegível, deve-se entregar a cartolina que contém o diagrama de estados final da unidade de controle.

Exemplo:

Estado: Decodifica

Neste estado o opcode e o function (guando 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. A explicação deverá consistir de todos os passos que compõem a execução de cada uma das instruções exigidas no projeto destacando todas as entidades envolvidas na sua execução.

Em cada simulação de instrução é importante que estejam presentes os valores dos registradores **PC, EPC, MDR, IR, Registrador 29 e Registrador 31**. Além desses, os **registradores do banco de 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 estar presente também a simulação (figura e descrição dos sinais envolvidos) do funcionamento de cada uma das entidades que forem **implementadas pelos alunos**, ou seja, cada unidade projetada pelos alunos deve ser simulada separadamente.

Exemplo:

Entidade: 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.

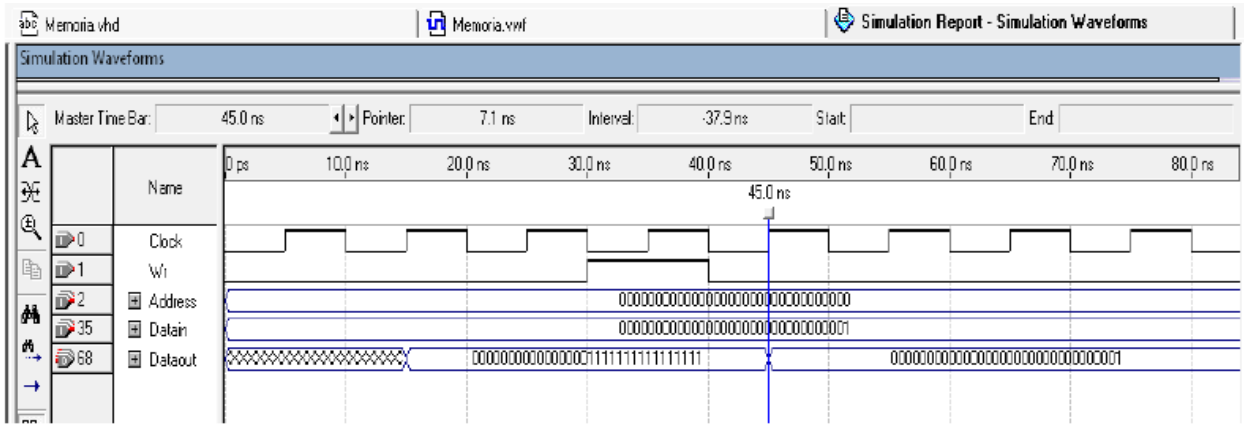


Figure 9 Snapshot da Simulação

Atenção:

a) Não se faz necessária a simulação das entidades que forem fornecidas 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:

O relatório deverá ser entregue impresso e encadernado na data especificada na página da disciplina. No dia da apresentação final do projeto, o relatório em formato eletrônico juntamente com a implementação deverão ser enviados até meia-noite. Os relatórios impressos deverão ser entregues na aula seguinte da disciplina.